[Array 15 (2022) 100244](https://doi.org/10.1016/j.array.2022.100244)

|  |  |  |
| --- | --- | --- |
|  | Contents lists available at [ScienceDirect](http://www.elsevier.com/locate/array) |  |
| Array |
| journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array) |
|  | | |

BLPnet: A new DNN model and Bengali OCR engine for Automatic Licence Plate Recognition   
Md. Saif Hassan Onim, Hussain Nyeem∗, Koushik Roy, Mahmudul Hasan, Abtahi Ishmam, Md.

Akiful Hoque Akif, Tareque Bashar Ovi   
*Department of EECE, Military Institute of Science and Technology (MIST), Mirpur Cantonment, Dhaka, 1216, Bangladesh*

|  |  |  |
| --- | --- | --- |
| A R T I C L E | I N F O | A B S T R A C T |
| *Keywords:*  ALPR  CNN  INCEPTION-V3  License plate detection NASNet Mobile  OCR |  | The development of the Automatic License Plate Recognition (ALPR) system has received much attention for the English license plate. However, despite being the sixth-largest population around the world, no significant progress can be tracked in the Bengali language countries or states for the ALPR system addressing their more alarming traffic management with inadequate road-safety measures. This paper reports a computationally efficient and reasonably accurate Automatic License Plate Recognition (ALPR) system for Bengali characters with a new end-to-end DNN model that we call Bengali License Plate Network (BLPnet). The cascaded architecture for detecting vehicle regions before vehicle license plate (VLP) is proposed to eliminate false positives, resulting in higher detection accuracy of VLP. Besides, a lower set of trainable parameters is considered for reducing the computational cost, making the system faster and more compatible for a real-time application. With a Convolutional Neural Network (CNN) based new Bengali OCR engine and word-mapping process, the model is characters-rotation invariant, and can readily extract, detect and output the complete license plate number of a vehicle. The model feeding with 17 frames per second (fps) of real-time video footage can detect a vehicle with the Mean Squared Error (MSE) of 0.0152, and the mean license-plate-character recognition accuracy of 95%. While compared to the other models, an improvement of 5% and 20% were recorded for the BLPnet over the prominent YOLO-based ALPR model and the Tesseract model for the number-plate detection accuracy and time requirement, respectively. |

**1. Introduction**

Automatic License Plate Recognition (ALPR) systems have received much attention in modern transportation services mainly for automatic management of traffic, parking, toll-station, and other road operations including surveillance and recognition of potential threats [1]. An ALPR system consists of three main phases: *frame-selection*, *character-segmentation*, and *optical character recognition* (OCR). The first phase verifies the existence of any Vehicle License Plate (VLP) in the in-put frame by extracting any possible character’s features from the frame. The second phase separates the characters from the background, followed by their recognition in the last phase. Such a system has more potential for the developing countries or states requiring higher road-safety measures and better traffic management.

One such potential group of the developing regions is the Bengali-speaking countries and states that still requires a promising ALPR system for the Bengali VLP application. Unlike English characters,

Bengali has more complex features, leaving its accurate recognition from a VLP more challenging. Despite being the six largest population around the world [2], no significant progress can be tracked in the Bengali language countries or states for the ALPR system, addressing their inadequate road-safety measures and poor traffic management. Besides, the performance of the prominent ALPR models developed for the English VLP is also unknown for the Bengali VLP recognition application.

Recent ALPR systems for English VLP captures the employment of promising machine learning models. Such models mainly detects the VLP in the given image (or a video frame) followed by the recognition of the text information on the plate. A variety of models is stemmed from the need for improving the classification accuracy, robustness to environmental artifacts, and computational efficiency. For exam-ple, the Convolutional Neural Network (CNN) based bounding box detectors were developed with regression algorithms [3–7], manual

∗ Corresponding au[thor.](mailto:saif@eece.mist.ac.bd)

*E-mail addresses:* [saif@eece.mist.ac.bd](mailto:saif@eece.mist.ac.bd) (M.S.H. Onim), [h.nyeem@eece.mist.ac.bd](mailto:h.nyeem@eece.mist.ac.bd) (H. Nyeem), [rkoushikroy2@gmail.com](mailto:rkoushikroy2@gmail.com) (K. Roy),   
[mahmud108974@gmail.com](mailto:mahmud108974@gmail.com) [(M. Hasan),](mailto:saif@eece.mist.ac.bd) [abtahiishmam3@](mailto:abtahiishmam3@gmail.com)[gmail.com (A. Ishmam),](mailto:h.nyeem@eece.mist.ac.bd) [m](mailto:h.nyeem@eece.mist.ac.bd)[ohammadaxif](mailto:mohammadaxif5717@gmail.com)[5717@gmail.com (M.A.H](mailto:rkoushikroy2@gmail.com). Akif), [ovitareque@gmail.com](mailto:ovitareque@gmail.com) [(T.B. Ovi).](mailto:mahmud108974@gmail.com)

<https://doi.org/10.1016/j.array.2022.100244>  
[Received 18 March 2022; Received in revise](https://doi.org/10.1016/j.array.2022.100244)d form 8 August 2022; Accepted 11 August 2022 Available online 17 August 2022

2590-0056/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

*M.S.H. Onim et al.*  *Array 15 (2022) 100244*

annotation [8], and transfer learning [9]. Other development with recurrent neural network (RNN) based architectures include the Bidi-rectional Long short-term memory (BiLSTM) based models [10], and real-time object detection algorithm, YOLO (You Only Look Once) and its variants-based models [11–13]. The end-to-end cascaded and unified architectures of the neural networks were also studied for the ALPR systems [14–17].

However, Unlike the English VLP, the effort in developing the ALPR systems for the Bengali VLP is particularly limited. Despite much interest in developing Bengali handwriting, scripts and character recognition in general, it has not captured the ALPR system yet. A few notable developments of the ALPR systems for the Bengali VLP include the feature extraction based on the digital curvelet transform [18], Tesseract OCR [19], and CNN with Adam optimizer [20].

In summary, no Deep Neural Networks (DNN) model can be tracked in the literature that can detect the Bengali VLP and recognize its characters simultaneously. A few models only focused on the Bengali character recognition [19,20], but their performance is unknown for ALPR system. Although Tesseract and BornoNet have relatively high character-recognition accuracy, they are not suitable for real-time ap-plications like ALPR due to higher processing time. Unlike the above models, Onim et al. [17] recently combined YoloV4 for detection of VLP, and Tesseract as OCR engine. However, the model requires reason-ably higher time for number plate detection and character recognition. All these mean that the models developed for either Bengali VLP detection or Bengali character recognition are generally limited with *low detection and recognition accuracy*, and *high computational complexity,* making them unsuitable for a real-time application.

In this paper, we, therefore, report an ALPR system with a new DNN model that we call Bengali License Plate Network (BLPnet) (Section 3). BLPnet is constructed to have three primary phases to significantly reduce the computational cost and false-positives making the system faster and more accurate by defining vehicle regions detection (in the first phase) before VLP recognition (in the second phase). Particularly, the contributions with the proposed three-phase ALPR system can be summarized as follows:

• The NASNet-Mobile backbone architecture is redefined in the first phase with more *dense* and *pulling* layers on the network-head to identify the vehicles more efficiently with a region of interest bounding-box (Section 3.1).

• An InceptionV3 architecture is customized in the second phase with new *dense* and *pulling* layers for a more accurate and faster detection of the VLP in the bounding-box region (Section 3.2).• Finally, a new Bengali-OCR engine (Section 3.3) is introduced in the third phase. Unlike the existing ALPR systems that cannot adequately tackle the artefacts like motion-blur and non-uniform shadow, the proposed engine employs de-blurring and Region Scalable Fitting (RSF) based segmentation, which are optionally invoked (*i.e.*, when characters are not recognized) to better tackle the intensity inhomogeneity.

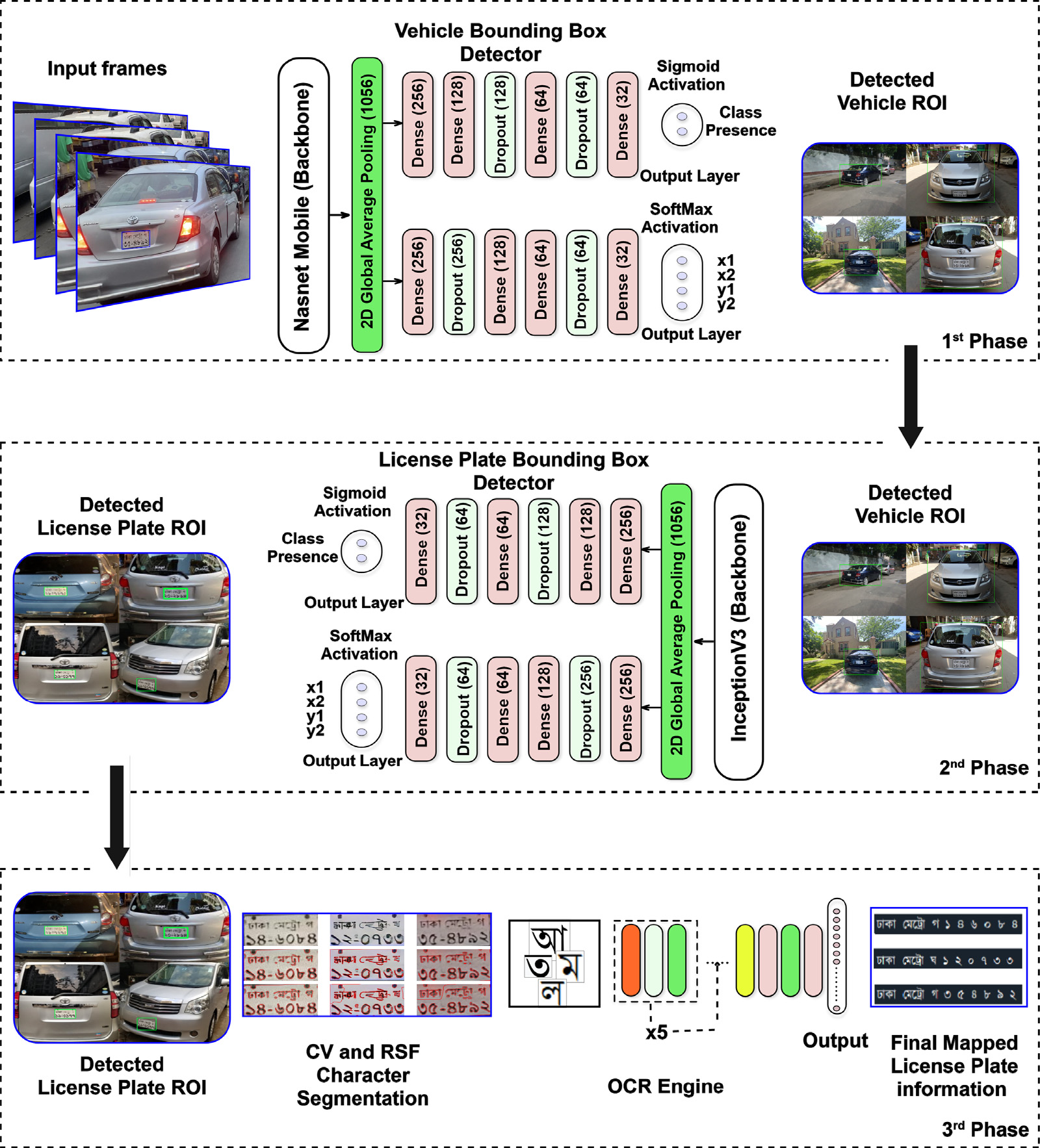
**2. Related ALPR systems**

Development of several prominent ALPR systems for English VLP can be tracked in the literature so far. Getting the VLP information requires the localization and detection of the VLP. In CNN-based ap-proach, features are extracted for VLP and then object localization is done with bounding box detectors and regression algorithms [4,6]. Silva et al. [7] and Laroca et al. [5] used bounding box regression to localize the object of interest with image coordinates using YOLO. These are One-stage detection networks and relatively faster than other detectors. But this approach is computationally inefficient as it requires training *darknet* backbone of over 27*𝑀* parameters.

Alternatively, VLP can also be semantically segmented using either a CNN or deep segmentation network. Bulan et al. [3] proposed a real-time complete CNN model having higher adaptability to tackle the

2

*M.S.H. Onim et al.*  *Array 15 (2022) 100244*

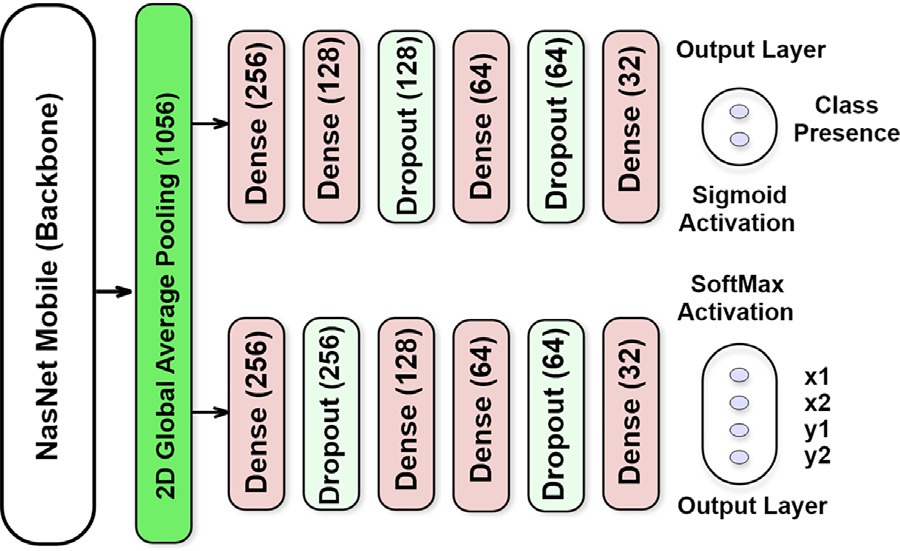


**Fig. 1.** Key processes of the proposed ALPR system.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| are fed to the VLP detection model along with the coordinates of the bounding-box. Later, the second phase detects and extracts the VLP contour, followed by the character localization and recognition in the final phase. Consideration of intensity-inhomogeneity resilient seg-mentation and effective deployment of these cascaded modules would reduce the computational cost and false-positive predictions, improving the accuracy and reliability of the system for a real-time application. These phases are defined in the Algorithm 1 and discussed below, with necessary mathematical modeling and more technical details. |  | balance hardware resources and processing speed. The TensorFlow (TF) v2.4 framework was used for the overall construction of our CNN model.  Unlike the YOLO algorithm [22] which calculates both class prob-ability and bounding box coordinates, the final layer of our proposed model only calculates the bounding box coordinates. The softmax acti-vation function defined in Eq. (1) helps find the normalized coordinates as an offset of the current grid cell. | | | | | | |
| *𝜎*(*𝑧*)*𝑖* = *𝑒𝑧 𝑖*  (1)  Here *𝑍* is the input vector indexed with *𝑖*,∑*𝑘* term and *𝑘* is the number of classes in the multiclass classifier. The backbone was pre-trained with the Imagenet dataset [23].  ∑*𝑘 𝑗*=1*𝑒𝑧*  *𝑗*=1*𝑒𝑧 𝑗*is the normalizing  During the training of the Bounding Box detector, the following loss function *𝜙* was optimized. The error in the coordinate position was formulated as the sum of squares, and the error in the box width and height was defined with the root of the sum of squares. | | | | | | |
| *3.1. An extended vehicle bounding-box detector* | 3 |
| A NASNet-Mobile is extended and used as the backbone of our DNN, followed by six hidden layers to improve the accuracy of vehicle detection (see Fig. 2). The NASNet-Mobile is more computationally effi-cient than its counterpart like the ResNet-based architecture that needs higher hardware configuration, making it unsuitable for large-scale deployment [21]. Thus, our model can be implemented on Field-Programmable Gate Array (FPGA) and other embedded devices to |
| *𝜙* =*𝜆𝑐𝑜𝑜𝑟𝑑* | *𝑆*2 ∑ | ∑ | P*𝑜𝑏𝑗 𝑖𝑗* | [(*𝑥𝑖* − *̂𝑥𝑖* | )2 + (*𝑦𝑖* − *̂𝑦𝑖* | )2] |

*M.S.H. Onim et al.*  *Array 15 (2022) 100244*

|  |
| --- |
| **Algorithm 1** 𝚝𝚛𝚊𝚒𝚗\_𝙱𝙻𝙿𝙽𝚎𝚝 (⋅) |
| 1: **Input:** *Dataset,* 𝙱𝙻𝙿𝙽𝚎𝚝 (**𝐖***,* **𝐛**)  2: **Output:** 𝚝𝚛𝚊𝚒𝚗𝚎𝚍\_𝙱𝙻𝙿𝙽𝚎𝚝 (⋅)  3: **Initialize:**  **𝐖** = 𝚛𝚊𝚗𝚍𝚘𝚖 (⋅)  **𝐛** = 𝚛𝚊𝚗𝚍𝚘𝚖 (⋅)   *𝐿𝑟* = 0.0001   *𝑒𝑝𝑜𝑐ℎ𝑠* = 300  4: [**𝐗**′*𝑡𝑟𝑎𝑖𝑛,* **𝐗**′*𝑡𝑒𝑠𝑡*] ← 𝚝𝚛𝚊𝚒𝚗\_𝚝𝚎𝚜𝚝\_𝚜𝚙𝚕𝚒𝚝 (*𝐷𝑎𝑡𝑎𝑠𝑒𝑡*) 5: **for all** *𝑥* ∈ **𝐗**′*𝑡𝑟𝑎𝑖𝑛***do**  10:  11: **end for**  12: **𝐗***𝑡𝑟𝑎𝑖𝑛* ← [**𝐗***𝑎* 13: **while** *𝑒𝑝𝑜𝑐ℎ* ≤ *𝑒𝑝𝑜𝑐ℎ𝑠* **do**  6:  7:  8:  9: **𝐗***𝑎*  **𝐗***𝑏*  **𝐗***𝑐*  **𝐗***𝑑*  **𝐗***𝑒 𝑡𝑟𝑎𝑖𝑛*← 𝚛𝚊𝚗𝚍𝚘𝚖\_𝚛𝚘𝚝𝚊𝚝𝚎 (**𝐗**′*𝑡𝑟𝑎𝑖𝑛*← 𝚛𝚊𝚗𝚍𝚘𝚖\_𝚜𝚑𝚒𝚏𝚝 (**𝐗***𝑎 𝑡𝑟𝑎𝑖𝑛*← 𝚛𝚊𝚗𝚍𝚘𝚖\_𝚏𝚕𝚒𝚙 (**𝐗***𝑏 𝑡𝑟𝑎𝑖𝑛*← 𝚛𝚊𝚗𝚍𝚘𝚖\_𝚌𝚘𝚗𝚝𝚛𝚊𝚜𝚝 (**𝐗***𝑐 𝑡𝑟𝑎𝑖𝑛*← 𝚛𝚊𝚗𝚍𝚘𝚖\_𝚋𝚕𝚞𝚛 (**𝐗***𝑑 𝑡𝑟𝑎𝑖𝑛,* **𝐗***𝑏 𝑡𝑟𝑎𝑖𝑛,* **𝐗***𝑐 𝑡𝑟𝑎𝑖𝑛,* **𝐗***𝑑 𝑡𝑟𝑎𝑖𝑛 𝑡𝑟𝑎𝑖𝑛*  *𝑡𝑟𝑎𝑖𝑛,* **𝐗***𝑒*  ) ) )  *𝑡𝑟𝑎𝑖𝑛*  )  )  *𝑡𝑟𝑎𝑖𝑛*]  14: Perform forward propagation with Eq. (3) 15: Calculate cost for FP from Eq. (2)  16: Perform backpropagation with Eq. (4) 17: Optimize cost function with Eq. (5)  19: **end while** 20: 𝚜𝚊𝚟𝚎 (*𝑡𝑟𝑎𝑖𝑛𝑒𝑑*\_*𝐵𝐿𝑃 𝑁𝑒𝑡*) [**𝐖**′*,* **𝐛**′] ← 𝚐𝚛𝚊𝚍𝚒𝚎𝚗𝚝\_𝚍𝚎𝚜𝚌𝚎𝚗𝚝 (**𝐖***,* **𝐛***,* **𝐗***𝑡𝑟𝑎𝑖𝑛, 𝜙*) 18: |



**Fig. 2.** Vehicle Bounding-box detector of the proposed ALPR system.

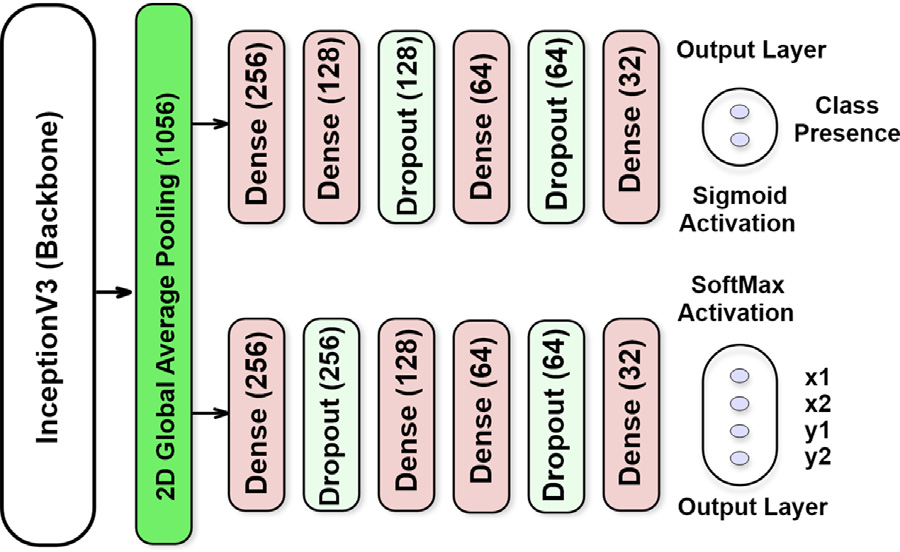
|  |
| --- |
| *𝑆*2  + *𝜆𝑐𝑜𝑜𝑟𝑑*∑∑P*𝑜𝑏𝑗 𝑖𝑗*[(√*𝑤𝑖* −√*̂𝑤𝑖*)2  + (√*ℎ𝑖* −√*̂ℎ𝑖*)2] (2)  Here *𝜆𝑐𝑜𝑜𝑟𝑑* is a random weight, P*𝑜𝑏𝑗* is the *𝑠𝑖𝑔𝑚𝑜𝑖𝑑* output from vehicle detection. *̂𝑥, ̂𝑦, ̂𝑤, ̂ℎ* are the predicted value and *𝑥, 𝑦, 𝑤, ℎ* are the ground truth. The P*𝑜𝑏𝑗 𝑖𝑗*  term is solely responsible for the false positive detection. The number of times P*𝑜𝑏𝑗 𝑖𝑗*equals 0 is the number of false positive for that instance. A few important processes and considerations of bounding box detector are now briefly discussed below. |

*3.1.1. Dataset collection*

For the training of the vehicle bounding-box detection model, we used the Cars dataset [24] of Stanford AI Lab. There are 16,185 images in the dataset, representing 196 different car classifications. The data is divided into 8144 training images and 8041 testing images, with about a 50-50 ratio between the two classes. Typically, classes are organized by Make, Model, and Year, such as 2012-Tesla-Model-S or 2012-BMW-M3-coupe. Each image in the dataset has also a bounding-box output value as an ideal reference that suits our requirement.

4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *M.S.H. Onim et al.* | | | Onim et al. [17] | |  |  | *Array 15 (2022) 100244* |
| **Table 1**  Comparison of the training hyper-parameters. | | |
| Model | Ours (BLPnet) | | Zou et al. [10] | Hendry &  Chen [11] | Li et al. [25] |
| Parameters  Backbone | Vehicle  NASNet  Mobile  2  32  300 × 300 SGD  300 | VLP  InceptionV3 | Vehicle  Darknet | VLP  Darknet | VLP  MobileNetV3 | VLP  Darknet | VLP – |
| Classes  Batch size  Image shape  Optimizer  Epochs | 2  64  200 × 200 Adam  30 | 4  32  224 × 224 Adam  100 | 2  64  128 × 128 Adam  3500 | 37 – 152 × 56  SGD – | 60 – 256 × 256 Adam  600 | 37  32  24 × 24  Adam – |



**Fig. 3.** Licence Plate Bounding-box detector of the proposed ALPR system.

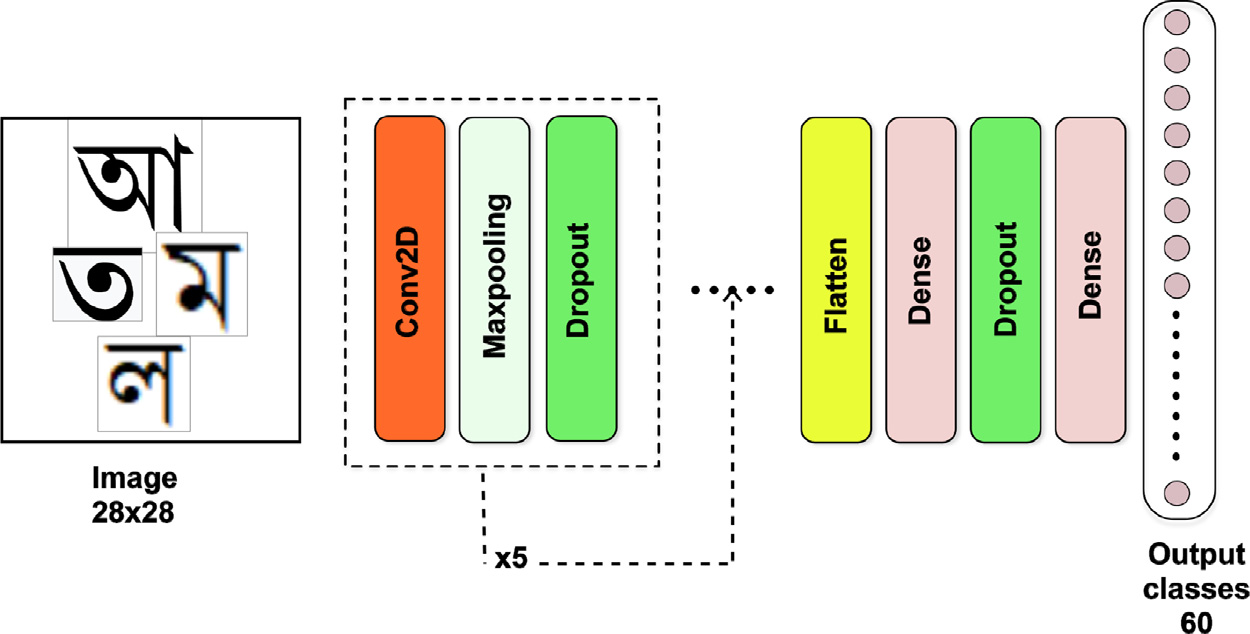
|  |
| --- |
| *𝛿𝑏𝑚,𝑛 𝛿𝜙*  = *𝑓*1−1 ∑ *𝑓*2−1 ∑*𝛿𝑋𝑖,𝑗*  *𝛿𝜙* ×*𝛿𝑋𝑖,𝑗*  *𝛿𝑏𝑚,𝑛*  = *𝑓*1−1 ∑ *𝑓*2−1 ∑*𝛥𝑙𝑜𝑐𝑎𝑙 𝑋* ×*𝛿𝑋𝑖,𝑗*  *𝛿𝑏𝑚,𝑛*   (4b)  Adam optimizer is defined in Eq. (5), where *𝜔* is the set of model weights, *𝜂* is the step size,*̂𝑚* and *̂𝑣* are first and second moment respectively and *𝜖* is a small value to avoid dividing by zero.  *𝜔𝑡*+1 = *𝜔𝑡* − √1−*𝛽𝑡*  *𝑣𝑡*  *𝜂*  2   + *𝜖* × 1 − *𝛽𝑡*   *𝑚𝑡*  1   (5)  We have used an annotated dataset of 1500 training and 300 validation images to train our model. From real-world video footage taken on roadways in Dhaka, Bangladesh, with varying road conditions, includ-ing high traffic congestion, the VLP were detected with reasonably higher accuracy (see Section 4). The training began with some itera-tions set at 6000. The average loss did not diminish significantly after 2000 iterations, and after every 1000 cycles, the training was capable of taking the weight back up. To reduce training duration, we thus used early stopping. |

*3.3. A new Bengali-OCR engine*

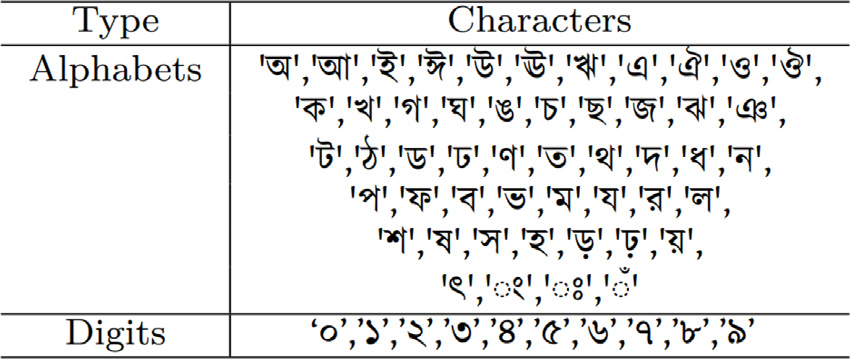
This step recognized characters in the extracted VLP from the previous step. Similar to [27], we considered the character-recognition as an object-recognition problem. By treating characters as objects, the character segmentation and recognition steps are integrated and consist of a total of 5 convolutional 2D layers with 16, 32, 64, 128 and 512 nodes. Each of them includes a preceding max-pooling layer with a pool size of 2 and a dropout layer with a drop rate of 20%. The dropout layers were used to prevent the model from over-fitting. The kernel size of 2 is considered with necessary padding. Relu activation function is used for all the backbone layers. Finally, a global-average pooling layer is included before the output layer, followed by two dense layers. Two important considerations of this model are the conditional use of de-blurring filter and intensity inhomogeneity invariant segmentation of the character.

5

*M.S.H. Onim et al.*  *Array 15 (2022) 100244*



**Fig. 4.** The newly developed OCR engine for BLPnet.



**Fig. 5.** Bengali character classes for the OCR training.

models: Li et al. [25], Hendry & Chen [11], Zou et al. [10], and Onim et al. [17]. This section starts with the analysis of the model complexity followed by the results are discussed in three main parts, namely, for vehicle detection, VLP detection and finally, for the OCR and word-mapping.

*4.1. Model complexity: Trainable parameters*

The complexity of a model is measured by the number of trainable parameters, *i.e.*, the number of neurons and the number and weights of connections between neurons from layer to layer. A dense layer with *m* input-nodes and *n* output-nodes will have a total of (*𝑛* + 1) × *𝑚* learnable parameters. The pooling layers and dropout layers do not learn anything. For convolutional layers with *p* feature maps in input and *q* feature maps as output, having a filter size of *𝑖*×*𝑗* will have total learnable parameters of (*𝑖* × *𝑗* × *𝑝* + 1) × *𝑞*.

The distribution of trainable parameters for the bounding box detec-tor across the network is noted in Table 2. The total number of trainable parameters is calculated using Eq. (8). Similarly, Table 3 shows the trainable parameters for our OCR engine. Here *Layers* denotes the name and operation of added layers, *Shape* denotes the input shape of tensors for that particular layer and finally the trainable parameter for that layer.

*𝑁𝑑𝑒𝑡𝑒𝑐𝑡𝑜𝑟* = 2 × ((1056 + 1) × 256

|  |  |
| --- | --- |
| + (256 + 1) × 128 + (128 + 1) × 64 + (64 + 1) × 32 + (32 + 1) × 2) | (8) |

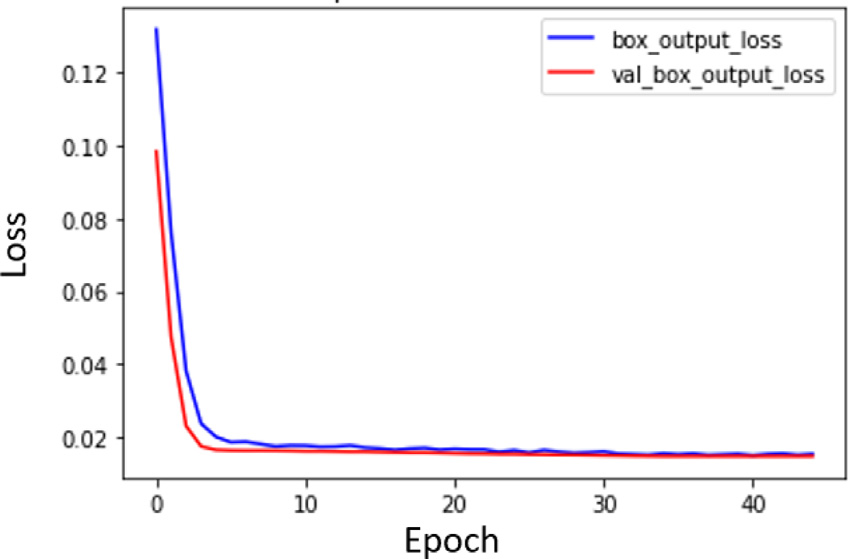
= 627*𝑘*

*4.2. Vehicle detection*

We trained the vehicle bounding-box model for vehicle detection for 1000 epochs with early stopping employed to stop the model training if the accuracy improvement is negligible. The hardware specifications

6

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *M.S.H. Onim et al.* | | | |  |  |  |  |  | *Array 15 (2022) 100244* |
| **Table 4**  Performance of vehicle detection with bounding-box. | | | |
| Box | | | Class | Box | Class | Validation loss | Validation box  Output  loss | Validation class  Output  loss | Validation  box  Output  MSE |
| Epochs | Loss | Output loss | Output  loss | Output MSE | Output  accuracy |
| 50  100  150  200  250  300  350  400 | 0.0056  0.0056  0.0056  0.00569 0.00569 0.00569 0.00569 0.00582 | 0.0176 0.0171 0.0169 0.0173 0.0176 0.0176 0.0176 0.0130 | 0.0059  0.0058  0.0058  0.0057  0.0056  0.0053  0.0052  0.0050 | 0.0176 0.0171 0.0169 0.0173 0.0176 0.0176 0.0176 0.0155 | 75.59  85.89  89.19  90.55  94.74  96.99  97.02  97.05 | 0.0057  0.0056  0.0055  0.0055  0.0054  0.0055  0.0055  0.0056 | 0.0175  0.0140  0.0138  0.0175  0.0175  0.0175  0.0175  0.0148 | 0.0057  0.0057  0.0056  0.0055  0.0055  0.0054  0.0054  0.0058 | 0.0175  0.0140  0.0138  0.0175  0.0175  0.0175  0.0175  0.0152 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 5**  OCR performance for the CV and RSF-based segmented characters. | | | |  |
| Segmentation model | No. of  characters extracted | Accuracy of OCR  (in %) | Time taken for OCR  (in seconds) | Time taken  for Tesseract (in seconds) |
| CV model | 4  5  6  8 | 90  83  81  80 | 0.302  0.395  0.432  0.502 | 0.402  0.548  0.701  0.705 |
| RSF model | 4  5  6  8 | 95  93  90  89 | 0.256  0.312  0.333  0.398 | 0.402  0.548  0.701  0.705 |

**Fig. 6.** Loss vs Epoch curve for the first 50 epochs. *4.4. Character recognition*

Fig. 6, we observe that the model can quickly converge to a lower loss value. After a while, the improvement gets very slow. The best training loss and MSE of the bounding-box were 0.0130 and 0.0155, respectively, and the validation loss and MSE were 0.0148 and 0.0152, respectively. Overall training parameters with their respective epochs are presented in Table 4. We observe in this table that after 300 epochs, the rate of change in loss and accuracy is becoming saturated which indicates the completion of model fitting. Compared to the YOLO model, the hyper-parameters were kept similar as shown earlier in Table 1. All these observations suggest that our proposed model would effectively detect the vehicles in the input video clips. The final predicted output of the model applied in random vehicles is shown in Fig. 7(a) with the successful demarcation of green boxes.

*4.3. VLP detection*

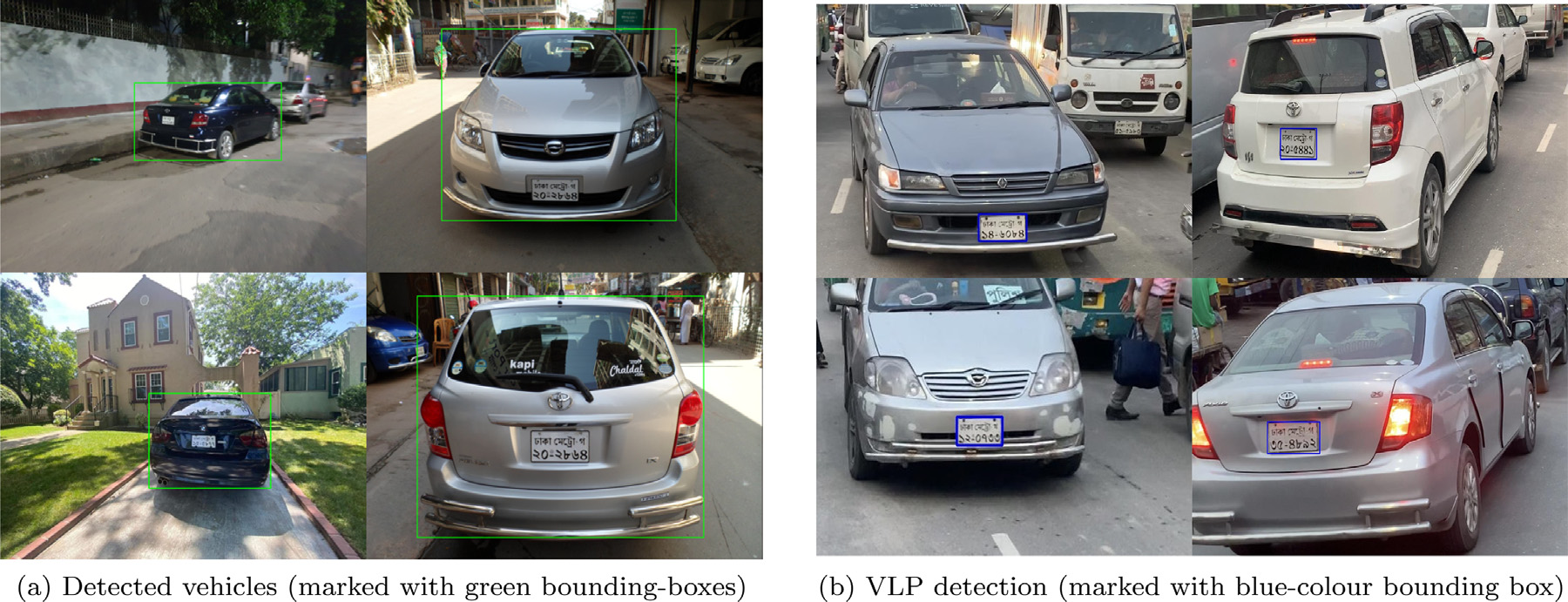
Our model was evaluated using both real-time and pre-recorded video clips. Table 1 shows the hyper-parameters used to train the network. Our model did not over-fit due to the use of adequate *dropout* and *pooling* layers, and thus, it took little time to train and converge to an optimum detection level. During the evaluation, the performance of the model was monitored in real-time.

Our algorithm successfully detected VLP as a few examples are illustrated in Fig. 7(b), where VLP is marked with a blue bounding box that was generated from the x-y coordinates and height-width values. A VLP was detected from the varying orientation of the detected vehicle and at that time, 17 frames per second processing speed were maintained on average. Such accurate detection was observed for all the testing video clips with resolution 1920 × 1080 and frames per second (*fps*) of 15 and 20.

Similar to the vehicle detection with the bounding box (Section 3.1), the VLP detection training loss and MSE of the bounding-box were 0.013 and 0.016, respectively and the validation loss and MSE were 0.015 and 0.015, respectively. No false-positive detection was recorded during the process, which justifies the effectiveness of the previous phase.

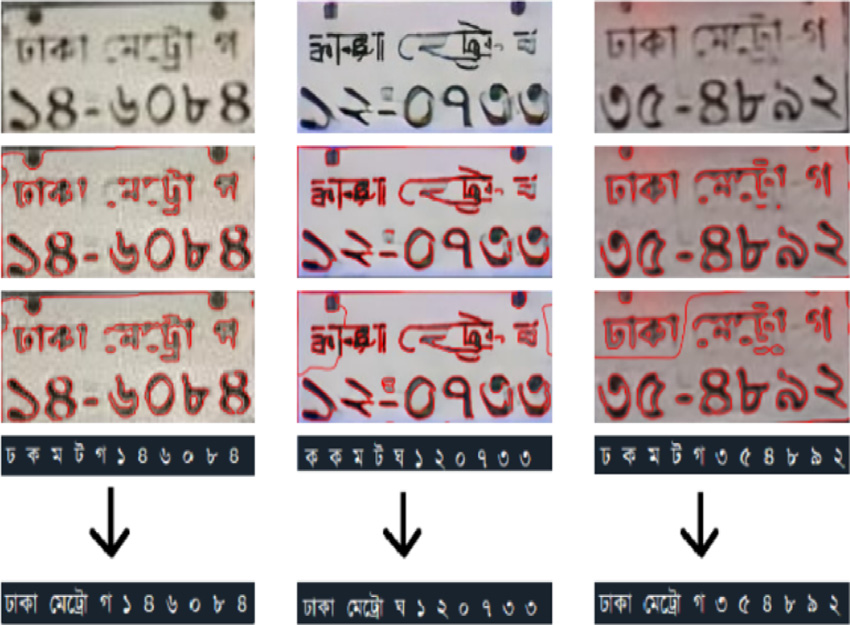
7

*M.S.H. Onim et al.*  *Array 15 (2022) 100244*



**Fig. 7.** Example of detected vehicles and VLP from real-time captured video footage with different orientations of the vehicles.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 6**  Comparison of features of the ALPR systems. | | |  |  |  |  |  |  |
| Model | Testing  accuracy | Dataset | VLP  False-Positive detection | Characters’rotation  invariant? | Trainable  parameters (*Million*) | Processing time  (s) | Recognizing  object | Hardware |
| Hendry & chen [11] | 78.2% | AOLP | – | No | 8.8 | 0.8–1.0 | VLP and  characters  (English) | Core i7  Nvidia GTX  970 4 GB GPU |
| Li et al. [25] | 94.85% 94.19% 88.38% | AOLP-AC AOLP-LE AOLP-RP | Heuristically minimized  with CNN | No | 1.0 | 2.0–3.0 | VLP and  characters  (English) | Core i5  Nvidia Tesla  k40c 4 GB GPU |
| Zou et al. [10] | 79.5% | AOLP,  PKU &  PLPD | – | Yes | 1.9 | – | Blurry text  recognition  (English) | Core i5  Nvidia Titan  12 GB GPU |
| Onim et al. [17] | 90.51% | Custom | Eliminated using 2-phase detection | No | 27 | 0.7–1.0 | VLP (English) | Core i5  Nvidia Tesla  T4 6 GB GPU |
| **BLPnet**  **(Ours)** | **95.0%** | **Cars**  (**Stanford AI Lab**) | **Eliminated using 2**-**phase detection** | **Yes** | **0.97** | **0.32–0.52** | **Real-time VLP & characters**  **(English)** | **Core i5**  **Nvidia Tesla**  **k80 6 GB GPU** |



**Fig. 8.** Example of input and outputs of the newly developed OCR engine (From *top*,

input images in *first row*, CV-segmented images in *second row*, RSF-segmented images

in *third row*, and OCRs and word-mapping in *fourth row*).

is observed to be more effective with lower-computational time and accuracy than the prominent ALPR systems.

Moreover, the model generates the actual license number of the vehicle from the recognized characters using our simple, yet effective mapping algorithm with a set of predefined cases of registration area-codes. The model also performed well without compromising accuracy to tackle challenging conditions that cause rotated, blurry or noisy

8

*M.S.H. Onim et al.*  *Array 15 (2022) 100244*

[3] [Bulan O, Kozitsky V, Ramesh P, Shreve M. Segmentation-and annotation-free license plate recognition with deep localization and failure identification. IEEE Trans Intell Transp Syst 2017;18(9):2351–63.](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb3)

[4] [Xu Z, Yang W, Meng A, Lu N, Huang H, Ying](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb3) [C, et al. Towards end-to-end license plate detection and recognition: A large dataset and baseline. In: Computer vision– ECCV 2018. Cham: Springer International Publishing; 2018, p. 261–77.](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb4)

[5] [Laroca R, Severo E, Zanlorensi LA, Oliveira LS, Gonçalves GR, Schwartz](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb4) WR, et al. A robust real-time automatic license plate recognition based on the YOLO detector. In: International joint conference on neural networks. 2018, p. 1–10. <http://dx.doi.org/10.1109/IJCNN.2018.8489629>.

[6] [Wang W, Yang J, Chen M, Wang P. A light cnn](http://dx.doi.org/10.1109/IJCNN.2018.8489629) for end-to-end car license plates detection and recognition. IEEE Access 2019;7:173875–83. [http://dx.doi.org/10. 1109/ACCESS.2019.2956357](http://dx.doi.org/10.1109/ACCESS.2019.2956357).

[7] [Silva SM, Jung CR. A flexib](http://dx.doi.org/10.1109/ACCESS.2019.2956357)le approach for automatic license plate recognition in unconstrained scenarios. IEEE Trans Intell Transp Syst 2021;1–11. [http://dx. doi.org/10.1109/TITS.2021.3055946](http://dx.doi.org/10.1109/TITS.2021.3055946).

[8] [Zhuang J, Hou S, Wang Z, Zha](http://dx.doi.org/10.1109/TITS.2021.3055946) Z-J. Towards human-level license plate recognition. In: Proceedings of the european conference on computer vision. 2018.

[9] Huang Z, Pan Z, Lei B. Transfer learning with deep convolutional neural network for sar target classification with limited labeled data. Remote Sens 2017;9(9):907. <http://dx.doi.org/10.3390/rs9090907>.

[10] [Zou Y, Zhang Y, Yan J, Jiang X, Huang T, Fan H, et al. A robust license plate](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb10)  [recognition model based on bi-lstm. IEEE Access 2020;8:211630–41.](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb10)

[11] [Hendry, Chen R-C. Automatic license plate recognition via slid](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb10)ing-window darknet-YOLO deep learning. Image Vis Comput 2019;87:47–56. [http://dx.doi. org/10.1016/j.imavis.2019.04.007](http://dx.doi.org/10.1016/j.imavis.2019.04.007).

[12] [Al-Shemarry MS, Li Y, Abdulla S.](http://dx.doi.org/10.1016/j.imavis.2019.04.007) [An efficient texture descriptor for the detection of license plates from vehicle images in difficult conditions. IEEE Trans Intell Transp Syst 2019;21(2):553–64.](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb12)

[13] [Bochkovskiy A, Wang C, Liao](http://refhub.elsevier.com/S2590-0056(22)00078-9/sb12) HM. YOLOv4: Optimal speed and accuracy of object detection. 2020, CoRR abs/2004.10934. [arXiv:2004.10934](http://arxiv.org/abs/2004.10934).

[14] Hsu G-S, Chen J-C, Chung Y-Z. Application-or[iented license pla](http://arxiv.org/abs/2004.10934)te recognition.

IEEE Trans Veh Technol 2013;62(2):552–61. [http://dx.doi.org/10.1109/TVT. 2012.2226218](http://dx.doi.org/10.1109/TVT.2012.2226218).

[15] Montazzolli S, Jung C. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In: 2017 30th SIBGRAPI conference on graphics, patterns and images. 2017, p. 55–62. [http://dx.doi.org/10.1109/](http://dx.doi.org/10.1109/SIBGRAPI.2017.14)  [SIBGRAPI.2017.14](http://dx.doi.org/10.1109/SIBGRAPI.2017.14).

9