License Plate Detection and Recognition

Pratik Pujari
Sardar Patel Institute of Technology
Department of Computer Engineering
Mumbai India

Adish Padalia
Sardar Patel Institute of Technology
Department of Information Techonology
Mumbai India

***Abstract*— The detection of Indian vehicles using their licence plates has been the most interesting and challenging research topic in recent years. In different countries, automobile licence plates have varied shapes and sizes, as well as distinct colours. This project provides a system for detecting and identifying vehicle licence plates, which will aid in the detection of authorised and unauthorised vehicle licence plates. This research proposes a technique based on simple but efficient morphological operation and Sobel edge detection algorithm. Using the bounding box method, this methodology is reduced to segment all of the letters and numbers utilised in the number plate. Following the segmentation of the numbers and characters on the number plate, the template matching approach is utilised to recognise the numbers and characters. The focus is given to accurately locate the number plate region in order to segment all of the numbers and letters in order to identify each number independently.**

Keywords— Number Plate Recognition (NPR), Optical Character Recognition (OCR), Image processing, Convolutional Neural Network (CNN), Image segmentation.

I. Introduction

Licence Plate Recognition (LPR) is a technology that combines image processing, character segmentation, and recognition to identify automobiles based on their licence plates. This system requires no additional hardware to be put on automobiles because it just uses licence plate information for identification. LPR technology is gaining traction, particularly in security and traffic control systems. Access control in buildings and parking lots, law enforcement, stolen car detection, traffic control, automatic toll collection, and marketing research are all common uses for Licence Plate Recognition Systems. For licence plate extraction, LPR apps use image processing and segmentation techniques, and each operation requires a significant amount of compute. Government rules standards used in licence plates can significantly reduce computational requirements while improving accuracy. In an ideal world, for harsh situations and serious challenges with regular visibility, special cameras, such as infrared cameras, would be available to address these aims and achieve superior results. This is due to the infrared lighting generating light reflection on the licence plate made of particular material, resulting in a distinct brightness in that portion of the image relative to the rest of it, making it easier to identify.

II. Related Work

License plate detection and segmentation are fundamental tasks in computer vision and image processing, with numerous research papers presenting unique approaches to these challenges. For example in [1], the authors present an innovative approach for License Plate Recognition (LPR) using a 30-layer Xception network to extract global and intermediate features. These features are used in a two-layer LSTM network with an attention mechanism, eliminating the need for image correction or character segmentation. A key contribution is the use of AsymCycleGAN to generate synthetic license plates for training. Their dataset (CLPD) comprises diverse license plates from various vehicles. Results show high accuracy, with 97.3%, 98.3%, and 91.9% on AOLP for different images and 88.2% and 76.8% for PKU and CLPD, using a 512 Channel CNN with 30 Xception layers, and training on three-fifths of the data. This approach simplifies LPR and performs strongly across various datasets. The authors of propose using an edge guided sparse attention network (EGSANet) to detect license plates in [2]. A VGG19 backbone network, an EGSA module, and a cascaded multi-task detection head comprise the EGSANet. The authors used 12 layers of the VGG19 network with the EGSA module instead of all 19 layers. EGSA has the advantage of using LP edge contours to solve the LP detection problem in real time. S. Rakhshani et al [3] introduced an encoder-decoder network to extract license plate features and employed eight parallel classifiers for character recognition. Their dataset included 11,000 license plates, and they achieved a 96% accuracy on 4,000 test images. The encoder-decoder pair processed grayscale images, with binary images as targets. Segmented characters were passed to eight parallel classifiers for character recognition, including numerals and characters. The dataset contained high-resolution images captured at varying times of day from a height of 15m. They trained the encoder on 1,000 ILPD11 dataset images for license plate binarization, resulting in an average recognition time of 123 milliseconds. The character recognition, trained on 7,000 ILPD11 dataset images, achieved an accuracy of 95.3%, while the method produced an average accuracy of 96.1% on the AOLP dataset. T. Vaiyapuri et al proposed using SSA-CNN, a Squirrel Search Algorithm (SSA)-based CNN, for character recognition [4]. To detect license plates, they used median filtering, edge detection, and morphological operators. Following that, they suggested using the Hough Transform to segment licence plates and SSA-based CNN to recognise characters. The authors reported F1 scores of 98% for the FZU Cars data set, 96.9% for the Stanford Cars data set, and 96.1% for the HumAIn data set.

In [5], the authors proposed a comprehensive approach focusing on the generation of synthetic images and license plate detection. They created synthetic images of 768 × 384 pixels, generating 20,000 positive and negative training images and 2,500 positive and negative validation images. For license plate detection, they introduced a feature extractor network and two sub-networks for localization and differentiation. Additionally, a character detection network processed 128 × 64 pixel inputs, producing character coordinates. Character recognition achieved an impressive 98.3% accuracy on platesmania.com images and F1 scores of 97.2% for AC, 98.9% for LE, and 98.4% for RP images on the AOLP dataset. Notably, their approach allowed flexible use as a plate or character detector by adjusting hyper parameters and leveraged synthetic data to streamline training efforts, ultimately demonstrating efficiency with average recognition times.

Gonçalves et al. [6] presented a deep multi-task network for real-time Automatic License Plate Recognition (ALPR), comprising two cascaded networks: a License Plate Detection (LPD) network for locating license plates and a License Plate Recognition (LPR) network for character recognition. The model achieved an impressive 99.3% accuracy on a public dataset, outperforming existing ALPR models, while demonstrating real-time performance on standard CPUs. This robust model has promising applications in traffic surveillance, congestion control, law enforcement, and can be further enhanced by expanding the training dataset and accommodating various license plate formats and international recognition.

Authors/ Year	Methodology	Dataset	Performance	Limitations/ Future work
Zhang et al. 2020	A tailored Xception network for feature extraction and a 2D-attention based RNN model for character decoding	AOLP Dataset (AC, LE, RP), PKU Dataset (G1, G2, G3, G4, G5)	AOLP Dataset Accuracy – 97.3% PKU Dataset Accuracy – 88.2%	The proposed model was unable to recognize some similar letters like “A” , “8” and “B”, “0”, “D”
Liang et al. 2022	Edge guided sparse attention network (EGSANet) used real time license plate detection	CCPD and AOLP Dataset	Accuracy (AOLP Dataset) AC – 99.6% LE – 99.8% RP – 99.5%	Bounding boxes for rotated or skewed could be optimized
Rakhshani et al. 2020	Deep stacked convolutional auto-encoder for detection of Iranian license plates	Over 11000 images collected from control cameras on the dual-carriageways	Correct character recognition rate (CCRR) – 96.56 %	May not effectively handle high-dimensional data due to the curse of dimensionality, which can lead to reduced performance
Vaiyapuri et al. 2021	Hough Transform, Squirrel Search Algorithm combined with CNN	Stanford Cars, FZU Cars and HumAIn 2019 Dataset	Accuracy – 97%	Cropping of image after localization could be improved
Bjorklund et al. 2019	Feature Extraction and Localization using CNN	40,000 synthetic images of Italian vehicles	Accuracy – 98.4%	Using natural training images can limit generalization as the sample distribution may not match deployment conditions, affecting the network's ability to classify new images
Gonçalves, L. et al. (2018)	Two deep learning networks: a license plate detection network (LPD) and a license plate recognition network (LPR)	Public dataset of license plate images	Accuracy of 99.3% on the test set	Model could be trained on a larger dataset of license plate images to improve its generalization ability. Model could be modified to recognize license plates from different countries.

Table 1: Comparison of related work

Table. 1 shows the detailed comparison the related work mentioned with categorizing into different section of evaluation.

III. Data Preprocessing

Data preprocessing is a critical component of the research presented in the referenced papers on license plate detection and segmentation. In the context of these studies, data preprocessing involves tasks such as image enhancement, noise reduction, and feature extraction. These processes are essential for preparing raw image data for subsequent analysis and classification.

A. Dataset

. We have two distinct datasets sourced from Kaggle to support my research and projects. The first dataset [6], is primarily used for training optical character recognition (OCR) models. This dataset provides valuable character images for the development of accurate and robust OCR systems, making it essential for tasks such as license plate text extraction and interpretation.

The second dataset [7] is specifically curated for car license plate detection. It includes a collection of images with annotated regions of license plates, making it a valuable resource for training and evaluating license plate detection algorithms. These datasets are crucial assets for advancing the fields of OCR and license plate recognition, providing the foundation for developing efficient and effective models for real-world applications.

B. Image Preprocessing

1) Grayscale Imaging

Grayscale images, also known as black and white images, are digital images in which each pixel is represented by a single value denoting the intensity of light or gray level. To convert a color image to grayscale, there are several methods, but one of the most straightforward approaches is to use the luminance or average method. The luminance method calculates the weighted average of the red, green, and blue (RGB) color channels to obtain the grayscale intensity value. The equation for converting an RGB image to grayscale using the luminance method is:

$$(G) = 0.299 * (R) + 0.587 * (G) + 0.114 * (B)$$

This equation uses the coefficients 0.299, 0.587, and 0.114, which are derived from the human eye's sensitivity to different colors.

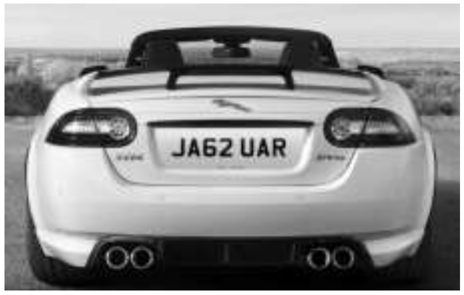

Fig 1: Grayscaled image of car with visible license plate

2) Morphological operations

In this process, we employ two morphological transformations. The Top-Hat transformation, defined as the difference between the original image and its opening, highlights small bright structures. In contrast, the Black-Hat transformation, calculated as the difference between the closing of the image and the original image, accentuates small dark structures. These operations, aimed at enhancing image details, are instrumental in revealing both bright and dark structural features within the image, making them valuable tools for a range of image processing applications.

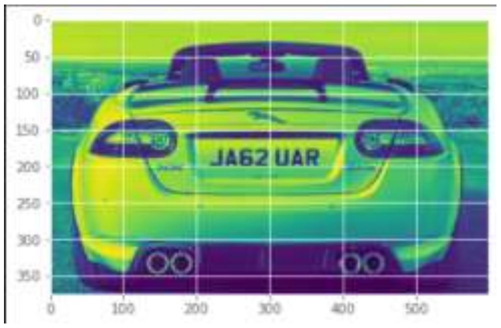

Fig 2: Morphological operations performed on the license plate image (Fig.1)

3) Adaptive Thresholding

In this image processing technique, Gaussian blurring is initially applied to a grayscale image to reduce noise and smooth it. Subsequently, adaptive thresholding is performed using the Gaussian-weighted mean of pixel values within a local neighborhood. The formula for adaptive thresholding is expressed as follows:

$T(x, y) = \{$

$maxValue, if\ I(x, y) - M(x, y) > C$

$0, otherwise$

$\}$

Here, $T(x, y)$ represents the threshold value at pixel (x, y) in the output binary image, $I(x, y)$ is the pixel value at position (x, y) in the input image $M(x, y)$ is the mean of the pixel values in the local neighborhood, and C is a user-defined constant. Adaptive thresholding effectively separates objects from the background, particularly in scenarios with varying lighting conditions.

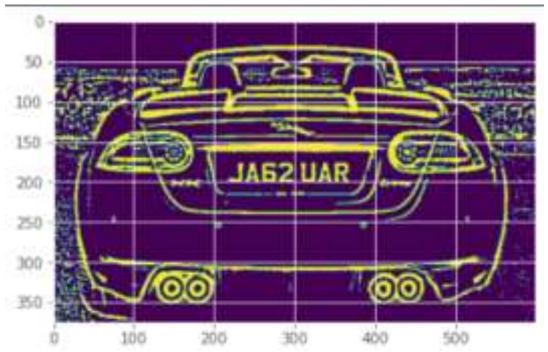

Fig 3: Adpative Thresholding performed on image (Fig.2)

C. Extracting and Visualizing Contours

a) Contour Extraction

Contours are a collection of points that define the boundaries of objects or shapes in an image. In mathematical representation, a contour C can be expressed as a set of points (x, y)

$$C = \{(x1, y1), (x2, y2), \ldots, (xn, yn)\}$$

b) Contour Visualization

To visualize the extracted contours, they can be drawn on a canvas. The canvas is represented as an image matrix, and the act of drawing contours can be expressed as a function D that modifies the canvas, where I is the canvas, C is the contour, and $color$ represents the line color:

$$I = D(I, C, color)$$

2) Character Contour Arrangement

a) Contour Extraction and Bounding Rectangles

Contours representing potential characters are identified, and bounding rectangles are calculated around them.

b) Characteristic-Based Filtering

Contours are filtered based on area, width, height, and aspect ratio.

c) Matching Contours

Contours that meet the criteria are matched based on spatial relationships, considering factors like distance, angle, area, width, and height differences.

d) Recursive Grouping

A recursive approach is used to group similar contours.

e) Visualization

The matched character contours are visualized, highlighting identified characters in the image.

After performing the possible contour operations on the image, bounding boxes of the contours are drawn

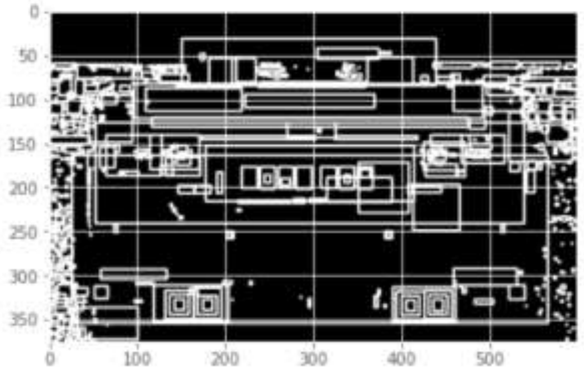

Fig 4: Contour Visualization on Fig 3

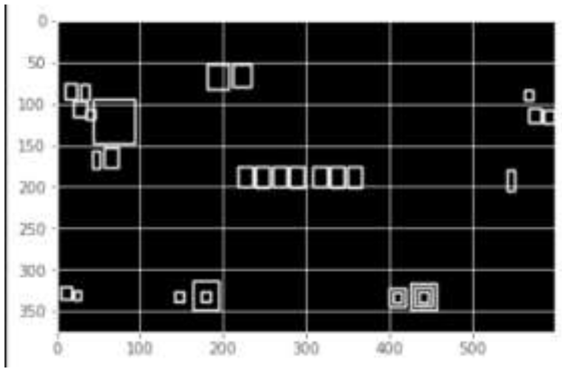

Fig 5: Possible contours visualized from Fig.4

After performing operations to get area almost similar to license plate digits, using recursive grouping 7 letters are grouped and found out.

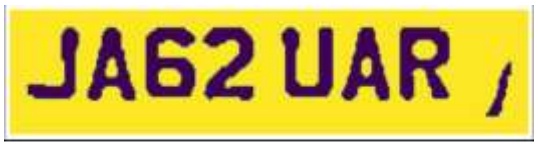

Fig 6: Licence plate extracted from original image from contours

D. Rotation and Skewing Morphed Images

In this process, matched character contours are examined to determine potential license plate regions in an image. First, the characters are sorted based on their x-coordinates, and the center of the potential license plate $plate\_cx$ and $\underline{plate\_cy}$ is calculated. The width $plate\_width$ of the potential plate is estimated as a factor of the distance between the leftmost and rightmost characters, and the height $plate\_height$ is calculated as a factor of the average character height. Angle $angle$ correction is applied to align the potential plate with the horizontal axis using a rotation matrix. The image is cropped to extract the potential license plate based on the estimated width and height. Filtering is then performed to ensure the cropped image's aspect ratio falls within a valid range. The result is a set of cropped images representing potential license plates, correctly aligned and ready for further processing, ensuring robustness in identifying license plates in images with varying orientations and layouts.

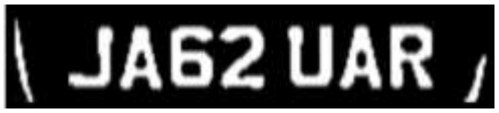

Fig 7: Skewed image straightened using correct implementation

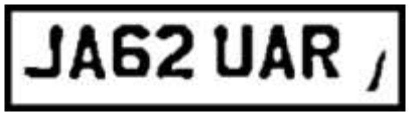

Fig 7: Resize and threshold in Fig 8

IV. METHODOLOGY

The proposed methodology encompasses a well-defined process for character recognition in images. It starts with an Input Image, which undergoes a sequence of steps for efficient recognition. The initial stage involves Preprocessing to enhance the image's quality and prepare it for further analysis. Next, Plate Region Extraction is carried out to pinpoint the area of interest containing the license plate. Subsequently, Characters and Number Segmentation isolates individual characters and numbers within the plate region. Feature Extraction is then applied to these segmented characters and numbers to capture their distinctive attributes.

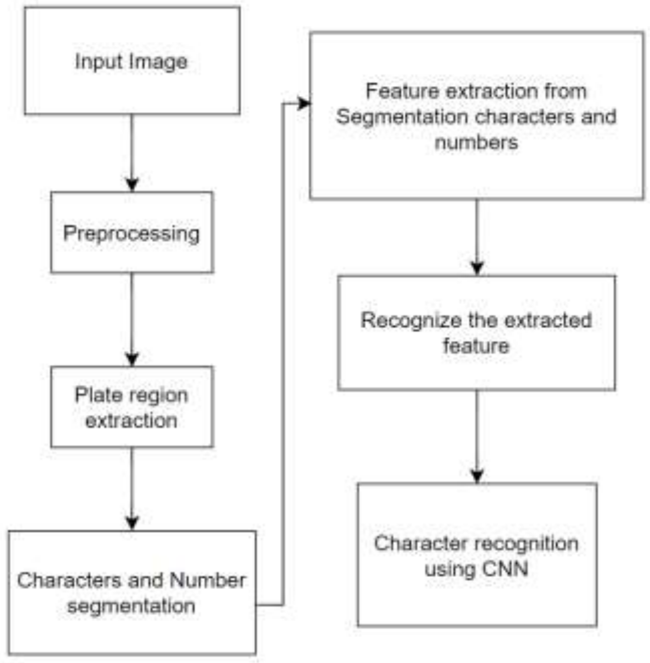
Input Image
Preprocessing
Plate region extraction
Characters and Number segmentation
Feature extraction from Segmentation characters and numbers
Recognize the extracted feature
Character recognition using CNN

Fig 8: Proposed Model Architecture

The recognized features are subjected to an advanced step of Character Recognition using Convolutional Neural Networks (CNN). This neural network architecture is specifically designed for image recognition tasks, ensuring accurate and reliable character recognition. The proposed methodology presents a systematic approach to transforming an Input Image into a structured format, ultimately leading to character recognition through the utilization of modern deep learning techniques.

A. Segmentation

This segment of code is responsible for segmenting individual characters from a license plate image. It begins by finding contours within the preprocessed license plate image. A set of potential character contours is filtered based on their dimensions, effectively isolating the characters on the license plate. Each character is then processed and resized to a consistent 24x44 pixel format with a black border, suitable for further classification. The final list of segmented characters is returned in the correct left-to-right order, providing the necessary input for character recognition. Mathematically, it involves calculating the dimensions of potential character contours, filtering these contours based on their size, and organizing them according to their x-coordinates, ensuring proper character segmentation for subsequent processing.

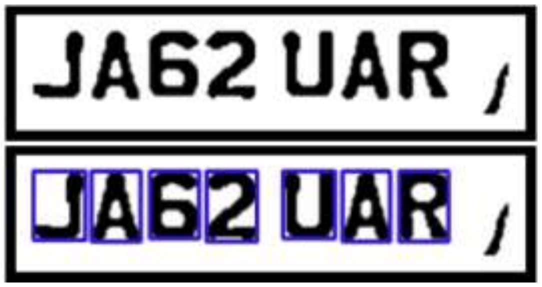

Fig 9: Segmentation image on the license plate

B. Training and Testing.

To train the neural network model for license plate character recognition, the dataset provided by Kaggle [14] was utilized. The training data was augmented using an *ImageDataGenerator* , which applied various transformations to the images, such as rescaling and shifting, to improve the model's robustness. This augmented dataset was then used for training the neural network. During training, the model architecture consisted of multiple Convolutional and Dense layers, followed by a final output layer with 36 units, each representing a different character class. The model was compiled with the sparse categorical cross-entropy loss function and the Adam optimizer.

After training, the model's performance was evaluated using a separate validation dataset. This dataset, also obtained from Kaggle, was processed in a similar manner to the training data. During testing, the model predicted characters from the license plates in the validation dataset, and its accuracy was measured to assess its recognition capabilities. The training and testing phases are crucial for ensuring that the model can accurately recognize characters on license plates, making it a valuable tool for license plate recognition applications.

C. Convolutional Neural Network Model (CNN)

The proposed CNN model consists of multiple layers, starting with Conv2D layers that apply convolution operations with varying filter sizes and numbers of filters, followed by activation functions and padding to maintain the spatial dimensions. The MaxPooling2D layer is then used for down-sampling. Afterward, the Flatten layer converts the 2D feature maps into a 1D vector. The model continues with dense layers, incorporating activation functions and dropout layers to prevent over fitting. The output layer consists of 36 units, each corresponding to a unique character class, and employs the softmax activation function for multi-class classification. The model is compiled using sparse categorical cross-entropy as the loss function and the Adam optimizer with a specific learning rate. This architecture is designed for character recognition tasks where there are 36 distinct character classes, typically corresponding to alphanumeric characters and some special symbols.

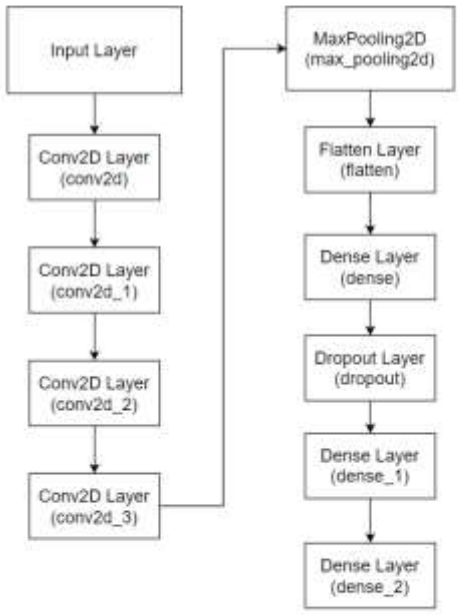

Fig 9: Convolutional Input, Hidden and Output Layer

The model is trained using data generators, which are suitable for handling large datasets. The key parameter here is the $batch\_size$, set to 4, which determines the number of samples processed in each training iteration. Additionally, a custom callback function named $stop\_training\_callback$ is used to control the training process based on specific conditions. The training process runs for a total of 20 epochs. The steps_per_epoch parameter specifies the number of steps (mini-batches) to complete an epoch, which is calculated based on the number of samples in the training data divided by the batch size. This code demonstrates the training of a deep learning model with specified parameters and the inclusion of a custom callback for enhanced training control.

D. Performance Metric

Performance metrics, such as loss and the F1 score, play a pivotal role in assessing the efficacy of machine learning models. The loss function quantifies the disparity between predicted and actual values, serving as a guide for the model to optimize its parameters. Mathematically, it can be represented as:

$$Loss = f(Yactual, Ypredicted)$$

. Here, Y_{Actual} represents the ground truth values, while $Y_{predicted}$ symbolizes the model's predictions. The choice of loss function depends on the specific problem, with common options including mean squared error, categorical cross-entropy, or others tailored to the task.

Furthermore, the F1 score, a prominent metric for classification tasks, gauges the balance between precision and recall. It is defined as:

$$F1\ Score = \frac{2(Precision * Recall)}{Precision + Recall}$$

Here, Precision reflects the ratio of true positive predictions to the sum of true positives and false positives. Recall signifies the ratio of true positive predictions to the sum of true positives and false negatives. A higher F1 score indicates a model's superior ability to correctly classify instances from a dataset while maintaining low false positive and false negative rates. It is particularly useful when dealing with imbalanced datasets, as it accounts for both false positives and false negatives.

V. Results

Over the course of 20 epochs, the model demonstrated substantial progress in its classification performance. The F1 score, a key performance metric, saw a remarkable increase from an initial 8.68% to an impressive 96.18%. This substantial rise in F1 score underlines the model's ability to achieve a balanced trade-off between precision and recall, resulting in more accurate classifications.

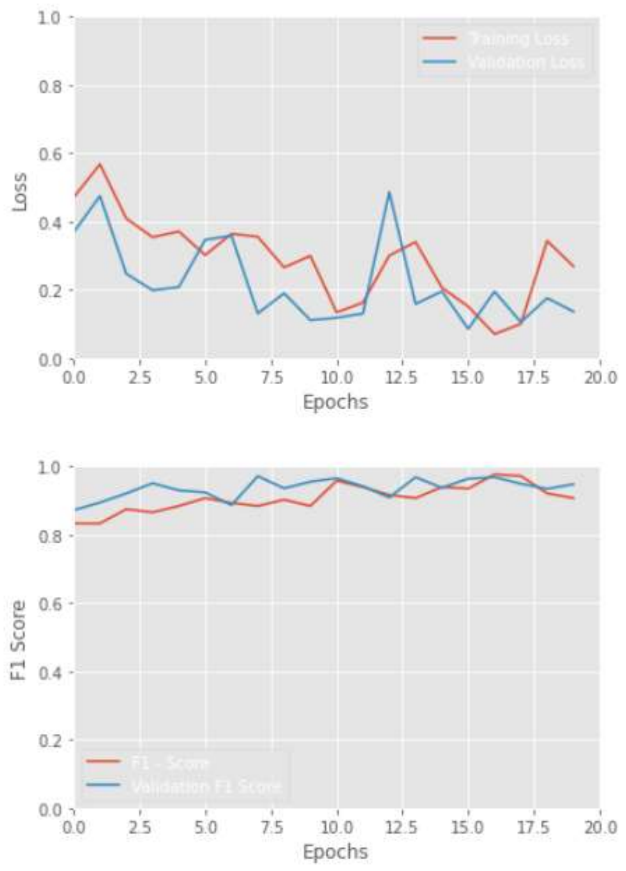

Fig 10: Plot history of Loss and F1-score over 20 epochs

Moreover, the model's loss function decreased from 338.86% to 9.26%, illustrating its capacity to minimize errors in predictions. Both accuracy and custom F1 scores on the validation data followed similar upward trends, indicating the model's proficiency in generalization and precision, ultimately achieving an impressive classification accuracy of

98.15%. These results highlight the model's exceptional ability to progressively refine its predictions, achieving a significant boost in classification accuracy.

VI. Conclusion

In summary, license plate detection and recognition is a multi-step process involving image preprocessing, contour extraction, character segmentation, and the crucial step of neural network modeling. A well-designed convolutional neural network (CNN) architecture, trained with extensive data and performance metrics like the F1 score, is key to the system's success.

The training and testing stages are pivotal, utilizing data augmentation and validation sets to improve the model's performance. The results highlight the system's success, with an F1 score increasing from 8.68% to 96.18% over 20 epochs, emphasizing its accurate classification abilities. With a validation accuracy of 98.15%, the system demonstrates strong generalization capabilities, making it valuable in real-world scenarios. This comprehensive process, from preprocessing to performance evaluation, ensures the robustness of the license plate detection system.

VII. Future Work

Future work in license plate detection and recognition could focus on enhancing the system's versatility. One avenue is extending the model to handle various types of license plates and font styles, accommodating different regions and countries. Additionally, implementing real-time processing capabilities for video streams or surveillance systems would be valuable. The integration of optical character recognition (OCR) technology could further enhance the extracted information's accuracy and utility. Lastly, research into reducing the model's computational complexity and memory requirements for deployment on edge devices or embedded systems would be beneficial. These directions can lead to a more adaptable and efficient license plate detection system for a broader range of applications.

References

[1] L. Zhang, P. Wang, H. Li, Z. Li, C. Shen, Y. Zhang, “A robust attentional framework for license plate recognition in the wild”, IEEE Transactions on Intelligent Transportation Systems, pp. 1-10, doi: 10.1109/TITS.2020.3000072, June 2020.

[2] J. Liang, G. Chen, Y. Wang, H. Qin, "EGSANet: edge-guided sparse attention network for improving license plate detection in the wild", Applied Intelligence, vol.52, no.4, pp.4458-4472, 2022.

[3] S. Rakhshani, E. Rashedi, H. Nezamabadi-pour, “Representation learning in a deep network for license plate recognition”, Multimedia tools and applications, 10.1007/s11042-019-08416-0, May 2020

[4] T. Vaiyapuri, S. Mohanty, M. Sivaram, I. Pustokhina, D. Pustokhin, K. Shankar, "Automatic vehicle license plate recognition using optimal deep learning model", Computers, Materials & Continua, no.62, nol.7, pp.1881- 1897, 2021

[5] T. Bjorklund, A. Fiandrotti, M. Annarumma, G. Francini, E. Magli, "Robust license plate recognition using neural networks trained on synthetic images", Pattern Recognition, vol.93, pp. 134-146, 2019

[6] G. Goncalves, M. Diniz, R. Laroca, D. Menotti, W. Schwartz, "Real-time automatic license plate recognition through deep multi-task networks", 31st IEEE SIBGRAPI Conference on Graphics, Patterns and Images, pp.110-117, 2018

[7] Kaggle Dataset for OCR Model Training: "ALPR Character Train" by FoolishBoi. Available at: https://www.kaggle.com/datasets/foolishboi/alpr-character-train/

[8] Kaggle Dataset for Car License Plate Detection: "Car Plate Detection" by AndrewMVD. Available at: https://www.kaggle.com/datasets/andrewmvd/car-plate-detection/