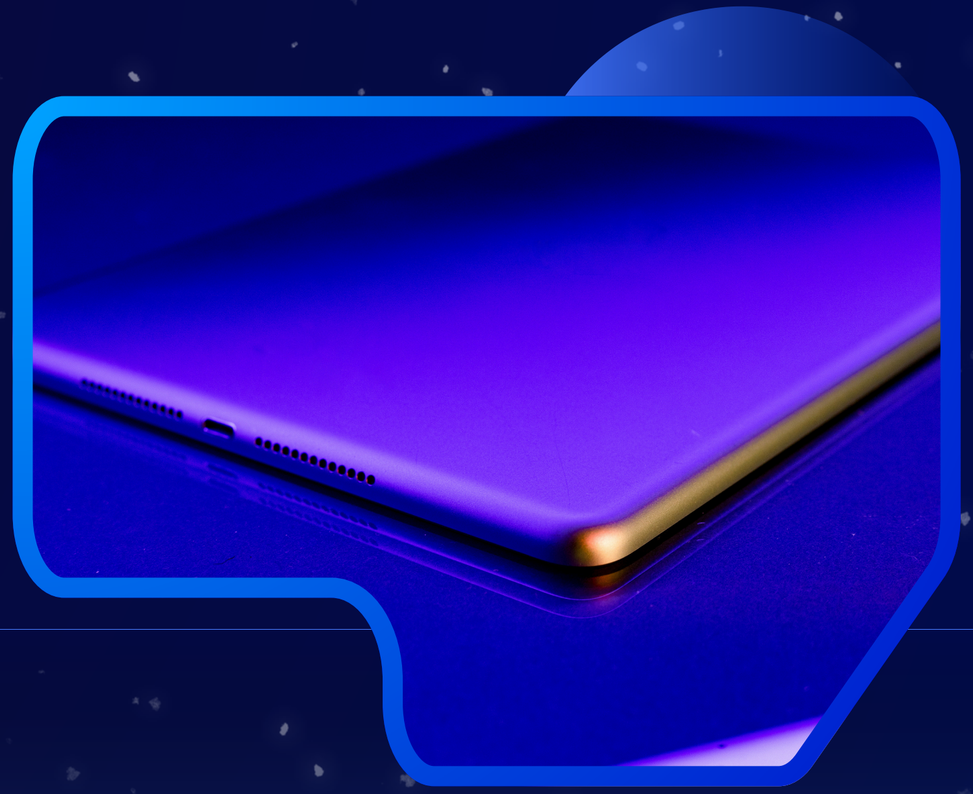
VISION-BASED VEHICLE ENTRY-EXIT TRACKING

Introduction



Traditional vehicle entry systems face significant bottlenecks: manual verification is labor-intensive and prone to error, while RFID solutions struggle with adoption barriers due to user costs. This project leverages rapid advancements in Deep Learning. To bridge this gap, offering a non-intrusive, automated alternative that utilizes existing camera infrastructure to deliver high-speed vehicle identification without requiring any additional hardware from the driver.

Dataset

Indonesian License Plate Detection

Indonesian License Plate Detection

Source : Roboflow Universe

IMG_20230408_133259_flipped.jpg | IMG_20230408_133132_flipped.jpg | IMG_20230408_133220_flipped.jpg

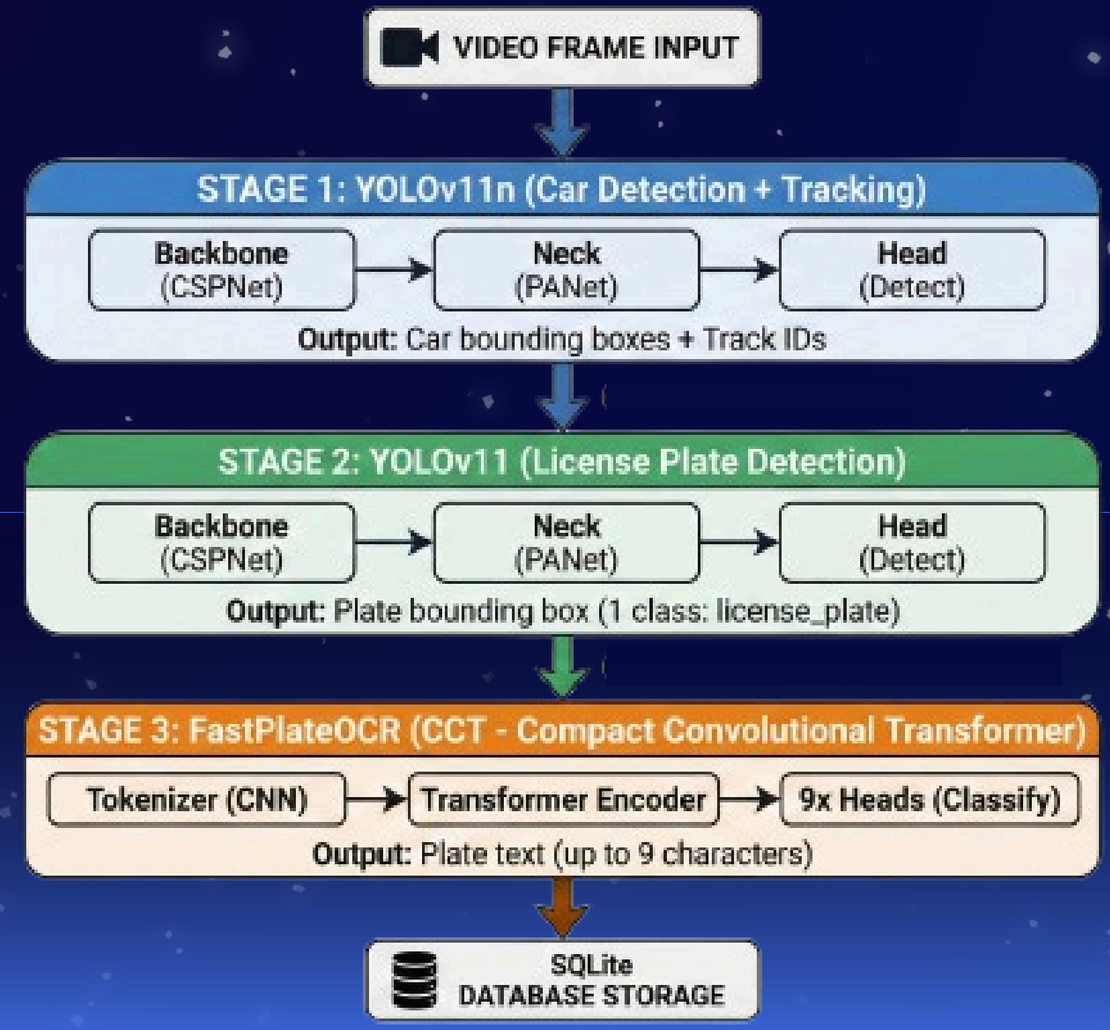
Preprocessing

- Images resized to 640x640 pixels (Stretch) to match model input.

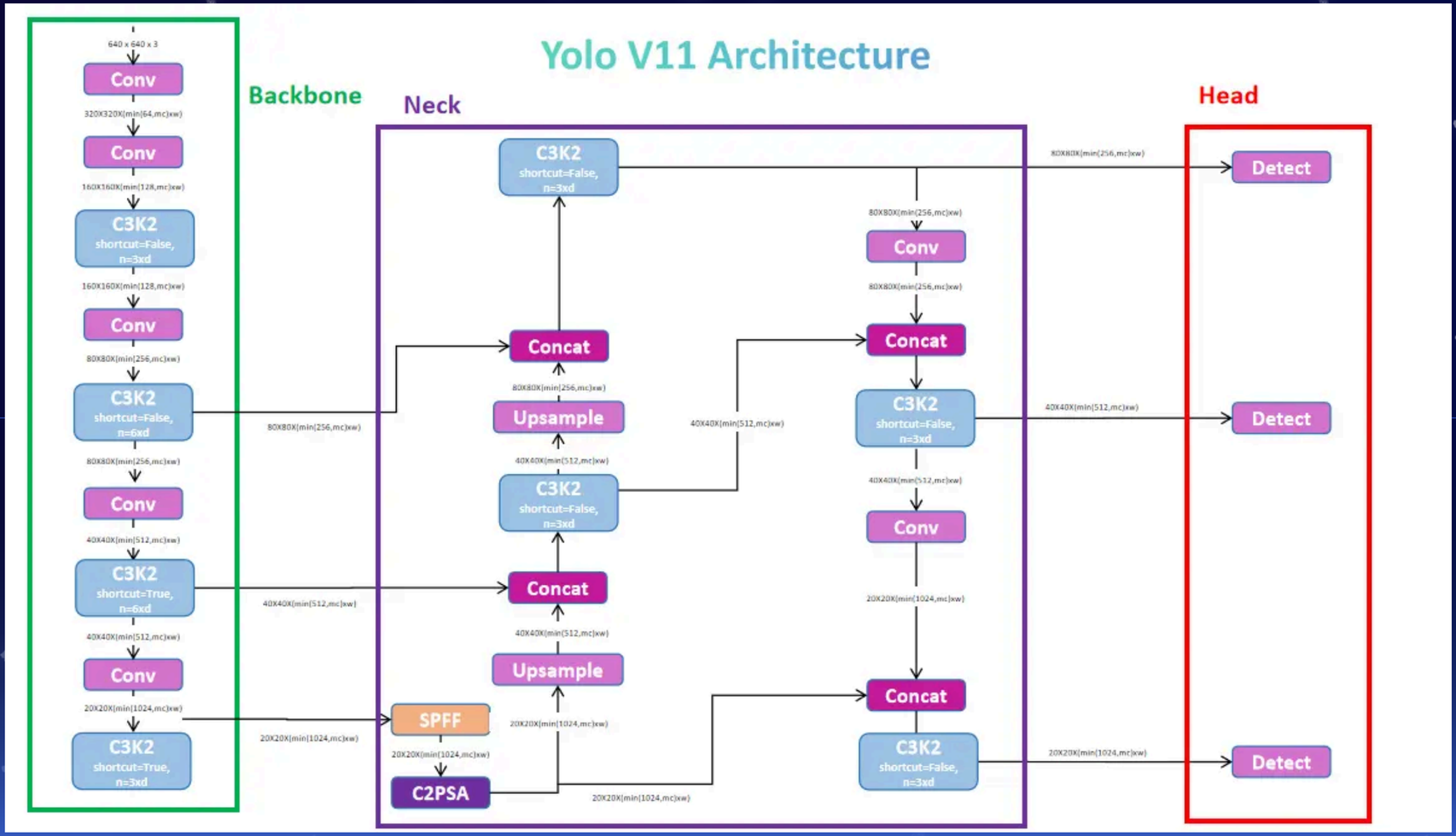
Augmentation Strategy

- Rotation: ±15°
- Shear: ±10° (Horizontal & Vertical)
- Brightness: ±20%
- Blur: Up to 0.8px
- Noise: Up to 1% of pixels

Model


VIDEO FRAME INPUT
STAGE 1: YOLOv11n (Car Detection + Tracking)
Backbone (CSPNet)
Neck (PANet)
Head (Detect)
Output: Car bounding boxes + Track IDs
STAGE 2: YOLOv11 (License Plate Detection)
Backbone (CSPNet)
Neck (PANet)
Head (Detect)
Output: Plate bounding box (1 class: license_plate)
STAGE 3: FastPlateOCR (CCT - Compact Convolutional Transformer)
Tokenizer (CNN)
Transformer Encoder
9x Heads (Classify)
Output: Plate text (up to 9 characters)
SQLite DATABASE STORAGE

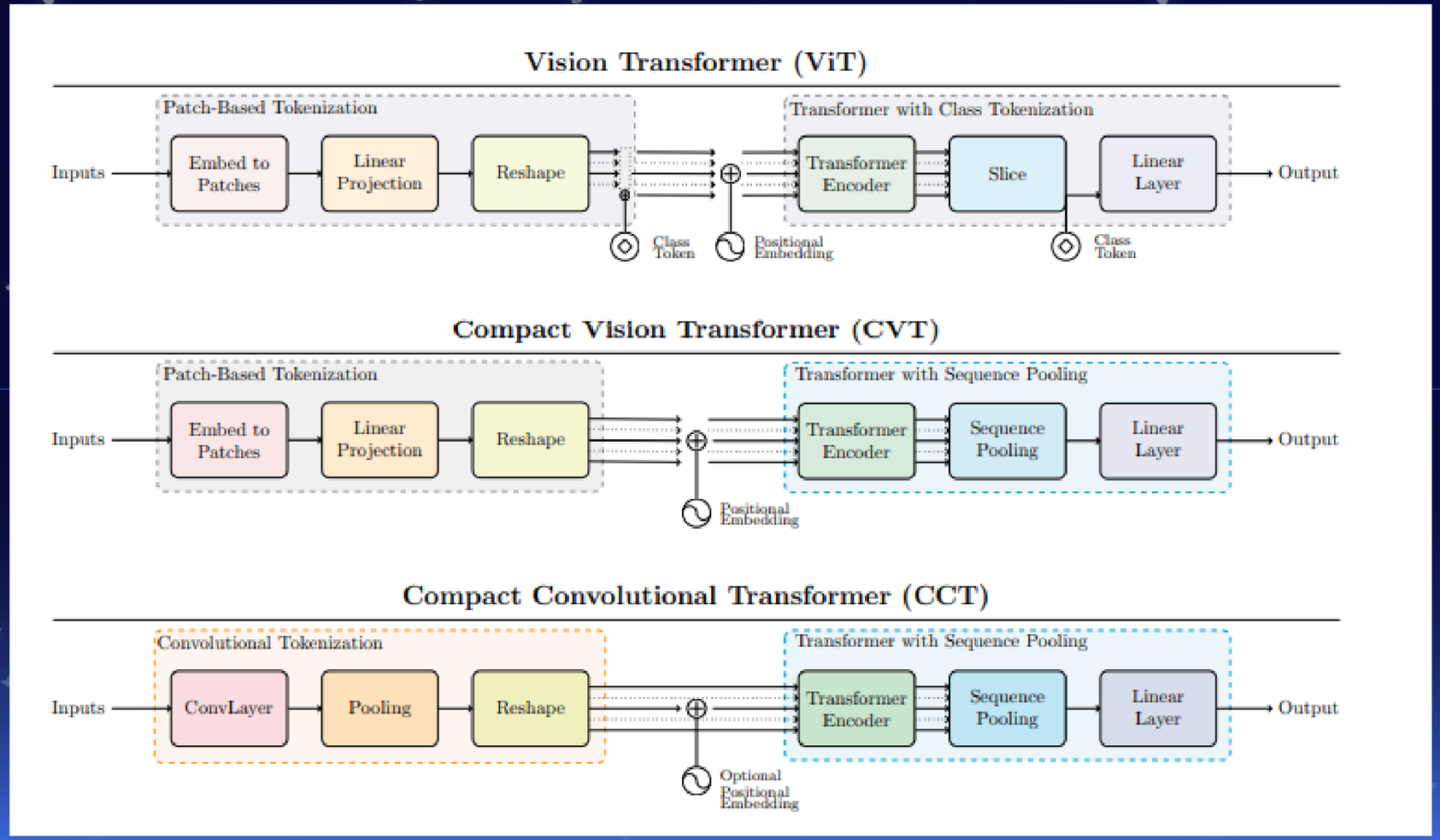
Yolov11n


Yolo V11 Architecture
Backbone
Neck
Head
640 x 640 x 3
Conv
320X320X(min(64,mc)xw)
Conv
160X160X(min(128,mc)xw)
C3K2
shortcut=False,
n=3xd
160X160X(min(128,mc)xw)
Conv
80X80X(min(256,mc)xw)
C3K2
shortcut=False,
n=6xd
80X80X(min(256,mc)xw)
Conv
40X40X(min(512,mc)xw)
C3K2
shortcut=True,
n=6xd
40X40X(min(512,mc)xw)
Conv
20X20X(min(1024,mc)xw)
C3K2
shortcut=True,
n=3xd
80X80X(min(256,mc)xw)
40X40X(min(512,mc)xw)
20X20X(min(1024,mc)xw)
C3K2
shortcut=False,
n=3xd
Concat
80X80X(min(256,mc)xw)
Upsample
40X40X(min(512,mc)xw)
C3K2
shortcut=False,
n=3xd
Concat
40X40X(min(512,mc)xw)
Upsample
20X20X(min(1024,mc)xw)
SPFF
20X20X(min(1024,mc)xw)
C2PSA
20X20X(min(1024,mc)xw)
80X80X(min(256,mc)xw)
Conv
80X80X(min(256,mc)xw)
Concat
C3K2
shortcut=False,
n=3xd
40X40X(min(512,mc)xw)
Conv
20X20X(min(1024,mc)xw)
Concat
C3K2
shortcut=False,
n=3xd
80X80X(min(256,mc)xw)
Detect
40X40X(min(512,mc)xw)
Detect
20X20X(min(1024,mc)xw)
Detect

Yolov11n

Component	Description
Backbone (CSPNet)	Extracts features from input image using Cross-Stage Partial connections
Neck (PANet)	Path Aggregation Network - fuses multi-scale features
Head (Detect)	Outputs bounding boxes, class probabilities, confidence scores
Tracker (BoT-SORT)	Assigns persistent IDs to detected objects across frames

Fast-plate-ocr



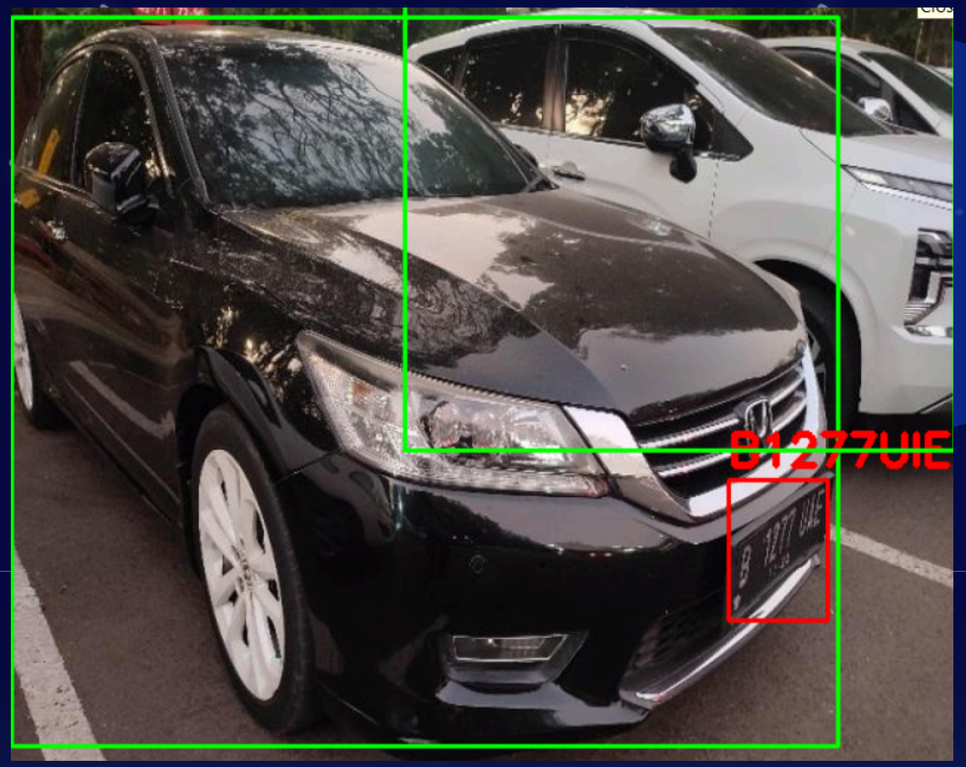
Fast-plate-ocr

Generic CCT	FastPlateOCR CCT
Single ConvLayer	**4 stacked Conv2D** layers
Standard MaxPool	**MaxBlurPooling2D** (anti-aliased)
LayerNorm	**DyT Normalization** (Dynamic Tanh)
Global pooling → 1 output	**Token Reducer → 9 outputs**
Single classification head	**9 parallel heads** (one per char)

Evaluation

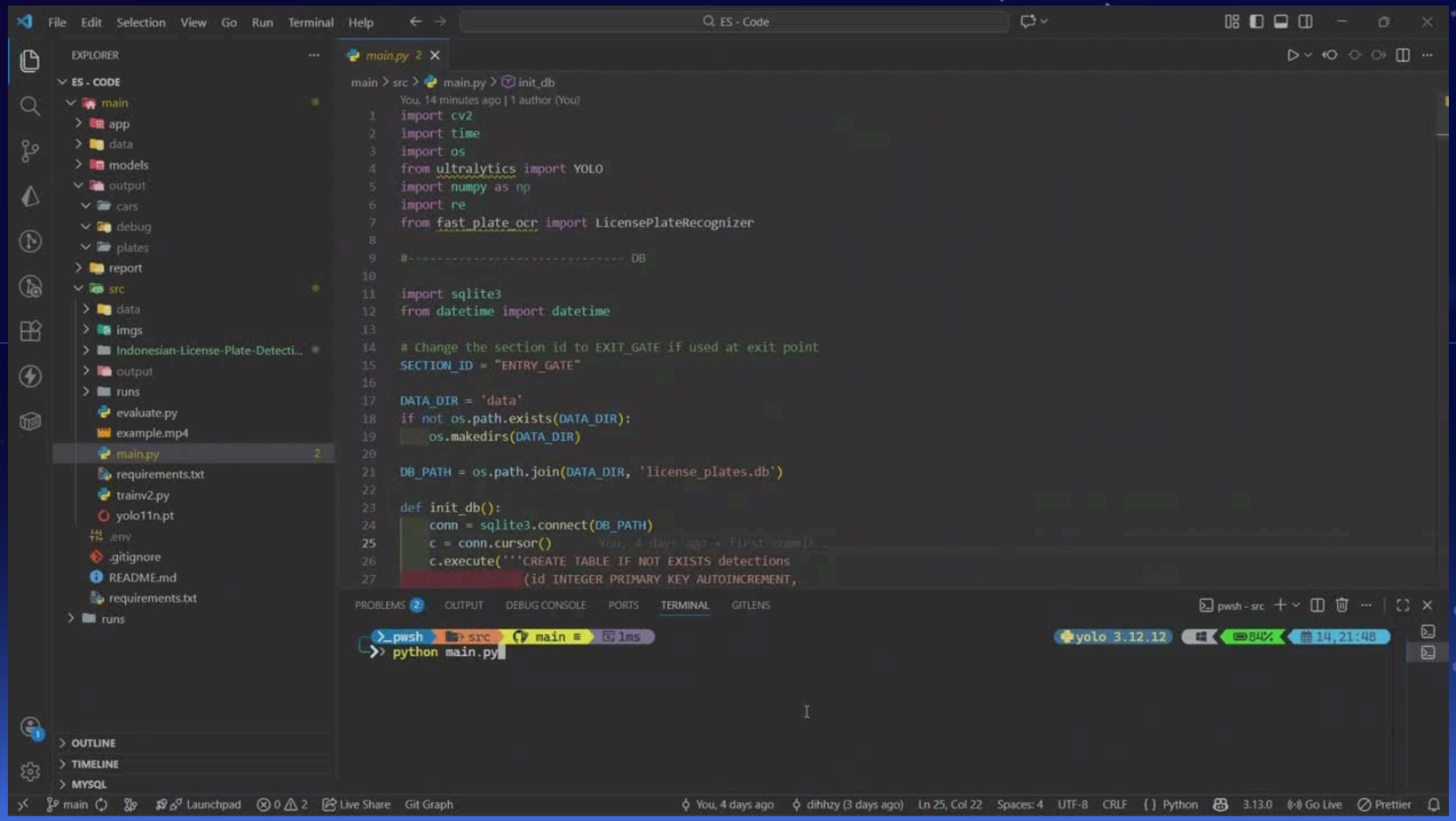
Metric	Score	Explaination
Plate Accuracy	97.22%	The percentage of plates where **every single character** was read correctly. This is the most important metric for real-world use.
Character Accuracy	99.15%	The average accuracy of recognizing individual letters/numbers (A-Z, 0-9).
Length Acc	98.84%	How often the model correctly predicted the number of characters on the plate (e.g., identifying a 7-digit plate as 7 digits).

Evaluation

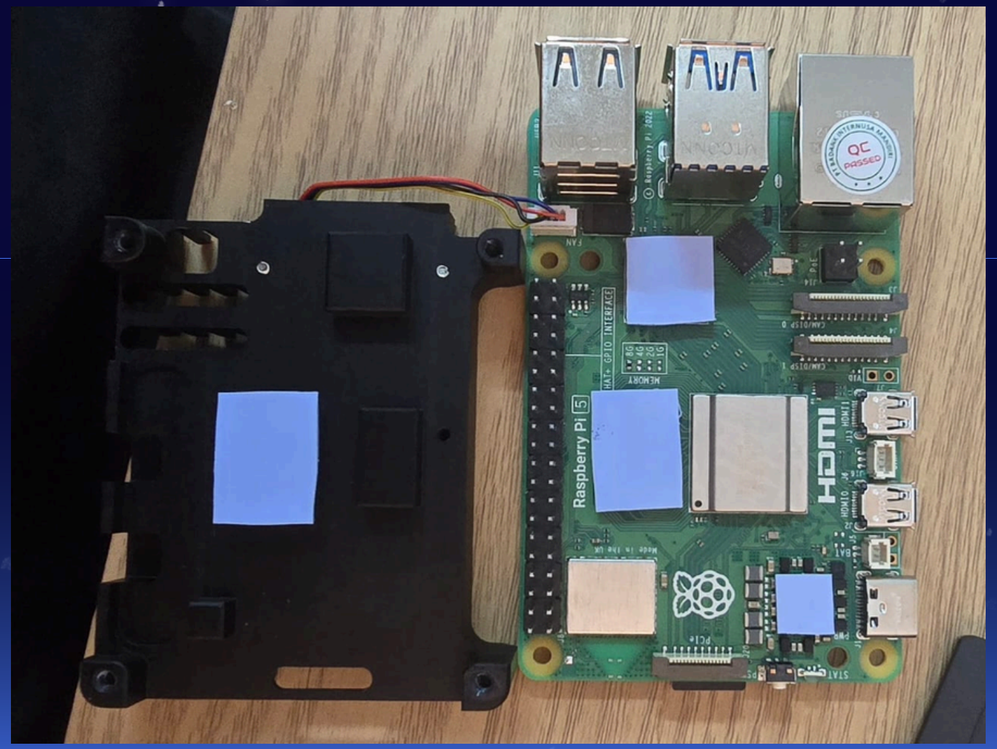


A manual evaluation was conducted for the model. On a test set of 34 images under suboptimal conditions. The system achieved perfect text recognition in 29 instances, while the remaining samples exhibited either minor discrepancies of 1-2 characters or complete detection failures. These results suggest that controlling environmental variables, particularly camera angle, is critical for maximizing OCR accuracy.

Demo


File Edit Selection View Go Run Terminal Help
ES - Code
EXPLORER
ES - CODE
main
app
data
models
output
cars
debug
plates
report
src
data
imgs
Indonesian-License-Plate-Detecti...
output
runs
evaluate.py
example.mp4
main.py
requirements.txt
trainv2.py
yolo11n.pt
.env
.gitignore
README.md
requirements.txt
runs
main.py 2
main > src > main.py > init_db
You, 14 minutes ago | 1 author (You)
import cv2
import time
import os
from ultralytics import YOLO
import numpy as np
import re
from fast_plate_ocr import LicensePlateRecognizer
#---------------------------- DB
import sqlite3
from datetime import datetime
# Change the section id to EXIT_GATE if used at exit point
SECTION_ID = "ENTRY_GATE"
DATA_DIR = 'data'
if not os.path.exists(DATA_DIR):
os.makedirs(DATA_DIR)
DB_PATH = os.path.join(DATA_DIR, 'license_plates.db')
def init_db():
conn = sqlite3.connect(DB_PATH)
c = conn.cursor()
c.execute('''CREATE TABLE IF NOT EXISTS detections
(id INTEGER PRIMARY KEY AUTOINCREMENT,
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
PORTS
TERMINAL
GITLENS
pwsh - src
_pwsh src main ≡ lms
yolo 3.12.12
84%
14, 21:48
python main.py
OUTLINE
TIMELINE
MYSQL
main
Launchpad
0 2
Live Share
Git Graph
You, 4 days ago
dihhzy (3 days ago)
Ln 25, Col 22
Spaces: 4
UTF-8
CRLF
Python
3.13.0
Go Live
Prettier

Demo


Raspberry Pi 5
HDMI
QC PASSED

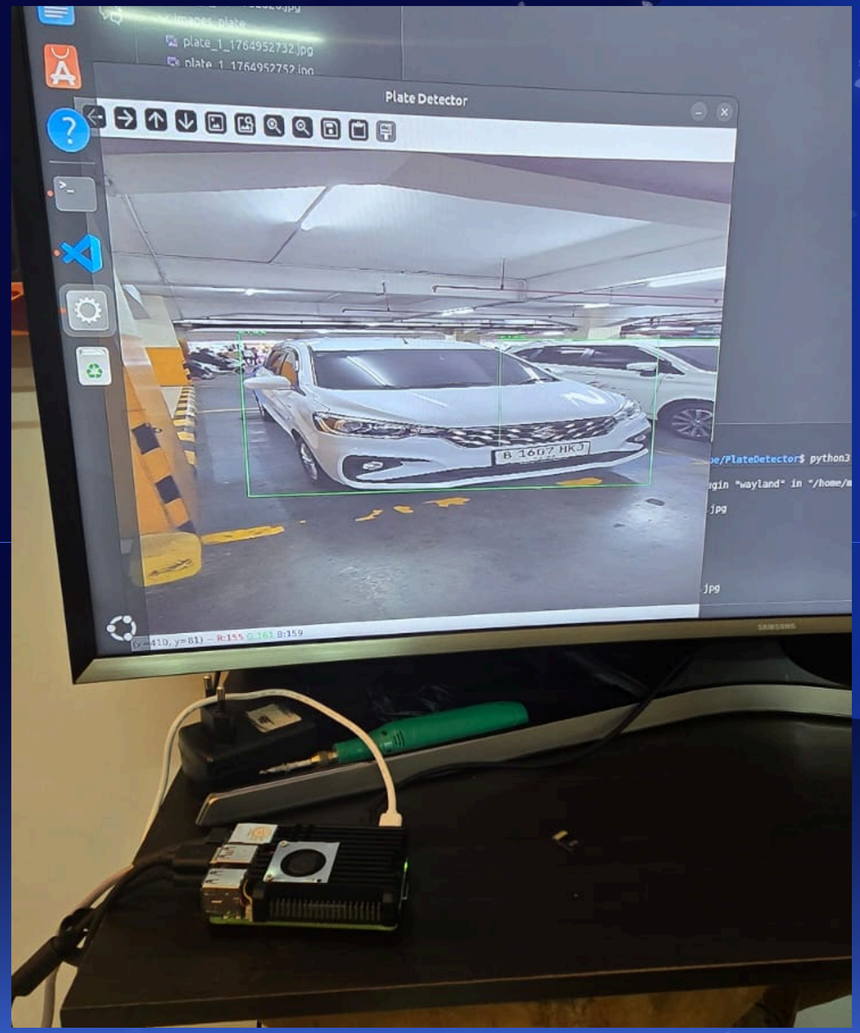


Plate Detector
plate_1_1764952752.jpg
B 1607 HKJ

Thank You

Refrences

https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71

Escaping the Big Data Paradigm with Compact Transformers https://arxiv.org/abs/2104.05704

https://ankandrew.github.io/fast-plate-ocr/latest/