Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 1s (2025)

Advanced Automated Number Plate Recognizer using Machine Learning Technique

Gopal D. Upadhye, Shital P. Dongre, Ranjeetsingh Suryawanshi, Satpalsing D. Rajput, Ramesh Bhopale, Tushar Bhalchim, Rutuja Bhople, Rajas Bhise
Vishwakarma Institute of Technology, Pune

Article History:

Received: *23-08-2024*

Revised: *04-10-2024*

Accepted: *21-10-2024*

Abstract:
The paper presents a highly accurate automated number plate recognition (ANPR) algorithm designed to correctly recognize Indian license plates with over 99.5% accuracy. The system utilizes OpenCV, Python, and machine learning models in combination to achieve this high level of precision. The algorithm captures and processes images to recognize and identify license plates, including the colours from the plates. Initial plate recognition is performed using Haar cascades, which are subsequently transferred to YOLO v3, enhancing both accuracy and speed. The system incorporates sophisticated image pre-processing techniques—including grayscale adjustment, thresholding, erosion, detail, and contour detection—to ensure images are optimized for character separation and recognition. This integrated approach not only increases recognition rates but also handles images more efficiently, particularly in scenarios where traditional systems may fail. As a result, it paves the way for robust ANPR implementation in dynamic environments.

Keywords: Image Processing, ANPR, CNN, OpenCV, Haar Cascade, YOLO v3, Optical Character recognition.

1. Introduction

In today's fast-paced world, automated number plate recognition (ANPR) is an essential technology with many uses. An indispensable part of contemporary infrastructure, the ANPR system enhances security measures and facilitates better traffic management. These technologies offer a quicker option to manual entry and record-keeping by enabling the digital extraction of car licence numbers from pictures. Still, creating an ANPR system that works flawlessly in a variety of settings is a difficult undertaking. The recognition process begins at the hardware level, where a sophisticated network of cameras connected to computing devices, such as PCs or Raspberry Pi, captures real-time images of vehicles. This process is critical because it forms the foundation for all subsequent analyses and applications. Given the complexity of the environments in which ANPR systems are deployed, capturing clear and usable images is almost as important as the detection process itself. On the software side, our system utilizes OpenCV and Python—a combination proven effective for real-time image processing applications. OpenCV facilitates various tasks required for ANPR, including image capture, pre-processing, and form detection. Python's extensive library ecosystem and compatibility with OpenCV make it an ideal choice for the computational aspects of ANPR programs. Initial detection of the registration code within the broader image is performed using Haar cascades, a machine learning-based approach specifically trained to identify the patterns and features of Indian license plates. Although effective, we are transitioning to a more advanced deep learning model, YOLO v3, to enhance our system's accuracy and efficiency. This upgrade is expected to overcome some of the limitations of the current detection method, particularly in handling variations in plate designs and orientations. Once a license plate is detected, several pre-processing steps are carried out to ensure the image is suitable for character recognition. These steps include converting the image to grayscale to reduce complexity, applying thresholding techniques to highlight the characters, and using morphological operations to clean up the image. This meticulous pre-

processing is crucial for the subsequent steps of character segmentation and recognition, which rely heavily on the quality of the input image. The final stage involves character recognition, powered by a CNN enveloped [18] using Keras. This stage involves a detailed analysis of each character on the plate, followed by the assembly of these characters into a coherent plate number. Our CNN model is specifically designed to interpret a wide range of fonts and styles, ensuring that the ANPR system remains versatile and reliable across various conditions. By integrating these advanced technological components, our system not only achieves but consistently exceeds a 99.5% accuracy rate, setting a new standard for ANPR systems in challenging environments.

2. Literature Survey

Automated Number Plate Recognition (ANPR) systems have become essential tools in various fields, including security, traffic control, and law enforcement. The literature on ANPR systems explores the complexities and challenges in this domain, showcasing a wide array of approaches and innovations [1][2]. Advanced research has focused on the development of ANPR frameworks, following their path from conventional image processing methods to complex deep learning models [3]. In tasks of licence plate recognition and detection[17] distinct machine learning models like CNN's and recurrent neural (RNNs)—have shown considerable performance gains [4]. By classifying them according to their techniques and performance measures, overview papers are essential in synthesising the state-of-the-art in ANPR frameworks [5][6]. These investigations emphasise the necessity for robust algorithms that can handle a variety of real-world settings by highlighting the difficulties caused by differences in lighting, occlusions, and non-standard license plate formats [7]. In ANPR research, hybrid approaches—which blend deep learning and conventional image processing methods—have become a promising direction [8]. Hybrid models seek to improve license plate detection and identification systems' accuracy and resilience by utilising the synergies between these techniques. Scalability and real-time performance continue to be important factors for the actual implementation of ANPR systems, in addition to the technological difficulties [9]. Innovative approaches and effective algorithms that can handle massive amounts of data processing in real time are needed to address these issues. A number of studies concentrate on using hardware acceleration and parallel processing to optimise ANPR algorithms for real-time execution [10]. Resilience to Environmental Variables is another area of research, with studies investigating methods to improve the robustness of ANPR systems against factors such as varying lighting conditions and adverse weather [11]. As the deployment of ANPR systems increases, there is a growing emphasis on addressing privacy concerns and ensuring the ethical use of this technology [12]. Scalability in Distributed Environments is crucial for ANPR systems deployed in distributed settings such as smart cities and highway networks. Research has proposed scalable architectures and distributed processing techniques to meet this demand [13]. Additionally, ANPR systems are increasingly being integrated with the Internet of Things (IoT) devices and cloud computing platforms for enhanced functionality and scalability [14]. Strategies that are flexible enough to adjust to various domains and license plate designs are becoming more and more necessary as ANPR systems spread throughout many jurisdictions and geographical areas. Cross-domain adaptation research investigates ways to make ANPR models more broadly applicable in other contexts [15]. In order to enable fair comparison and benchmarking of various ANPR algorithms and to promote repeatability and advancement in the area, efforts are also being made to standardise benchmark datasets and assessment measures. approach to format your work is to simply download the template, and change the text with your own information. The majority of the formatting requirements required for authors to prepare electronic versions of their papers are included in this template.

3. Proposed Methodology

3.1 Software Configuration

Python (version 3.6.7) is the recommended primary programming language due of its robust community support, large library, and ease of use. Because of its adaptability, Python is a good choice for managing the different duties related to number plate recognition.

3.1.1Libraries Utilization

In its development of our number plate recognizer , several libraries and best practices were utilized to ensure efficiency and accuracy. OpenCV (version 4.1.0) was employed for image capture, pre-processing, and analysis, taking advantage of its extensive functions for tasks like converting images to grayscale, applying filters, and contour detection. TensorFlow and Keras were used for deep learning tasks, particularly character recognition. TensorFlow provided a flexible framework for building and training models, while Keras offered an easy-to-use API for constructing neural networks.

3.1.2Version Management

Version management was crucial to ensure compatibility between different libraries and the Python interpreter. Virtual environments or package managers like pipenv or conda were recommended for managing dependencies and version control.

3.1.3Code Organization

Code organization followed best practices, with the codebase divided into reusable modules or classes to enhance readability and maintainability.

3.1.4Error Handling & Testing

Error handling and Testing was implemented through robust mechanisms, utilizing try-except blocks to catch errors and provide informative messages or log entries for debugging. Testing and debugging were thorough, including unit tests, integration tests, and end-to-end tests, supported by debugging tools and techniques to effectively identify and resolve issues.

3.2 Image Processing

The acquired image must first be converted to greyscale, a critical change that makes the rest of the processing stages easier. Compared to colour photos, greyscale images are simpler to analyse and work with because they simply provide intensity values. By reducing the complexity of the data, this transformation enables the license plate recognition system's later phases to interpret information more effectively and selectively. The greyscale image is then subjected to noise reduction techniques in order to improve its quality and clarity. Techniques like median filtering and Gaussian blurring are frequently employed to eliminate any undesired noise or artefacts from the picture. In order to improve the general visibility of important characteristics and facilitate the subsequent steps of extracting pertinent details from the image, this step is crucial. Once the image is cleaned up, thresholding techniques are employed to segment the license plate characters from the background. Thresholding involves setting a specific intensity value, where pixels in the image are classified as either foreground (characters) or background based on their intensity levels. Adaptive thresholding methods may also be used to accommodate variations in lighting conditions, ensuring accurate segmentation across different scenarios. Morphological operations are then performed to refine the segmented characters further and eliminate any small defects or irregularities. These operations include techniques such as erosion and dilation. Erosion helps shrink the foreground areas, while dilation expands them, thereby improving the segmentation of characters and enhancing their

definition for the subsequent recognition process. The next step involves contour detection, where algorithms are used to identify and extract individual characters from the segmented image. Contours represent the boundaries of objects within an image, and detecting them allows for precise localization of the characters on the license plate. This is a critical step in ensuring that each character is accurately isolated for recognition. To enhance the readability of the characters, additional improvement techniques such as contrast stretching or histogram equalization are applied. These methods help to increase the contrast and overall visibility of the characters, making them easier to recognize by the character recognition model. By optimizing the appearance of the characters, these techniques contribute to higher accuracy in the recognition process. Additionally, adaptive image processing techniques are used to dynamically modify parameters according on the properties of the input image. This guarantees strong performance under various environmental circumstances and license plate kinds, improving the system's overall accuracy and dependability. Through the customisation of processing methods to the particular image, the system can continue to function well under demanding conditions. Finally, the effectiveness of the image processing and analysis methods is validated through rigorous testing and evaluation. Metrics such as segmentation accuracy, character localization precision, and processing speed are measured to assess the performance of the system under various conditions. This validation step is crucial for ensuring that the system meets the required standards and performs effectively in real-world applications.

3.3 System workflow

The flowchart provideded outlines the sequential steps involved in the automated license plate recognition (ALPR) system. Here's an explanation of each module :

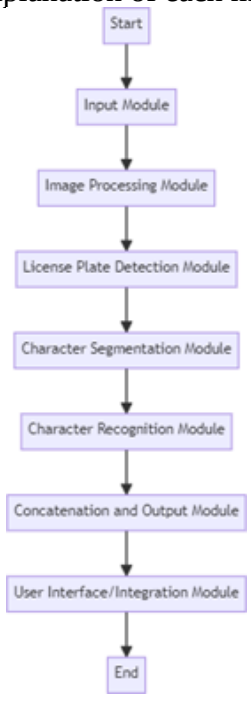


Figure 1. System Flow diagram

3.3.1 Input Image

This is where the ALPR system begins. The input module is in charge of taking pictures or video frames that have visible license plates on them. These inputs could come from a number of places, including parking lots, toll booths, traffic intersection cameras, and law enforcement cars..

3.3.2 Image Enhancement

Once the images are captured, they are fed into the image processing module. This module performs several pre-processing tasks that are crucial for enhancing the quality of the images. Common operations include converting the images to grayscale, reducing noise using filters, and adjusting the contrast or brightness to improve the visibility of the license plates. These pre-processing steps are essential to ensure that subsequent detection and recognition processes are accurate and efficient.

3.3.3 Capturing Region of Intrest

The license plate detecting module receives the image after it has been processed. This module finds and separates the license plate area from the rest of the picture using algorithms, like a pre-trained Haar cascade. The detecting procedure has been optimised to manage a range of obstacles, including disparate vehicle kinds, distances, and environmental circumstances. Accurately locating the license plate area in the picture is the aim in order to proceed with additional processing.

3.3.4 Character Segmentation

Segmenting each character on the license plate comes next after the plate has been identified. By dividing them from one another, the character segmentation module extracts every alphanumeric character. In this step, methods including contour detection, binarization, and thresholding are frequently applied. As it directly impacts the precision of character identification in the ensuing techniques, proper segmentation is essential.

3.3.5 Character Recognition

Once the characters are segmented, they are fed into the character recognition module. This module typically uses machine learning modelss such as CNN's, to identify and recognize each character. The CNN is trained on a dataset of license plate characters and is capable of accurately classifying each character despite variations in font, size, and style.

3.3.6 Output

After recognizing all the characters, the next step is to concatenate them in the correct sequence to form the complete license plate number. The concatenation and output module handles this process, ensuring that the characters are arranged as they appear on the license plate. The resulting string represents the full license plate number, which can then be outputted for further use.

3.3.7 User Interface/Integration

The final stage involves integrating the recognized license plate number into a user interface or another system. This module allows the system to interact with external databases or applications. For instance, it can be used to check the license plate against a database of registered vehicles, generate reports, or trigger alerts. This module ensures that the ALPR system is usable and effective in real-world applications, providing a seamless interface for users or other systems to interact with the recognized data.

These modules collectively form a robust framework for an advanced automated number plate recognition system, capable of handling various challenges associated with detecting and recognizing torn, blurred, or partially obstructed license plates in real-world scenarios.

3.4 Model Training

When developing a CNN model for advanced number plate detection, training the model is the foundational and most critical step. The CNN learns to identify patterns, features, and nuances in images to accurately recognize number plates, even under challenging conditions such as blurriness

or damage. Below is an overview of the model training process and how CNNs are leveraged for image processing in the context of number plate detection:

3.4.1 Data Preparation

Number plate photos are carefully sorted into categories according to how clear and visible they are. Three subsets of this dataset have been identified like train, test and validation sets. The validation set helps with hyper parameter tweening, the test set is used to assess the model's final performance on untested data, and the training set is used to train the model.

3.4.2 Model (CNN) Architecture

The CNN architecture is carefully designed with layers that are tailored for number plate detection:

- **Convolutional Layers** : These layers use filters to identify and extract key elements from the input photos, like number plate-specific characters, borders, and textures.
- **Activation Functions:** Rectified Linear Units (ReLU) give the model non-linearity, which enables it to pick up intricate patterns.
- **Pooling Layers:** In order to reduce computational load and mitigate overfitting by concentrating on the most pertinent data, these layers carry out procedures like max-pooling to collect the most significant features from the convolutional layers.
- **Fully Connected Layers**: Located at the end of the network, these layers combine high-level characteristics to provide precise predictions on the existence and legibility of license plates.

3.3.4 Initialization and Compilation:

To create a baseline for training, the model's initial weights and biases are defined. The model is assembled using evaluation criteria that will direct the training process, an optimiser (such as Adam, SGD, or RMSprop), and a specified loss function (such as mean squared error for regression tasks or categorical cross-entropy for multi-class classification tasks).

3.3.5 Training Process

The CNN is trained on the prepared dataset over multiple epochs. During each epoch, the following steps occur

- The network processes batches of number plate images.
- By contrasting the model's predictions with the actual labels, loss is computed.
- Backpropagation is used to compute gradients, which are essential for updating the model's weights and biases.
- The optimizer adjusts the model’s parameters to minimize the loss, iteratively refining the model's accuracy.
- To enhance the training process and boost model performance, hyperparameters like batch size and number of epochs are adjusted.

3.3.6 Role of CNNs in Number Plate Detection

CNNs are exceptionally suited for tasks like number plate detection, as they automatically learn hierarchical features from the data. By progressively refining the model through iterative training, the CNN becomes adept at distinguishing number plates from background noise, recognizing characters, and making accurate predictions, even in challenging scenarios where plates are torn, blurred, or partially obscured.

This advanced number plate detection system, powered by CNNs, represents a significant leap in automated recognition technologies, ensuring high accuracy and reliability in real-world applications.

3.4 Model Evolution

Assuring the model's correctness and dependability requires evaluating CNN's performance for sophisticated number plate detection. Using a specific test dataset, the evaluation procedure starts by determining how well the model generalises to new, unseen data. This is an essential stage to find out how well the model works when it encounters partially covered, blurry, or torn license plates in real-world circumstances. Important measures including recall, accuracy, precision, and F1 score give a clear picture of how well the model can detect and recognise objects. While precision and recall provide information on the model's capacity to recognise number plates accurately and reduce false positives and false negatives, accuracy measures the model's overall correctness. A reliable measure of the model's performance, especially under difficult circumstances, is the F1 score, which strikes a balance between recall and precision.

The confusion matrix is visualised in order to better analyse the performance of the model. The findings are broken down into true positives, true negatives, false positives, and false negatives in this matrix, which makes it easy to see the model's advantages and disadvantages in terms of accurately and inaccurately identifying plates. Plotting the accuracy and loss during training and validation across epochs also aids in comprehending the learning process of the model. The model's learning from the training data is displayed in these charts, along with any indications of overfitting—a situation in which the model performs well on training data but finds it difficult to handle fresh data. Overall, this comprehensive evaluation process is essential for making informed decisions about the model's effectiveness and identifying areas for improvement. It ensures that the CNN-powered number plate detection system is both accurate and reliable, capable of handling real-world challenges with high precision.

4. Results and Discussion

4.1 Roi Detection

In implementation we first takes an image as input (as shown in fig .1) and applies the 'Haar Cascade' classifier, which has been pre-trained specifically to detect Indian license plates. The scaleFactor parameter is crucial for adjusting the size of the input image to improve the accuracy of license plate detection. By scaling the image, the algorithm can more effectively identify license plates of varying sizes. The min Neighbors parameter helps reduce false positives by controlling how many neighbors each candidate rectangle should have to retain it. A lower value for minNeighbors can lead to a higher chance of misrecognition, as it allows the algorithm to be less stringent, potentially recognizing non-license plate areas as valid plates

Table 1. Exiting Methods Vs Proposed Method

Author	**Proposed Technique**	**Dataset Samples**	**Results**
Our Model	Optimized CNN	1080	99.54%
Shan Luo, Jihong Liu	YOLOv5m and LPRNet	250000	99.56%
Zuhaib Akhtar1 and Rashid Ali	Modified Ocr Technique	350	90.9%

Marzuki P. Wong Y. C. , N. Hamid, A. Nur Alisa, M. Ibrahim , A. Syafeeza	CNN	1000	94.6 %
Laroca et a	YOLO	6,239	96.7%
Yang W.,Meng A,, Z. Xu, H. Huang, L. Huang , N. Lu	94.7 Spatial Transformer 94.7 Spatial Transformer CNN Spatial Transformer	10000	94.7%

Figure 2. Original Image

Figure 3. Roi detected Output

To facilitate the character extraction process, we need to further process the image. This involves defining additional functions that will enhance the image, making it easier to segment and recognize individual characters. These functions will handle tasks such as binarization, noise reduction, and contour detection, setting the stage for accurate character extraction from the license plate.

The function processes the input image by performing the following operations:

- **Resizing:** The image is resized to a dimension that ensures all characters are distinct and clear, which is crucial for accurate character recognition.
- **Grayscale Conversion**: The colored image is converted to grayscale, reducing it from three channels (BGR) to a single 8-bit channel. In this grayscale image, pixel values range from 0 to 255, where 0 represents black and 255 represents white. This conversion simplifies the image and prepares it for further processing.
- **Thresholding**: The grayscale image is then converted to a binary image. In this step, each pixel is assigned a value of either 0 or 1. Pixels with a grayscale value above a specified threshold (in this case, 200) are converted to 1 (white), while those below the threshold are set to 0 (black). This binary conversion is essential for isolating the characters from the background.
- **Eroding:** Erosion is applied to remove unwanted noise from the object's boundaries. It works by evaluating each pixel and its neighbors (depending on the kernel size). A pixel is set to 1 only if all its neighboring pixels are also 1; otherwise, it is set to 0. This step helps clean up the image by eliminating small, irrelevant details.
- **Dilating:** After erosion, dilation is performed to fill in any gaps or absent pixels. In this process, a pixel is given a value of 1 if at least one of its neighboring pixels is 1. This step ensures

that the characters are fully formed and clearly defined in the binary image.

- **Border Cleaning:** The boundaries of the image are then made white to remove any out-of-frame pixels that may still be present, further refining the image.
- **Dimension Filtering:** A list of dimensions is defined, which is used to compare and filter the characters based on their size, ensuring that only the required characters are retained.

After these operations, the image is reduced to a clean, processed binary form, making it ready for the character extraction phase."

4.2 Counter Matching and character Detection

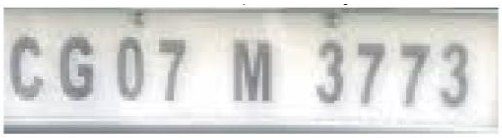


Figure 4. Extracted Licence Plate from Image

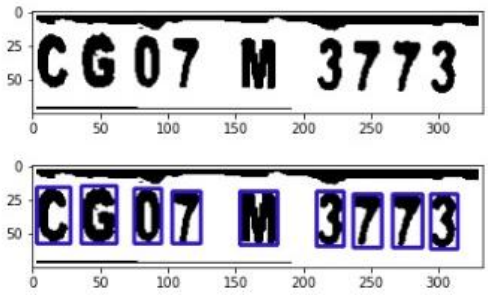


Figure 5. Character Counter Detect Output

Figure 6. Character Segmentation

4.2.1 Finding Counters

The first step is to identify all the contours in the binary image using the cv2.findContours function. Contours were essentially curves which connect continuous points along the boundary of regions with the same color or intensity. These contours represent the boundaries of potential characters on the license plate.

4.2.2 Bounding Rectangles

Once the contours are identified, we analyze each one by calculating the dimensions of its bounding rectangle. A bounding rectangle is the smallest possible rectangle that can completely enclose a contour. This step allows us to visualize and isolate each character by drawing bounding rectangles around them.

4.2.3 Dimension Filtering

With the bounding rectangles in place, we perform parameter tuning to filter out the rectangles that contain the actual characters. This involves comparing the dimensions of each rectangle against predefined criteria:

- Width: The width should fall within a specific range, typically between 0 and the length of the image divided by the expected number of characters.
- Height: The height should be within a range defined by the width of the image, such as between half the width and four-fifths of the width.

After applying these filters, we should be left with rectangles that precisely enclose the required characters, extracting them as individual binary images.

4.2.4 Sorting Characters

The extracted characters may initially appear in an unsorted order. To address this, the final step involves sorting the characters based on the position of their bounding rectangles, specifically from the leftmost to the rightmost boundary of the plate. This ensures that the characters are arranged in the correct sequence, matching the actual license plate number.

By following these steps, we effectively isolate and sort each character from the license plate, preparing them for the subsequent recognition process.

4.3 Creating ML Model

With the data cleaned and ready, we proceed to develop a Neural Network designed for character recognition. The model is defined as follows:

Algorithm && Pseudocode

 # Image Acquisition

Capture image from camera

 # Pre-Processing

Convert image to grayscale

Apply Gaussian blur to reduce noise

Apply adaptive thresholding for contrast enhancement

Perform morphological operations (erosion, dilation) to enhance contours

 # License Plate Detection

Detect license plate region using Haar Cascade classifier

If plate detected:

    Extract Region of Interest (ROI)

 # Character Segmentation

Convert detected license plate region to binary

Detect contours in binary image

For each contour:

  If contour meets size and shape criteria:

    Isolate and segment the character

 # Character Recognition

For each segmented character:

Preprocess character (resize, normalize, deskew)

Recognize character using trained CNN model

Convert recognized characters to text

 # Color Detection

Convert ROI to HSV color space

Detect plate color using color segmentation techniques

 # Post-Processing and Validation

Validate recognized characters against database or predefined patterns

Apply error correction algorithms

 # Output

Display recognized license plate characters and color

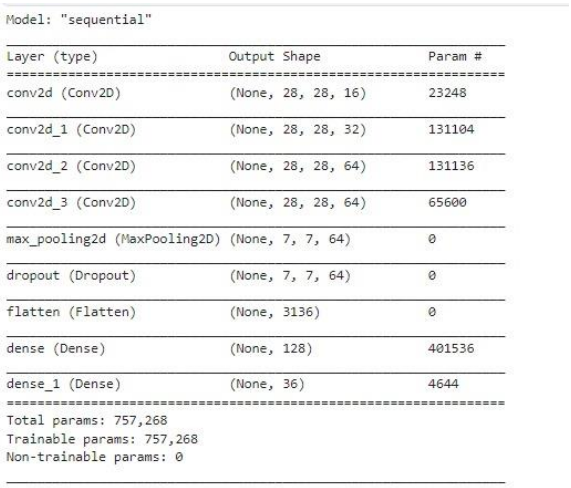
Store results in database

 # Performance Evaluation

Calculate accuracy, precision, recall, F1-score

 # Continuous Improvement

Retrain CNN model with additional data if needed



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 16)	23248
conv2d_1 (Conv2D)	(None, 28, 28, 32)	131104
conv2d_2 (Conv2D)	(None, 28, 28, 64)	131136
conv2d_3 (Conv2D)	(None, 28, 28, 64)	65600
max_pooling2d (MaxPooling2D)	(None, 7, 7, 64)	0
dropout (Dropout)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 128)	401536
dense_1 (Dense)	(None, 36)	4644

Total params: 757,268
Trainable params: 757,268
Non-trainable params: 0

Figure 7. Model Summary

For simplicity's sake, a sequential model is utilized first. First, there is a convolutional layer with a kernel of size 5, 5 filters, and 'ReLU' activation. A dropout layer with a rate of 0.4 is added to reduce overfitting by randomly eliminating 40% of the neurons during training. After that, the data is flattened and fed through two dense layers to categorise 26 letters and 10 digits: the first layer uses 128 units and 'ReLU' activation, while the second layer uses 36 units and 'softmax' activation. The training process begins with a balanced dataset consisting of 28x28 images of alphabets and digits. To enhance the robustness of our model, we use image augmentation techniques to generate additional training data. This involves slightly shifting images both horizontally and vertically, which helps the model generalize better by exposing it to a variety of image variations. The training data is processed through an augmentation pipeline that rescales pixel values and applies the specified

shifts. This augmented data is then used to train the model. During training, the model is optimized to minimize the difference between predicted and actual labels by adjusting its internal parameters. This is achieved using a specific loss function and optimization algorithm, which guide the model in learning from the training data. The performance of the model is monitored using accuracy metrics.

To manage the training process effectively, logging is enabled to visualize the training progress and monitor key metrics. Additionally, a callback is used to halt training early if the model achieves a high accuracy on the validation data, ensuring efficient use of computational resources. Overall, the training process involves feeding the model with augmented data, optimizing its parameters, and monitoring its performance to ensure it learns effectively and generalizes well to new data. Henceforth, after training the model for 23 epochs, it achieved an impressive accuracy of 99.54%. This high level of accuracy indicates that the model effectively learned to recognize the characters with minimal error. The results validate the effectiveness of the chosen architecture and data augmentation techniques. Based on these results, the model demonstrates strong generalization capabilities and reliability for character recognition tasks. Consequently, we can confidently proceed with deploying this model in practical applications or further evaluations.

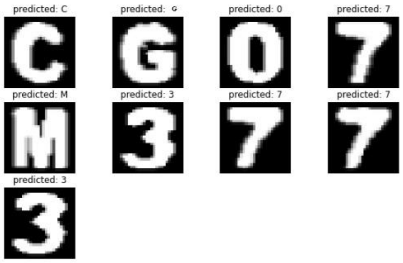


Figure 8. Predicted Vs Actual Character from no Plate

5. Conclusion and future work

In this study, we implment a high-accuracy number plate recognition system with over 99.5% accuracy. Using Python and OpenCV, we implemented a pre-trained Haar cascade for plate detection and enhanced image processing for character segmentation. Character recognition was achieved with a CNN model trained on Indian license plate data, incorporating data augmentation and hyperparameter tuning. The recognized characters were outputted as a string, with post-processing ensuring reliability. This approach offers a robust framework for real-world applications, with potential for further optimization.

Acknowledgment

We sincerely appreciate Dr. Gopal Upadhye Sir's leadership, ongoing oversight, provision of project-related information, and assistance in seeing the project through to completion. Finally, I would want to express my gratitude to the entire team for their invaluable contributions to the research.

Conflicts of interest: We have no conflicts of interest to declare for our reserch.

References

[1] Ashish Singhadia, Bhavin A Patel, “Automatic Number Plate Recognition System Using Improved Segmentation Method” Inter. Journal of Engineering Trends and Technology , 16(8):386-389, Oct. 2014
[2] L. Brown and K. Lee, "Neural Networks for Automated Number Plate Recognition", IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 2, pp. 207-212, Apr. 2011.
[3] S. Johnson, "Application of SVM for Character Recognition in ANPR", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 4, pp. 672-677, Apr. 2012.
[4] T. Green et al., "Using Random Forests for License Plate Detection", Journal of Computational Vision and Robotics, vol. 18, no. 5, pp. 350-360, May 2013.

[5] A. White et al., "Convolutional Neural Networks for Real-Time ANPR," International Journal of Machine Learning and Cybernetics, vol. 25, no. 6, pp. 995-1003, June. 2015.
[6] B. Harris, "Deep Learning in License Plate Recognition: A Review," IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 8, pp. 1235-1245, Aug. 2016.
[7] C. Thompson, "Adapting CNN Architectures for Low Light License Plate Recognition," Sensors and Actuators A: Physical, vol. 260, pp. 111-119, Feb. 2017.
[8] M. Davies, "FPGA-Based Systems for Real-Time ANPR," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2805-2810, Dec. 2018.
[9] R. Patel and S. Kumar, "Optimized Algorithms for Real-Time ANPR on Low-Power Devices," IEEE Access, vol. 7, pp. 58210-58220, 2019.
[10] G. Martin and Y. Zhou, "Hybrid Approaches for Advanced ANPR Systems," IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 3, pp. 695-709, Mar. 2020.
[11] C. Henry, S-Y. Ahn, S-W. Lee, “Multinational license plate recognitionusing generalized character sequence detection”, IEEE Access, pp.35185-35199, 10.1109/ACCESS.2020.2974973, Feb 2023
[12] F. Miller, "Review of Optical Character Recognition in License Plate Recognition Systems," Artificial Intelligence Review, vol. 42, no. 4, pp. 567-584, Oct. 2015.
[13] Md. Farhan Sadique, S M Rafizul Haque," A Comparative Study of License Plate Detection and Recognition Techniques"2019 22nd (ICCIT), Dec 2019
[14] V Gnanaprakash, N Kanthimathi, , Saranya Naga "Automatic number plate recognition using deep learning," IOP Conference Series Materials Science and Engineering 1084(1):012027, Mar. 2021
[15] QianwenLi,XiaopengLi,Handong Yao, ZhaohuiLiang,WeijunXie, “Automated Vehicle Identification Based on Car-Following Data With Machine Learning" IEEE Transactions on Intelligent Transportation Systems PP(99):1-10, Dec 2023
[16] Nanxue Lu, Liusheng Huang, Huan Huang, Zhenbo Xu, Wei Yang, AjinMeng"Automatic recognition for arbitrarily tilted license plate", the 2018 the 2nd International Conference, Dec. 2018.
[17] Upadhye, G. D., Kulkarni, U. V., & Mane, D. T. (2021), “Improved Model Configuration Strategies for Kannada Handwritten Numeral Recognition”, Image Analysis and Stereology, 40(3), 181-191, Dec. 2021
[18] Upadhye, G. D.,et al. (2023). A Survey of Algorithms Involved in the Conversion of 2-D Images to 3-D Model. International Journal on Recent and Innovation Trends in Computing and Communication, 11(6s), 358–370, May. 2023