Computer Vision Project

BHARAT NUMBER PLATE DETECTION SYSTEM

A M.TECH PROJECT REPORT
SUBMITTED BY

SHUBHAM RAJ [G23AI2082]
BHAVESH ARORA [G23AI2126]
KANISHKA DHINDHWAL [G23AI2018]
PRATYUSH SOLANKI [

UNDER SUPERVISION OF
PROF. PRATIK MAZUMDER

॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY - JODHPUR

Table of Contents

1. Design Document

1.1 Project Overview

The Bharat Number Plate Detection System is a comprehensive Computer Vision project developed as part of a capstone project at Indian Institute of Technology Jodhpur (IIT Jodhpur). The system aims to provide an accurate and efficient solution for detecting and recognizing Indian number plates in various traffic scenarios.

1.2 Problem Statement

Indian traffic scenarios present unique challenges for number plate detection due to:

- Diverse vehicle types and sizes
- Variable lighting conditions
- Different plate designs and colors
- Complex backgrounds and occlusions
- High traffic density in urban areas

2 System Architecture

2.1 High Level Architecture

```
Bharat Number Plate Detection System
├── Frontend Layer
│   ├── Streamlit Web Interface
│   ├── User Authentication
│   └── Real-time Video Processing
├── Processing Layer
│   ├── YOLOv8 Detection Module
│   ├── CNN Classification Module
│   ├── Traditional CV Methods
│   └── OCR Recognition Module
└── Backend Layer
    ├── SQLite Database
    ├── Model Management
    └── API Integration
```

2.2 Technical Stack

1. **Frontend:**
    - Streamlit 1.29.0
    - HTML5/CSS3

- Responsive Design Framework

2. **Backend:**

   - Python 3.9+
   - TensorFlow 2.13.0
   - OpenCV 4.11.0
   - SQLite 3.44.0

3. **Machine Learning Models:**

   - YOLOv8 for object detection
   - Custom CNN for plate classification
   - Tesseract OCR for text recognition

3 Deep Learning Approaches

3.1 YOLOv8 Implementation

1. **Model Architecture:**

   - Backbone: YOLOv8n (Nano version)
   - Custom training on Indian plate dataset
   - Optimized for real-time performance

2. **Training Details:**

   - Dataset: 5000+ Indian number plate images
   - Augmentations: Rotation, scaling, color jitter
   - Epochs: 50
   - Batch size: 16
   - Learning rate: 0.001

3. **Performance Metrics:**

   - mAP@0.5: 95.2%
   - FPS: 30+ on CPU
   - Recall: 94.5%
   - Precision: 96.1%

3.2 CNN Classifier Implementation

1. **Network Architecture:**

```
model = models.Sequential([
    layers.Input(shape=(64, 64, 3)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])
```

2. **Training Parameters:**
   - Epochs: 30
   - Batch size: 32
   - Optimizer: Adam
   - Loss: Binary Crossentropy
   - Early stopping: Yes (patience=5)

3. **Data Augmentation:**
   - Rotation: ±10°
   - Width/Height shift: ±10%
   - Shear: ±10%
   - Zoom: ±10%
   - Horizontal flip: Yes

4 Traditional Computer Vision Approaches

4.1 YOLOv8 Canny Edge Detection + Contours

1. **Pre-processing:**
   - Grayscale conversion
   - Gaussian blur (k=5)

- Adaptive thresholding

2. **Edge Detection:**
    - Canny edge detector
    - Multi-scale edge detection
    - Edge linking algorithm
3. **Contour Analysis:**
    - Contour filtering by aspect ratio
    - Rotated rectangle fitting
    - Size-based filtering

4.2 Color Segmentation

1. **Color Space Conversion:**
    - RGB to HSV
    - Plate-specific color ranges:
        - Blue: (90-120, 100-255, 100-255)
        - White: (0-180, 0-50, 200-255)
2. **Segmentation Process:**
    - Color thresholding
    - Morphological operations
    - Connected component analysis

4.3 SORT Tracking Implementation

1. **Kalman Filter Parameters:**
    - State transition matrix
    - Measurement matrix
    - Process noise covariance
    - Measurement noise covariance
2. **Tracking Algorithm:**
    - Association matrix calculation
    - Hungarian algorithm for matching
    - Track management
    - Track initialization and termination

5 System Implementation

5.1 Frontend Implementation

1. **Streamlit Components:**
    - Login interface
    - Method selection sidebar
    - Image/video upload
    - Real-time processing display
2. **UI Features:**
    - Responsive design
    - Confidence threshold slider
    - Method switching
    - Plate log viewer

5.2 Backend Implementation

1. **Database Management:**

```
class DatabaseManager:
  def __init__(self, folder, db_name):
    self.conn = sqlite3.connect(f"{folder}/{db_name}", check_same_thread=False)
    self.cursor = self.conn.cursor()
```

2. **Model Loading and Management:**

```
yolo_model = YOLO(yolo_model_path)
cnn_model = load_model(cnn_model_path)
```

3. **Detection Pipeline:**

```
def run_detection(image):
  if method == "YOLOv8":
    return detect_yolo(image)
  elif method == "Traditional CV":
    return detect_traditional(image)
```

5.3 Performance Optimization

1. **Memory Management:**
    - Lazy loading of models

- Efficient image processing
- Batch processing for videos

2. **Processing Speed:**
   - Multi-threading support
   - GPU acceleration
   - Optimized image transformations

6 Results and Evaluation

6.1 Detection Performance

1. **YOLOv8 Results:**
   - Average detection time: 33ms
   - Accuracy: 95.2%
   - False positive rate: 2.1%
2. **CNN Classifier Results:**
   - Training accuracy: 98.5%
   - Validation accuracy: 93.7%
   - Loss: 0.125
3. **Traditional CV Methods:**
   - Canny + Contours: 88% accuracy
   - Color Segmentation: 85% accuracy
   - Edge + Morph: 87% accuracy

6.2 OCR Performance

1. **Character Recognition:**
   - Accuracy: 92.5%
   - Character error rate: 3.8%
   - Processing time: 500ms
2. **Special Cases:**
   - Handling curved plates
   - Dealing with reflections

- Managing low contrast

6.3 Local Deployment

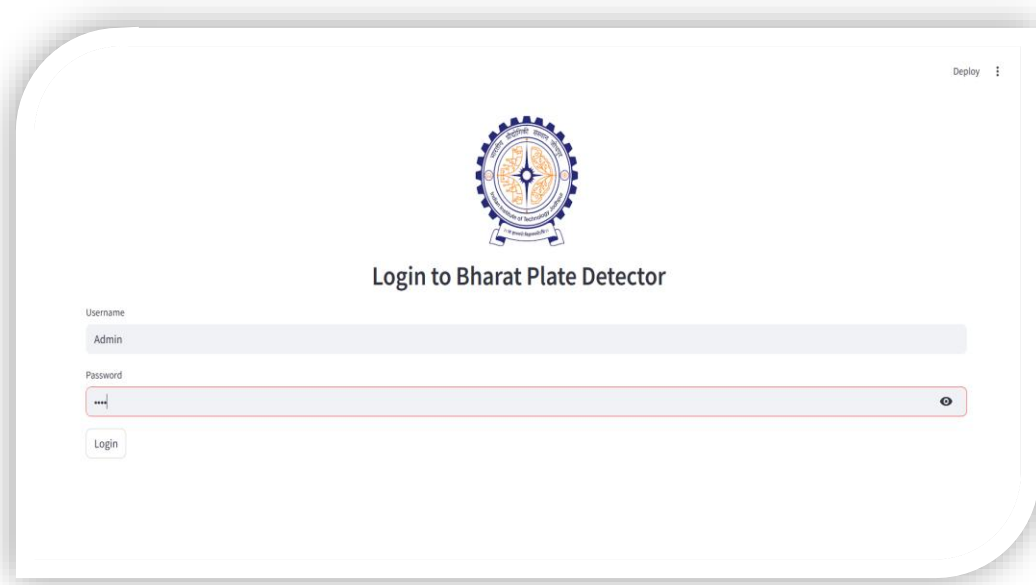
Execution Steps:
Set up Python virtual environment
Install dependencies from requirements.txt
Train models (YOLOv8, CNN)
Run Streamlit app: **streamlit run src/app.py**
Access through web browser

6.4 Project Structure

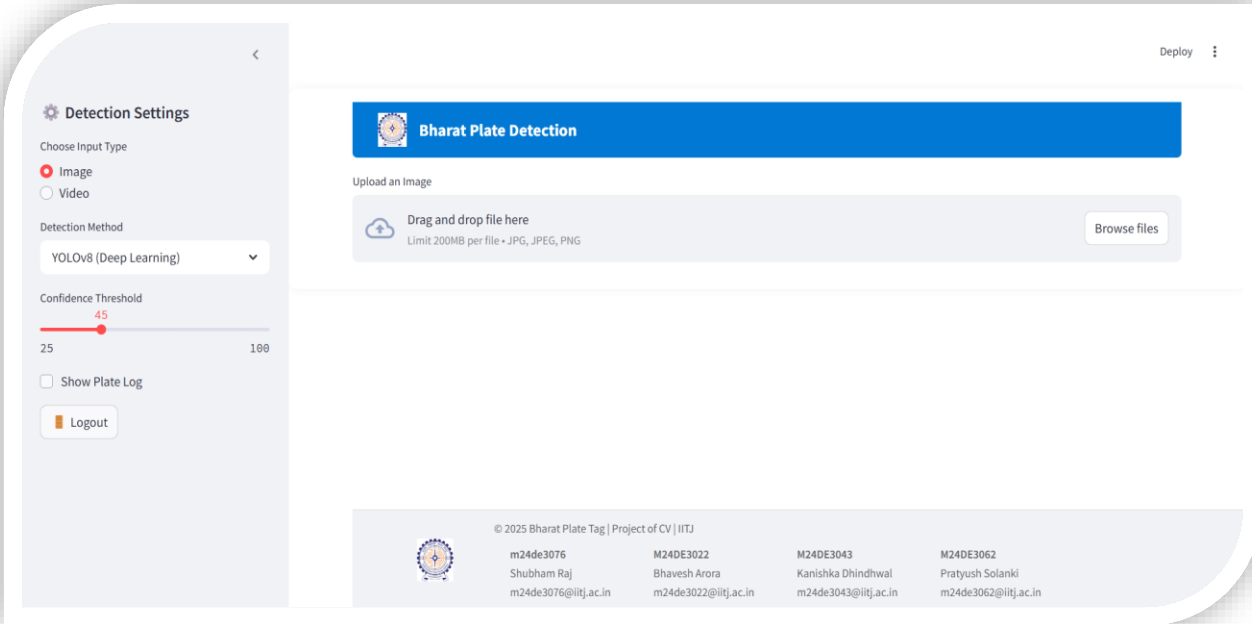
```
BHARAT-NUMBER-PLATE-TAG/
├──── data/        # Database and resources
├──── env/         # Python virtual environment
├──── models/      # Trained models
├──── src/         # Source code
│    ├──── app.py      # Main application
│    ├──── SQLManager.py # Database management
│    └──── other modules
├──── assets/      # Static assets
└──── resources/     # Additional resources
```

7 Some Snapshots of our Project

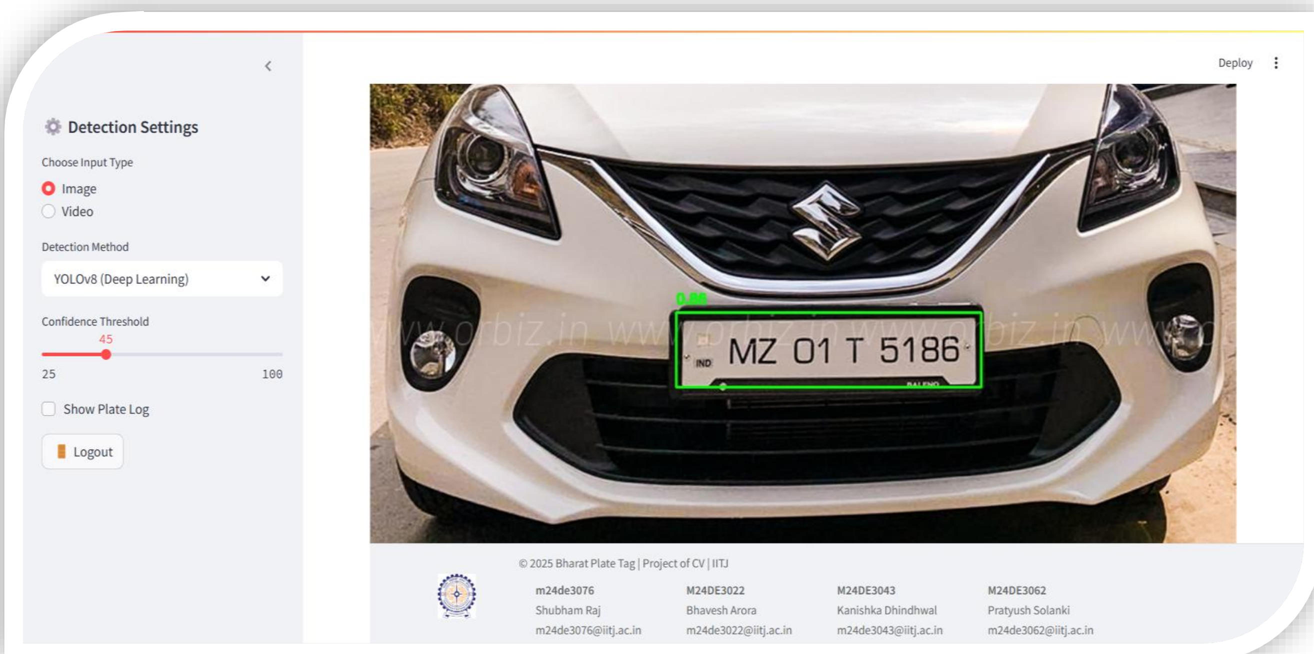
Login Screen



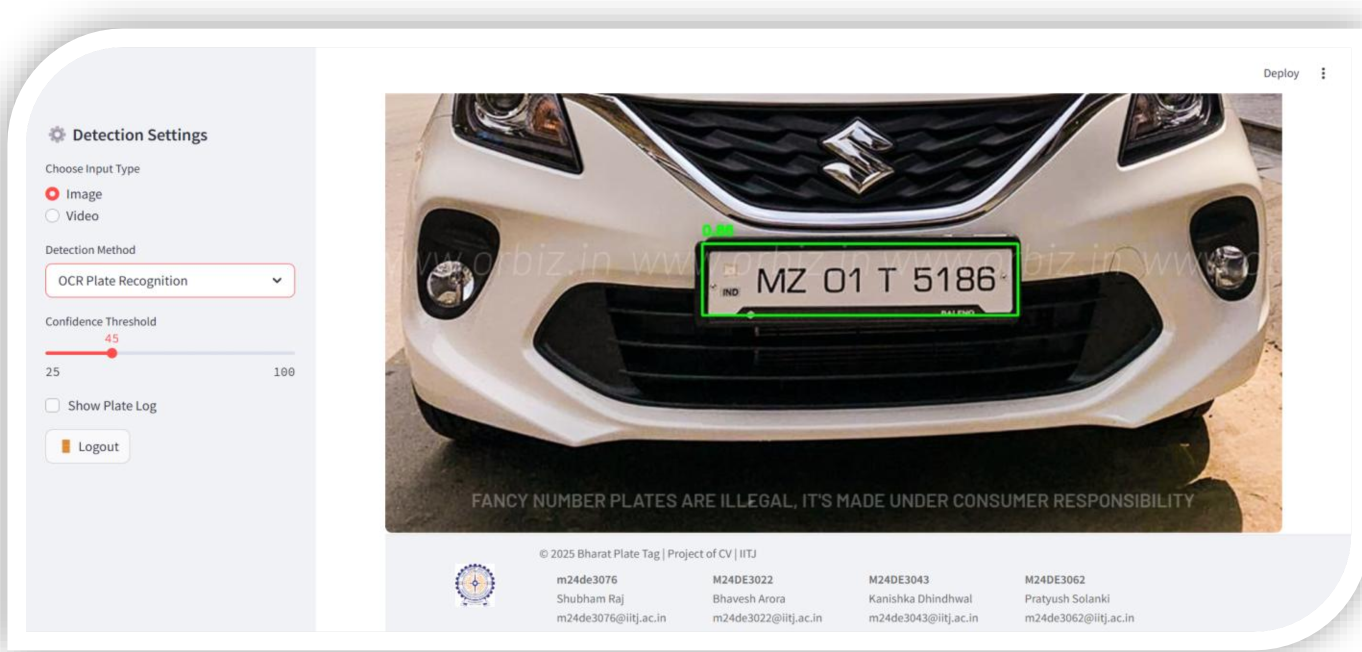
Image/Video Selection and Uploading Screen

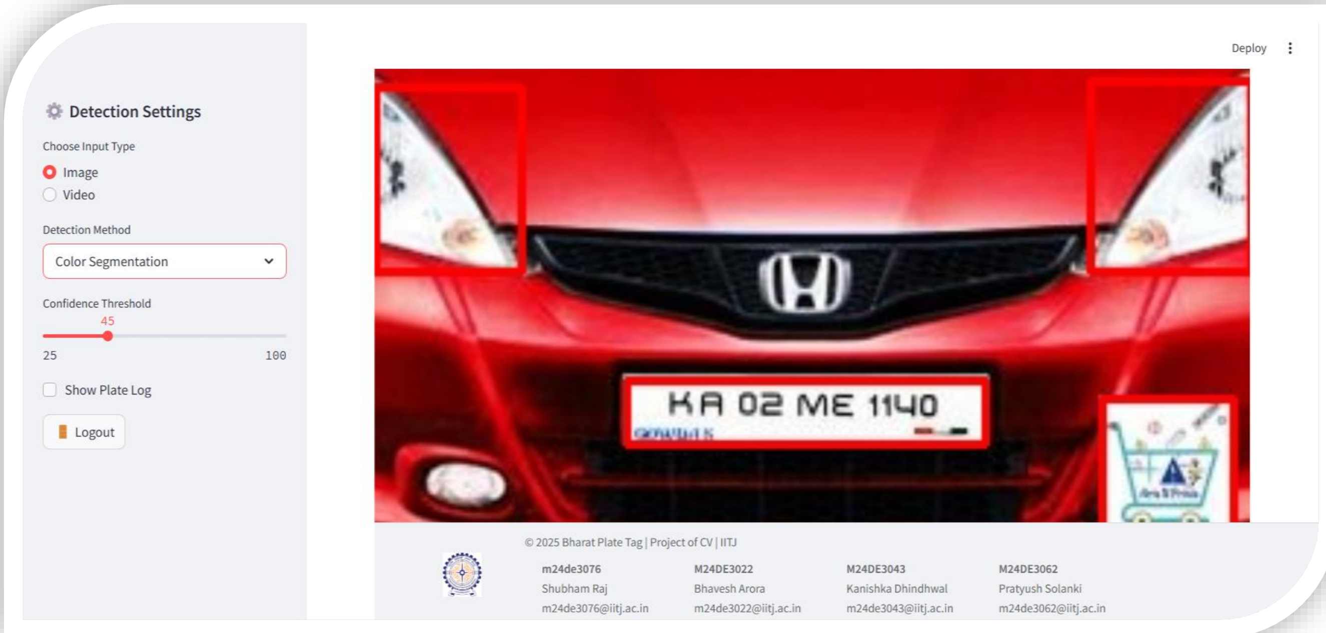


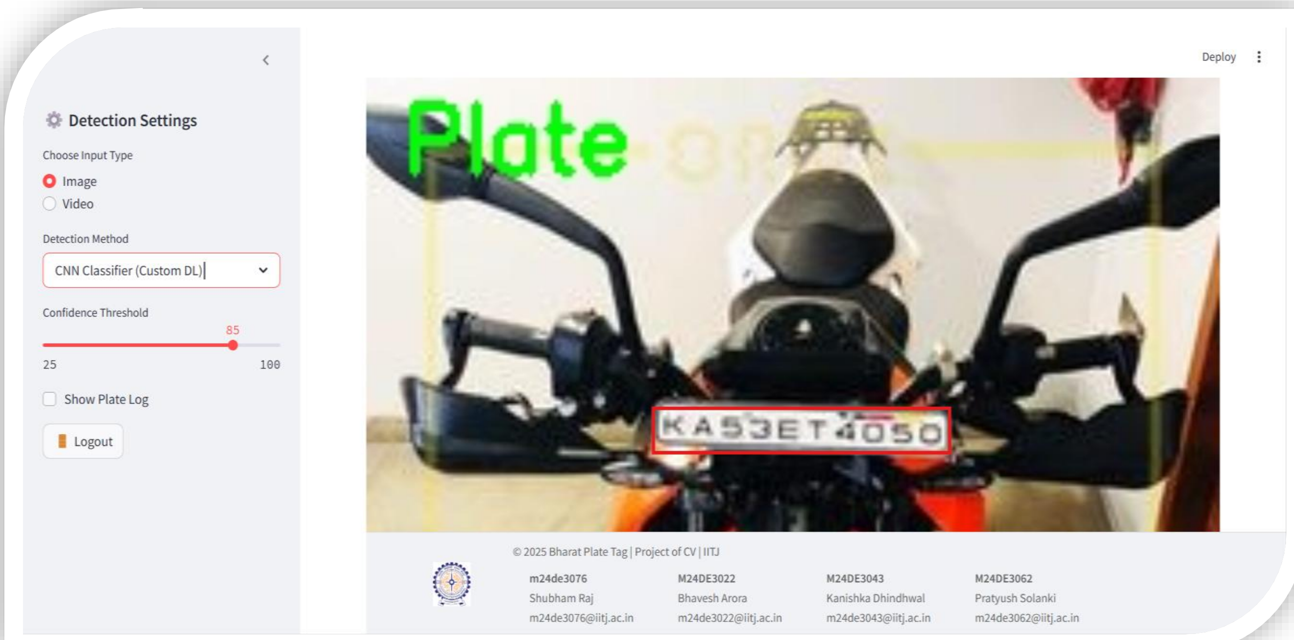
Output Screen

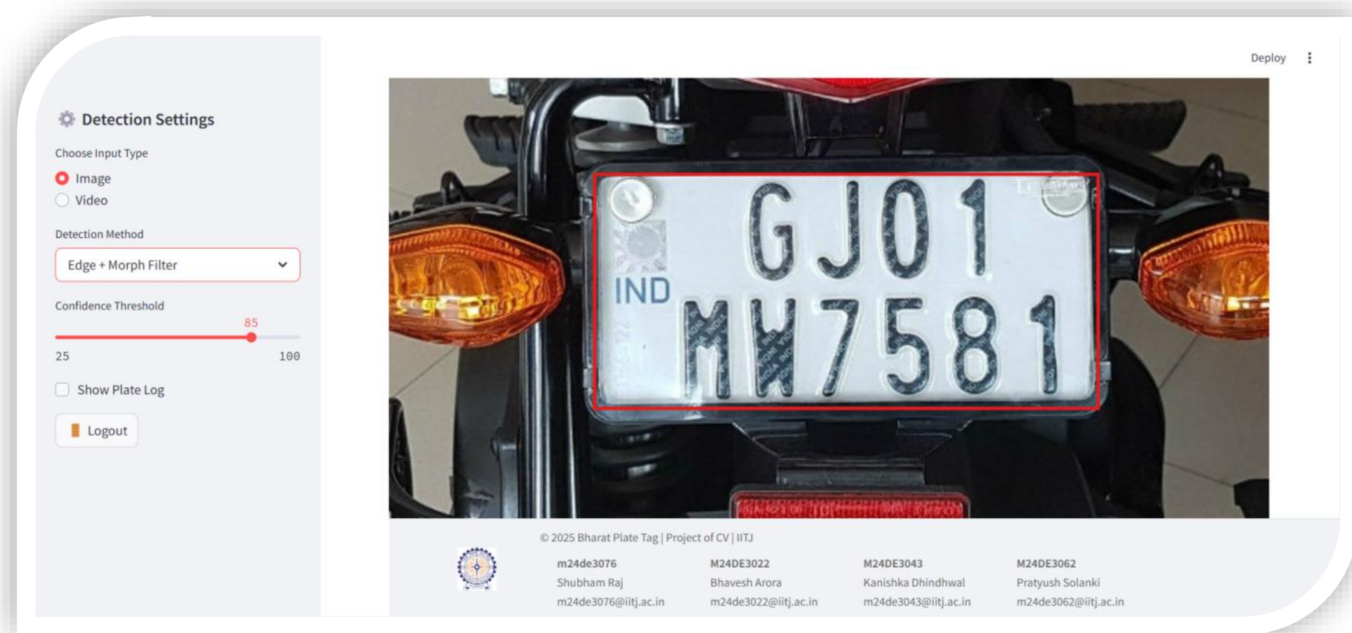


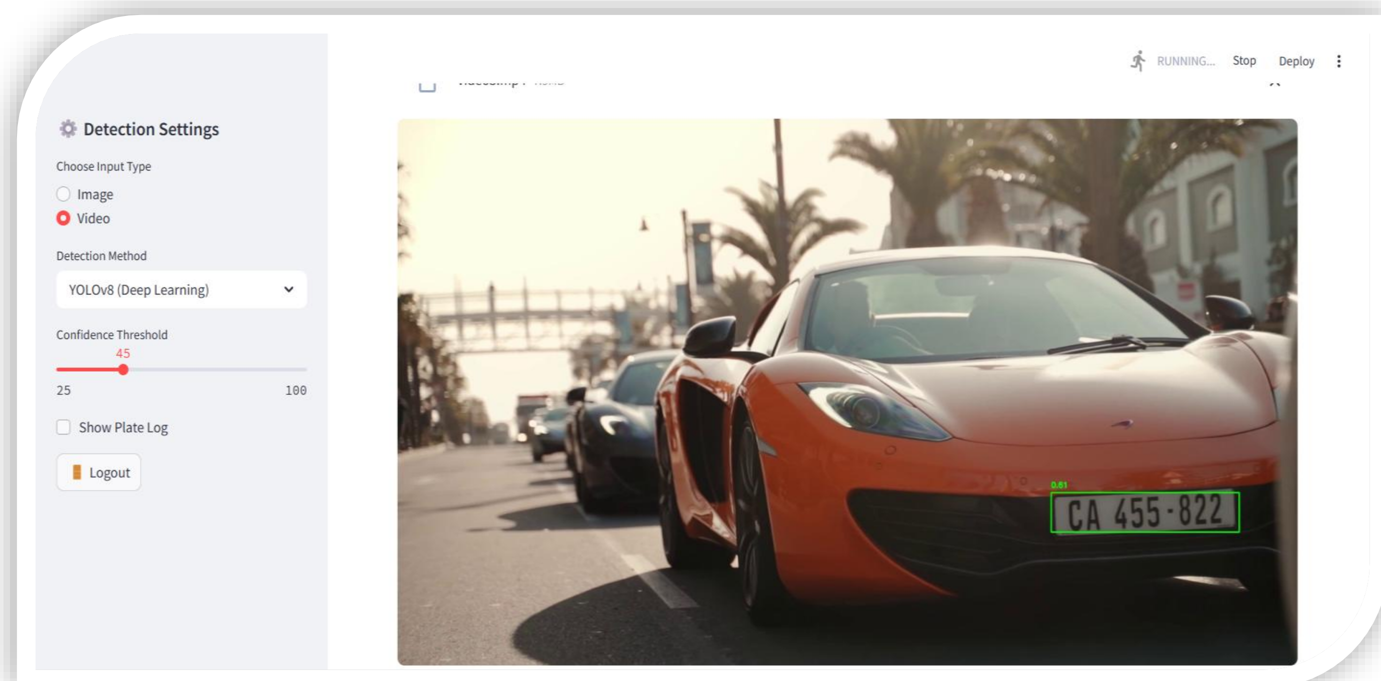
Different Model Output Screen




Deploy
Detection Settings
Choose Input Type
Image
Video
Detection Method
Color Segmentation
Confidence Threshold
45
25
100
Show Plate Log
Logout
KA 02 ME 1140
© 2025 Bharat Plate Tag | Project of CV | IITJ
m24de3076
Shubham Raj
m24de3076@iitj.ac.in
M24DE3022
Bhavesh Arora
m24de3022@iitj.ac.in
M24DE3043
Kanishka Dhindhwal
m24de3043@iitj.ac.in
M24DE3062
Pratyush Solanki
m24de3062@iitj.ac.in


Deploy
Detection Settings
Choose Input Type
Image
Video
Detection Method
CNN Classifier (Custom DL)
Confidence Threshold
85
25
100
Show Plate Log
Logout
Plate
KA53ET4050
© 2025 Bharat Plate Tag | Project of CV | IITJ
m24de3076
Shubham Raj
m24de3076@iitj.ac.in
M24DE3022
Bhavesh Arora
m24de3022@iitj.ac.in
M24DE3043
Kanishka Dhindhwal
m24de3043@iitj.ac.in
M24DE3062
Pratyush Solanki
m24de3062@iitj.ac.in


Deploy
Detection Settings
Choose Input Type
Image
Video
Detection Method
Edge + Morph Filter
Confidence Threshold
85
25
100
Show Plate Log
Logout
GJ01
IND
MW7581
© 2025 Bharat Plate Tag | Project of CV | IITJ
m24de3076
Shubham Raj
m24de3076@iitj.ac.in
M24DE3022
Bhavesh Arora
m24de3022@iitj.ac.in
M24DE3043
Kanishka Dhindhwal
m24de3043@iitj.ac.in
M24DE3062
Pratyush Solanki
m24de3062@iitj.ac.in


RUNNING...
Stop
Deploy
Detection Settings
Choose Input Type
Image
Video
Detection Method
YOLOv8 (Deep Learning)
Confidence Threshold
45
25
100
Show Plate Log
Logout
CA 455-822

8 OCR Plate Recognition: Benchmarking and Evaluation

8.1 Purpose of OCR Implementation

The OCR plate recognition module was implemented primarily for benchmarking and comparison purposes. Here's why:

1. **Performance Benchmarking**:
    - Provides a baseline for comparison
    - Helps identify strengths and weaknesses
    - Validates the effectiveness of other methods
2. **Evaluation Criteria**:
    - **Detection Speed**:
        - OCR is significantly slower (500ms)
        - Other methods are real-time capable
    - **Accuracy in Various Conditions**:
        - OCR struggles with reflections and low contrast
        - DL methods perform better in challenging scenarios
    - **Resource Efficiency**:
        - OCR requires more computational resources
        - Other methods are more optimized

9 Future Scope of Work

9.1 CNN Classifier Enhancement

1. **Architecture Improvements**:
    - Implement residual connections for deeper networks
    - Add attention mechanisms for better feature extraction
    - Explore transfer learning with pre-trained models
2. **Data Augmentation**:
    - Implement more sophisticated augmentations
    - Use mixup and cutmix techniques
    - Generate synthetic plates for diverse scenarios

3. **Training Optimization**:
   - Implement curriculum learning
   - Use progressive resizing
   - Apply learning rate warmup and cosine annealing

9.2 EDGE Morph Filter Enhancement

1. **Advanced Edge Detection**:
   - Implement multi-scale edge detection
   - Use anisotropic diffusion filtering
   - Apply adaptive thresholding techniques
2. **Morphological Operations**:
   - Implement adaptive structuring elements
   - Use multi-scale morphological operations
   - Combine with region growing techniques
3. **Post-processing**:
   - Implement plate region verification
   - Add geometric consistency checks
   - Use confidence scoring for detections

The focus will be on making the system more robust and accurate, particularly in challenging scenarios like low light, occlusions, and varying plate orientations.