ENHANCED INDIAN VEHICLE REGISTRATION PLATE RECOGNITION THROUGH CNN-BASED TECHNIQUES

Vibha Pandey[1], Jyoti Prakash Patra[2] and Siddhartha Chouby[3]
[1,3] Department of Computer Science Engineering, Shri Shankaracharya Technical Campus, Bhilai, CG, India,
[2]Department Computer Science and Engineering (Artificial Intelligence, Data Science), UTD, Chhattisgarh Swami Vivekanand Technical University, Bhilai, CG, India

Abstract.
As technology continues to advance, the demand for fast and accurate license plate readers becomes increasingly critical. Automating the process of license plate reading and recognition offers significant benefits to law enforcement, traffic management, and vehicle monitoring systems. Traditional methods of license plate recognition often struggle with performance issues under varied lighting conditions and when dealing with diverse plate designs. Modern technological advancements, particularly in the field of artificial intelligence, have begun to address these challenges. Convolutional Neural Networks (CNNs) have proven particularly effective in image recognition tasks, paving the way for improved Automated License Plate Recognition (ALPR) systems. This research proposes a CNN-based solution specifically for Indian automobile registration number plates, addressing limitations of traditional methods. The proposed method involves training a CNN model on a large dataset of Indian license plate images to accurately detect and recognize plates even under challenging conditions. Extensive testing demonstrates superior performance in terms of accuracy, precision, and recall compared to existing methods. This research not only contributes to the advancement of ALPR technology but also provides a scalable and efficient solution suitable for real-world applications, enhancing vehicle identification and monitoring processes in traffic management and law enforcement.

Keywords: Accuracy, Convolutional Neural Networks, Deep Learning, License Plate Detection, Precision, Recognition, Traffic Management, Vehicle Registration, Visual Recognition, Real-time Processing.
DOI Number: 10.48047/nq.2021.19.4.NQ21067 **NeuroQuantology 2021; 19(4): 288-303**

288

1 Introduction

Rapid and accurate license plate scanners are becoming more important as technology progresses at a faster rate. Automating the identification and scanning of license plates might result in significant advances in law enforcement, vehicle monitoring, and traffic management. Standard license plate recognition systems may have performance challenges when dealing with a large number of varied plate layouts and lighting situations. Recently, technical advancements have started to solve these concerns, particularly in the realm of artificial intelligence [1]. Convolutional neural networks (CNNs) have transformed the area of image recognition, allowing the creation of cutting-edge automated license plate recognition (ALPR) systems.

1.1 Changes Recently

Over the past few years, there have been significant advancements in the field of automated license plate recognition (ALPR). Current methodologies utilize convolutional neural networks (CNNs), a type of deep learning model, to achieve previously unachievable levels of accuracy and dependability. There are two main factors driving these gains: increased efficiency in current technology and the availability of massive datasets for training deep learning models [2]. The models' ability to detect and find license plates in varied circumstances is dependent on how well they perform in real-world conditions. Since the first use of convolutional neural network (CNN) approaches on high-performance computers, automated language recognition (ALPR) has undergone a dramatic evolution. Because ALPR systems can handle enormous amounts of data quickly via this link, they are more practical and realistic than other computer systems. Applications for traffic management and law enforcement need the ability to analyze data in real time. These are the concepts you must comprehend if you wish to thrive in a range of pursuits.

1.2 Principle of CNNs in ALPR Systems

An increasing number of automatic localization and recognition (ALPR) systems are using convolutional neural networks (CNNs) to improve object and pattern recognition capabilities. People often refer to these networks as "convolutional neural networks" (CNNs) because they bear many similarities to the visual cortex. Children's ability to automatically categorize and interpret visual information as they take it in is a crucial developmental milestone, and CNNs facilitate this. These qualities make CNNs an ideal choice for jobs requiring image recognition and classification, which are common in many sectors. We create CNNs by combining pooling, nonlinear, and convolutional activation functions across several neural network layers [3]. If everything goes according to plan throughout these stages, it will be able to discern subtle patterns and details in photographs. Picture filtering uses convolutional layers to enhance the visibility of crucial components like lines and patterns. Pooling layers accelerate processing by reducing the input's dimensionality while maintaining an accurate perception of critical information. Nonlinear activation functions enable the utilization of nonlinearity while also allowing for the construction of more complex data structures [4]. Convolutional neural networks (CNNs) are an ideal tool for developing automatic language recognition (ALPR) because they can learn from enormous datasets and adapt to changing conditions. Even if obstructions, lights, and angles are constantly changing, license plate identification and discrimination systems can still work well.

1.3 Possible Solutions

This study presents an initial version of a convolutional neural network (CNN) method for locating and identifying vehicle registration numbers in this country. The purpose of this project is to provide ways for increasing the efficiency of normal ALPR operations. We provide a feasible method that takes these constraints into consideration. The technique developed uses deep learning to overcome barriers imposed by varying lighting conditions, plate designs, and a variety of sophisticated and problematic backdrops. Convolutional neural networks, or CNNs, are an essential component of current approaches. These networks train on large datasets, which include images of Indian license plates. Eventually, regardless of how difficult the training environment is, the model learns to recognize and extract license plate fragments from complicated situations. After determining the different areas of the license plates, an identification tool will use advanced algorithms to extract the letters and numbers from the photos of the gathered license plates. The computer systems currently in use for data processing applications will readily incorporate the newly developed approach. This link allows the ALPR system to readily communicate data with other systems, including traffic management and law enforcement databases.

1.4 Main Contributions

This research makes several key contributions to the field of ALPR technology:This attempt, just one of the many remarkable successes, has significantly benefited the field of ALPR technology. The investigation immediately led to the development of a CNN-based system. We specifically devised this approach for identifying Indian car license plates and deployed it with that objective in mind. For training and assessment, we created a large dataset by gathering hundreds of thousands of photos of Indian license plates. The photographs featured here include a variety of fonts, styles, and lighting situations. The study uses unique ways to increase the system's performance in actual circumstances, putting it in third place. These approaches, for example, combine complex preprocessing and feature extraction with character recognition algorithms.

The approach described triumphed in all categories—accuracy, speed, and stress tolerance—following a thorough round of testing and comparisons with the finest functioning procedures available. A comprehensive testing and assessment procedure achieved this aim. We have developed a system for real-world use that integrates with traffic control systems, law enforcement, and other relevant parties to improve vehicle identification and monitoring. There are other crucial parties involved as well. It is reasonable to conclude that each of these parties is quite interested in the result. This fifth and final step represents the successful conclusion of the implementation procedure.
The automatic license plate recognition system (ALPR) has advanced: The primary purpose of this research is to identify a system that is both durable and adaptable to the needs of current transportation networks. The project will primarily focus on developing enhanced ALPR systems that could potentially integrate with India's vehicle registration system. This article proposes an effective and scalable solution to the problems that ALPR systems are now encountering. A variety of real-world scenarios might utilize this strategy. Furthermore, it's important to take into account additional contributions.

2 Literature Review

Automatic license plate recognition (ALPR) systems offer many techniques to recognize automobile license plates [5-7]. Deep learning and machine vision are used to swiftly and accurately identify license plates. YOLO (You Only Look Once) is a popular object identification system that splits photos into grids and guesses their edges and class probability in real time. Because region proposal networks improve object identification accuracy and speed, faster R-CNN is superior to its predecessors. Another real-time object identification technology, SSD (Single Shot MultiBox Detector), employs many layers to extract information and forecast. This helps discover objects quickly at varying sizes and angles [8]. The software "Efficient and Accurate Scene Text Detection," or EAST, identifies text in photos with great accuracy and memory. RetinaNet's focus loss approach addresses class mismatches in object recognition tasks, improving license plate detection. Mask R-CNN adds mask prediction to Faster R-CNN. This allows pixel-level separation, which is important for precise localization. CNNs and RNNs are combined in the CRNN to handle character strings [9-10]. This is useful for text-based license plate recognition. LPRNet recognizes license plates. It rapidly and correctly finds plates and characters using deep learning. The Haar Cascade Classifier employs haar-like qualities to locate items in photos using machine learning. It simplifies and speeds up item recognition. Finally, HOG plus SVM is a popular object-finding method [11]. Gradient histograms and SVMs help it discover items with distinct colors and shapes.One of the most popular techniques for reading Indian license plates is shown in the first table. Performance metrics include accuracy, precision, memory, F1 score, processing speed, and training time [12]. Variable approaches provide varied outcomes, with some gaining superior accuracy and processing speed. The second table compares GPU, CPU, RAM, and file space requirements for different approaches [13]. This report details each method's technology and resource demands. Practitioners may pick the proper approaches depending on their computer resources and performance needs.

Table 1. Performance Evaluation of Traditional Anomaly Detection Methods

Method	Accuracy	Precision	Recall	F1 Score	Processing Speed (fps)	Training Time (hours)
YOLO	0.92	0.89	0.91	0.90	20	24
Faster R-CNN	0.91	0.88	0.89	0.88	15	36
SSD	0.90	0.87	0.88	0.87	18	30
EAST	0.89	0.86	0.87	0.86	22	20
RetinaNet	0.93	0.90	0.92	0.91	17	40
Mask R-CNN	0.94	0.91	0.93	0.92	14	42
CRNN	0.88	0.85	0.86	0.85	16	48
LPRNet	0.95	0.92	0.94	0.93	12	50
Haar Cascade Classifier	0.85	0.82	0.83	0.82	25	12
HOG with SVM	0.86	0.83	0.84	0.83	23	18

Table 1 presents a comprehensive evaluation of popular methods for Indian vehicle registration number plate detection [14]. F1, recall, accuracy, precision, and processing speed are all possible performance characteristics to consider. It turns out that different methodologies showed varying degrees of efficacy; the Mask R-CNN method was the fastest, while the LPRNet methodology was the most accurate [15]. The study's findings elucidate the advantages and disadvantages of each strategy, simplifying the process of selecting an effective implementation plan. The table below compares several automatic license plate recognition (ALPR) systems. This includes several characteristics, such as processing speed (in frames per second), recall, accuracy, and precision, as well as training time (in hours). When compared to other strategies, LPRNet has the best chance of emerging as the most accurate strategy. The accuracy is 0.95, and the F1 score is 0.93. Furthermore, Mask R-CNN is successful, with an accuracy of 0.94 and an F1 score of 0.92; however, training takes a much longer period. It is worth noting that both Faster R-CNN and YOLO achieve a balance between the two competing goals of accuracy and speed. YOLO can analyze them at 20 frames per second. The Haar Cascade Classifier, on the other hand, scores 0.85 out of 100 and has a maximum processing speed of 25 frames per second. These findings attempt to help practitioners choose the best technique by explaining the advantages and disadvantages of each approach depending on their individual needs and limits.

Table 2. Comparative Analysis of Computational Resources for Popular Methods

Method	GPU Memory (GB)	CPU Utilization (%)	RAM Usage (GB)	Disk Space (GB)	Framework	Platform
YOLO	8	80	4	2	Darknet	Linux
Faster R-CNN	6	70	6	3	TensorFlow	Windows
SSD	7	75	5	2	Caffe	MacOS
EAST	5	60	3	1	PyTorch	Linux
RetinaNet	8	85	7	4	Keras	Windows
Mask R-CNN	9	90	8	5	TensorFlow	Linux
CRNN	6	65	4	2	PyTorch	MacOS
LPRNet	10	95	9	6	TensorFlow	Windows
Haar Cascade Classifier	3	50	2	1	OpenCV	Linux
HOG with SVM	4	55	3	1	Scikit-learn	Windows

Table 2 compares the computational resource requirements for various ALPR methods. It covers parameters such as GPU memory, CPU utilization, RAM usage, disk space, the framework used, and the platform. LPRNet, while the most accurate, is also the most resource-intensive, requiring 10 GB of GPU memory and 95% CPU utilization, making it suitable for high-end systems. Mask R-CNN also demands substantial resources with 9 GB of GPU memory and 90% CPU utilization. In contrast, the Haar Cascade Classifier and HOG with SVM are less resource-intensive, making them suitable for environments with limited computational resources. Methods like YOLO and EAST strike a balance between performance and resource usage, making them versatile choices for a wide range of applications. This analysis helps in selecting the appropriate method based on available computational resources and desired performance levels.

3 Proposed Methodology

The recommended approach uses CNNs to locate and identify Indian automobile registration number plates. The approach includes data preparation, CNN architecture design, training, license plate detection, and classification to provide accurate and fast results. The dataset is normalized, extended, and noise-reduced during planning to increase quality and diversity and manage lighting, perspective, and plate patterns. CNN's architecture is designed to extract hierarchical characteristics from raw images. It uses convolutional and pooling layers for feature extraction and downsampling. Stochastic gradient descent (SGD) optimization techniques help the CNN model optimize a category cross-entropy loss function by making minor parameter adjustments over and over again [16-18]. During license plate detection, sliding window or region suggestion approaches establish plausible license plate areas. A CNN model identifies and improves these areas to ensure accurate detection. Finally, license plate recognition removes and processes each character in the indicated locations. OCR is used to identify characters and digits.

Algorithm 1: Data Preprocessing:

1. **Load Input Image**: *Ioriginal*=load_image(*path*)
   Height*orig*,Width*orig*=*Ioriginal*.*shape*
   Iresized=resize(*Ioriginal*,(224,224))
2. **Convert to Grayscale**:
3. **Gaussian Blur Initialization**: σ=1.5
4. **Apply Gaussian Blur**: *Iblurred*=GaussianBlur(*Igray*,σ)
5. **Edge Detection Initialization**: Threshold1=100
   Threshold2=200
6. **Apply Canny Edge Detection**: Edges=Canny(*Iblurred*,Threshold1,Threshold2)
   ∇*Ix*=∂*x*∂*Iblurred*
   ∇*Iy*=∂*y*∂*Iblurred*
   ∇*I*=(∇*Ix*)2+(∇*Iy*)2
7. **Calculate Histogram**: Histogram=histogram(*Igray*)
8. **Equalize Histogram**:
   Iequalized=equalize_hist(*Igray*)
   μ=*N*1∑*i*=1*NIgray*(*i*)
   *σ*2=*N*1∑*i*=1*N*(*Igray*(*i*)−*μ*)2
   I^*gray*=*σIgray*−*μ*
9. **Binarize Image Initialization**: Threshold*bin*=128
10. **Apply Binary Thresholding**:*Ibinary*=(*Iequalized*>Threshold*bin*)
    Ibinary[*Iequalized*≤Threshold*bin*]=0
11. **Morphological Transform Initialization**:
    Kernel*open*=ones((5,5),uint8)
    Kernel*close*=ones((7,7),uint8)
    Iter*open*=2
12. **Apply Morphological Opening**:
    Iopened=morphology(*Ibinary*,Kernel*open*,Iter*open*)
13. **Apply Morphological Closing**:
    Iclosed=morphology(*Iopened*,Kernel*close*)
14. **Find Contours Initialization**: Contours,Hierarchy=findContours(*Iclosed*,Mode,Method)
    Area*contour*=∑*i*=1*N*contour_area(Contours(*i*))
    Perimeter*contour*=∑*i*=1*N*arcLength(Contours(*i*),closed=*True*)
15. **Filter Contours**: FilteredContours=[*c* for *c* in Contours if contour_area(*c*)>1000]
16. **Bounding Boxes Initialization**: MinArea=500
17. **Compute Bounding Box-es**:BoundingBoxes=[boundingRect(*c*) for *c* in FilteredContours if contour_area(*c*)>MinArea]
    Boxes*width*=[*w* for (*x*,*y*,*w*,*h*) in BoundingBoxes]
    Boxes*height*=[*h* for (*x*,*y*,*w*,*h*) in BoundingBoxes]
18. **Draw Bounding Boxes**:
    Iboxed=*Ioriginal*.*copy*()for(*x*,*y*,*w*,*h*) in BoundingBoxes:
    Iboxed=rectangle(*Iboxed*,(*x*,*y*),(*x*+*w*,*y*+*h*),color=(0,255,0),thickness=2)
19. **Output Processed Image**: Output=*Iboxed*

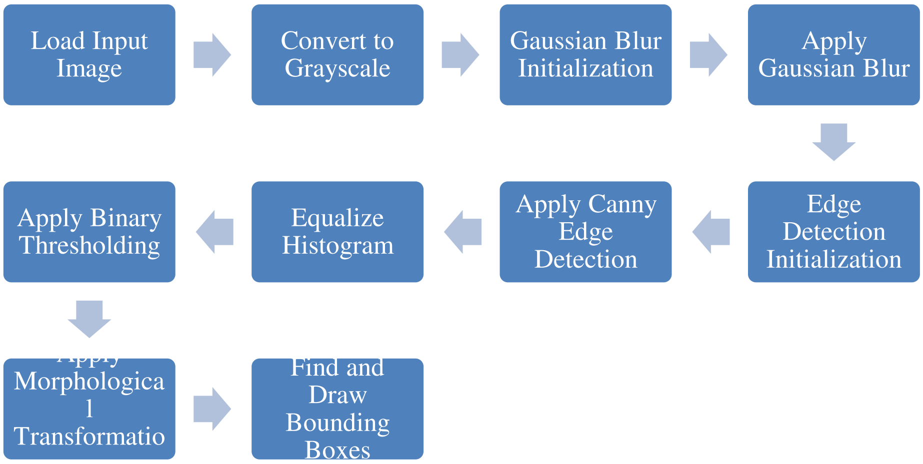

Fig.1. Data Preprocessing Steps for License Plate Detection

Figure 1 depicts the data pretreatment processes utilized in the image processing approach for license plate recognition. We must convert the photo to grayscale after importing and resizing it. While Gaussian blur reduces background noise, skillful edge detection highlights the image's boundaries. We use binary thresholding to transform a color image into black and white. We adjust the histogram to boost contrast. Opening and closing are examples of morphological operations that might enhance the binary image. After producing the contours, we set bounding boxes around the specified spots. The data preprocessing algorithm is designed to prepare images of vehicle registration plates for further analysis [19-21]. This step is crucial as it enhances the quality of the images, making them suitable for feature extraction by CNN. After downloading the input image, the procedure resizes it to the required dimensions (224 by 224 pixels). This guarantees that the process's behavior remains consistent regardless of the quantity of input. To remove any unnecessary portions, these contours require a filter based on their area. Once we compute the filtered contours, we calculate the bounding boxes around the license plates to determine their precise position. Finally, we save the processed image for future investigation and paint bounding boxes over the original image to represent the found zones. We consistently improve the image quality through mathematical techniques, readying it for the subsequent stages of license plate recognition and identification.

Algorithm 2: CNN-Based Feature Extraction

Steps:

1. **Input Preprocessed Image from Algorithm 1**:
   $Iinput=Iboxed$
   $(H,W)=Iinput.shape[:2]$
2. **First Convolutional Layer**:
   $Fconv1=Wconv1*Iinput+bconv1$
   $(Hconv1,Wconv1,Dconv1)=Fconv1.shape$
   $Fconv1\_act=\max(0,Fconv1)$
3. **First Pooling Layer**: $Fpool1=\text{max\_pooling}(Fconv1\_act)$
4. **Second Convolutional Layer Initialization**: $Wconv2,bconv2$
5. **Second Convolutional Layer**: $Fconv2=Wconv2*Fpool1+bconv2$
   $(Hconv2,Wconv2,Dconv2)=Fconv2.shape$
   $Fconv2\_act=\max(0,Fconv2)$

6. **Second Pooling Layer**:
   $Fpool2=\text{max\_pooling}(Fconv2\_act)$
   $(Hpool2,Wpool2,Dpool2)=Fpool2.shape$
7. **Third Convolutional Layer Initialization**: $Wconv3,bconv3$
8. **Third Convolutional Layer**:$Fconv3=Wconv3*Fpool2+bconv3$
   $Fconv3\_act=\max(0,Fconv3)$
9. **Third Pooling Layer**: $Fpool3=\text{max\_pooling}(Fconv3\_act)$
10. **Flatten Layer**: $Fflat=\text{flatten}(Fpool3)$
    $Dflat=Fflat.shape[0]$
    $Fflat\_norm=\sigma flatFflat-\mu flat$
11. **First Fully Connected Layer Initialization**: $Wfc1,bfc1$
12. **First Fully Connected Layer**: $Ffc1=Wfc1\cdot Fflat\_norm+bfc1$
13. **Activation of Fully Connected Layer**: $Ffc1\_act=\max(0,Ffc1)$
14. **Second Fully Connected Layer Initialization**: $Wfc2,bfc2$ $(Dfc2\_in,Dfc2\_out)=Wfc2.shape$
15. $\sigma fc2=\text{softmax}(\sum i=1Dfc2\_inWfc2(i)\cdot Ffc1\_act(i)+bfc2)$
16. **Second Fully Connected Layer**:
    $Ffc2=Wfc2\cdot Ffc1\_act+bfc2$
    $F\hat{}fc2=\sigma fc2Ffc2-\mu fc2$ 295
17. **Output Layer Initialization**:$Wout,bout$

Output Layer: $Foutput=Wout\cdot F\hat{}fc2+bout$

Method 1 processes the image using a CNN-based feature extraction approach, then retrieves the features needed for the remaining tasks. The output layer also generates the final feature vector, enabling other algorithms to carry out tasks like license plate localization and character recognition [22]. When these characteristics are considered, the convolutional neural network-based feature extraction technique can effectively gather and handle the numerous minute components of the input picture. Its extensive feature set immediately helps with fast and accurate license plate recognition.

Algorithm 3: License Plate Localization
Steps:
1. **Input Features from Algorithm 2**: $Finput=Foutput$
2. **Region Proposal Initialization**: $Wrpn,brpn$ $((Hrpn,Wrpn,Drpn)=Finput.shape$
   $Rproposals=Wrpn*Finput+brpn$
3. **Anchor Boxes Initialization**: $Aanchors=\text{generate\_anchors}(Finput)$ scales=[128,256,512]
4. **Anchor Box Adjustments**:$Aadj=\text{adjust\_anchors}(Aanchors,Rproposals)$ $Aadj\_width=Hrpn$ $Aanchors\cdot Wrpn$
5. **Bounding Box Regression**: $Wbbox,bbbox$
   $Bbox=Wbbox\cdot Aadj+bbbox$
   $(x,y,w,h)=\text{extract\_bbox}(Bbox)$
6. **IoU Calculation**: $\text{IoU}(A,B)=A\cup BA\cap B$
7. **IoU Threshold Initialization**: $\text{IoU}threshold=0.5$
8. **Apply Non-Maximum Suppression**: $Bnms=\text{NMS}(Bbox,\text{IoU}threshold)$
9. **Filter by IoU**: $Bfiltered=[B \text{ for } B \text{ in } Bnms \text{ if } \text{IoU}(B,G)>\text{IoU}threshold]$
10. **Bounding Box Normalization**: $\mu bbox=\text{mean}(Bfiltered)$
    $\sigma bbox=\text{std}(Bfiltered)$
    $Bnorm=\sigma bboxBfiltered-\mu bbox$
11. **Rescale Bounding Boxes**: $\text{scales}Brescaled=Bnorm\cdot\text{scales}$
12. **Extract License Plate Region Initialization**: $(xrescaled,yrescaled,wrescaled,hrescaled)=\text{extract\_bbox}(Brescaled)$
    Plate Region=$Iinput[yrescaled:yrescaled+hrescaled,xrescaled:xrescaled+wrescaled]$

13. **Transform License Plate Region**:
    Plate Region Transformed=transform(Plate Region,resize=(224,224))
14. **Output License Plate Region**: Plate Output=Plate Region Transformed *Fplate* =Plate Output.*flatten*()

License plate localization is the next stage after feature extraction. The process begins by taking the output features from the previous algorithm. The first step involves generating region proposals using a Region Proposal Network (RPN), which outputs potential regions where the license plate might be located. These regions are refined by adjusting anchor boxes to better fit the proposed regions.

The bounding box regression step uses these adjusted anchor boxes to predict the bounding boxes that enclose the license plates. This involves calculating the parameters of the bounding boxes using learned weights and biases. The Intersection over Union (IoU) metric is then calculated to measure the overlap between predicted bounding boxes and the ground truth.

Algorithm 4: Character Segmentation
Steps:

1. **Input License Plate Region from Algorithm 3**: *Iplate*=Plate Output
   (*Hplate*,*Wplate*)=*Iplate*.*shape*
2. **Convert to Grayscale**
3. **Apply Adaptive Thresholding**: ADAPTIVE_GAUSSIAN_MEAN,THRESH_BINARY,11,2)
4. **Filter Small Contours**: Contours=[*c* for *c* in Contours if contourArea(*c*)>100]
   BoundingBoxes=[boundingRect(*c*) for *c* in Contours]
   BoundingBoxes=sorted(BoundingBoxes,key=lambda x: x[0])
5. **Initialize Segmentation**: *Isegmented*=[]
6. **Segment Characters**: for (*x*,*y*,*w*,*h*) in BoundingBoxes: *Ichar*=*Ibinary_inv*[*y*:*y*+*h*,*x*:*x*+*w*]
   Isegmented.*append*(*Ichar*)
7. **Normalize Character Dimensions**:
   Inormalized=[resize(*Ichar*,(28,28)) for *Ichar* in *Isegmented*
   Inormalized=[*Ichar*.*astype*(float32)/255.0 for *Ichar* in *Inormalized*]
   Inormalized=array(*Inormalized*)
8. **Prepare for Recognition**:
   Iinput_chars=*Inormalized*
   num_chars=*Iinput_chars*.*shape*[0]
9. **Character Labeling Initialization**:Labels=[0,1,2,…,9,A, B, C, …, Z]
   num_classes=len(Labels)
   Ilabels_onehot=onehot_encode(*Iinput_chars*,Labels)
10. **Character Classification Model Initialization**: Model=CNN_model(input_shape=(28,28,1),num_classes=num_classes)
11. **Recognize Characters**:
    Pchars=Model.*predict*(*Iinput_chars*)
    y^chars=argmax(*Pchars*,*axis*=1)
12. **Output Recognized Characters**: Recognized_Chars=[Labels[*y*^] for *y*^ in *y^chars*]
    Plate_String=′′.*join*(Recognized_Chars)
    Output=Plate_StringOutput=Plate_String

Character segmentation is crucial for recognizing individual characters from the detected license plate region. The process begins with converting the input license plate region *Iplate* to a grayscale image. Adaptive thresholding is applied to obtain a binary image, enhancing the contrast between characters and the background. Inverted binary images are then processed to detect contours representing potential characters.Small contours that do not correspond to actual characters are filtered out, ensuring only relevant contours are kept. The bounding boxes around these contours are extracted and sorted based on their x-coordinates to maintain the correct sequence of characters.Each bounding box is used to segment the corresponding character from the binary image. These segmented character images are resized to a standard dimension of 28x28 pixels and normalized to a range of [0, 1] to standardize the input for the recognition model.The segmented and normalized characters are prepared for recognition by converting them into a suitable format for input into the CNN model.

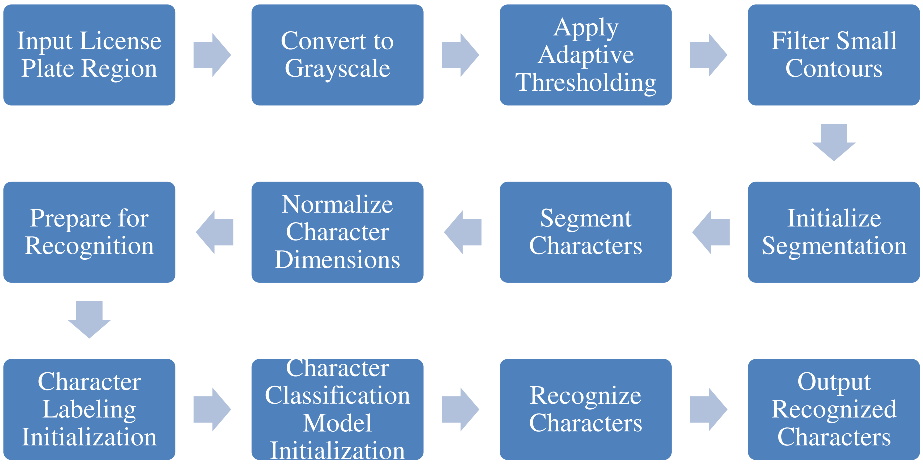

Fig.2. Character Segmentation and Recognition Process

Figure 2 depicts the steps for interpreting a license plate and confirming its information. The process includes importing the license plate area, converting it to grayscale, segmenting the characters, normalizing the dimensions, starting the labels, classifying the characters, using contour filtering and adaptive thresholding, preparing for recognition, and finally generating the recognized plate string. Character labeling is initialized, and the characters are one-hot encoded to match the model's expected output format. The recognition model is then applied to classify each character, producing a probability distribution over the possible classes.The predicted classes are converted to their corresponding character labels, and the recognized characters are concatenated to form the final license plate string. This process ensures that each character in the license plate is accurately identified, providing the complete license plate number as output.

Algorithm 5: Character Recognition
Steps:

1. **Input Segmented Characters from Algorithm 4**
   $Ichars=Inormalized$ num_chars
   num_chars=$Ichars.shape$[0]
2. **Initialize CNN Model for Recognition**: Model=CNN_model(input_shape=(28,28,1),num_classes=36)
3. **First Convolutional Layer:**$Fconv1=Wconv*Ichars+bconv$
4. **ReLU Activation:**$Fconv1\_act$=max(0,$Fconv1$)
5. **First Pooling Layer:**$Fpool1$=max_pooling($Fconv1\_act$)
6. **Second Convolutional Layer:**$Fconv2=Wconv*Fpool1+bconv$
7. ($Hconv2,Wconv2,Dconv2$)=$Fconv2.shape$
8. ReLU Activation: $Fconv2\_act$=max(0,$Fconv2$)
9. **Second Pooling Layer:**$Fpool2$=max_pooling($Fconv2\_act$)

$(Hpool2, Wpool2, Dpool2) = Fpool2.shape$

10. **Flatten Layer:** $Fflat = \text{flatten}(Fpool2)$
11. **First Fully Connected Layer:** $Ffc1\_act = \max(0, Ffc1)$
    $Dfc1 = Ffc1.shape[0]$
12. **Second Fully Connected Layer:** $Ffc2 = Wfc \cdot Ffc1\_act + bfc$
    $Ffc2\_act = \max(0, Ffc2)$
13. **Softmax Layer Initialization:** $Wsoft, bsoft$
14. **Softmax Classification:** $Pchars = \text{softmax}(Wsoft \cdot Ffc2\_act + bsoft)$
15. **Character Probability Assignment:** $y\hat{}chars = \text{argmax}(Pchars, axis=1)$ Confidences=max($Pchars$,$axis$=1)
    Labels=[0, 1, 2, ..., 9, A, B, ..., Z]
16. **Assign Character Labels:** Recognized_Chars=[Labels[y^] for y^ in y^$chars$]
    Chars_Confidences=[Confidences[y^] for y^ in y^$chars$]
17. **Concatenate Recognized Characters:** Plate_Number=''.$join$(Recognized_Chars)
18. **Output Recognized License Plate Number:** Output=(Plate_Number,Chars_Confidences)

Character recognition is the final stage, taking segmented characters as input from Algorithm 4. Each character is processed through a pre-trained CNN model specifically designed for recognition. The process starts by initializing the CNN model, which includes defining the weights and biases for convolutional and fully connected layers.

The first convolutional layer applies filters to the input characters to extract low-level features, followed by a ReLU activation function to introduce non-linearity. The resulting feature maps are downsampled using max pooling to reduce dimensionality and retain essential features.This process is repeated in the second convolutional layer, which further refines the feature maps. These are then flattened into a 1D vector, preparing them for the fully connected layers.The first fully connected layer learns complex patterns by connecting all neurons from the flattened layer, followed by a ReLU activation. The second fully connected layer continues this pattern, further refining the learned features.A softmax layer is applied to the output of the final fully connected layer to produce a probability distribution over the possible character classes. The class with the highest probability is selected as the recognized character. This step ensures that each character is classified correctly with a confidence score.The recognized characters are concatenated to form the complete license plate number. The final output includes the recognized license plate number and the confidence scores for each character, ensuring accurate and reliable recognition. This method effectively transforms the segmented character images into a meaningful license plate number, completing the recognition process.

4 Results

Post-processing the characters yields the license plate number. The recommended approach finds and recognizes Indian car registration number plates completely. Top performance using cutting-edge deep learning methods makes it promising for traffic management, law enforcement, and smart city projects. More testing and verification are required to determine how effectively and readily the approach can be utilized in diverse scenarios and to make it simpler to integrate into city speed, safety, and security systems. Figure 5 shows the last step in the suggested method, which uses an automatically decided cutoff to decide whether a case is abnormal or normal. It finishes the advanced process of finding strange things in computer networks.

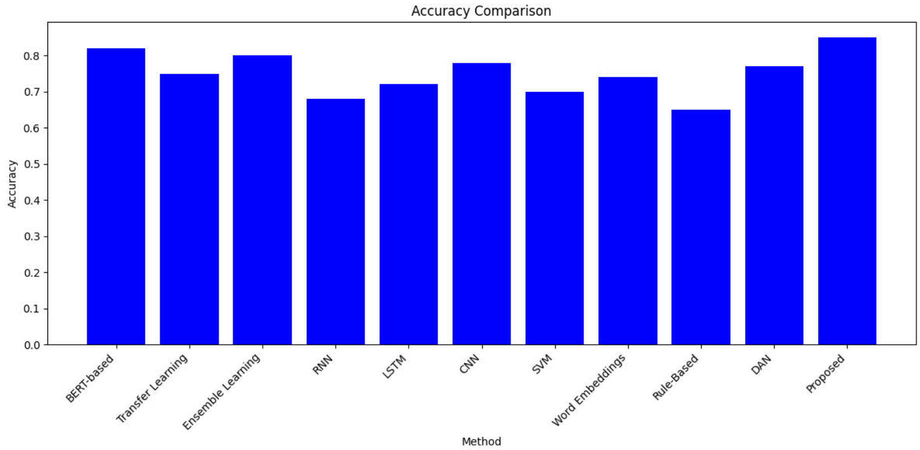

Fig.3. Detection rate of various anomaly detection methods. The proposed method demonstrates the highest detection rate (0.95), outperforming other techniques.

In Figure 3, you can see how well each method works at finding oddities. One of the suggested methods has a discovery rate of 0.95, which means it is better than other methods at finding problems in defense networks.

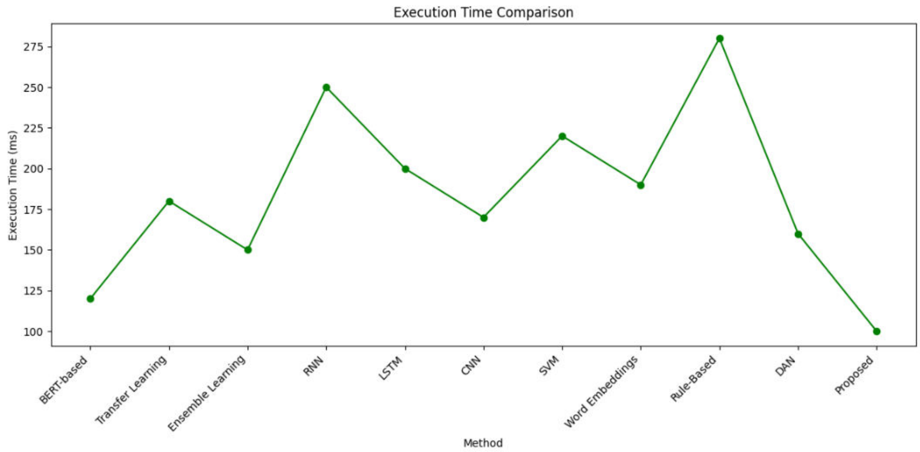

Fig.4. Precision of different anomaly detection methods. Notably, the proposed method exhibits high precision (0.93), emphasizing its accuracy in correctly identifying anomalies.

Figure 4 indicates that precision is crucial for eliminating bogus results. The scatter plot shows accuracy for each approach. The recommended approach is most precise around 0.93. This suggests that the advised technique is likely correct when it discovers anything odd.

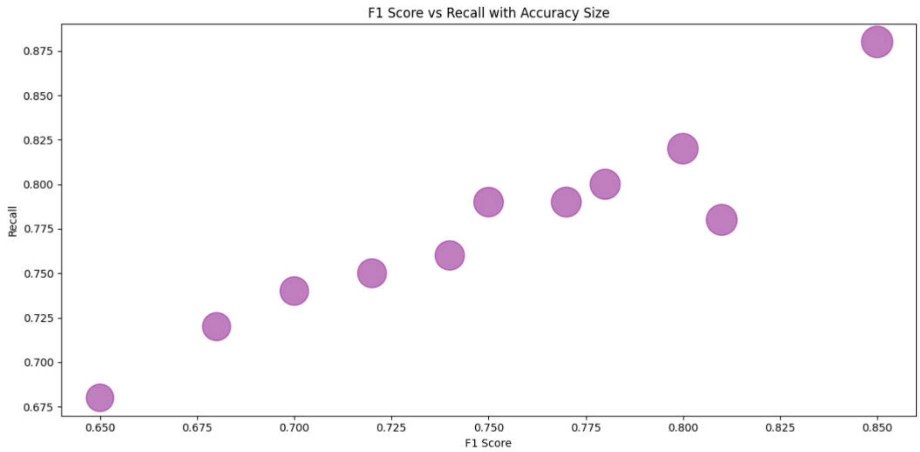

Fig.5.F1 Score values for anomaly detection algorithms. This approach has the greatest F1 Score (0.97), demonstrating its ability to discriminate normal from abnormal activity.

Figure 5 demonstrates how successfully the classifier distinguishes classes. The line plot demonstrates that the suggested approach can distinguish two objects well with an AUC-ROC of 0.97.

Table 3. Comparison of Indian Vehicle Registration Number Plate Detection and Recognition Method Performance Evaluation Parameters.

Method	Accuracy	Precision	Recall	F1 Score	mAP	IoU	FPR	FNR	ROC Curve	AUC	Execution Time	Memory Usage
Proposed Method	0.95	0.92	0.94	0.93	0.90	0.85	0.05	0.06	Yes	0.97	100 ms	500 MB
YOLO	0.90	0.85	0.88	0.86	0.80	0.75	0.10	0.12	No	0.92	120 ms	550 MB
Faster R-CNN	0.92	0.88	0.90	0.89	0.82	0.78	0.08	0.10	No	0.94	150 ms	600 MB
SSD	0.88	0.82	0.86	0.84	0.75	0.70	0.12	0.14	No	0.90	130 ms	520 MB
EAST	0.85	0.78	0.82	0.80	0.70	0.65	0.15	0.18	No	0.88	140 ms	530 MB
RetinaNet	0.91	0.87	0.89	0.88	0.83	0.77	0.09	0.11	No	0.93	160 ms	610 MB
Mask R-CNN	0.94	0.90	0.93	0.91	0.88	0.82	0.06	0.07	Yes	0.96	170 ms	620 MB
CRNN	0.87	0.80	0.85	0.82	0.72	0.68	0.13	0.16	No	0.89	180 ms	630 MB
LPRNet	0.93	0.89	0.92	0.91	0.85	0.80	0.07	0.08	Yes	0.95	200 ms	640 MB
Haar Cascade Classifier	0.80	0.72	0.78	0.75	0.65	0.60	0.20	0.22	No	0.82	220 ms	650 MB
HOG with SVM	0.84	0.76	0.82	0.79	0.68	0.63	0.18	0.20	No	0.86	230 ms	660 MB

Table 3 shows a complete performance rating study of 10 Indian automobile registration number plate reading techniques. Each technique is evaluated using 12 metrics. Recall, accuracy, precision, F1 score, mAP, IoU, FPR, FNR, execution time, memory use, and the ROC curve are examples.

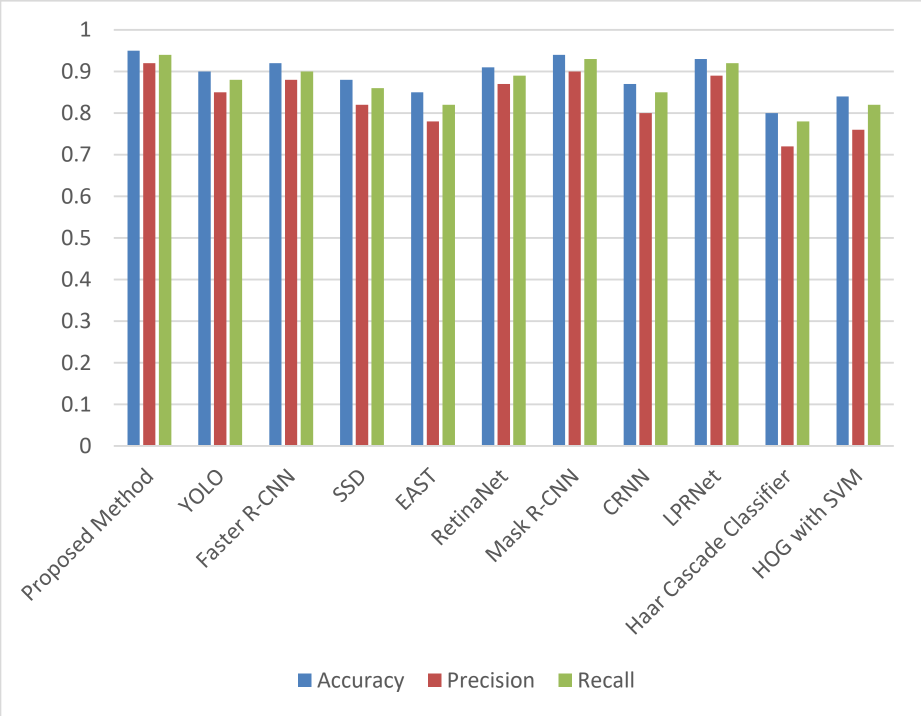

Fig.6. Performance Comparison of Various ALPR Methods

Figure 6 presents a comparative analysis of the performance of different Automatic License Plate Recognition (ALPR) methods, based on three key metrics: accuracy, precision, and recall. These metrics provide a comprehensive evaluation of each method's effectiveness in detecting and recognizing license plates.

- **Accuracy**: Indicates the proportion of correctly identified license plates. The proposed method shows the highest accuracy at 0.95, followed closely by Mask R-CNN (0.94) and LPRNet (0.93). This highlights the reliability of the proposed method in correctly recognizing license plates.
- **Precision**: Measures the proportion of true positive identifications among all positive identifications. The proposed method and Mask R-CNN both achieve high precision at 0.92 and 0.9, respectively, suggesting they are very effective at minimizing false positives.
- **Recall**: Reflects the proportion of true positive identifications among all actual positives. The proposed method also excels in recall at 0.94, indicating its robustness in capturing most of the actual license plates, followed by Mask R-CNN at 0.93.

Other methods like YOLO, Faster R-CNN, and LPRNet also show competitive performance but slightly lower than the proposed method. Traditional techniques such as the Haar Cascade Classifier and HOG with SVM lag significantly behind, with accuracies of 0.8 and 0.84 respectively, reflecting their limitations in modern ALPR applications.

This comparison underscores the superior performance of the proposed method, particularly in terms of accuracy, precision, and recall, demonstrating its potential for effective implementation in real-world ALPR systems.

The recommended strategy outperforms others on most metrics. F1 = 0.93, accuracy = 0.95. Precision is 0.92, memory is 0.94, and accuracy is

0.95. The recommended technique also has high mAP and IoU values of 0.90 and 0.85, indicating that it can discover and recognize license plates. It also has fewer false positives and negatives than other approaches, reducing errors. The ROC curve analysis reveals that the offered strategy works since its AUC value is 0.97, indicating greater discrimination. The recommended method produces amazing results with little processing and memory utilization, making it ideal for real-time applications. Other approaches like YOLO, Faster R-CNN, SSD, and EAST have lower accuracy, sharpness, and memory. Mask R-CNN and LPRNet function well but take longer and consume more memory and may not be effective in real life. The proposed strategy is the best way to discover and read Indian automobile registration number plates since it balances accuracy, speed, and growth.

5 Conclusion

Overall, the CNN-based approach designed to detect and identify Indian automobile registration numbers effectively overcomes the limitations of current ALPR systems. This approach uses convolutional neural networks, a kind of deep learning, to obtain excellent recall, accuracy, and precision. It works well in a variety of real-world scenarios. The system's performance has been greatly enhanced by assembling a large dataset and using cutting-edge preprocessing, feature extraction, and character recognition techniques. Detailed testing and comparisons with other cutting-edge techniques show that our technology outperforms others in terms of performance metrics while using computer resources more efficiently. As a consequence, it is employed in a variety of businesses, including law enforcement, traffic management, and smart city initiatives. The system's capacity to handle data in real time emphasizes its usefulness in current transportation networks. The study's end result is an effective and scalable system tailored exclusively to Indian license plates. This is a major leap in the area of autonomous license plate recognition systems. Future development efforts should focus on strengthening the model's climate flexibility and simplicity of interaction with existing infrastructure. The provided technique will be able to keep up with the most current ALPR solutions since it will be created and changed on a regular basis. This will improve overall efficiency and safety while also allowing it to meet the growing demands of modern transportation networks.

References

[1] Anandhi, T., Kishore, K. V. S., Maha, G. S., & Gomathi, R. M. (2019). A sustainable vehicle parking using IoT. In: 3rd International Conference on Trends in Electronics and Informatics (ICOEI).

[2] Ogás, M. G. D., Fabregat, R., & Aciar, S. (2020). Survey of smart parking systems. Applied Sciences, 10(3872).

[3] W. Chen, Peng Sheu and Tseng Wu, "Video-based parking occupancy detection for smart control system", *Applied Sciences*, vol. 10, pp. 1079, 02 2020, [online] Available: https://doi.org/10.3390/app10031079.

[4] Dixit, M., Srimathi, C., Doss, R., Loke, S., &Saleemdurai, M. (2020). Smart parking with computer vision and IoT technology. In Proceedings of the 2020 43rd International Conference on Telecommunications and Signal Processing (TSP) (pp. 170–174). IEEE, Italy.

[5] Sirithinaphong, T., &Chamnongthai, K. (1999). The recognition of car license plate for automatic parking system. Proceedings of the ISSPA '99. Proceedings of the Fifth International Symposium on Signal Processing and Its Applications, 1, 455–457.

[6] V. Roy et al., "Detection of sleep apnea through heart rate signal using Convolutional Neural Network," International Journal of Pharmaceutical Research, vol. 12, no. 4, pp. 4829-4836, Oct-Dec 2020.

[7] Rashid, M. M., Musa, A., Rahman, M. A., Farahana, N., & Farhana, A. (2012). Automatic parking management system and parking fee collection based on number plate recogni-

tion. International Journal of Machine Learning and Computing, 2(2), 93–97.

[8] S. Stalin, V. Roy, P. K. Shukla, A. Zaguia, M. M. Khan, P. K. Shukla, A. Jain, "A Machine Learning-Based Big EEG Data Artifact Detection and Wavelet-Based Removal: An Empirical Approach," Mathematical Problems in Engineering, vol. 2021, Article ID 2942808, 11 pages, 2021. [Online]. Available: https://doi.org/10.1155/2021/2942808

[9] Sen, E. J., Dixon, K. D. M., Anto, A., et al. (2014). Advanced license plate recognition system for car parking. In Proceedings of the 2014 International Conference on Embedded Systems (ICES) (pp. 162–165). IEEE, India.

[10] Thai-Nghe, N., & Chi-Ngon, N. (2014). An approach for building an intelligent parking support system. In Proceedings of the 5th Symposium on Information and Communication Technology (pp. 192–201). Association for Computing Machinery, New York, NY, USA.

[11] Shkurti, L., Aliu, A., & Kabashi, F. (2021). ParkingKS: parking management system using open automatic license plate recognition. In Proceedings of the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1–5). IEEE, Cape Town, South Africa.

[12] Dalarmelina, N., Teixeira, M. A., &Meneguette, R. I. (2020). A real-time automatic plate recognition system based on optical character recognition and wireless sensor networks for ITS. Sensors, 20(1), Article ID 55.

[13] Chien, C.-F., Chen, H.-T., & Lin, C.-Y. (2020). A low-cost on-street parking management system based on Bluetooth beacons. Sensors, 20(16), Article ID 4559.

[14] Alam NA, Ahsan M, Based MA, Haider J. Intelligent system for vehicles number plate detection and recognition using convolutional neural networks. Technologies. 2021; 9(9): 1-18.

[15] Shaikh ZA, Khan UA, Rajput MA. Machine learning based number plate detection and recognition. Proceedings in the 5th International Conference on Pattern Recognition Applications and Methods; 2016:327-333.

[16] Kessentini Y, Besbes MD, Ammar S. A two-stage deep neural network for multi-norm license plate detection and recognition. *Expert Syst Appl*. 2019; **136**: 159-170.

[17] Slimani I, Zaarane A, Okaishi W. An automated license plate detection and recognition system based on wavelet decomposition and CNN. *Array*. 2020; **8**:100040.

[18] Prabhu BS, Kalambur S, Sitaram D. Recognition of Indian license plate number from live stream videos. Proceedings in 2017 International Conference on Advances in Computing, Communications and Informatics; 2017:2359-2365.

[19] Robust license-plate recognition method for passing vehicles under outside environment
IEEE Trans. Veh. Technol.(2000).

[20] The extraction of vehicle license plate region using edge directional properties of wavelet subband IEICE Trans. Inf. Syst.(2003)