AMRITA VISHWA VIDYAPEETHAM

EOC -2 AND MFC-2

LICENSE PLATE RECOGNITION USING YOLO AND CNN

TEAM -MEMBERS

CH.VAISHNAVI KRISHNA-CB.SC.U4AIE24108

NETHRA.R-CB.SC.U4AIE24147

SAHAANA SHRI.S.K -CB.SC.U4AIE24149

CH.MOUNIKA-CB.SC.U4AIE24169

INTRODUCTION:

- *In this project, we built a system that can automatically detect and read vehicle license plates from images or videos. We used YOLO, a fast object detection algorithm, to find the license plate in an image.*

- *Then, we used CNN (Convolutional Neural Network) to recognize the characters on the plate. This kind of system is useful in places like toll booths, parking lots, and traffic monitoring.*

OBJECTIVES:

- *To detect vehicle license plates accurately using the YOLO object detection algorithm.*
- *To recognize characters on the license plate using a Convolutional Neural Network (CNN).*
- *To build a real-time, efficient system for automatic license plate recognition in traffic and security applications.*

LITERATURE REVIEW:

S.NO	Title	Author	year	Algorithms used	Key Contributions	Accuracy
1	Chinese License Plate Recognition Using a Convolutional Neural Network	Z. Zhao, Sh. Ma, W. Han, Y. Yang, X. Wang	2008	CNN	Early segmentation-free CNN for character recognition; demonstrated robustness across varied plate styles and lighting.	93.5%

3	Number Plate Recognition Based on Support Vector Machines	L. Zheng & X. He	2006	svm	Employed SVM for per-character classification, analyzed different kernels, and showed SVM outperforms inductive learning approaches.	96.34%
4	Segmentation-free Vehicle License Plate Recognition using ConvNet-RNN	T. K. Cheang, Y. S. Chong & Y. H. Tay	2017	ConvNet + RNN	Proposed an end-to-end model that combines CNN feature extraction with RNN sequence modeling to avoid explicit character segmentation.	96.57%

METHODOLOGY:

DETECTING BOUNDING BOXES USING YOLO :

1.YOLO Detection (Single Forward Pass):

- YOLO performs object detection in a single forward pass.
- It divides the image into a grid and simultaneously predicts bounding boxes, confidence and class probabilities.

2. Anchor Boxes:

- Internally, YOLO uses anchor boxes—predefined shapes that help predict bounding boxes of various aspect ratio.
- Each grid cell predicts multiple boxes using these anchors to better match real-world object shapes.

3.IoU (Intersection over Union):

- During training, YOLO uses IoU to compare predicted boxes with ground truth boxes.
- Boxes with IoU > 0.5 (usually) are considered valid detections.

4.NMS (Non-Maximum Suppression)

- YOLO applies NMS automatically to suppress overlapping boxes.
- Keeps only the box with the highest confidence score among overlapping boxes.

5.Extract Bounding Box Coordinates:

- Gets the top-left (x1, y1) and bottom-right (x2, y2) of the bounding box.
- This box surrounds the detected license plate in the image.

6.Crop the License Plate:

- Crops the region of interest from the image — the license plate only.
- Can be used later for CNN.

7.Output:

Preprocessing :

Before a CNN can recognize the characters on a license plate, the input image must be:
Free of noise and irrelevant details,
Standardized in size and intensity,
Cropped into individual, clean characters.

- Grayscale Conversion
- Binarization and Inversion
- Adaptive Thresholding
- Contour Detection and Filtering
- Character Cropping and Alignment
- Normalization and Channel Stacking

1. Grayscale Conversion

- The input image is converted from a 3-channel RGB color image to a single-channel grayscale image.The conversion reduces the computational load and focuses only on the shape and structure of the plate.

Code:

```
gray=cv2.cvtColor(plate_image,cv2.COLOR_BGR2GRAY)
```

2. Binarization and Inversion

- The grayscale image is converted into a binary image(black and white).
- The binary image is then inverted to ensure the characters appear white on a black background, which helps in contour detection.

Code: _,binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

- binary_inverted = 255 - binary

3.Adaptive Thresholding:

- Adaptive thresholding is applied to handle varying lighting conditions across the image. The grayscale image is blurred using cv2.GaussianBlur() to reduce noise, and then cv2.adaptiveThreshold() is used to apply different thresholds for local regions.
- It focuses on local pixel intensity variations, which is important when characters appear in shadows or under uneven lighting

4.Contour Detection and Filtering:

- External contours are detected from the thresholded image, which likely represent individual characters.
- These contours are filtered based on geometrical properties such as height, width, area, and aspect ratio to eliminate unwanted elements like noise or small objects.

Code:

- **contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)**

5. Character Cropping and Resizing

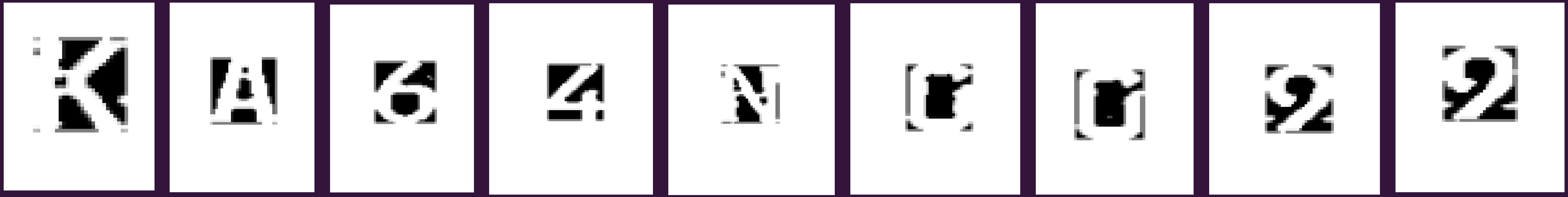
- The filtered character regions are cropped from the grayscale image and resized to a fixed size (50×50 pixels) to maintain uniformity for CNN input.
- **Code: char_resized = cv2.resize(char_img, (50, 50), interpolation=cv2.INTER_AREA)**

6.Normalization and Reshaping

- The pixel values are normalized to the range [0, 1] to improve model learning and performance.
- The grayscale image is then expanded to 3 channels (RGB) and reshaped to fit the CNN input structure.

Code:

```
norm_img = canvas.astype("float32") / 255.0
norm_img = np.stack((norm_img,) * 3, axis=-1)
```



CNN:

1. **Input Layer:**
    - Dimension: The network accepts color images of size 64x64 with three channels (RGB).
    - Purpose: This fixed input size ensures consistency across the dataset and serves as the foundation for all subsequent layers.
    - Input(shape=(64, 64, 3))

2.Convolutional Layers:

- **Activation Function :** ReLU (Rectified Linear Unit) makes negatives 0 and keeps positives, helping learn complex patterns by introducing non-linearity.
- **Padding:** To preserve dimensions, ensuring that important edge features are not lost.
- **Conv2D(32, (3, 3), activation='relu', padding='same')**

3.Pooling Layers:

- Purpose : reduce the dimensions of feature maps while preserving the most important features.
- **MaxPooling2D(2, 2)**

4.Flattening Stage:

- The Flattening layer converts the 3D feature maps into a 1D vector.
- **Flatten()**

5.Dense (Fully Connected) Layers:

- The Dense layer is a fully connected layer that learns high-level features, enabling complex decision-making for classification.
- **Dense(128, activation='relu')**

6.Dropout Regularization:

- The Dropout layer randomly deactivates a fraction of neurons during training to prevent overfitting.
- **Dropout(0.5)**

7.Output Layer:

- The Output layer uses a softmax activation function to produce a probability distribution over all character classes, enabling the model to predict the most likely character.
- **Dense(num_classes, activation='softmax')**

OUTPUT:


```
Final Detected Text: KA64N0099
```

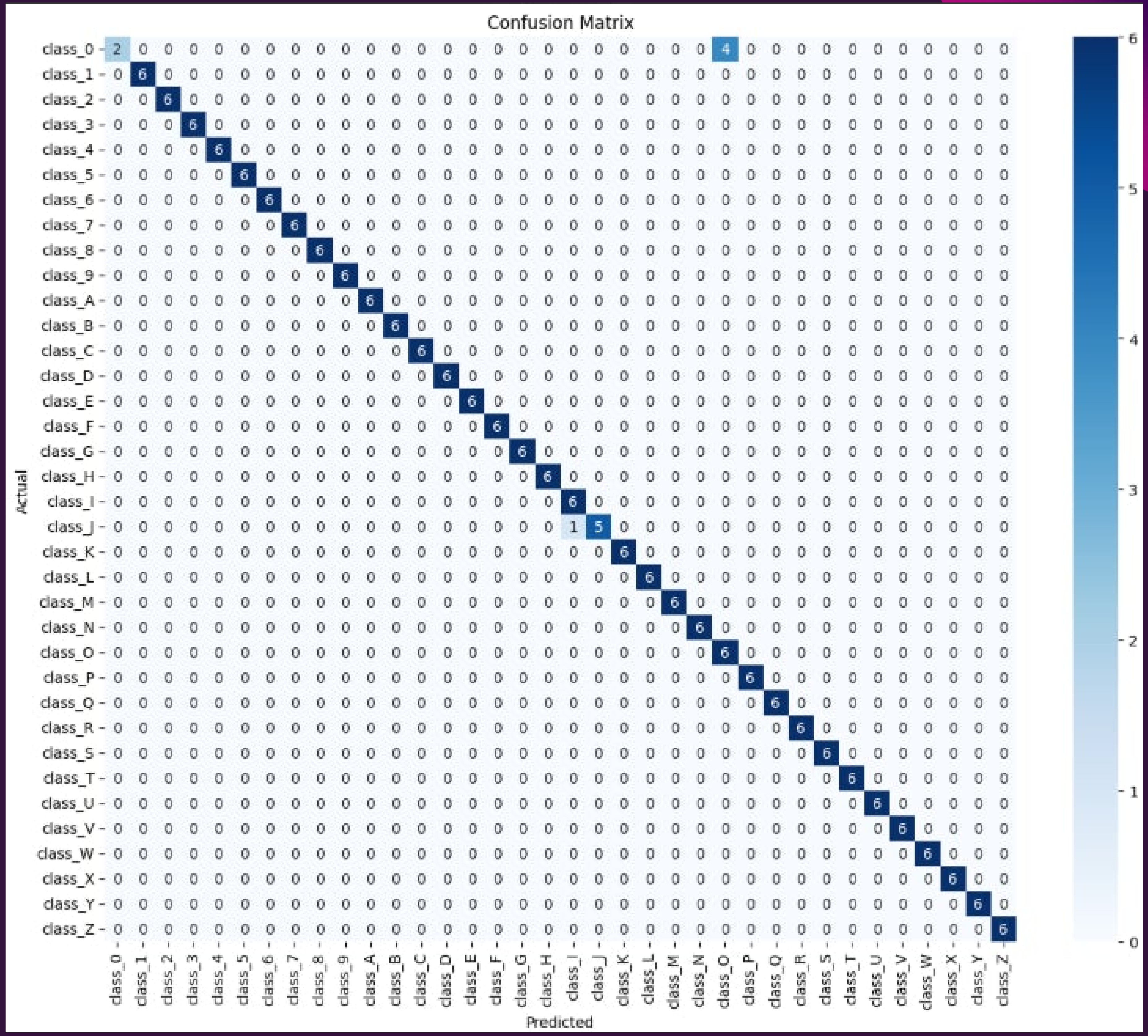
RESULTS & ANALYSIS:

Accuracy : 97.69%

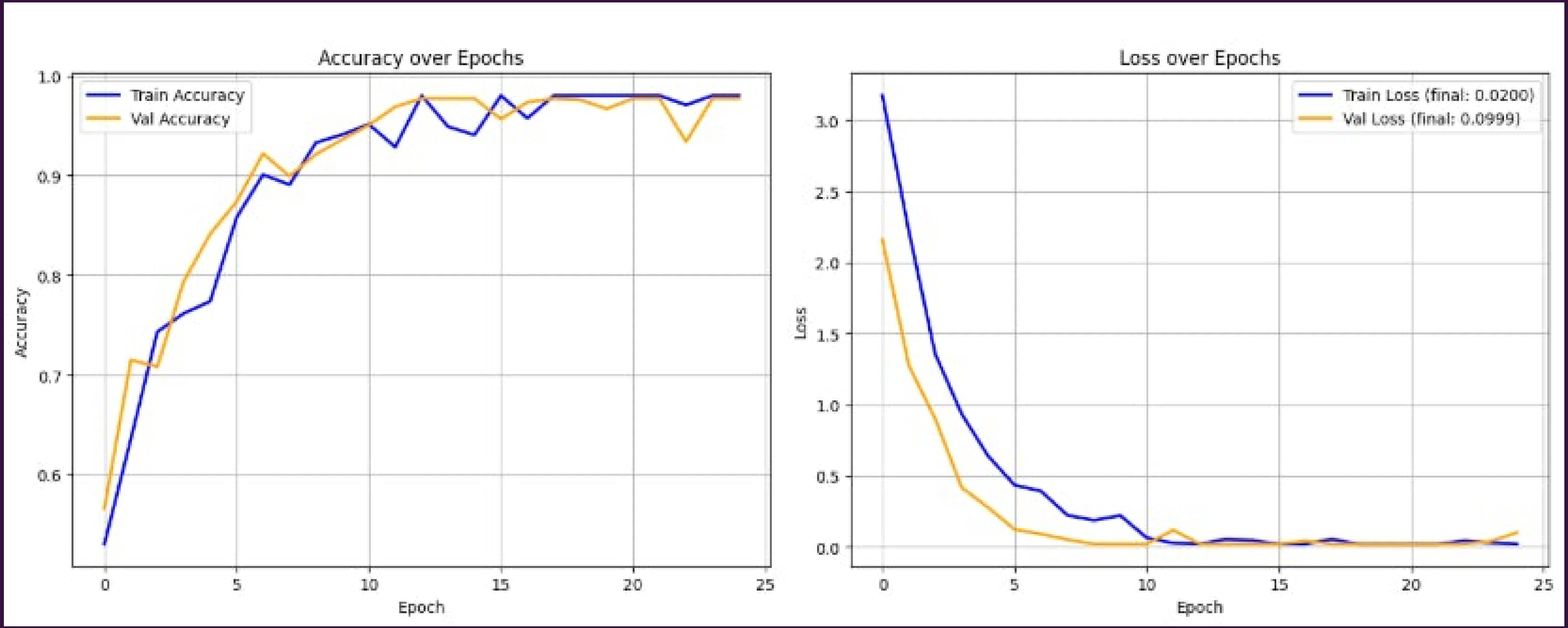
Precision : 98.49%

Recall : 97.69%

F1-Score : 97.45%



Model Accuracy and Model Loss Graph



FUTURE WORK:

1.Integration with Real-Time Video Streams:

- The system can be deployed with live CCTV or drone footage, enabling real-time vehicle tracking and license plate monitoring using continuous YOLO inference.

CONCLUSION:

- In this project, we successfully developed a robust License Plate Recognition System using a combination of YOLOv8 for license plate detection and a custom CNN model for character recognition.
- Through proper dataset preparation, model training, and integration, our system achieved an impressive 97.6% accuracy, demonstrating high efficiency and reliability in real-world scenarios.

THANK YOU