### **Number Plate Detection**

Number Plate Detection is a computer vision technique that finds and reads vehicle license plates from images or video. It uses tools like image processing and Optical Character Recognition (OCR) to detect the plate and extract the text on it. This is helpful for traffic monitoring, toll collection, parking systems, and law enforcement.

### **LLM (Large Language Model):**

A Large Language Model (LLM) is a type of artificial intelligence trained on a vast amount of text. It can understand and generate human-like language. LLMs are used in chatbots, search engines, and assistants to respond smartly to questions and conversations.

### **How LLMs Are Used in Number Plate Detection:**

LLMs are not used for detecting plates directly, but they can help in:

1. **Text Correction:** Fixing errors in plate text that OCR might misread.
2. **Pattern Matching:** Verifying that the plate format is valid (like two letters followed by numbers).
3. **Context Awareness:** Understanding which text is likely to be the plate, especially when OCR reads extra surrounding text.
4. **Integration:** Working alongside OCR systems to make sense of text in traffic records or police databases.

### **Current Usage of Number Plate Detection:**

Where number plate detection is used:

1. **Traffic Monitoring:** To identify vehicles breaking traffic rules.
2. **Toll Systems:** Automatically reading plates for payment at toll booths.
3. **Parking Management:** Checking plate numbers for entry and exit in parking areas.
4. **Security:** Tracking stolen vehicles or monitoring restricted zones.
5. **Law Enforcement:** Assisting police in locating vehicles of interest.
6. **Smart Cities:** Integrating with surveillance systems for public safety.
7. **Border Control:** Identifying vehicles crossing international borders.

### **Future Evolution: Number Plate Detection**

1. **Higher Accuracy:** Better detection even in poor lighting, weather, or blurry images.
2. **Real-Time Performance:** Faster recognition in live video for instant traffic monitoring.
3. **Multimodal Verification:** Cross-checking with GPS, vehicle color, and make for higher reliability.
4. **Wider Integration:** Built into smart traffic lights, toll booths, and surveillance systems.
5. **AI-Based Prediction:** Detect suspicious patterns or stolen vehicles automatically.
6. **Privacy & Regulation:** Focus on protecting vehicle owner data and usage laws.
7. **Global Format Handling:** Recognize plates from different countries with diverse formats and fonts.

### **Problem Statement**

**Main Objective:**

The goal of this program is to:

* Take an image of a vehicle,
* Detect and locate the number plate,
* Read and extract the text from the plate using OCR.

**How it works (Simple Steps):**

1. Loads an image from the dataset.
2. Uses image processing to find the number plate area.
3. Applies OCR (EasyOCR) to extract text.
4. Draws a box around the plate and shows detected number.

### **Existing System Limitations**

1. **Low Accuracy:** May miss plates in poor quality or angled images.
2. **Limited Scope:** Works only on clean, centered plates.
3. **No Real-Time Use:** Processes images one-by-one, not live video.
4. **Font Issues:** Struggles with fancy or distorted plate fonts.
5. **No Context Use:** Can't verify if the detected text matches real formats.
6. **Doesn’t Handle Multiple Plates:** Detects only one or misses some in complex images.
7. **Language Limitation:** Works only with standard English-alphabet plates.

### **Proposed System**

1. Load image input (JPEG/PNG) from dataset.
2. Detect the plate area using image filters or OCR regions.
3. Use EasyOCR to read text from the detected plate region.
4. Highlight detected plates on image with bounding boxes.
5. Display plate number(s) along with the original image.

**Why it's better:**

* Reads multiple plate formats
* Uses real OCR, not random guesses
* Can be extended for live video or video frame processing
* Works with more fonts, sizes, and conditions

### **Pipeline of the Project**

**Dataflow for Number Plate Detection:**

1. **Data Collection**  
   └──> Load images of vehicles from dataset
2. **Preprocessing**  
   └──> Convert to grayscale  
   └──> Apply filters for better plate visibility
3. **Text Detection**  
   └──> Use EasyOCR to detect text regions in the image
4. **Text Extraction**  
   └──> Extract actual plate numbers from those regions
5. **Output Visualization**  
   └──> Show image with detected plate text and highlight boxes

**CODING:**

pip install openai-whisper librosa scikit-learn

# Mount Google Drive

from google.colab import drive

drive.mount('/content/drive')

# Import libraries

import os

import cv2

import easyocr

import matplotlib.pyplot as plt

from google.colab.patches import cv2\_imshow

# Initialize EasyOCR reader

reader = easyocr.Reader(['en'])

# Path to your dataset folder in Google Drive

dataset\_path = '/content/drive/MyDrive/Indian\_Number\_Plates/Sample\_Images'

# Function to detect and highlight number plate text

def detect\_number\_plate(image\_path):

    img = cv2.imread(image\_path)

    result = reader.readtext(img)

    number\_plate\_texts = []

    for detection in result:

        bbox, text, \_ = detection

        number\_plate\_texts.append(text)

        # Draw rectangle around detected text

        top\_left = tuple(map(int, bbox[0]))

        bottom\_right = tuple(map(int, bbox[2]))

        cv2.rectangle(img, top\_left, bottom\_right, (0, 255, 0), 2)

        cv2.putText(img, text, top\_left, cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (255, 0, 0), 2)

    return img, number\_plate\_texts

# Function to analyze all images in the dataset folder

def analyze\_dataset(dataset\_path):

    detected\_texts = {}

    for image\_filename in os.listdir(dataset\_path):

        if image\_filename.lower().endswith(('.png', '.jpg', '.jpeg')):

            image\_path = os.path.join(dataset\_path, image\_filename)

            print(f"\nProcessing {image\_filename}...")

            result\_img, plate\_text = detect\_number\_plate(image\_path)

            detected\_texts[image\_filename] = plate\_text

            print(f"Detected Text for {image\_filename}: {plate\_text}")

            cv2\_imshow(result\_img)

    return detected\_texts

# Run the analysis

detected\_texts = analyze\_dataset(dataset\_path)

# Print all detected texts

print("\nAll Detected Texts:")

for filename, texts in detected\_texts.items():

    print(f"{filename}: {texts}")

### **Libraries Used:**

1. **openai-whisper**
   * Transcribes speech from audio.
   * Detects spoken language.
2. **librosa**
   * Extracts audio features like **MFCC** and **pitch**.
3. **numpy**
   * Performs numerical operations and feature combinations.
4. **scikit-learn**
   * LabelEncoder: Converts emotion labels into numbers.
   * SVC: Classifies emotions using Support Vector Classifier.
5. **moviepy**
   * Converts video (MP4) to audio (WAV) for processing.

### **Architecture Used:**

* **Whisper AI**:
  + Pretrained model for speech recognition.
  + Converts audio to text + identifies language.
* **SVM Classifier**:
  + Classifies emotions from extracted audio features.
* **Not a Traditional LLM**, but Whisper works as a transcription tool that feeds into an emotion classification pipeline.

### **Dataset Used:**

* **Simulated Dataset**:
  + 13 **MFCC features** + 13 **pitch features** per sample.
  + Randomly assigned emotion labels.
  + Emotion classes: happy, sad, angry, neutral.

### **Limitations:**

1. **Simulated Data** – No real emotion samples used.
2. **Limited Accuracy** – Not robust for real-world data.
3. **Basic Emotion Classes** – Only detects 4 emotions.
4. **No Text Analysis** – Ignores meaning in transcribed speech.
5. **Simple Features Only** – Lacks advanced prosody or semantic analysis.
6. **No Evaluation** – Accuracy/metrics not tested.
7. **Offline Only** – No real-time/live audio support.

### **Future Enhancements:**

1. **Use Real Emotional Speech Datasets**.
2. **Expand Emotion Categories** – Include fear, surprise, boredom, etc.
3. **Enable Real-Time Detection** – Via live microphone input.
4. **Add Richer Audio Features** – Intonation, energy, tempo, etc.
5. **Text-Based Emotion Detection** – Use LLMs like GPT for context.
6. **Multimodal Analysis** – Combine voice, text, facial expression.
7. **User Interface** – Build a web/app interface for users.
8. **Personalized Emotion Models** – Adapt to each user’s speech patterns.
9. **Multilingual Support** – Recognize emotions across various languages.
10. **Deploy as Cloud API/Service** – Allow global access.

### **Conclusion:**

This project shows a **basic audio-based emotion detection system**, combining Whisper AI for transcription and an SVM classifier trained on simulated features. While it's a **good prototype**, its reliance on fake data and limited scope restricts its real-world usability. **With real datasets, deeper feature extraction, and multimodal inputs**, the system could evolve into a powerful, real-time emotional intelligence tool for diverse applications.