How to build your own license plate recognition system

An AI agent swarm project: setting up ordinary CCTV cameras, choosing models and a measurable path to the 99% goal.

Jacek Szybicki

01

A PROJECT FROM AN AI AGENT SWARM

Problem and goal

Can an LPR system be built that correctly reads 99% of plates at a barrier-controlled entrance, on a single ARM single-board computer with an NPU, without a GPU, and with ordinary CCTV cameras? The commission covered **33 questions in 8 areas**, including models, architecture, data for Poland and Europe, cameras, hardware and large approach angles.

Research result: a system architecture design, CCTV camera setup parameters, model selection, and a plan for training and measuring performance.

SCALE OF THE MAIN STUDY

1,273 / 11,460
Agents' research checks / model calls

78.5 million / 1,404
Tokens / sources

16,854 of 18,849
Statements accepted by the verifier (89.4%)

19,018 / 1,470
Chapter analysts' work: statements read / cited

02

COURSE OF THE STUDY

How the system worked

A custom swarm running on the self-hosted **Qwen3.8-27B** model posed competing hypotheses and tested them against scientific publications, repositories and technical documentation. The verifier accepted statements tied to a verbatim quote from the source. Of 49 hypotheses, the system marked 26 as refuted, including the CPU-only variant for two 4 MP cameras under the assumed conditions.

My role

I defined the problem, the requirements and the architecture variants. I designed the commission for the swarm, supervised the run and assessed the result. **I rejected the first version**, because it summarized manufacturers' claims instead of answering how to build the system.

- I changed the questions from "what is best" to "how to build and measure it".
- I required the analysts to read the complete set of scientific and technical statements before writing.

The swarm collected the data and verified it automatically. AI analysts (Claude) wrote the chapters from the statement database, and Python scripts performed the calculations. The second version delivered the documentation needed to start building a prototype.

The swarm ran on 2 RTX PRO 6000 cards for 3.7 hours. GPU rental cost USD 13.08.



Figure 1. Swarm hypotheses view: the outcome of source assessment and of competing architecture proposals. Competing hypotheses, their assessment status and the argumentation gathered by the agents. (interface in Polish)

03

IMAGE ANALYSIS ARCHITECTURE

Seven stages of image analysis

Different models and rules handle separate tasks. The OCR receives a cropped and rectified plate; the result for a pass is built from many observations.

01 Vehicle detection

YOLOX-Tiny INT8, 640 × 384 input, planned 12.5 fps. The vehicle is located on a downscaled image, and the plate is searched for within its area at full resolution. Vehicle detection benchmark cited in the study: 99.92% recall on eight datasets [1].

02 Plate detection and rectification

The detector finds four corners and a single- or double-row layout. A homography from the four points straightens the oblique plate before OCR.

03 Quality gates

Before recognition, character size, sharpness, blur and angle are checked. Usable crops go into fusion, not every frame regardless of quality.

04 Two OCR models from different families

The character detector also handles double-row layouts; the second model is a fast 9-position classifier trained from the plate text. In the study [2], Fast-OCR reached 97.2% versus 93.5% for CRNN on OpenALPR-EU, and 59.6% versus 31.7% on the difficult CD-HARD set.

05 Multi-frame fusion

Combining character probability distributions from about 16 usable crops of a single pass at 30 km/h. Temporal fusion limits the errors of single observations [9]; the second OCR is meant to help with repeatable errors. The inspiration for assessing agreement between models is [8].

06 Decoding with Polish plate syntax

Templates derived from the regulation restrict the permissible strings. Correction is to apply only to 7 pairs of confusable characters, not to arbitrary rewriting of the OCR result.

07 Calibrated confidence

Logistic regression on 16 features and isotonic calibration determine the decision: accepting the read, an additional check against the authorized list, or routing to the exception path.

04

SETTING UP ORDINARY CCTV CAMERAS

What image the camera must deliver

Mounting geometry, plate resolution and exposure time are part of the recognition design, not settings left to chance.

Character and plate resolution

The design adopts a minimum of **8 × 16 px per character** and a target of **12 × 24 px**, based on an analysis of image resolution [3]. Publication [4] shows the scale of the problem: for 16 × 48 px crops the tested OCR achieved about 2% full reads.

156 px	188 px	266 px
Head-on	45°	60°

Plate angle / required plate width in the image, at least

Above 31.7° the limit becomes the foreshortening of the plate's width, not its height.

Setting up an ordinary CCTV camera

- **Image:** 4 MP, 25 fps, H.265. Varifocal lens; horizontal field of view no greater than 47°, to keep the required pixels at the end of the reading window.
- **Mounting in line with the barrier:** height 1.3–2.5 m; lateral offset up to 2.5 m; reading window 2–9 m in front of the camera.
- **Exposure:** time limited by vehicle motion and angle, controlled by the SBC instead of being left to the camera's automatic exposure.

Adopted blur criterion: $t \le 0.0045 / (v \cdot \tan \alpha)$, where t is in seconds, v in m/s, and α is the angle used in the geometry model. At 30 km/h: about 0.94 ms for 30°, 0.54 ms for 45° and 0.31 ms for 60°.

- **Night:** the design assumes 850 nm IR, a monochrome image, WDR and BLC off, and a reading angle of up to 30°. The literature [5] points to the influence of plate construction and retroreflectivity and of camera settings on night capture; it concerns studies of British plates.

Matching the computation to the RK3588

For deployment, convolutional models in INT8 on the NPU and hardware image decoding were chosen. The Frigate documentation [6] gives 18 ms for YOLOX-Tiny INT8 on the RK3588 using 3 NPU cores. The design estimates handling one crop at 15.2 ms, and 39.7 ms in the conservative variant; these cover different scopes of computation than the vehicle detector's time.

The baseline variant omits PARSeq and the other transformers considered because of the time budget and the deployment path [7]. YOLOX was also chosen for its Apache 2.0 license [12].

05

MEASURING PERFORMANCE AND TRAINING

How to measure the path to 99%

The goal concerns a correct read of the whole pass. A high score of a single model or of character reading is not enough to describe the quality of the whole system.

Stage loss budget

The design gives up basing the result on a single frame alone. One reference point was 86.4% of reads per frame for a Polish-plate model [11]. The losses are broken down into five separately recorded categories:

LOSS CATEGORY	ADOPTED LIMIT
Read not triggered	0.05%
Too few usable crops	0.15%
Random OCR error after fusion	0.10%
Persistent per-vehicle error: dirt, bolt, sticker	0.20%
Correct read rejected by the confidence threshold	0.50%
Product of the stage survival rates	**99.003%**

With this split of the loss budget, the product of the stage survival rates gives 99.003%. An additional goal is no more than 0.1% of passes misrecognized.

Proposed prototype acceptance

- 3,000 passes counted independently of the system; no more than 20 incorrect reads and zero misreads.
- For 2,980 correct results out of 3,000, the exact one-sided lower 95% confidence bound is about **99.03%** (Clopper-Pearson).

The event log distinguishes the absence of an accepted result from the acceptance of a wrong number. For zero misreads in 3,000 trials, the one-sided upper 95% bound is about 0.10%.

Data and fine-tuning on site

- **Start:** 100,000 synthetic crops of Polish plates, CCPD for detector pretraining, and at least 5,000 unique plates from own recordings for version 1.
- **Automatic labels:** at least 3 agreeing reads and agreement with the teacher model on the server. The label of a pass is also transferred to its smaller and blurred crops.
- **Updates:** fine-tuning on the server after every 2,000 new passes, INT8 QAT, and a comparison of the new model in observation mode before deployment on the device.

Value of the study: concrete parameters and a measurement plan that make it possible to move from a technology review to implementation and controlled validation of the system.

06

STUDY
DOCUMENTATION

Research trail and sources

Sources cited in the study documentation

01 Laroca et al. – plate detection and recognition.
arxiv.org/abs/1909.01754

02 Laroca et al. – Leveraging Model Fusion for Improved License Plate Recognition.
arxiv.org/abs/2309.04331

03 Low-Power License Plate Detection and Recognition on a RISC-V Multi-Core MCU-Based Vision System.
arxiv.org/abs/2607.09768

04 Nascimento et al. – Toward Advancing License Plate Super-Resolution in Real-World Scenarios.
raysonlaroca.github.io/papers/nascimento2025toward-published.pdf

05 Rhead et al. – plate construction, IR reflectivity and ANPR settings.
uhra.herts.ac.uk/id/eprint/14235/

06 Frigate – Rockchip Supported Models.
docs.frigate.video/configuration/object_detectors/#rockchip-supported-models

07 llama.cpp – discussion of RK3588 performance, issue #722.
github.com/ggml-org/llama.cpp/issues/722

08 Consensus Entropy – agreement of multiple VLM models in OCR.
arxiv.org/abs/2504.11101

09 Multi-Task Deep Learning Framework – temporal validation of reads.
arxiv.org/abs/2607.03131

10 DOI:
doi.org/10.1109/icicc66840.2025.11199678

11 A CNN-RNN hybrid approach for Polish license plate recognition.
mgv.sggw.edu.pl/article/view/10236

12 YOLOX – Apache 2.0 license.
github.com/Megvii-BaseDetection/YOLOX/blob/main/LICENSE