Array 15 (2022) 100244

Contents lists available at ScienceDirect

Array

journal homepage: 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

ARTICLE INFO

Keywords:
ALPR
CNN
INCEPTION-V3
License plate detection
NASNet Mobile
OCR

ABSTRACT

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 input 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 example, the Convolutional Neural Network (CNN) based bounding box detectors were developed with regression algorithms [3–7], manual

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

https://doi.org/10.1016/j.array.2022.100244
Received 18 March 2022; Received in revised form 8 August 2022; Accepted 11 August 2022
Available online 17 August 2022

annotation [8], and transfer learning [9]. Other development with recurrent neural network (RNN) based architectures include the Bidirectional 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 applications 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 reasonably 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 approach, 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 $27M$ 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 scene-variations requiring minimum possible human-intervention. An image-based classification was proposed with weak-Snow and strong-CNN classifiers. The first classifier was initially used for the region-tagging of the possible number-plates, and the strong-CNN refined the classification outputs for the actual number-plates Zhuang et al. [8] proposed such a method where they segmented the VLP and the characters in it. This approach also needs a manual annotation of characters and LP. Besides these segmentation-based networks, segmentation-free Networks have also been proposed. Wang et al. [6] proposed a Convolutional Recurrent Neural Network (CRNN) and a multitask-license-plate detection and recognition (MTLPR) model. Zou et al. [10] used Bi-LSTM and directly localized characters without segmentation.

Both end-to-end cascaded and unified architectures have been used for the ALPR system. Li et al. [16] used end-to-end trainable LP detection model that remains error-prone to challenging conditions and also depends heavily on architectural design. On the other hand, cascaded architectures have the flexibility of tuning and testing. Hsu et al. [14] and Montazzolli et al. [15] proposed a cascaded architecture for real time VLP detection and recognition. Huang et al. [9] introduced Transfer learning into their CNN model to tackle the limited labeled data for a high detection accuracy of VLP of 93%. Later, with a modified YOLOv3 model with 10 CNNs, Chen et al. [11] attempted to detect and recognize the VLP with an accuracy of about 98% and 78%, respectively in a variety of conditions (*i.e.*, rainy backgrounds, darkness and dimness, and varied colors and saturation of photos).

In addressing several other environmental artefacts like poor-contrast and noisy images resulting from foggy, distortion and dusty conditions, Al-Shemarry et al. [12] improved the accuracy of VLP detection using a binary descriptor with contrast enhancement, leaving their proposed system limited to real-time application with poor frame-rate and higher computational cost. To tackle the motion artefacts (*i.e.*, blurry input images), Zou et al. [10] recently proposed a Bi-LSTM model with robust blur-kernel estimation that offered a reasonably lower accuracy of 79.55%. Another recent development with object bounding-box detection for ALPR system is YOLOv4 model [13], which is a deep convolutional network that can localize VLPs with immense frame rate and high accuracy.

Besides, on Bengali character detection and recognition, Majumdar et al. [18] proposed a feature extraction based on the digital curvelet transform. Separate k-nearest neighbor classifiers were trained using the curvelet coefficients of an original image and its morphologically changed copies. The overall recognition accuracy of 96.8% was reported while trained for twenty popular Bengali fonts and tested for different font sizes. Later, Hasnat et al. [19] attempted to introduce Bengali script in Tesseract OCR. Tesseract can learn new languages or alphabets with training, along with existing languages for character recognition.

Recently, Rabby et al. [20] presented a new 13-layer CNN model called BornoNet for Bengali character recognition. The model was designed with two sub-layers optimized using the Adam optimizer Despite having relatively higher accuracy in the models for Bengali character recognition, their applications to an ALPR system is not yet studied. In addressing the potential gap of having a promising ALPR system for Bengali VLP, we, therefore, develop and present a real-time End-to-End DNN based model that we call BLPnet.

3. A new ALPR system with BLPnet

This section presents a newly developed BLPnet model for the ALPR system. As illustrated in Fig. 1, the model consists of three main phases that are cascaded together to detect the vehicle, VLP and recognize the characters in it. It takes real-time video frames as input consisting of vehicles and their surroundings. In the first phase, the vehicle region is detected and distinguished using a bounding-box as the smallest possible rectangle with vertical and horizontal sides that surround the vehicle captured in a frame. The frames containing the bounding boxes

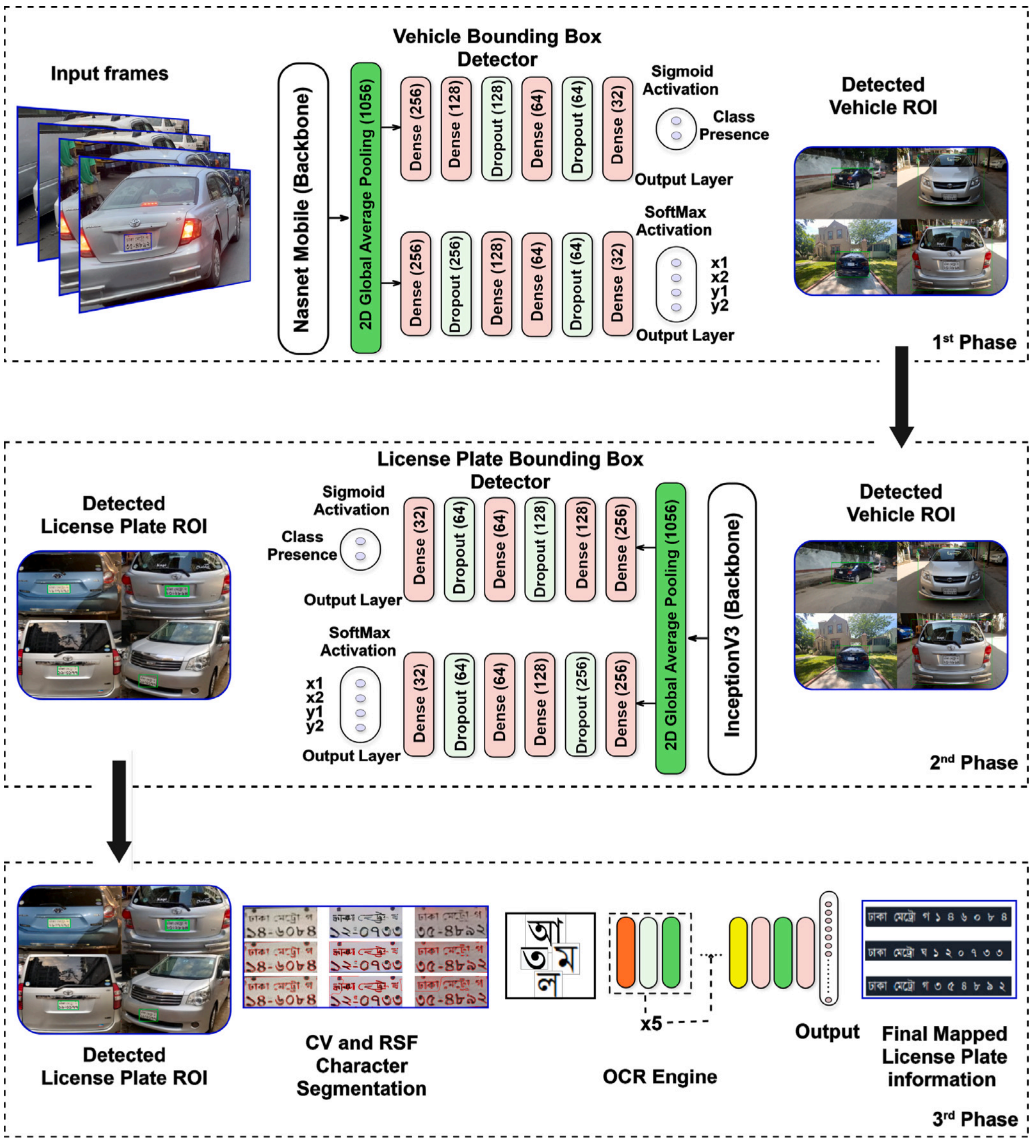


Fig. 1. Key processes of the proposed ALPR system.

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 segmentation 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.

3.1. An extended vehicle bounding-box detector

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 efficient 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 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 probability and bounding box coordinates, the final layer of our proposed model only calculates the bounding box coordinates. The softmax activation function defined in Eq. (1) helps find the normalized coordinates as an offset of the current grid cell.

$$\sigma(z)_i = \frac{e_i^z}{\sum_{j=1}^{k} e_j^z} \tag{1}$$

Here Z is the input vector indexed with i, $\sum_{j=1}^{k} e_j^z$ is the normalizing term and k is the number of classes in the multiclass classifier. The backbone was pre-trained with the Imagenet dataset [23].

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.

$$\phi = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{P}_{ij}^{obj} \left[\left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2 \right]$$

Algorithm 1 train_BLPNet(·)

```
1: Input: Dataset, BLPNet $(\mathbf{W}, \mathbf{b})$
2: Output: trained_BLPNet(·)
3: Initialize:
     $\mathbf{W}$ = random(·)
     $\mathbf{b}$ = random(·)
     $Lr = 0.0001$
     $epochs = 300$
4: $[\mathbf{X}'_{train}, \mathbf{X}'_{test}]$ ← train_test_split(Dataset)
5: for all $x \in \mathbf{X}'_{train}$ do
6:     $\mathbf{X}^a_{train}$ ← random_rotate $(\mathbf{X}'_{train})$
7:     $\mathbf{X}^b_{train}$ ← random_shift $(\mathbf{X}^a_{train})$
8:     $\mathbf{X}^c_{train}$ ← random_flip $(\mathbf{X}^b_{train})$
9:     $\mathbf{X}^d_{train}$ ← random_contrast $(\mathbf{X}^c_{train})$
10:    $\mathbf{X}^e_{train}$ ← random_blur $(\mathbf{X}^d_{train})$
11: end for
12: $\mathbf{X}_{train} \leftarrow [\mathbf{X}^a_{train}, \mathbf{X}^b_{train}, \mathbf{X}^c_{train}, \mathbf{X}^d_{train}, \mathbf{X}^e_{train}]$
13: while $epoch \le epochs$ do
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)
18:    $[\mathbf{W}', \mathbf{b}']$ ← gradient_descent $(\mathbf{W}, \mathbf{b}, \mathbf{X}_{train}, \phi)$
19: end while
20: save(trained_BLPNet)
```

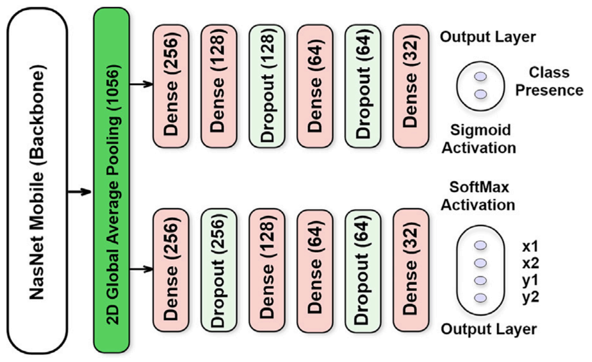


Fig. 2. Vehicle Bounding-box detector of the proposed ALPR system.

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{P}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{2}$$

Here λ_{coord} is a random weight, $\mathbb{P}_{ij}^{obj}$ is the *sigmoid* output from vehicle detection. $\hat{x}, \hat{y}, \hat{w}, \hat{h}$ are the predicted value and x, y, w, h are the ground truth. The $\mathbb{P}_{ij}^{obj}$ term is solely responsible for the false positive detection. The number of times $\mathbb{P}_{ij}^{obj}$ 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.

3.1.2. Pre-processing

We considered a total of 8041 training images and passed them through two preprocessing stages. At first, label-encoding and training-validation split are done with `train_test_split(·)` in Step 4 of Algorithm 1. In this case, we made an 80–20 split. After the splitting, image augmentation was done to simulate challenging conditions which the real-time traffic video might face. The applied augmentations include: rotate, shift, flip, varying contrast and blur with `random_rotate(·)`, `random_shift(·)`, `random_flip(·)`, `random_contrast(·)` and `random_blur(·)`. These augmentation processes are mentioned in Step 5–Step 12 of Algorithm 1. After augmentation, the model is trained and tuned.

3.1.3. Transfer learning (TL)

A pre-trained model (NASNet-Mobile) as the backbone architecture of the proposed model ensures optimized training. Unlike the isolated learning paradigm, transfer learning allows extending the pre-learned features to adequately address the new related challenges. Particularly, an average pooling layer usually can extract the pre-trained model's edge detection capability. From the final layer of shape $(10 \times 10 \times 1056)$, an average pooling layer is thus added, followed by a group of fully connected dense layers and dropout layers. Finally, the output layers identify the object and generate the bounding box coordinates. Thus, our model was pre-trained with over fourteen million images from the ImageNet dataset, which is now capable of categorizing images into over 1000 different classes with transfer learning ability. As a result, it has accumulated a library of rich feature sets for a wide variety of classes.

3.1.4. Hyper-parameters and tuning

The training hyperparameters are given in Table 1. Finally, the parameters with the best accuracy have been chosen after several trials and errors. The optimizer that suits best for the VLP detection is Stochastic gradient descent (SGD). We have also tweaked several parameters of the optimizer for our dataset and defined early stopping to reduce the training time. Additionally, Reduced Learning Rate functionality has been used to further expedite the training process.

3.2. VLP detection with Inception-v3

We have also extended the Inception-v3 DNN architecture and employed it as the backbone of the second phase process of BLPnet for VLP detection, as illustrated in Fig. 3. Inception-v3 is a CNN designed from the Inception family that transports label-information to a lower level of the network using label-smoothing, factorized 7 × 7 convolutions, an auxiliary classifier, and batch-normalization for layers in the side head. Inception architecture being 42 layers deep has 2.5 times lower computational cost than that of GoogLeNet and can function well even when memory and computational budgets were limited [26].

Forward propagation of training the neural network follows Eq. (3), where σ is the activation function, X, W, and b are the inputs, weights, and biases respectively and $\mathbb{O}$ is the output after the activation.

$$\mathbb{O}_{i,j} = \sigma \sum_i \sum_j \{W_{i,j} \times X_{i,j} + b_{i,j}\} \tag{3}$$

Backward propagation with gradient descent follows Eq. (4), where $\frac{\delta\phi}{\delta W_{m,n}}$ and $\frac{\delta\phi}{\delta b_{m,n}}$ are the derivatives of the loss function *w.r.t* weights and biases respectively and Δ is the local gradient.

$$\frac{\delta\phi}{\delta W_{m,n}} = \sum_{i=0}^{f1-1} \sum_{j=0}^{f2-1} \frac{\delta\phi}{\delta X_{i,j}} \times \frac{\delta X_{i,j}}{\delta W_{m,n}} = \sum_{i=0}^{f1-1} \sum_{j=0}^{f2-1} \Delta_X^{local} \times \frac{\delta X_{i,j}}{\delta W_{m,n}} \tag{4a}$$

Table 1
Comparison of the training hyper-parameters.

Model	Ours (BLPnet)		Onim et al. [17]		Zou et al. [10]	Hendry & Chen [11]	Li et al. [25]
Parameters	Vehicle	VLP	Vehicle	VLP	VLP	VLP	VLP
Backbone	NASNet Mobile	InceptionV3	Darknet	Darknet	MobileNetV3	Darknet	–
Classes	2	2	4	2	37	60	37
Batch size	32	64	32	64	–	–	32
Image shape	300 × 300	200 × 200	224 × 224	128 × 128	152 × 56	256 × 256	24 × 24
Optimizer	SGD	Adam	Adam	Adam	SGD	Adam	Adam
Epochs	300	30	100	3500	–	600	–

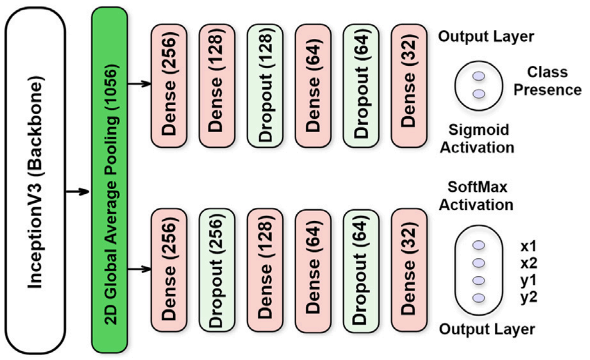


Fig. 3. Licence Plate Bounding-box detector of the proposed ALPR system.

$$\frac{\delta\phi}{\delta b_{m,n}} = \sum_{i=0}^{f1-1}\sum_{j=0}^{f2-1} \frac{\delta\phi}{\delta X_{i,j}} \times \frac{\delta X_{i,j}}{\delta b_{m,n}} = \sum_{i=0}^{f1-1}\sum_{j=0}^{f2-1} \Delta_X^{local} \times \frac{\delta X_{i,j}}{\delta b_{m,n}} \tag{4b}$$

Adam optimizer is defined in Eq. (5), where ω is the set of model weights, η is the step size, $\hat{m}$ and $\hat{v}$ are first and second moment respectively and ϵ is a small value to avoid dividing by zero.

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{\frac{v_t}{1-\beta_2^t}} + \epsilon} \times \frac{m_t}{1-\beta_1^t} \tag{5}$$

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, including high traffic congestion, the VLP were detected with reasonably higher accuracy (see Section 4). The training began with some iterations 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 deblurring filter and intensity inhomogeneity invariant segmentation of the character.

Blurring is known to be a key challenge in VLP recognition resulting from wrong focusing or vehicle motion that requires restoration of the blurred images for higher accuracy in character recognition. A set of filters like Wiener filter, Total Variation (TV) deblurring, Haar deblurring and a combination of both is considered for restoring any blurred image to its estimated original version. For an input image, $Im[n]$, let the output of the filter is $x[n]$ as in (6a), where $G(z)$ is the filter function, N is the number of past taps with coefficient a, $s[n]$ is the reference signal, and $e[n]$ is the residual error. The criteria to minimize the mean squared error (MSE) for the coefficient is defined in (6b), where $E[.]$ is the expectation operator. On the other hand, Haar deblurring [28] uses a deconvolution filter and reduce the MSE as in (7), where $I(i,j)$ is the expected and $G(i,j)$ is the desired deblurred images. In our model, we have designed a conditional deblurring process invoked when not enough characters are detected in the OCR. An iterative approach is employed for this process with a varying threshold based on the Fast-Iterative Shrinkage Thresholding Algorithm (FISTA) [29] in the VLP image.

$$x[n] = \sum_{i=0}^{N} a_i \times \mathrm{Im}[n-1] \tag{6a}$$

$$a_i = \mathrm{argmin}\left(E\left[e^2[n]\right]\right) \tag{6b}$$

$$MSE = \frac{1}{m \times n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(I,J) - G(i,j)]^2 \tag{7}$$

Additionally, we have considered the intensity inhomogeneity invariant segmentation of the characters. Thus, we extracted the characters from the VLP with two segmentation models: the Chan & Vese (CV) model and the Region-Scalable Fitting (RSF) model (see Table 5 in Section 4). As most of the images for VLP have very high contrast, CV model performed well in approximating the intensities of the foreground or object and background. In the case of intensity inhomogeneity, the CV model used the RSF model and failed to segment the characters.

The complete architecture is illustrated in Fig. 4. The output layer contains 60 classes to capture the total number of Bengali alphabets and digits. The output layer uses *Softmax* as an activation function. We have also used CMATERdb version 3.1.2 for training the alphabets for the digits. A total alphabet-images of 15000 and digits of 500 have been split as follows: 9900 images for training, 2600 images for validation, and 3000 images for testing.

The images are then augmented so that the model can identify images under non-ideal conditions. The following augmentations are considered: (*i*) 10% shift in left and 10% shift in right; (*ii*) 10% shift in vertical and 10% shift in horizontal; (*iii*) 7.5° rotation in left and 7.5° in right; (*iv*) zoom range 20%; and, (*v*) Shear range 20%. Besides, each character illustrated in Fig. 5 is resized to 16 × 16 after segmentation before passing through the network. As a result, this network can work with variable size characters.

4. Results and analysis

We now present the performance of the proposed BLPnet model for the ALPR system, and compare its performance with some related

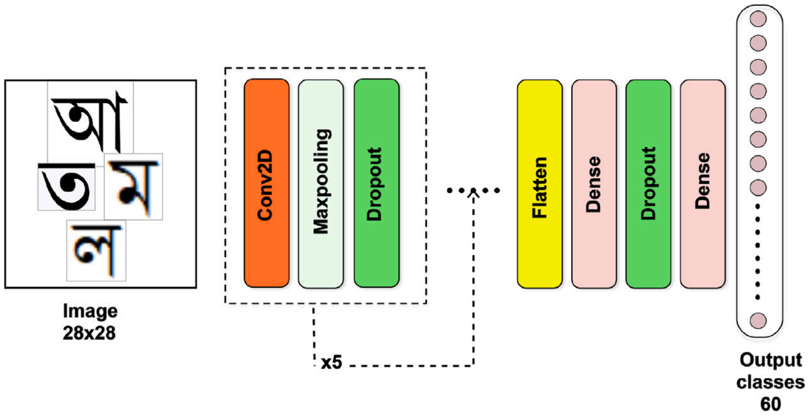


Fig. 4. The newly developed OCR engine for BLPnet.

Type	Characters
Alphabets	'অ','আ','ই','ঈ','উ','ঊ','ঋ','এ','ঐ','ও','ঔ', 'ক','খ','গ','ঘ','ঙ','চ','ছ','জ','ঝ','ঞ', 'ট','ঠ','ড','ঢ','ণ','ত','থ','দ','ধ','ন', 'প','ফ','ব','ভ','ম','য','র','ল', 'শ','ষ','স','হ','ড়','ঢ়','য়', 'ৎ','ং','ঃ','ঁ'
Digits	'০','১','২','৩','৪','৫','৬','৭','৮','৯'

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 $(n+1)\times m$ 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 $i \times j$ will have total learnable parameters of $(i \times j \times p + 1) \times q$.

The distribution of trainable parameters for the bounding box detector 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.

$$\begin{aligned} N_{detector} &= 2\times((1056+1)\times 256 \\ &+ (256+1)\times 128 + (128+1)\times 64 \\ &+ (64+1)\times 32 + (32+1)\times 2) \\ &= 627k \end{aligned} \tag{8}$$

Table 2
Trainable parameters for bounding box detector.

Layers	Shape	Trainable parameters
Global_AveragePooling2D	(None, 1056)	0
Dense	(None, 256)	270592
Dense	(None, 256)	270592
Dense	(None, 128)	32896
Dropout	(None, 256)	0
Dropout	(None, 128)	0
Dense	(None, 128)	32896
Dense	(None, 64)	8256
Dense	(None, 64)	8256
Dropout	(None, 64)	0
Dropout	(None, 64)	0
Dense	(None, 32)	2080
Dense	(None, 32)	2080
Dense	(None, 2)	66
Dense	(None, 4)	132
Total		627k

Table 3
Trainable parameters for OCR engine.

Layers	Shape	Trainable parameters
Conv2D	(None, 63, 63, 16)	80
MaxPooling2D	(None, 31, 31, 16)	0
Dropout	(None, 31, 31, 16)	0
Conv2D	(None, 30, 30, 32)	2080
MaxPooling2D	(None, 15, 15, 32)	0
Dropout	(None, 15, 15, 32)	0
Conv2D	(None, 14, 14, 64)	8256
MaxPooling2D	(None, 7, 7, 64)	0
Dropout	(None, 7, 7, 64)	0
Conv2D	(None, 6, 6, 128)	32896
MaxPooling 2D	(None, 3, 3, 128)	0
Dropout	(None, 3, 3, 128)	0
Conv2D	(None, 2, 2, 256)	131328
MaxPooling 2D	(None, 1, 1, 256)	0
Dropout	(None, 1, 1, 256)	0
Flatten	(None, 256)	0
Dense	(None, 256)	65792
Dense	(None, 512)	131584
Dropout	(None, 512)	0
Dense	(None, 60)	25650
Total		397K

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 for training the model are Tesla K80 2496 CUDA cores GPU with 12 GB GDDR5 VRAM, Processor @2.3 GHz (4 core, 8 threads), and RAM of 48 GB.

We chose the criteria for our evaluation matrices of bounding-box coordinates are MSE and accuracy. From the loss vs epoch curve in

Table 4
Performance of vehicle detection with bounding-box.

Epochs	Loss	Box Output loss	Class Output loss	Box Output MSE	Class Output accuracy	Validation loss	Validation box Output loss	Validation class Output loss	Validation box Output MSE
50	0.0056	0.0176	0.0059	0.0176	75.59	0.0057	0.0175	0.0057	0.0175
100	0.0056	0.0171	0.0058	0.0171	85.89	0.0056	0.0140	0.0057	0.0140
150	0.0056	0.0169	0.0058	0.0169	89.19	0.0055	0.0138	0.0056	0.0138
200	0.00569	0.0173	0.0057	0.0173	90.55	0.0055	0.0175	0.0055	0.0175
250	0.00569	0.0176	0.0056	0.0176	94.74	0.0054	0.0175	0.0055	0.0175
300	0.00569	0.0176	0.0053	0.0176	96.99	0.0055	0.0175	0.0054	0.0175
350	0.00569	0.0176	0.0052	0.0176	97.02	0.0055	0.0175	0.0054	0.0175
400	0.00582	0.0130	0.0050	0.0155	97.05	0.0056	0.0148	0.0058	0.0152

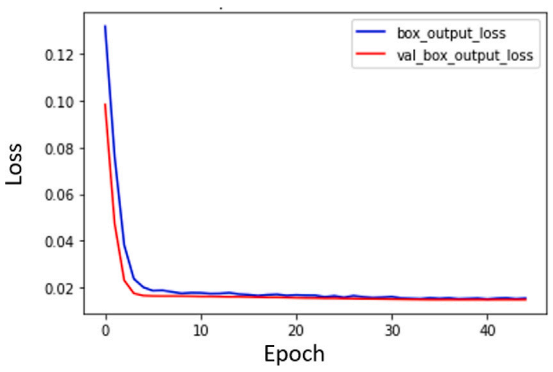


Fig. 6. Loss vs Epoch curve for the first 50 epochs.

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.

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	90	0.302	0.402
	5	83	0.395	0.548
	6	81	0.432	0.701
	8	80	0.502	0.705
RSF model	4	95	0.256	0.402
	5	93	0.312	0.548
	6	90	0.333	0.701
	8	89	0.398	0.705

4.4. Character recognition

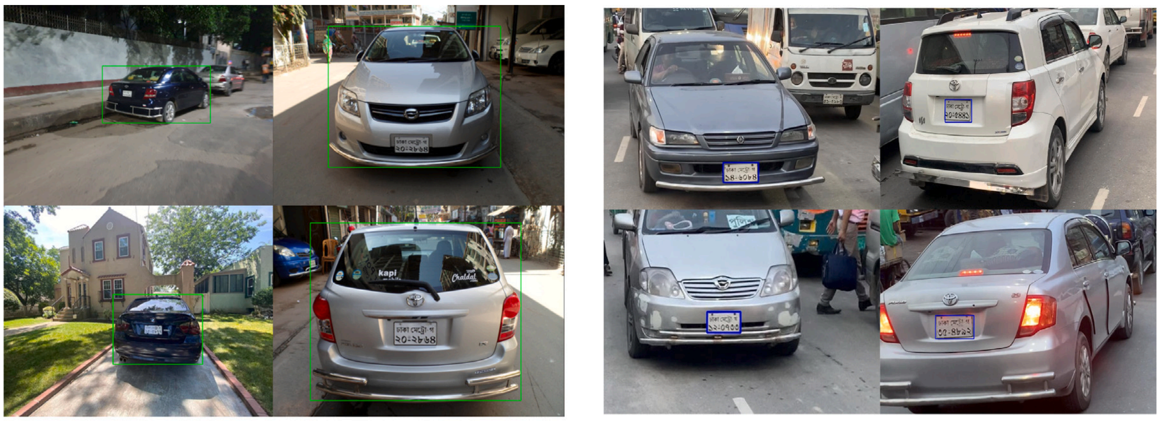
The overall procedure can be broken down into three steps. First is the preprocessing step, where the image is filtered based on the sharpness. Later, in character segmentation, the fast active contour model segments the characters as objects. Finally, a word mapping follows the character recognition. Here, for each group of key characters recognized, a predefined word is mapped for them. Our model's generated character output is shown in Fig. 8.

Our proposed OCR engine demonstrated reasonably better accuracy, as it treated the characters as objects. The accuracy varies from 80% to 95% based on the number of characters extracted. The time it takes to extract a character varies between 0.302 to 0.502 seconds. Initially, the model was weak against some vowels. This weakness was tackled successfully with word mapping (see Fig. 8). Here, detected letters are turned into words with predefined symbols. It is seen that similar license plate reading is captured after the segmentation of both RSF and CV models. Detailed performance analysis is given in Table 5. The accuracy decreases with the increase in the number of characters.

The performance of the proposed model is compared with the other prominent models in Table 6. Considering the nullified False-Positives, lower computational complexity, reduced trainable parameters, and reasonably higher accuracy, the proposed model has demonstrated a higher potential for a real-time ALPR application.

5. Conclusion

The development of an ALPR system is a timely requirement for modern transportation services. 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 inadequate road-safety measures and poor traffic management. To this end, we have presented a computationally efficient and reasonably accurate BLPnet model for a new ALPR system. The model is designed to efficiently and correctly return the VLP number by considering a three-phase top-to-bottom approach. This means, the model starts by detecting the vehicle first, and then any possible VLP followed by the recognition of characters in the VLP. This consideration



(a) Detected vehicles (marked with green bounding-boxes) (b) VLP detection (marked with blue-colour bounding box)

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**

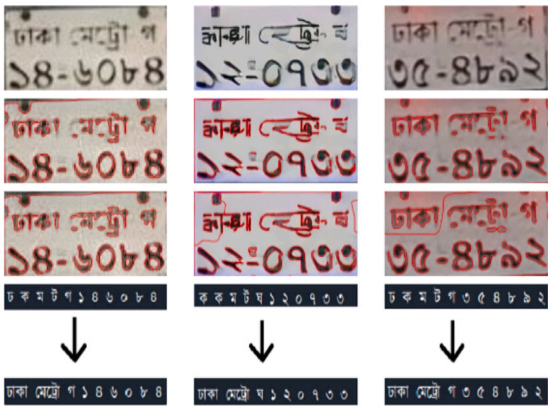


Fig. 8. Example of input and outputs of the newly developed OCR engine (From *top*, input images in *first row*, CV-segmented images in *second row*, RSF-segmented images in *third row*, and OCRs and word-mapping in *fourth row*).

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 frames at the input. The preliminary results with the reasonably faster and accurate performance of the model suggest that it would be promising for the real-time ALPR application, with a continuing development in the future.

CRediT authorship contribution statement

Md. Saif Hassan Onim: Writing – original draft, Conceptualization, Methodology, Visualization, Validation. **Hussain Nyeem:** Project administration, Resources, Methodology, Writing – review & editing, Supervision. **Koushik Roy:** Software, Visualization, Investigation. **Mahmudul Hasan:** Data curation, Validation. **Abtahi Ishmam:** Software, Investigation. **Md. Akiful Hoque Akif:** Validation, Investigation. **Tareque Bashar Ovi:** Resources, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Du S, Ibrahim M, Shehata M, Badawy W. Automatic license plate recognition (ALPR): A state-of-the-art review. IEEE Trans Circuits Syst Video Technol 2012;23(2):311–25.
[2] What are the top 200 most spoken languages? (2021). https://www.ethnologue.com/guides/ethnologue200.

[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.

[4] Xu Z, Yang W, Meng A, Lu N, Huang H, Ying 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.

[5] Laroca R, Severo E, Zanlorensi LA, Oliveira LS, Gonçalves GR, Schwartz 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 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.

[7] Silva SM, Jung CR. A flexible 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.

[8] Zhuang J, Hou S, Wang Z, Zha 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 recognition model based on bi-lstm. IEEE Access 2020;8:211630–41.

[11] Hendry, Chen R-C. Automatic license plate recognition via sliding-window darknet-YOLO deep learning. Image Vis Comput 2019;87:47–56. http://dx.doi.org/10.1016/j.imavis.2019.04.007.

[12] Al-Shemarry MS, Li Y, Abdulla S. 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.

[13] Bochkovskiy A, Wang C, Liao HM. YOLOv4: Optimal speed and accuracy of object detection. 2020, CoRR abs/2004.10934. arXiv:2004.10934.

[14] Hsu G-S, Chen J-C, Chung Y-Z. Application-oriented license plate recognition. IEEE Trans Veh Technol 2013;62(2):552–61. 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/SIBGRAPI.2017.14.

[16] Li H, Wang P, Shen C. Toward end-to-end car license plate detection and recognition with deep neural networks. IEEE Trans Intell Transp Syst 2019;20(3):1126–36. http://dx.doi.org/10.1109/TITS.2018.2847291.

[17] Onim MSH, Akash MI, Haque M, Hafiz RI. Traffic surveillance using vehicle license plate detection and recognition in Bangladesh. In: 2020 11th International conference on electrical and computer engineering. 2020, p. 121–4. http://dx.doi.org/10.1109/ICECE51571.2020.9393109.

[18] Majumdar A. Bangla basic character recognition using digital curvelet transform. J Pattern Recognit Res 2007;2(1):17–26.

[19] Hasnat M, Chowdhury MR, Khan M, et al. Integrating bangla script recognition support in tesseract OCR. 2009.

[20] Rabby ASA, Haque S, Islam S, Abujar S, Hossain SA. Bornonet: Bangla handwritten characters recognition using convolutional neural network. Procedia Comput Sci 2018;143:528–35.

[21] Bai L, Zhao Y, Huang X. A CNN accelerator on FPGA using depthwise separable convolution. IEEE Trans Circuits Syst II Express Briefs 2018;65(10):1415–9.

[22] Redmon J, Divvala SK, Girshick RB, Farhadi A. You only look once: Unified, real-time object detection. 2015, Corr abs/1506.02640. arXiv:1506.02640.

[23] Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. 2009, p. 248–55. http://dx.doi.org/10.1109/CVPR.2009.5206848.

[24] Krause J, Stark M, Deng J, Fei-Fei L. 3D object representations for fine-grained categorization. In: 4th international IEEE workshop on 3D representation and recognition (3dRR-13), Sydney, Australia. 2013.

[25] Li H, Wang P, You M, Shen C. Reading car license plates using deep neural networks. Image Vis Comput 2018;72:14–23. http://dx.doi.org/10.1016/j.imavis.2018.02.002.

[26] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 2818–26.

[27] Henry C, Ahn SY, Lee S-W. Multinational license plate recognition using generalized character sequence detection. IEEE Access 2020;8:35185–99. http://dx.doi.org/10.1109/ACCESS.2020.2974973.

[28] Do TD, Cui X, Nguyen THB, Kim H, Nguyen VH. Blind deconvolution method using omnidirectional gabor filter-based edge information. 2019, CoRR abs/1905.01003. arXiv:1905.01003.

[29] Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imaging Sci 2009;2(1):183–202. http://dx.doi.org/10.1137/080716542.