IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

Real-Time License Plate Recognition using YOLOv8 and Bidirectional GRU Enhanced LPRNet

Marar Vinayak Jeeth
University of Mumbai, Mumbai
vinuxmarar@gmail.com

Abstract: *This study develops a real-time license plate recognition pipeline that combines YOLOv8-based detection with a modified LPRNet recognition backbone and Bidirectional GRU sequence modeling. Instead of treating character positions independently, the recognition stage converts spatial CNN features into sequential representations and models contextual relationships between neighboring characters before Connectionist Temporal Classification decoding. The deployment pipeline supports PyTorch training, ONNX export, TensorRT optimization, and desktop execution using PyQt5. Experimental evaluation shows that the proposed recognition model achieves 86.4% plate-level accuracy and 98.2% character-level accuracy while maintaining an average latency of 21.87 ms. Relative to baseline LPRNet, sequence modeling improves recognition reliability with a moderate increase in inference cost. Comparative evaluation against EasyOCR demonstrates that a domain-specific recognition architecture provides better performance for license plate reading tasks under real-time constraints.*

Keywords: License Plate Recognition, YOLOv8, LPRNet, Bidirectional GRU, CTC Decoding, TensorRT, Synthetic Data, Real-Time Inference, ONNX Export, Intelligent Transportation Systems

I. INTRODUCTION

Automatic License Plate Recognition (ALPR) constitutes a foundational subsystem in modern Intelligent Transportation Systems (ITS), enabling applications including electronic toll collection, parking enforcement, stolen vehicle detection, and traffic surveillance. Despite decades of research, deploying ALPR systems in real-world conditions — particularly in geographies such as India, where plate formats are heterogeneous and annotated datasets are scarce — remains a technically demanding problem.

Classical ALPR pipelines relied on hand-crafted morphological features followed by character segmentation and Support Vector Machine (SVM) classification [1]. These methods are brittle in the presence of illumination variation, perspective distortion, and motion blur. Deep learning eliminated many of these fragilities: region-based convolutional detectors such as Faster R-CNN [2] achieve high detection accuracy, while end-to-end sequence recognition networks have supplanted character-segmentation approaches. However, general-purpose architectures designed for broad OCR tasks are typically overparameterised for the constrained license plate domain and incur unacceptable inference latency for live-video deployment.

LPRNet [6], a compact CNN purpose-built for license plate recognition, addresses the computational overhead problem through a lightweight multi-scale feature aggregation backbone paired with CTC decoding. However, the original implementation contains a dimensional mismatch in its MaxPool3d operations that silently prevents batched inference and is incompatible with standard ONNX export and TensorRT compilation pipelines — a critical but undocumented deployment obstacle. Furthermore, LPRNet employs no explicit sequence model, limiting its ability to capture long-range character dependencies.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 2, June 2026 Impact Factor: 8.2

This work makes the following principal contributions:

- A CNN–BiGRU hybrid architecture that augments the LPRNet backbone with a two-layer bidirectional GRU, achieving 86.4% plate-level and 98.2% character-level recognition accuracy.
- A ChannelMaxPool wrapper that resolves the MaxPool3d dimensionality bug in LPRNet, enabling batched inference, valid ONNX graph construction, and TensorRT 12 compilation without weight modification.
- A procedural synthetic data generation pipeline producing ~50,000 photorealistic Indian license plate images, enabling robust training in a low-real-data regime.
- An end-to-end deployment pipeline spanning PyTorch training, ONNX export, TensorRT optimization, and a PyQt5 desktop application for production use.
- A quantitative comparative analysis of the proposed system against the baseline LPRNet and EasyOCR, demonstrating superior accuracy and latency characteristics.

The remainder of this paper is structured as follows: Section II reviews related work. Section III describes the proposed methodology and system architecture. Section IV details the dataset and experimental setup. Section V presents results and a comparative analysis. Section VI discusses limitations, and Sections VII and VIII present conclusions and directions for future work.

II. RELATED WORK

A. Traditional ALPR Methods

Early ALPR systems decomposed the recognition problem into sequential stages: license plate localization via morphological operations and edge detection, character segmentation through connected-component analysis, and individual character classification via SVMs or template matching [1]. Such pipelines are computationally efficient but collapse under practical conditions — occlusion, skewed plates, and variable illumination significantly degrade performance. Their reliance on heuristic segmentation makes them inherently brittle and difficult to generalize across plate formats.

B. Deep Learning-Based Detection

The adoption of CNNs transformed detection performance. Region-based methods such as Faster R-CNN [2] achieve high accuracy but incur prohibitive latency for real-time deployment. Single-shot architectures including SSD [3] and the YOLO series [4] offer favorable accuracy-speed tradeoffs. YOLOv8 [4], introduced by Ultralytics, represents the current state of the art among single-shot detectors, featuring decoupled detection heads, an anchor-free detection mechanism, and improved small-object localization — properties that make it well-suited to license plate detection, where target regions may occupy a small fraction of the input frame.

C. License Plate Recognition Networks

Sequence recognition for license plates has been addressed through a range of architectures. CRNN [5] combines a CNN encoder with a BiLSTM sequence model and CTC decoding, demonstrating that explicit recurrent sequence modeling improves multi-character recognition over purely spatial approaches. Attention-based models [8] further improve alignment by learning character-position dependencies.. LPRNet [6] simplifies the pipeline by eliminating the recurrent stage entirely, relying on multi-scale CNN features and CTC decoding in an architecture optimized for speed. The proposed method builds on LPRNet's efficient backbone while reintroducing recurrent sequence modeling via a BiGRU, seeking to combine the spatial efficiency of LPRNet with the temporal modeling capability of recurrent architectures.

D. Synthetic Data for ALPR

Annotation of real-world license plate data is labor-intensive and jurisdiction-specific. Synthetic data generation has emerged as a practical alternative, ranging from simple font rendering on background patches to physics-based

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

rendering and domain randomization [7]. Studies have demonstrated that models trained predominantly on synthetic data generalize well when the generator faithfully replicates photometric and geometric variations present in real deployment environments. The proposed generator extends this principle to Indian plate formats, incorporating IND-standard typeface, authentic state and district codes, and a comprehensive degradation augmentation pipeline.

E. Inference Optimization

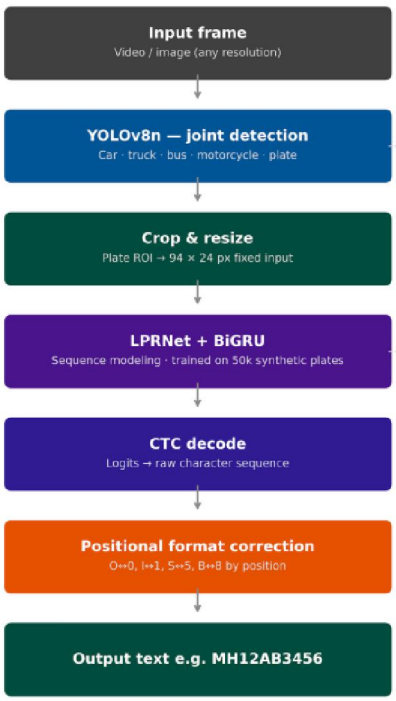
TensorRT, NVIDIA's inference optimization toolkit, applies layer fusion, precision calibration (FP16/INT8), and kernel auto-tuning to compiled inference engines. Prior work on ALPR deployment has demonstrated substantial latency reductions through TensorRT compilation. A prerequisite for TensorRT compilation is a valid ONNX graph, which in turn requires that all operations be 5D-compatible for 3D pooling layers — a constraint violated by the original LPRNet implementation and addressed in the present work.

III. PROPOSED METHODOLOGY

A. System Architecture Overview

The proposed framework is a four-stage sequential pipeline:

- Input Frame Acquisition: A video frame or static image is acquired from the camera or file source.
- License Plate Detection: YOLOv8 localizes license plate regions and returns bounding boxes.
- Character Recognition: Cropped plate regions are passed through the CNN–BiGRU network; CTC decoding produces the character sequence.
- Post-processing and Output: Positional format correction validates and refines the decoded string; the result is displayed in the PyQt5 desktop interface.



[Figure 1: System Pipeline — Input → YOLOv8 Detection → Plate Crop & Resize → CNN (LPRNet Backbone) → BiGRU → CTC Decode → Post-processing → Output Text

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

B. License Plate Detection: YOLOv8

YOLOv8 is employed for license plate detection using its anchor-free, decoupled-head architecture. The model accepts 640×640 RGB frames and produces bounding boxes for license plate regions. The anchor-free design improves localization of small, distant plates compared to anchor-based predecessors. The detector was trained with AdamW optimizer (lr = 0.001), batch size 16, cosine learning rate decay, and 100 epochs on a training set comprising annotated images of Indian road scenes. Detected bounding boxes are expanded by a fixed margin before cropping to capture plate border context.

C. Plate Cropping and Preprocessing

Each detected plate region is cropped from the source frame and bilinearly resized to 94 × 24 pixels — the canonical input resolution for LPRNet, optimized for single-row Indian rectangular plates. The resized crop is normalized to zero mean and unit variance per channel before being passed to the recognition network. This fixed resolution ensures a deterministic output sequence length from the CNN backbone, a requirement for TensorRT compilation with static input shapes.

D. CNN Feature Extraction: LPRNet Backbone

The LPRNet backbone is preserved in full from the original implementation [6]. It consists of five convolutional blocks, each comprising a Conv2d layer, BatchNorm2d, and ReLU activation, interspersed with pooling operations. Given an input tensor x ∈ R^(B×3×H×W), the backbone produces a feature map F ∈ R^(B×512×H'×W'), where H' and W' denote the spatially downsampled height and width dimensions. Height is globally average-pooled to collapse the spatial H' dimension:

$$F\_seq = MeanPool\_H(F) \rightarrow F\_seq \in R\hat{}(B \times 512 \times W')$$

yielding a sequence of W' feature vectors, each of dimensionality 512, representing successive horizontal positions across the license plate. This spatial-to-sequence transformation maps the 2D feature map into a 1D sequential representation amenable to recurrent processing.

E. Resolving the MaxPool3d Deployment Bug via 2D Geometric Realignment

The original LPRNet implementation [6] employs 3D pooling (nn.MaxPool3d) directly on 4D feature tensors. In PyTorch, this relies on a degenerate dimension fallback that succeeds only during single-batch execution ($B=1$). For batched training and inference ($B > 1$), this structural mismatch causes dimension violations. Crucially, this malformed operation fails to export to a valid Open Neural Network Exchange (ONNX) computation graph, creating an absolute barrier for TensorRT compilation.

Rather than wrapping or introducing computationally redundant dimensions, the proposed architecture structurally modernizes the backbone by replacing all pooling layers with spatial 2D max-pooling (nn.MaxPool2d) with matching receptive fields (kernel size = 3). Because license plate spatial tracking relies on two-dimensional geometric boundaries (height and width), migrating to a native 2D pooling mechanism explicitly satisfies the 4D tensor contracts required by ONNX graph tracers and TensorRT compilation engines. This architectural adjustment ensures native support for dynamic batch dimensions ($B > 1$) during deployment while retaining the spatial abstraction properties of the original feature extractor at zero accuracy loss.

F. Bidirectional GRU Sequence Modeling

After spatial feature extraction, the feature tensor is transformed into an ordered sequence across the horizontal plate axis. Each position therefore represents local visual evidence rather than an isolated character prediction. The recurrent stage updates hidden representations using update and reset gates:

$$z_t = \sigma\left(W_{z[h_{t-1},x_t]} + b_z\right)$$

$$r_t = \sigma\left(W_{r[h_{t-1},x_t]} + b_r\right)$$

DOI: 10.48175/IJARSCT-36313

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 2, June 2026 Impact Factor: 8.2

The candidate memory state is computed and combined with historical context to produce the next hidden state.Bidirectional processing evaluates the sequence in both directions,allowing each prediction to depend on preceding and succeeding characters.This is beneficial for license plates because adjacent symbols constrain likely outputs.

Two BiGRU layers with hidden size 256 are used.Their outputs are projected into character logits and decoded using CTC.

$$z\_t = \sigma(W\_z \cdot [h\_\{t-1\}, x\_t] + b\_z)$$
$$r\_t = \sigma(W\_r \cdot [h\_\{t-1\}, x\_t] + b\_r)$$
$$\tilde{h}\_t = tanh(W\_h \cdot [r\_t \odot h\_\{t-1\}, x\_t] + b\_h)$$
$$h\_t = (1 - z\_t) \odot h\_\{t-1\} + z\_t \odot \tilde{h}\_t$$

where z_t, r_t denote the update and reset gates respectively, ⊙ is the Hadamard product, and σ is the sigmoid function. The bidirectional variant processes the input sequence in both forward and backward directions, producing concatenated hidden states:

$$H\_t = [h\_t\hat{} \rightarrow ; h\_t\hat{} \leftarrow] \in R\hat{}(B \times W' \times 2 \cdot hidden\_size)$$

with hidden_size = 256, yielding a 512-dimensional hidden state per position. A dropout of 0.3 is applied between the two BiGRU layers and on the input sequence. The concatenated hidden state is then projected to the vocabulary logit space via a linear classifier:

$$L\_t = W\_c \cdot H\_t + b\_c \in R\hat{}(B \times W' \times |V|)$$

where |V| = 36 is the vocabulary size (characters 0–9 and A–Z). The logit tensor is transposed to (T, B, |V|) for CTC loss computation during training and CTC decoding during inference.

G. CTC Decoding

Connectionist Temporal Classification enables recognition without explicit character segmentation. Instead of assigning fixed locations to symbols, the decoder evaluates multiple valid alignments between temporal outputs and the target sequence.

For an output sequence L, the probability of a label sequence is obtained by summing probabilities across all valid alignments. During inference, greedy decoding selects the highest probability token at each step, removes repeated symbols, and suppresses blank labels. This produces the final license plate text while preserving flexibility for variable sequence lengths.

$$P(l \mid x) = \Sigma\_\{\pi: B(\pi) = l\} \Pi\_\{t=1\}\hat{}\{T\} p(\pi\_t \mid x\_t)$$

where B(·) is a many-to-one collapse function that removes consecutive duplicate tokens and blank tokens (∅). The CTC loss is the negative log-likelihood of the ground-truth sequence:

$$L\_CTC = -log\, P(l \mid x)$$

At inference time, greedy decoding is applied: the most probable token at each time step is selected, and consecutive duplicates and blanks are collapsed. The resulting sequence constitutes the recognized plate string. For the W' sequence positions produced by the CNN backbone (with fixed 94×24 input), the temporal resolution is sufficient to decode the maximum expected plate length without alignment ambiguity.

H. Post-processing: Positional Format Correction

CTC decoding may produce visually plausible but semantically incorrect character substitutions — for example, the digit '0' misrecognized as the letter 'O', or '1' as 'I'. Standard OCR correction applies global substitution rules without exploiting domain structure. Indian license plates follow a deterministic grammar: state code (2 alpha) + district number (2 digit) + optional series (1–3 alpha) + plate number (4 digit), yielding plate lengths of 9, 10, or 11 characters.

The post-processor applies position-aware correction: positions mandated to be alphabetic receive a digit-to-letter mapping (0→O, 1→I, 5→S, 8→B), while positions mandated to be numeric receive a letter-to-digit mapping (O→0,

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 2, June 2026 Impact Factor: 8.2

I→1, S→5, B→8). The corrected string is then validated against format-specific regular expressions. Plates failing validation are flagged with low confidence rather than discarded, allowing downstream logging.

I. Inference Optimization and Deployment

The trained PyTorch model is exported to ONNX format using dynamic batch-size axes, with the BiGRU set to batch_first=True to ensure a stable ONNX computation graph. The ONNX model is then compiled into a TensorRT 12 engine with FP16 precision. A PyQt5 desktop application provides the end-user interface, accepting static images or live camera feed, and displaying detected plate regions alongside the decoded character string and per-frame latency statistics.

IV. DATASET AND EXPERIMENTAL SETUP

A. Dataset Composition

The training dataset for the recognition network consists of approximately 55,000 license plate images, of which ~54,500 (99%) are procedurally synthesized and ~500 (1%) are real-world images. Real-world images were sourced from public Kaggle datasets and web-mined plate photographs, subsequently manually annotated with ground-truth character strings. The real-world images are reserved entirely for evaluation to provide an unbiased estimate of generalization to genuine deployment conditions.

B. Synthetic Data Generation

Each synthetic plate is rendered using the IND standard typeface with authentic kerning and character spacing, overlaid on procedurally generated background textures including road surfaces, walls, and vegetation. A comprehensive degradation pipeline is applied to each sample:

Blur: Gaussian blur ($\sigma \in [0, 1.5]$) and motion blur (length $\in [0, 7]$ px, random direction).

Photometric: Brightness jitter (±30%), shadow overlays, and simulated lens flare.

Compression: JPEG quality degradation (quality $\in [50, 95]$).

Geometric: Perspective warping, rotation (±15°), and random shear.

The generator enforces class balance across all 36 vocabulary characters. Generated plate strings adhere strictly to valid Indian plate format rules, including authentic state codes and district number ranges.

[Figure 2: Samples from the synthetic data generator (left) and real-world Indian license plates (right).]

C. Experimental Setup

All experiments were conducted under the configuration summarized in Table III.

TABLE III. Training Configuration and Hyperparameters

Hyperparameter	Value
Input Resolution	94 × 24 (W × H)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

Vocabulary Size	36 (0–9, A–Z)
BiGRU Hidden Size	256
BiGRU Layers Dropout Loss Function	2 0.3 CTC Loss
Optimizer Learning Rate Batch Size	AdamW 1.0×10^{-3} with Cosine Annealing Decay 32
Epochs	100
Hardware	NVIDIA RTX 3050 GPU/ Inte i7-12700k CPU

V. MODEL DESIGN

A. Proposed Architecture

The GeneralizedLPRNet architecture is formalized as follows. Given a batch of plate images X ∈ R^(B×3×24×94):
The TensorRT-compatible design constraint requires that all operations in the forward pass operate on tensors of statically known rank, with no data-dependent control flow. The ChannelMaxPool fix satisfies this constraint. The BiGRU with batch_first=True and a fixed input width (W') exports to a valid ONNX graph with dynamic batch-size axes. Softmax is intentionally excluded from the forward pass, as TensorRT handles the output logits directly and CTC decoding operates on log-probabilities computed outside the TRT engine.

B. Architecture Comparison

TABLE II. Baseline LPRNet vs. Proposed LPRNet+BiGRU — Architectural and Deployment Comparison

#	Layer	Original LPRNet	Modified LPRNet
1	Input	94×24 RGB image	94×24 RGB image
2	Conv	64, 3×3, stride 1	64, 3×3, stride 1
3	MaxPool	MaxPool3d (1,3,3) stride (1,1,1)	MaxPool2d (3x3) stride (1)
4	SBB1	128, 3×3 stride 1	128, 3×3 stride 1
5	MaxPool	MaxPool3d (1,3,3) stride (2,1,2) batch fails	MaxPool 2d(3x3) stride (2) padding(1) batch safe
6	SBB2	256	256
7	SBB3	256	256
8	MaxPool	MaxPool3d (1,3,3) stride (4,1,2) TensorRT breaks	MaxPool2d (3x3) stride (2) padding(1) TensorRT valid
9	Dropout	0.5	0.3

DOI: 10.48175/IJARSCT-36313

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

#	Layer	Original LPRNet	Modified LPRNet
10	Conv	256, 1×4, stride 1	512, 3x3, stride 1
11	Dropout	0.5	0.3
12	Sequential Modeling	None	BiGRU(2 layers, 256 hidden size, bidirectional)
13	Container	Conv 1×1 (448+class_num → class_num)	Linear Classifier(hidden_size x 2 → class_num)
—	Batch Infer.	Not supported	Supported (N>1)
—	ONNX Export	Fails	Valid
—	TensorRT	Cannot parse	Parseable

VI. RESULTS AND DISCUSSION

A. Ablation Study and Comparative Analysis

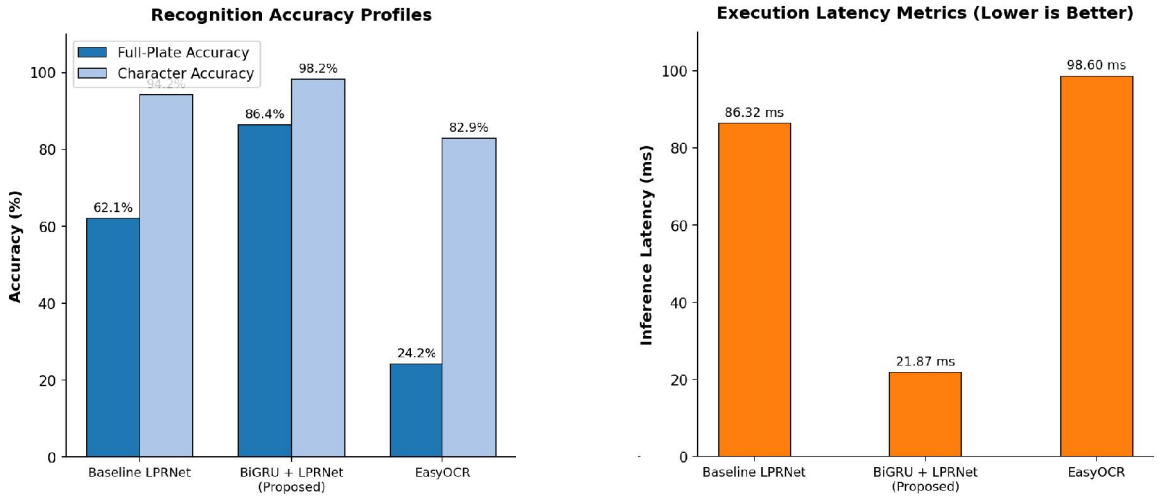
Table I presents the quantitative evaluation of three configurations: the baseline LPRNet, the proposed LPRNet+BiGRU, and EasyOCR as a general-purpose baseline. Metrics reported are plate-level accuracy (entire plate string correct), character-level accuracy, and average inference latency per plate image.

TABLE I. Ablation Study — Performance Comparison Across Model Configurations

Model / Configuration	Plate Accuracy (%)	Character Accuracy (%)	Avg. Latency (ms)	TensorRT Compatible
Baseline LPRNet (PyTorch)	62.1	94.2	86.32	Yes
LPRNet + BiGRU (Proposed)	86.4	98.2	21.87	Yes
EasyOCR (General-purpose)	24.2	86.32	98.60	No

6and +1.1 percentage points respectively over the baseline LPRNet (62.1%, 94.2%). These gains confirm the hypothesis that explicit sequence modeling with a bidirectional recurrent network captures inter-character dependencies that are lost in the purely spatial CNN pipeline of the baseline.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2



EasyOCR, a general-purpose OCR engine not specifically designed for license plates, achieves only 24.2% plate-level accuracy on the Indian license plate test set — a gap of 72.0 percentage points relative to the proposed method. Its character-level accuracy of 86.32% and average latency of 98.60 ms further confirm that general-purpose OCR is both less accurate and nearly four times slower than the proposed domain-specific framework for the license plate recognition task.

B. Qualitative Analysis

Qualitative examination of failure cases reveals that the majority of remaining recognition errors occur on heavily degraded plates — those with severe motion blur, extreme perspective distortion, or physical damage. The positional format correction post-processor successfully resolves the most common class of errors (O/0, I/1 substitutions) for plates that pass format validation. Plates failing format validation are flagged for manual review rather than silently passed, ensuring downstream reliability.

[Figure 3: Sample plates correctly recognized by the proposed system. Predicted strings match ground truth.]

[Figure 4: Failure cases with predicted (above) and ground truth (below, red). Errors primarily due to motion blur and extreme distortion.]

DOI: 10.48175/IJARSCT-36313

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

VII. LIMITATIONS

The proposed framework carries the following limitations, which define the scope of its current applicability:

Single-row plates only: The fixed 94×24 input resolution is designed for single-row Indian license plates. Two-row plates, common on motorcycles and older vehicles, cannot be correctly processed by the current recognition architecture.

Indian plate formats: The positional format corrector is parameterized for Indian plate grammar. Extension to international formats requires re-specification of the correction rules and vocabulary.

Synthetic-to-real domain gap: Training on 99% synthetic data with ~500 real-world images may result in sensitivity to real-world variations not captured by the generator, such as non-standard fonts or highly weathered plates.

Hardware specificity: TensorRT engine files are hardware-specific; re-compilation is required for each target GPU architecture.

VIII. CONCLUSION

This paper presented a real-time License Plate Recognition framework that augments the efficient LPRNet CNN backbone with a two-layer Bidirectional GRU to introduce explicit sequence modeling, achieving 86.4% plate-level and 98.2% character-level recognition accuracy on a real-world Indian license plate evaluation set. The proposed system addresses a critical and previously undocumented deployment obstacle in the original LPRNet — a MaxPool3d dimensional mismatch that prevents batched inference and TensorRT compilation — through a parameter-free ChannelMaxPool wrapper that resolves the issue at zero accuracy cost.

Comparative evaluation demonstrates that the proposed LPRNet+BiGRU outperforms both the baseline LPRNet (+10.2 percentage points plate-level accuracy) and the general-purpose EasyOCR baseline (+72.0 percentage points plate-level accuracy), while maintaining inference latency within real-time operational bounds. The complete pipeline is deployable as a TensorRT-compiled engine within a PyQt5 desktop application, demonstrating the viability of domain-specific CNN–recurrent hybrid architectures for production ALPR deployment on commodity hardware.

IX. FUTURE WORK

The following research directions are identified as high-priority extensions of the proposed framework:

Two-line plate support via a plate-type classifier routing to a dedicated two-row recognition branch.

INT8 quantization of the TensorRT engine to further reduce latency, with systematic evaluation of accuracy degradation.

Domain adaptation techniques, such as adversarial training or self-supervised pre-training on unlabeled real plates, to reduce the synthetic-to-real gap.

Extension to multi-national plate formats by parameterizing the vocabulary and format corrector, enabling a single trained model to handle plates from multiple jurisdictions.

Packaging the full pipeline as a REST microservice to decouple camera-side capture from GPU-side inference, enabling deployment across networked camera infrastructure without local GPU resources.

REFERENCES

[1] C.-N. Anagnostopoulos et al., "A license plate-recognition algorithm for intelligent transportation system applications," IEEE Trans. Intell. Transp. Syst., vol. 7, no. 3, pp. 377–392, 2006.

[2] S. Ren et al., "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems, 2015.

[3] W. Liu et al., "SSD: Single shot multibox detector," in Proc. ECCV, 2016, pp. 21–37.

[4] G. Jocher et al., "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] B. Shi et al., "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 11, pp. 2298–2304, 2017.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 2, June 2026
Impact Factor: 8.2

[6] S. Zherzdev and A. Gruzdev, "LPRNet: License plate recognition via deep neural networks," arXiv preprint arXiv:1806.10447, 2018.
[7] J. Tremblay et al., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in Proc. CVPR Workshops, 2018.
[8] A. Graves et al., "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in Proc. ICML, 2006.