Journal of Information Systems Engineering and Management
2025, 10(2)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Advanced License Plate Recognition with Squeezenet Effeciency

Shajan Jacob[1*], M.K Jeyakumar[2]
[1] *Department of Computer Science and Engineering, Noorul Islam Centre for Higher Education Kumaracoil, Tamil Nadu, India*
[2] *Department of Computer Science and Engineering, Noorul Islam Centre for Higher Education Kumaracoil, Tamil Nadu, India*

ARTICLE INFO

Received: 10 Oct 2024

Revised: 14 Dec 2024

Accepted: 26 Dec 2024

ABSTRACT

License Plate Recognition (LPR) holds immense importance within the world of Intelligent Transportation Systems (ITS) due to its diverse applications. This study explores the implementation of a SqueezeNet model for LPR, addressing the critical role of intelligent transportation systems. By utilizing deep learning (DL) and image processing techniques, the purpose of the study is to enhance the accuracy and efficiency of automatic license plate recognition (ALPR). The characters on license plates are identified and recognized from a variety of vehicle images using the SqueezeNet architecture, which is well-known for its lightweight construction and computational effectiveness. The main steps in this study includes image processing to optimize the input quality, followed by character segmentation and recognition using the trained SqueezeNet model. The experimental results illustrates that the model achieves 99.65% of accuracy, thereby highlighting the model's efficiency in real world applications such as automated toll collection and traffic monitoring. This study underscores the transformative potential of modern DL approaches in advancing ALPR systems.

Keywords: License plate recognition, SqueezeNet model, Intelligent transportation system, Image recognition, Deep learning, Object detection.

1. INTRODUCTION

Tracking and managing license plate identification is a crucial aspect of Intelligent Transportation Systems (ITS). They are frequently utilized in high-efficiency roadway services, access control security, automatic driving, and the prevention of auto theft [1]. LPR technology has become a cornerstone in modern transportation management systems, offering a robust solution for vehicle identification shown in Figure 1. As urban areas continue to grow and vehicular traffic increases, the need for accurate and efficient methods to monitor and manage vehicles is more critical than ever. LPR systems automatically read and interpret license plates using advanced image processing and machine learning techniques, facilitating applications such as toll collection, , automated traffic law enforcement, parking management and vehicle access control.

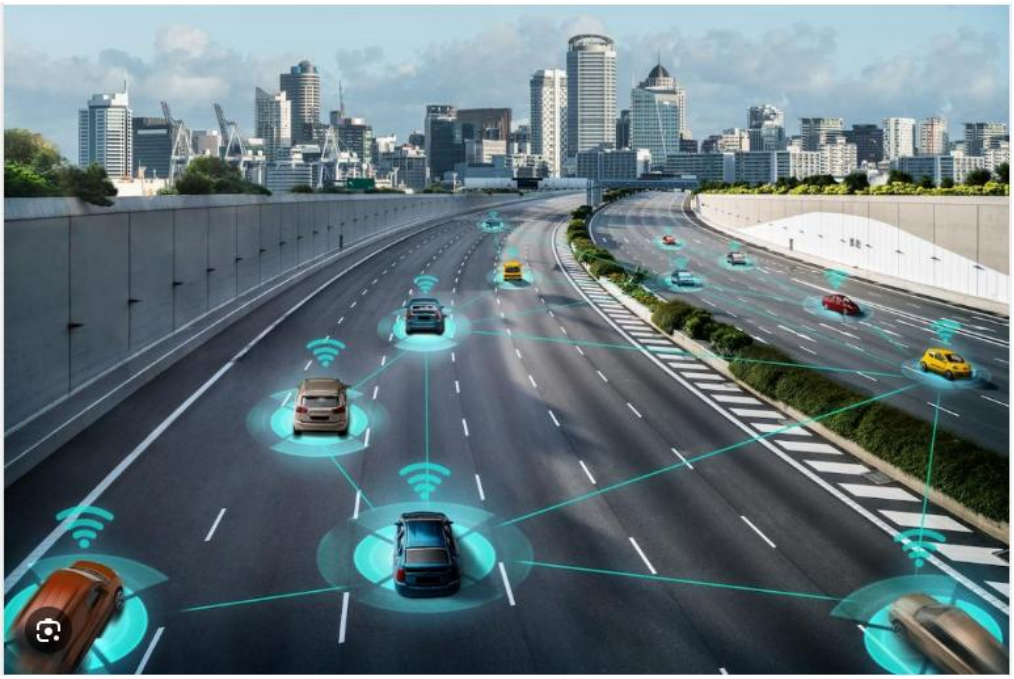

Fig. 1: Visualization of detection and recognition of license plate.

In license plate recognition task, the characters on the license plates can have different languages. Additionally, vehicle license plates can vary in sizes, shapes, orientations, conditions, and colors, making them difficult for detection as shown in Figure 2 [2]. A complete ALPR system depends on both License plate detection and character recognition. The three steps of an, ALPR pipeline is character recognition, character segmentation, and detection of License Plate [3]. The process of license plate detection is identifying the license plate from a specific image. From the detected license plate, individual characters are separated using character segmentation, and each segmented character is then classified using character recognition. Since the first two processes have a direct impact on the character identification stage, they are essential for accurate ALPR. If the license plate localization fails in the first stage it leads to failures in subsequent stages. To address this issue, some studies combine the character segmentation and recognition steps into a single object recognition process. [4].

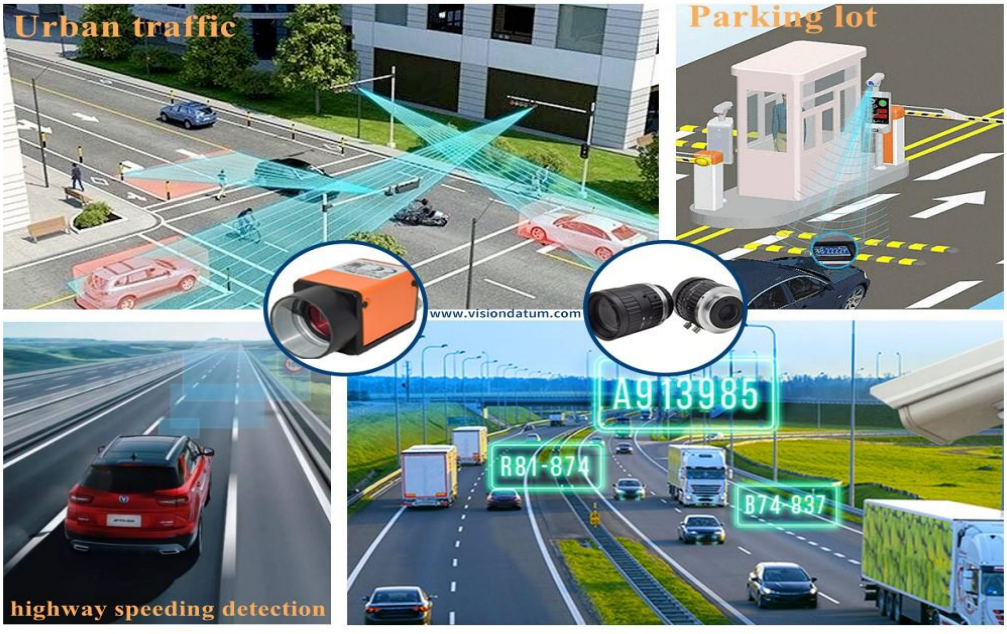

Fig. 2: Various application for License Plate Recognition

Traditional methods of LPR often depend on computationally intensive models that require significant resources, making them less practical for real-time applications on resource-constrained devices. The core of LPR technology involves the combination of several intricate processes. These include capturing images, enhancing and preparing these images, separating individual characters from the license plate, and finally using Optical Character Recognition (OCR) to identify and read the characters. These systems capture license plate images from vehicles in motion and transform them into digital text [5]. The effectiveness of LPR systems depends heavily on their ability to accurately process images under varying conditions, such as different lighting, weather, and speeds. Traditional methods often struggle with these challenges, necessitating the need for the creation of more advanced algorithms and models. Recent advancements in DL have significantly increased the capabilities of LPR systems. Convolutional neural networks (CNNs), in particular, have given great promise in enhancing the accuracy and speed of LPR [6]. However, challenges remain, such as dealing with fake objects in complex environments, weather conditions, illumination issues, and varying human factors like vehicle speed and driver behaviors. In this study, the SqueezeNet model is trained and assessed for LPR using a dataset of images of vehicle. SqueezeNet is renowned for its capability in reducing the number of parameters compared to traditional DL architectures without compromising accuracy [7]. It employs a unique design to extract meaningful features from the input data. The SqueezeNet model's prediction accuracy is evaluated using a test dataset and a range of hyperparameters. This study examines the use of SqueezeNet in LPR, showing how modern DL can revolutionize the field by providing reliable, real-time vehicle identification solutions. The proposed work offers some key contributions as follows:

- To perform character segmentation and recognition and studied the numerous kinds of pre-processing methods that can be applied to improve these processes.
- To enhance image quality from the input sources, utilizing the preprocessing techniques to accurately extract the region of interest (ROI).
- To create an effective vehicle LPR system using the SqueezeNet model, capable of accurately recognizing license plates.

The next sections of the study as follows: The current approaches are discussed on section 2. The methodology suggested is illustrated in section 3. Section 4 presents the results obtained. The paper concludes in Section 5.

2. LITERARTURE REVIEW

Sultan et al. (2023) [8] addressed the complex challenges of LPR, which arise from diverse license plate shapes, designs, non-standard templates, angle variations, irregular outlines, and occlusion. To overcome these issues, a new LPR method is proposed consisting of three distinct steps. Initially, vehicles in the input images were detected using the Faster R-CNN. Then morphological operations were employed to determine the area of the license plate within the vehicle detected. Finally, the recognition of the license plate characters was performed using a DL network. A detailed simulations on the CCPD AOLP and PKU databases demonstrated the efficacy of the proposed method, achieving mean recognition accuracies of 99%, 96.0231%, and 98.7%, respectively. However, the method faced limitations in scenarios where light reflection from vehicle surfaces onto the license plate area reduced the recognition accuracy.

Kaur et al. (2022) [9] created an effective ALPR system that utilized CNN for the recognition of character. To enhance the quality of input images, the system integrated the pre-processing techniques and morphological operations. It demonstrated versatility by effectively recognizing multi-line, multi-font and skewed license plates, it performed well in night mode across different vehicle types. The proposed CNN technique attained an impressive overall accuracy of 98.13%. However, the system faced a notable limitation in recognizing multiple number of license plates within a single frame, as the CNN struggled to differentiate between the area of license plate from the vehicle's grills.

Alghyaline (2022) [10] introduced a precise ALPR system designed for Jordanian license plates, employing a two-stage CNN architecture based on the YOLO3 framework. The YOLO3 network underwent modifications to adopt a shallow architecture, enhancing its capability to efficiently detect small objects, thereby optimizing its performance specifically for license plate recognition. Additionally, the study utilized a set of arrays data structures to monitor the license plates linked to vehicles and to effectively filter out incorrect detections. The study also introduced a new dataset, JALPR dataset, for evaluation purposes. Experimental findings demonstrated the efficiency of the proposed method in identifying Jordanian license plates, achieving an impressive recognition accuracy of 87%.

Lee et al. (2022) [11] conducted a study on optimizing training parameters for the YOLOv3 model to enhance its performance in tasks involving detection of license plates. By categorizing the parameters, they aimed to minimize the requirements for the design of experiments (DOE) runs, while gaining valuable observations into the interactions among YOLOv3 parameters, just looking for optimal training settings. Through a series of simple DOE experiments, it demonstrates significant improvements in the ALPR performance of YOLOv3 without altering the CNN model itself, precisely for the license plate detection tasks. The YOLOv3 training parameters are strategically adjusted, resulting in achieving an average precision (AP) of 99% in detecting Malaysian vehicle license plates.

Shahidi Zandi and Rajabi (2022) [12] proposed a deep convolutional neural network (DCNN) framework for recognizing Iranian license plates, utilizing two distinct CNN models. The first model, YOLOv3, was employed to determine license plates in the input images, while the second model, Faster R-CNN, was used to identify and classify the characters within the detected plates. The study utilized datasets specific to YOLOv3 and Faster R-CNN, achieving remarkable performance metrics. The YOLOv3 network attained a mean average precision (mAP) of 99.6%, recall of 98.26%, accuracy of 98.08%, and an average detection speed of 23 milliseconds. The Faster R-CNN network demonstrated a recall of 98.97%, precision of 99.9%, and accuracy of 98.8% on the developed dataset. The suggested system proved that it has the ability of accurately recognizing the license plates even in challenging situations, such as the presence of unwanted data on the plates.

Zou et al. (2022) [13] suggested a two-stage LPR algorithm utilizing YOLOv3 and an Improved LPR Net (ILPRNET). The dataset for this study was sourced from Chinese City Parking. In the primary stage, YOLOv3 was utilized to determine and extract the location of the license plate. In the next stage, ILPRNET was employed to localize the characters of license plate in achieving precise recognition by combining a CNN encoder with a 2D attentional-based recognizer. The test results demonstrated that the proposed algorithm performed effectively across various complicated situations. Specifically, in sub-datasets such as CCPD-Weather, CCPD-Base, CCPD-Challenge, CCPD-DB, and CCPD-FN, the recognition accuracy reached 99.2%, 98.5%, 98.1%, 86.2%, and 97.8%, demonstrating the robustness and accuracy of the algorithm.

Huang et al. (2021) [14] proposed an ALPRNet, a unified neural network designed for the recognition and identification of license plates of mixed styles. Two fully convolutional one-stage object indicators were used by ALPRNet to effectively recognize and categorize characters as well as license plates. Evaluation on mixed style license plate datasets as well as datasets with single style plates revealed promising results, with ALPRNet achieving an impressive accuracy of 98.21%. The study demonstrated that the proposed network outperformed existing methods, highlighting its effectiveness in accurately recognizing license plates with mixed styles.

Alam et al. (2021) [15] developed an advanced system for identifying and recognizing vehicle number plates using CNN. Using a digital camera, the system first takes an image of a car. It then extracts the frame containing the number plate. To improve the clarity of the segmented number plate, a super-resolution approach was utilized. This technique enhanced a low-resolution image to a higher resolution by reconstructing pixel details through the convolutional layer of CNN. The characters in the number plates were then segmented using the bounding box method. For recognition, the CNN extracted and classified the features of the characters segmented. The system was evaluated using the VLPR vehicle dataset, with 700 vehicle images applied for testing. The CNN obtained accuracy of 98.2% on the validation set and demonstrated consistent performance with 98.1% accuracy on the testing set. Additionally, the method demonstrated versatility by being applicable to number plates written in different languages.

The study by Izidio et al. (2020) [16] introduced an innovative approach to Brazilian LPR utilizing CNNs optimized for embedded systems. Employing the Tiny YOLOv3 architecture, the system efficiently detected license plates from captured images, demonstrating dependence to variations in angle, lighting, and noise. This robust detection required only a single forward pass per network, significantly enhancing processing speed. Evaluated with real license plate images, the methodology achieved a recognition rate of 98.43%, with an average processing time of 2.07 seconds per license plate on a Raspberry Pi3. Furthermore, the incorporation of an ensemble CNN model improved the recognition accuracy to an impressive 99.53%, though with a slightly increased average processing time of 4.88 seconds. This advancement emphasized the potential to deploy high-accuracy and rapid LPR systems on resource-constrained devices.

Zou et al. (2020) [17] addressed the challenge of LPR under non-restrictive conditions by developing a strong recognition model designed to handle various challenging scenarios such as dark, bright, and rotated conditions. In order to extract the full features from the license plates, the model first activates the regional features of the characters. Also, it employed Bi-LSTM to find each of the character using the background location information, and then utilized 1D-Attention to increase the relevant features of the character, while diminishing the irrelevant ones, thereby ensuring the effective acquisition of character features. Extensive experimental results validated the model's superior performance under these challenging conditions. Specifically, in the Chinese City Parking Dataset (CCPD) sub-datasets including CCPD-Tilt, CCPD-FN, CCPD-Base, CCPD-Challenge, CCPD-DB, and CCPD-Weather, the recognition rates achieved were 98.5%, 99.3%, 98.6%, 96.4%, and 86.6% respectively, demonstrating the model's effectiveness and robustness.

Pustokhina et al. (2020) [18] presented an efficient Vehicle LPR (VLPR) model known as the OKM-CNN model, which combines CNN for recognition with optimal K-means (OKM) clustering-based segmentation. The framework operated through three primary stages: license plate detection, using the OKM clustering technique for segmentation which is enhanced by the Krill Herd (KH) algorithm, and recognition of license plate number using a CNN. This method was rigorously tested across three datasets: The HumAIn 2019 Challenge dataset, Stanford Cars, and FZU Cars. The experimental results demonstrated the efficacy of the OKM-CNN model, which attained 0.981 of maximum accuracy on the applied datasets.

Sajed et al. (2020) [19] presented a new method for recognizing license plate characters that ignored traditional techniques and binarization, instead utilizing a DL algorithm. This method extracted features from the license plate characters using a deep encoder-decoder network, and then fed those data to eight parallel classifiers for recognition. The method was evaluated using a database including11,000 license plate images obtained from a functioning surveillance system on a dual carriageway. The proposed method exhibited an accurate character recognition rate of 96% on a test set including 4,000 photos, thus establishing it as a viable and competitive alternative to current methods.

Varma et al. (2020) [20] introduced an innovative image processing system for detecting and recognizing Indian number plates, designed to handle challenges such as noise, low illumination, cross angles, and non-standard fonts. The system utilized a series of image processing techniques, followed by number plate segmentation with contours applied at the borders. The ROI was filtered and de-skewed before applying a K-nearest neighbor (KNN) algorithm for character recognition. The suggested techniques achieved improved outcomes by accurately identifying 98% of the license plates from a dataset consisting of 101 plates, including both the Indian and international plates. Additionally, these methods accurately recognized over 96.2% of the characters on these plates.

In their study, Hassan Onim et al. (2020) [21] proposed the utilization of a YOLOv4 object identification model that was particularly trained to identify car license plates in Bangladesh. Additionally, Tesseract OCR was employed to accurately recognize the characters on the license plates. The study also demonstrated a Graphical User Interface (GUI) that was created using the Tkinter Python library. The YOLOv4 model was trained to attain 90.50% of mean average precision (mAP) and exhibited efficient performance on a TESLA T4 GPU, processing real-time video footage at an average rate of 14 frames per second (fps). This system effectively combined license plate detection and character recognition, providing a practical solution for real-time application in Bangladesh.

Omar et al. (2020) [22] presented a cascaded DL method to create an effective system for detecting and recognizing license plates that is specific to the cars in northern Iraq. The method utilized numerous preprocessing techniques, including adaptive image contrast enhancement and Gaussian filtering, to enhance the input images effectively. A deep encoder-decoder network architecture was used for segmentation, and two different CNN models processed the divided license plate regions to identify the city and recognize Arabic numbers. To facilitate experimental validation, a fresh dataset of license plates was created. The method suggested was examined using recall, and precision for identification, and classification accuracy for recognition. The results of the study showed the efficacy of the method, as seen by recall, F-measure scores and precision, of 92.10%, 91.01%, and 94.43%, respectively. Additionally, the classification accuracies for Arabic numbers and city labels were 92.26%, and 99.37% respectively. However, the method's dependency on semantic segmentation and character segmentation was noted as a limitation, as these processes directly impacted the final recognition outcomes.

The current approaches identify several significant research gaps. Differentiating in between the area of the license plate and vehicle grills remains a difficult task when trying to identify several license plates within a single frame. Light reflection from vehicle surfaces onto the license plate area reduces recognition accuracy, while diverse shapes, designs, and non-standard templates of license plates significantly impact detection and recognition performance. Computational efficiency in real-time applications is crucial, emphasizing the need for high accuracy and fast processing speeds, especially for systems with limited resources. Developing a system that generalizes across different languages and regions remains a significant challenge in ALPR research. There is also a necessity for optimizing training parameters specifically for license plate detection tasks to enhance performance. Issues such as occlusions and angle variations continue to affect ALPR system performance, highlighting the need for more robust recognition models. Furthermore, real-world challenges like angle variations, occlusions, and reflections persist, and a universal solution adaptable to all global variations is still lacking. Lastly, developing robust segmentation techniques that minimize errors and improve recognition rates is essential for advancing the field.

3. METHODOLOGY

The block diagram depicted in Figure 3 illustrates the recognition of a license plate using a SqueezeNet model. The model receives input images from a pre-determined directory that includes a variety of multi-font images of license plates, car images, and images taken in different lighting conditions. The input image is converted to grayscale to reduce computational complexity. A Gaussian blur is then applied to the grayscale image to reduce the noise. Adaptive thresholding is performed to create a binary image, segmenting it into foreground and background. The binary images are enhanced by morphological processes like dilation and erosion, which remove small noise and connecting disjointed elements in characters on license plates. The ROI area of the license plate is selected from the morphologically processed image. Contours of the ROIs are found to identify the correct position and boundary of the license plate. The SqueezeNet model, a lightweight CNN, is trained using an alphabet and number dataset to identify individual characters on the license plate. Once trained, the model is tested on new images to evaluate its performance. The final output is the recognized license plate number, a sequence of characters identified by the trained SqueezeNet model from the input image. This integrated approach combines image processing techniques for detecting and isolating license plates with deep learning methods for character recognition. The goal is to achieve accurate and rapid license plate recognition.

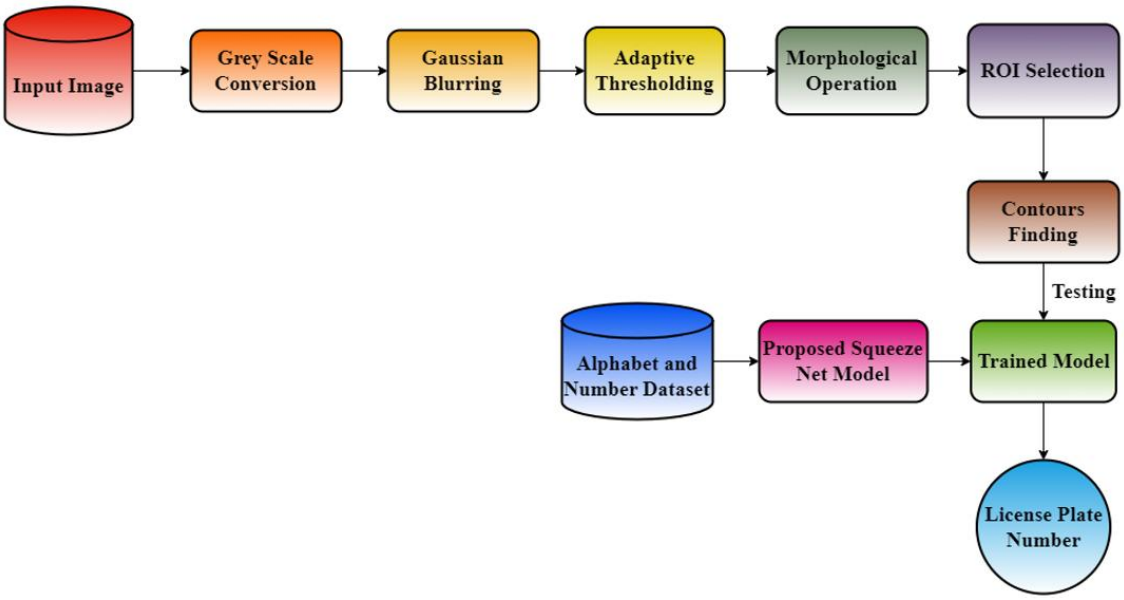

Fig. 3: Block Diagram of Proposed Method

3.1 Image Acquisition

Image acquisition involves capturing or uploading images of vehicles and their license plates into the system for processing. This is the initial step where the user triggers the process allowing the system to begin recognizing and analyzing the license plates. The Figure 4 represents the images considered for the extraction process and the cropped image for the extraction process.

Fig. 4: Input original image and Cropped image for extraction

3.1.1 Triggering the system

The system initiates its operation when the user presses the “Upload Image” button. This is a manual trigger to start the process of license plate recognition. The user interface with an interface such as web or desktop application, allowing them to select and upload an image containing the vehicle license plate.

3.1.2 Image Folder

A folder named “test_data” is created in the current directory to store the images of vehicles and license plates for processing. This folder serves as a centre location for all images that the system will process. By organizing images in this manner, it simplifies the management and retrieval of images during processing.

3.1.3 Image Variety

The "images" folder contains a diverse set of images, including variety of vehicles such as trucks cars, and motorcycles. It also features multi-line license plate images, license plates with different font styles and sizes, and images taken under various lighting conditions such as daylight, nighttime, and artificial lighting. This diversity is crucial for training and testing the ALPR system to ensure it can accurately recognize license plates under different scenarios.

3.1.4 Dataset Considerations

Existing datasets used by other ALPR systems may have limitations such as insufficient variety, poor image quality, or a lack of real-world diversity. To address these issues, a custom "images" folder has been created with a diverse set of images. This diversity includes various vehicle types, different license plate designs, and images captured under various lighting conditions. By overcoming the limitations of existing datasets, this approach aims to make the ALPR system more robust and versatile. The diverse images in the "test_data" folder are intended to address these drawbacks, ensuring the ALPR system performs well across different conditions.

3.1.5 Execution Steps

The user selects an image file and uploads it to the system via the "Upload Image" button. Upon receiving the image, the system stores it in the "test_data" folder and initiates the LPR process. The uploaded image undergoes preprocessing to increase the quality of the image. Once the license plates are detected, the characters on the plate are segmented. The segmented images are converted from the image format to text format. Finally, recognized characters are validated and formatted according to standard license plate formats.

3.2 Preprocessing

3.2.1 Gray Scale Conversion

Grayscale conversion transforms a color image into shades of gray, preserving intensity information crucial for license plate recognition. This process involves reducing the color depth of the image, typically using weighted averages of the RGB (red, green, blue) color channels while emphasizing green due to human vision sensitivity. The grayscale image generated has pixel values range between 0 and 255. Grayscale images simplify computational tasks in license plate recognition, improving speed and reducing complexity by providing a uniform representation of the image's features. The input image after the gray scale conversion is depicted in Figure 5.

Fig. 5: Input image after Gray Scale Conversion

3.2.2 Gaussian Blurring

In license plate recognition, Gaussian blurring is employed as a technique to reduce noise and detail in images. It involves convolving the image with a Gaussian kernel, resulting in a smooth, blurred effect that helps improve the accuracy of subsequent processing steps. The amount of blurring is controlled by the kernel's standard deviation σ. In the proposed approach, Gaussian blurring is applied to the image using a 5 X 5 kernel, enhancing the capability of the system to detect and identify the license plates accurately by reducing image noise and enhancing the clarity of plate features. Figure 6 represents the input image after Gaussian blurring.

Fig. 6: Input image after Gaussian blurring

Gaussian filtering involves convolving each point of the input array with the Gaussian equation, as specified in equation 1. All of these points are added up to create the output array. One way to express a one-dimensional Gaussian function mathematically as:

$$G(m) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{-m^2}{2\sigma^2}} \qquad (1)$$

This function is used in two dimensions for processing images; it is effectively the sum of two one-dimensional functions. Mathematically, it is shown as:

$$G(m,n) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{-(m^2+n^2)}{2\sigma^2}} \qquad (2)$$

The Gaussian distribution's standard deviation is represented by σ, with m and n representing the distances from the origin on the horizontal and vertical axes.

3.2.3 Adaptive Thresholding

Adaptive thresholding plays a significant role in LPR by enhancing the ability to accurately segment the license plate characters from the background under varying lighting conditions. Adaptive thresholding determines the threshold for each pixel based on the local neighborhood, making it more flexible and robust to lighting variations as shown in Figure 7. To increase the edges estimated and produce a binary edge image E_a, adaptive thresholding is applied to the previously generated greyscale edge image E_y. After generating the greyscale edge image E_y using an adaptive thresholding method. An integral image E_l is created by summing the pixel values from the upper-left corner to each

pixel in E_y. Subsequently, a binary edge image E_a, is produced by applying a threshold to each pixel $p\ (m,n)$ in the integral image E_l, where the threshold value is computed from a local window within E_l. $E_a\ (m,n)$ is calculated as follows.

$$E_a(m,n) = \begin{cases} 255 & if\ E_l\ (m,n)\ \geq \beta\varpi\ (m,n) \\ 0 & else\ E_l\ (m,n)\ <\ \beta\varpi\ (m,n) \end{cases} \quad (3)$$

Here β represents the coefficient which is used to regulate the threshold and $\varpi\ (m,n)$ denotes the average of all pixel's values within the local $t_\omega \times t_\omega$ window surrounding pixel $p\ (m,n)$, that is,

$$\varpi\ (m,n) = \frac{1}{t_\omega \times t_\omega} \sum_{\substack{-t_\omega/2<r<t_\omega/2 \\ -t_\omega/2<s<t_\omega/2}} E_l(m+r, n+s) \quad (4)$$

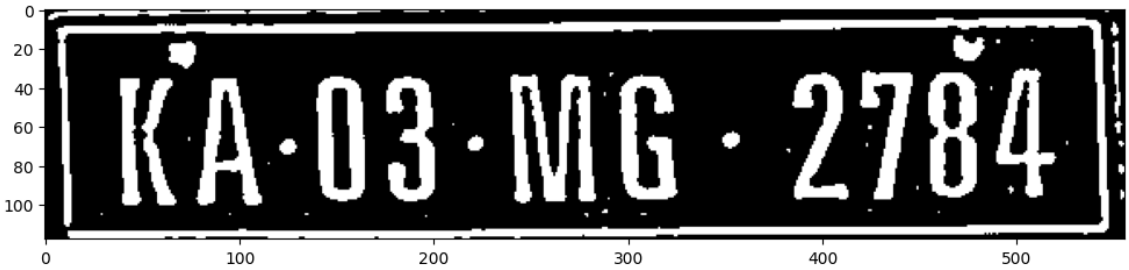

Fig. 7: Input image after Adaptive thresholding

3.2.4 Morphological Operation

The computational cost of edge detection algorithms is increased by matrix multiplications. Thus, in order to minimize computational complexity, morphological processes have been used in place of edge detection. The majority of the large connected background groups are removed after binarization. Still, there is unwanted data in an image that can affect the plate detection accuracy. In license plate recognition, morphological operations are used to enhance the features of a grayscale image. This process begins by creating a 3x3 rectangular structuring element. The top-hat transformation is then applied to highlight small bright features by subtracting the image opened from the original. Conversely, the black-hat transformation highlights small dark features by subtracting the original image from the image closed. By adding the top-hat result to the original image, bright details are enhanced. Subtracting the black-hat result from this enhanced image reduces dark details, thus improving the overall contrast and aiding in the accurate recognition of license plate characters. The morphological operations for top hat and black hat transformations are:

Top-Hat transformation is,

$$A - (A \circ B) = A - [(A \ominus B) \oplus B] \quad (5)$$

Black-Hat transformation is,

$$(A \cdot B) - A = [(A \oplus B) \ominus B] - A \quad (6)$$

3.3 Segmentation of Alphanumeric Characters

In license plate recognition, the process of finding all the contours in an input image is crucial for identifying potential characters. The function 'cv2.findContours' is employed to detect all the contours within the image. Contours are curves that connect all adjacent points on a border with the same color or intensity. Once these contours are identified, each one is examined individually to calculate the dimensions of its bounding rectangle, which is the smallest rectangle that can fully contain the contour. By tuning parameters and applying filters, only those rectangles that are likely to contain characters are retained. For this some dimension comparison will be performed by accepting those rectangles that have:

Width in the range 0 = (length of image) / (number of characters) (7)

Their length falls within the range from half the width of the image to four-fifths of the width of the image. Following this step, all the characters are extracted as binary images, making them ready for further processing in the recognition system.

3.4 SqueezeNet Model for Recognition

SqueezeNet is a DCNN architecture that focuses on efficiency in terms of model size and inference speed. It achieves these goals by implementing several innovative design choices that significantly reduce the number of parameters compared to traditional architectures like AlexNet. SqueezeNet is specifically designed to minimize the quantity of parameters within the network. It is important that fewer parameters translate to a smaller model size, means less memory usage and faster inference times. One of the primary techniques employed by SqueezeNet is the extensive use of 1x1 convolutional filters. These filters are employed to decrease the number of input channels before adding larger convolutional filters. This step is crucial as it decreases the computational load and the number of parameters in the subsequent layers.

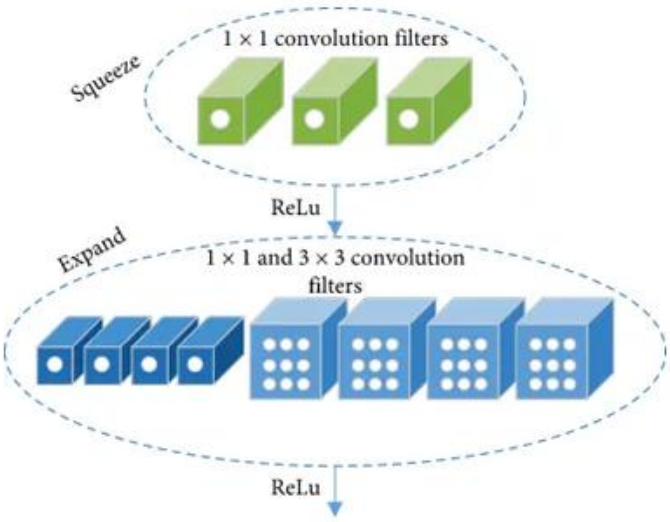

Fig. 8: Fire module of SqueezeNet

The fire module is the essential component of the SqueezeNet. It consists of two main layers such as squeeze layer and expand layer as shown in Figure 8. The squeeze layer uses 1x1 convolutions to reduce the number of input channels. By compressing the data, it prepares the feature map for the next layer without losing essential information. The expand layer processes the data after the squeeze layer by combining 1x1 and 3x3 convolutional filters. The 1x1 filters capture fine-grained features, while the 3x3 filters capture more complex patterns. The outputs from these convolutions are concatenated to form the final output of the fire module. Instead of using fully connected layers, SqueezeNet employs global average pooling. In traditional CNNs, fully connected layers contribute significantly to the total number of parameters. By replacing these with global average pooling, SqueezeNet reduces the parameter count dramatically. Global average pooling works by taking the average of each feature map, resulting in a single value per feature map, which is then used for categorization. It not only decreases the parameters but also helps in reducing overfitting and speeds up the inference process.

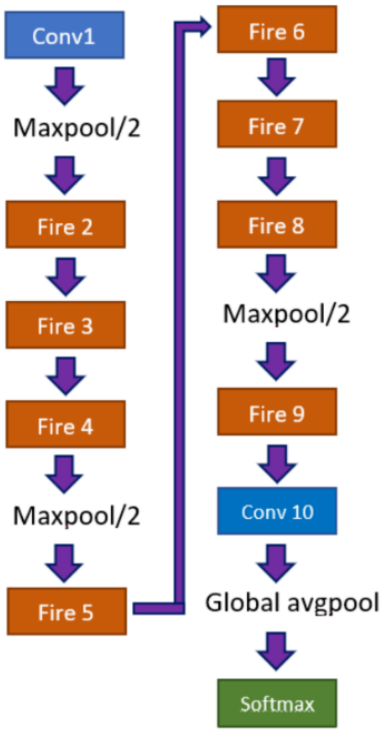

Fig. 9: Macro architecture of SqueezeNet

The architecture of SqueezeNet is illustrated in Figure 9. The proposed SqueezeNet model receives the segmented license plate image as an input. The initial layers of SqueezeNet starts with a convolutional layer applying 64 filters, each of size 3×3. Stride which reduces the spatial dimensions is set to 2 for the input image. After the convolution, a Rectified Linear Unit (ReLU) activation function is applied to bring non-linearity into the model. The activation layer is followed by a max pooling layer with a pool size of 3x3 and stride of 2, thereby reducing the spatial dimensions and adding translation invariance. As of said above the core of the architecture consists of a series of fire modules, which are designed to lessen the number of parameters while maintaining model performance. The squeeze phase uses a 1 x 1convolution with 16 filters. This is followed by a ReLU activation function. The expand phase gets splitted into two parallel paths. In the first path 1x1 convolution is applied with 64 filters. In the second path 3x3 convolution is applied with 64 filters. Both the paths are followed by ReLU activation function. The outputs from the 1×1 and 3×3 convolutions are concatenated along the depth dimension, combining the features extracted by both paths. Fire Modules 2 and 3 maintain 16 filters for the squeeze layer and 64 filters for each path in the expand layer. Also, Fire Modules 4 and 5 increased the number of filters to 32 for the squeeze layer and 128 for each path in the expand layer. The Max pooling layers are strategically placed after every two fire modules to progressively minimize the spatial dimensions of the feature maps, making the model more manageable and ensuring that higher-level features are retained while reducing the amount of computation required. The flatten layer transforms the 3D tensor output generated by the fire modules and pooling layers into a 1D vector. The fully connected layer consists of 128 units and it uses ReLU function for learning complex representations from the features extracted by the convolutional and fire modules. A dropout rate of 0.4 is applied in order to reduce overfitting. During training, this layer randomly sets 40% of the input units to zero, preventing the model from becoming too dependent on any particular set of features. The final layer is a dense layer with 36 units, corresponding to the characters (letters and numbers) on a license plate. The model utilizes a softmax activation function to generate a probability distribution across classes, enabling the model to determine the characters that appear on the license plate.

The model is compiled using the categorical cross-entropy loss function. In license plate recognition, categorical cross-entropy quantifies the difference between the predicted probability distribution across the characters. The Adam optimizer is chosen for training. A learning rate of 0.00001 is set for the optimizer. A small learning rate ensures that the model learns gradually, which helps in achieving better accuracy and preventing the model from overshooting. The model is trained over 50 epochs. Training for multiple epochs allows the model to learn and refine its weights gradually. Figure 10 illustrates the architecture of the work proposed, while Table 1 provides a summary of the model proposed.

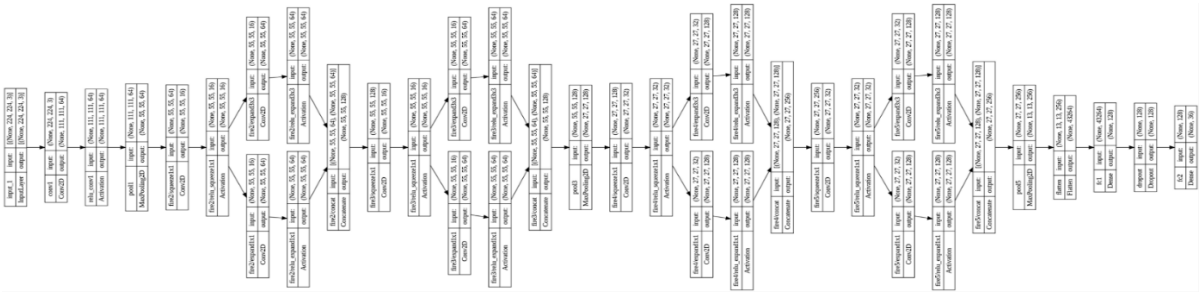

Fig. 10: Architecture of the Proposed model

Table. 1. Proposed Model Summary

Total Parameters	5662980
Trainable Parameters	5662980
Non-Trainable Parameters	0

The proposed model algorithm is outlined below

Algorithm

Input:

- *Pre-trained SqueezeNet Model*
- *Alphabet and Number Dataset for training*

Output: *License Plate Character recognition*

Begin:

Load and preprocess data:

1. *Collect dataset:* $D = \{(X_i, y_i)\}_{i=1}^{N}$, *where* X_i *is an input image.*
2. *Convert input image* X_i' *to grayscale:* $X_{gray} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$
3. *Applying Gaussian Blur to* X_{gray}*:* $X_{blur} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$
4. *Apply adaptive thresholding to* X_{blur} *:* $X_{thresh} = \mu(x, y) - C$
5. *Perform Morphological operations on* X_{thresh}*:* $X_{morph} = c_{dilated} = dilation\ (A, B)$

   $c_{eroded} = erosion(c_{dilated}, B)$

6. *Segmentation of the processed image and contours are obtained*

Define Base Models:

1. *Load SqueezeNet model*
2. *Input:* $224 \times 224 \times 3$

   Conv2D (64, (3,3), activation='relu')

   Maxpooling2D (pool size= (3,3))

   Fire Modules: *Fire2: Squeeze=16, Expand=64*

   Fire 3: Squeeze=16, Expand=64

   Maxpooling2D (pool size= (3,3))

   Flatten: *Flatten ()*

   Fully Connected Layers:

   Dense (128, activation='relu')

   Dropout (0.4)

   Dense (36, activation='softmax')

Model Compilation and Training:

1. *Compile each model M:*
   optimizer=Adam ()
   loss=categorical_crossentropy
   metrics=[accuracy]
2. *Train: M.fit (X_{train} , y_{train} ,validation_data= (X_{val}, y_{val}), epochs=num_epochs, batch_size, callbacks= [earlystopping, customcallback])*
3. *Display plate number and plot each character in the predicted label*

Save the Model

End

4. EXPERIMENTAL RESULTS

4.1 Software and Hardware Setup

The models development and training were carried out using Google Colaboratory, utilizing Python along with the Keras framework. The Colab environment was configured with TensorFlow, a GPU, 12.75 GB of RAM, 68.50 GB of disk space, and a 64-bit version of Windows 10. The efficiency of the model proposed was examined based on its predictions on the test dataset. Hyperparameters of the deep neural network (DNN) were determined through empirical methods, as outlined in Table 2, and plays an important role in the learning process. To identify the model that delivers the best classification performance, a wide range of variables were tested and evaluated.

Table. 2. Hyperparameters Specifications

Hyperparameters	Values
Loss Function	Categorical Crossentropy
Batch Size	16
Activation function	Softmax, Relu
Optimizer	Adam
Number of epochs	50
Learning rate	0.00001
Dropout	0.4

4.2 Evaluation Metrics

The performance of the model is shown to be improving over time in the accuracy plot, indicating an improved capacity to recognize license plates. Simultaneously, the error rate decreases on the loss plot as the model gains experience, signifying efficient parameter optimization. Figures 11 and 12 show the accuracy and loss plots of the model. In the initial epoch, the framework demonstrates a strong starting accuracy of 95.83%, which is quite high for an initial training phase. This shows that the model has a good initial grasp of recognizing license plates. By the final epoch, the accuracy has improved to 97.34%, showing that the framework has effectively learned from the data and has become more adept at LPR over the course of training. In the initial epoch, the model starts with a loss value of 0.1048, indicating the level to which the model's predictions differ from the actual labels at the beginning of training. As training increases, the loss value generally decreases, showing that the model is learning and improving its accuracy in recognizing license plates. By the final epoch (Epoch 50), the loss reduces significantly to 0.0678, indicating that the model has optimized its parameters and minimized prediction errors. The performance of the model is impressive with an accuracy of 99.65%, indicating overall correctness in its prediction.

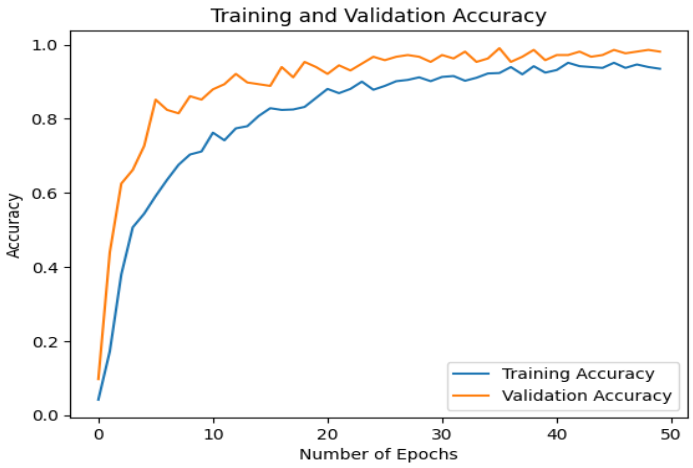

Fig. 11: Accuracy plot of the proposed model

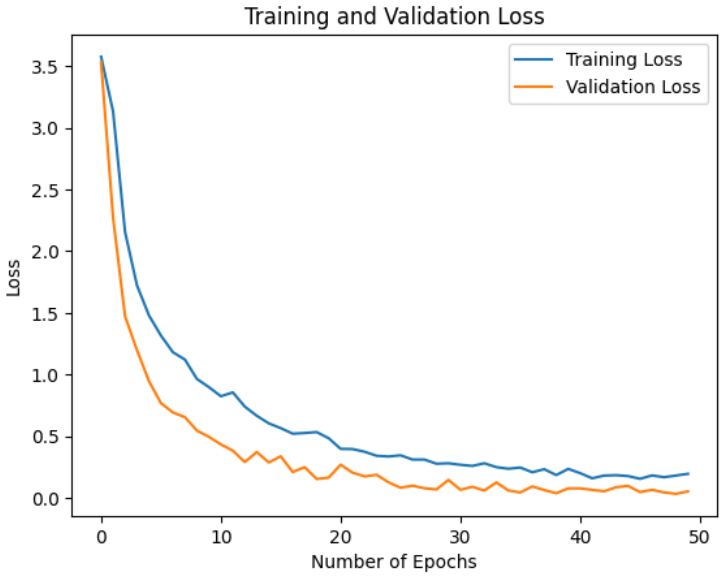

Fig. 12: Loss plot of the proposed model

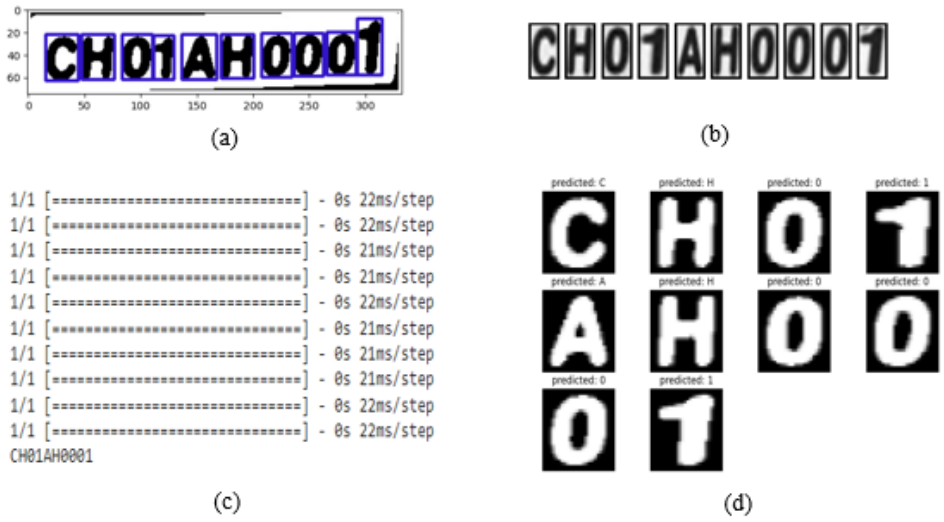

Fig. 13: (a) Sample image for testing, (b) Gray scale conversion, (c) Iteration over each character image, (d) Predicted output

A randomly selected image is taken from the images folder. The image undergoes a series of processing steps: it is resized to a fixed size and converted to grayscale, then smoothened to obtain a binary image. The boundaries are adjusted by adding a white background, removing noise and unnecessary elements located around the areas of license plate. The ROI is cropped and resized to a standard size. The stored and processed character images are returned for further analysis and recognition tasks. Each character image is resized to a standard size and iterated over an alphanumeric dataset. Each processed image is reshaped into a format suitable for model prediction. The predicted characters are collected, eventually forming the recognized license plate number. As depicted in Figure 13, this successful method underscores the model's effectiveness and dependability in correctly identifying the license plate. Table 3 presents an accuracy comparison between the proposed and current approaches and Figure 14 represents the graphical representation of the suggested method with the existing works.

Table 3. Comparison of proposed method and current approaches

REFERENCE	MODEL	DATASET	RESULTS
[10]	YOLO3	JALPR	Recognition accuracy of 87% in Jordanian plates
[13]	YOLOv3, ILPRNET	CCPD	Recognition accuracies of 99.2%, 98.1%, 98.5%, 97.8%, and 86.2%.
[14]	ALPRNet	Mixed style license plate datasets	Accuracy: 98.21%;
[15]	CNN	VLPR vehicle dataset	Validation set accuracy: 98.2% Testing set accuracy: 98.1%
[16]	Tiny YOLOv3	Custom dataset	Accuracy: 99.23% processing time: 4.88s
[19]	Deep encoder-decoder network	license plate images: 11,000	Character recognition rate: 96%
[20]	KNN	Dataset: 101 plates	Detected number plates: 98% Character recognition accuracy: 96.2%
Proposed SqueezeNet Model		**Vehicle images**	**Accuracy: 99.65%**

The table 3 presents a comparative analysis of various LPR models, highlighting their performance across different datasets. Alghyaline's (2022) YOLO3 model achieves an 87% recognition accuracy on Jordanian plates. Zou et al. (2022) report recognition accuracies ranging from 86.2% to 99.2% using YOLOv3 and ILPRNET on the CCPD dataset. Huang et al. (2021) demonstrate a 98.21% accuracy with ALPRNet on mixed style license plate datasets. Alam et al. (2021) achieve validation and testing accuracies of 98.2% and 98.1%, respectively, using a CNN on the VLPR vehicle dataset. Izidio et al. (2020) report a 99.53% accuracy and a processing time of 4.88 seconds using Tiny YOLOv3 on a custom dataset. Sajed et al. (2020) achieve a 96%-character recognition rate with a deep encoder-decoder network on 11,000 license plate images, while Varma et al. (2020) report a 98% detection rate and a 96.2%-character recognition accuracy using KNN on a dataset of 101 plates. The proposed SqueezeNet model demonstrates superior performance with an accuracy of 99.65% on a diverse set of vehicle images, indicating its high effectiveness and robustness in LPR tasks.

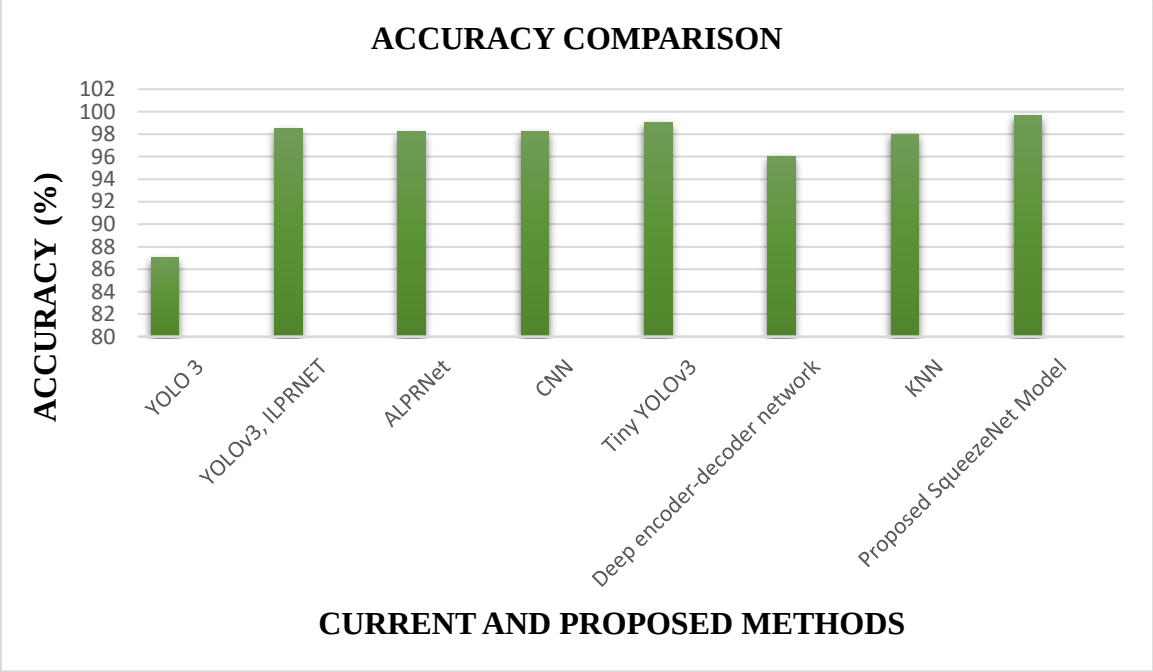

Fig. 14: Accuracy analysis of SqueezeNet model with existing approaches

5. CONCLUSION

The study aims to create an accurate and efficient vehicle LPR system using the SqueezeNet model. It addresses the complexity of license plate detection and character recognition by employing image preprocessing techniques like grayscale conversion, Gaussian blurring, adaptive thresholding, and morphological operations. These steps enhance the quality and clarity of the input images, enabling accurate detection and segmentation of license plate characters. The SqueezeNet model, known for its lightweight architecture and efficiency, is trained to identify these characters. Results show that the model performs exceptionally well, achieving an accuracy of 99.65%. The findings indicate that the SqueezeNet model, combined with effective preprocessing methods, can provide a robust and reliable solution for real-time license plate recognition, overcoming tasks such as varying lighting conditions and different plate designs.

REFERENCE

[1] Guerrero-Ibáñez, J., Zeadally, S., & Contreras-Castillo, J. (2018). Sensor technologies for intelligent transportation systems. *Sensors*, *18*(4), 1212.

[2] Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2012). Automatic license plate recognition (ALPR): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, *23*(2), 311-325.

[3] Henry, C., Ahn, S. Y., & Lee, S. W. (2020). Multinational license plate recognition using generalized character sequence detection. *IEEE Access*, *8*, 35185-35199.

[4] Shashirangana, J., Padmasiri, H., Meedeniya, D., & Perera, C. (2020). Automated license plate recognition: a survey on methods and techniques. *IEEE Access*, *9*, 11203-11225.

[5] Al-Shemarry, M. S. J. (2020). *Developing new techniques to improve licence plate detection systems for complicated and low quality vehicle images* (Doctoral dissertation, University of Southern Queensland).

[6] Cao, Y. (2024). Investigation of a convolutional neural network-based approach for license plate detection. *Journal of Optics*, *53*(1), 697-703.

[7] Ucar, F., & Korkmaz, D. (2020). COVIDiagnosis-Net: Deep Bayes-SqueezeNet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images. *Medical hypotheses*, *140*, 109761.

[8] Sultan, F., Khan, K., Shah, Y. A., Shahzad, M., Khan, U., & Mahmood, Z. (2023). Towards automatic license plate recognition in challenging conditions. *Applied Sciences*, *13*(6), 3956.

[9] Kaur, P., Kumar, Y., Ahmed, S., Alhumam, A., Singla, R., & Ijaz, M. F. (2022). Automatic License Plate Recognition System for Vehicles Using a CNN. *Computers, Materials & Continua*, *71*(1).

[10] Alghyaline, S. (2022). Real-time Jordanian license plate recognition using deep learning. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), 2601-2609.

[11] Lee, Y. Y., Halim, Z. A., & Ab Wahab, M. N. (2022). License plate detection using convolutional neural network–back to the basic with design of experiments. *IEEE Access*, *10*, 22577-22585.

[12] Shahidi Zandi, M., & Rajabi, R. (2022). Deep learning-based framework for Iranian license plate detection and recognition. *Multimedia Tools and Applications*, *81*(11), 15841-15858.

[13] Zou, Y., Zhang, Y., Yan, J., Jiang, X., Huang, T., Fan, H., & Cui, Z. (2022). License plate detection and recognition based on YOLOv3 and ILPRNET. Signal, Image and Video Processing, 16(2), 473-480.

[14] Huang, Q., Cai, Z., & Lan, T. (2021). A single neural network for mixed style license plate detection and recognition. *IEEE Access*, *9*, 21777-21785.

[15] Alam, N. A., Ahsan, M., Based, M. A., & Haider, J. (2021). Intelligent system for vehicles number plate detection and recognition using convolutional neural networks. *Technologies*, *9*(1), 9.

[16] Izidio, D. M., Ferreira, A. P., Medeiros, H. R., & Barros, E. N. D. S. (2020). An embedded automatic license plate recognition system using deep learning. *Design Automation for Embedded Systems*, *24*(1), 23-43.

[17] Zou, Y., Zhang, Y., Yan, J., Jiang, X., Huang, T., Fan, H., & Cui, Z. (2020). A robust license plate recognition model based on bi-lstm. *IEEE Access*, *8*, 211630-211641.

[18] Pustokhina, I. V., Pustokhin, D. A., Rodrigues, J. J., Gupta, D., Khanna, A., Shankar, K., ... & Joshi, G. P. (2020). Automatic vehicle license plate recognition using optimal K-means with convolutional neural network for intelligent transportation systems. *Ieee Access*, *8*, 92907-92917.

[19] Rakhshani, S., Rashedi, E., & Nezamabadi-pour, H. (2020). Representation learning in a deep network for license plate recognition. *Multimedia Tools and Applications*, *79*(19), 13267-13289.

[20] Ganta, S., & Svsrk, P. (2020). A novel method for Indian vehicle registration number plate detection and recognition using image processing techniques. *Procedia Computer Science*, *167*, 2623-2633.

[21] Onim, M. S. H., Akash, M. I., Haque, M., & Hafiz, R. I. (2020, December). Traffic surveillance using vehicle license plate detection and recognition in bangladesh. In *2020 11th International conference on electrical and computer engineering (ICECE)* (pp. 121-124). IEEE.

[22] Omar, N., Sengur, A., & Al-Ali, S. G. S. (2020). Cascaded deep learning-based efficient approach for license plate detection and recognition. *Expert Systems with Applications*, *149*, 113280.

[23] Y. Hole, S. Hole, L. P. Leonardo Cavaliere, B. Nair, M. Hasyim and H. B. Bapat, (2023) "Blockchain Usages in Hospitality Management," 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2023, pp. 2798-2801, doi: 10.1109/ICACITE57410.2023.10183291.

[24] Y. Hole, S. Hole, A. A. Ayub Ahmed, E. Efendi, I. Ibrahim and M. Hasyim, (2023) "Internet of Things Issues and Challenges," 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2023, pp. 1370-1373, doi: 10.1109/ICACITE57410.2023.10183221.