Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:698
doi: 10.56294/sctconf2024698

Categoría: STEM (Science, Technology, Engineering and Mathematics)

ORIGINAL

Bio Inspired Approach on Automatic License Plate Recognition Technique

Enfoque bioinspirado sobre la técnica de reconocimiento automático de matrículas

Mahalakshmi S[1], Dheeba J[2]

[1]Research Scholar, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore.
[2]Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore.

Cite as: S M, J D. Bio Inspired Approach on Automatic License Plate Recognition Technique. Salud, Ciencia y Tecnología - Serie de Conferencias. 2024;3:698. https://doi.org/10.56294/sctconf2024698

Submitted: 06-12-2023 **Revised:** 22-02-2024 **Accepted:** 15-04-2024 **Published:** 16-04-2024

Editor: Dr. William Castillo-González

ABSTRACT

With increasing range of vehicles in our day-to-day life, managing conveyance is one among the main problem faced by urban areas. Automatic Number Plate Recognition (ANPR) technology may be a tool that is applied to good cities in parking management systems and toll booths on highways to beat this downside. ANPR is employed to localize the license plates then extracting the text from the image, segmented each character and recognize the characters. Various localisation algorithms, segmentation and character recognition algorithms were used to complete the process. The primary objective of our research is to develop a model for number plate identification utilizing bio inspired neural network model and compare with existing neural network models based on different illumination, tilted images blurred and shaded conditions. In this research, we used spiked neural network, a third-generation neural network model, to construct an automatic number plate recognition model inspired by biotechnology. The model shows 70 % accuracy in normal images. The model would be tested for neuromorphic data sets for SNN model to enhance the SNN performance.

Keywords: Algorithms; Biotechnology; Computational Neural Networks; Urban Area.

RESUMEN

Con una gama cada vez mayor de vehículos en nuestra vida diaria, la gestión del transporte es uno de los principales problemas que afrontan las zonas urbanas. La tecnología de reconocimiento automático de matrículas (ANPR) puede ser una herramienta que se aplique en las buenas ciudades en los sistemas de gestión de estacionamiento y en las cabinas de peaje de las autopistas para superar este inconveniente. ANPR se emplea para localizar las matrículas y luego extraer el texto de la imagen, segmentar cada carácter y reconocer los caracteres. Se utilizaron varios algoritmos de localización, segmentación y reconocimiento de caracteres para completar el proceso. El objetivo principal de nuestra investigación es desarrollar un modelo para la identificación de matrículas utilizando un modelo de red neuronal bioinspirado y compararlo con modelos de redes neuronales existentes basados en diferentes condiciones de iluminación, imágenes inclinadas, borrosas y sombreadas. En esta investigación, utilizamos una red neuronal con picos, un modelo de red neuronal de tercera generación, para construir un modelo de reconocimiento automático de matrículas inspirado en la biotecnología. El modelo muestra una precisión del 70 % en imágenes normales. El modelo se probaría en busca de conjuntos de datos neuromórficos para el modelo SNN para mejorar el rendimiento del SNN.

Palabras clave: Algoritmos; Biotecnología; Redes Neurales de la Computación; Área Urbana.

INTRODUCTION

Automated number/license plate recognition system (ANPR/ ALPR) is a type of intelligent transportation system (ITS) technology that can extricate each vehicle as distinct by recognizing the letters of the number plates. The default number identification system detects many types of applications to fit without controlling access to a collection point or parking area or border. ANPR technology has seen the rapid acquisition and dissemination of many agencies around the world to improve their enforcement, investigative, and security aspects. This helps to gather vehicle details that minimize annoying and time-consuming lab or. ANPR is particularly useful for government agencies in tracking down stolen vehicles, traffic offenders, crime-related vehicles, or other popular vehicles. Other applications include parking management, traffic monitoring, automatic street ticket issuance, automatic toll payment, and maintenance.

ANPR work as follows:

- Detect vehicles
- Localizes plate
- Extract characters
- Recognize characters

In ANPR, the camera captures images of the car, and then, utilizing the imaging process, the computer interprets and recognizes the information on the number plate, and then the image will be extracted and as a result the letters of the number will be identified.

Generally, to identify a vehicle number, the number plate must be drawn from the vehicle image. Accurate location detection of number plate is an important step in the process of character recognition. ANPR can be used to automatically open a barrier in a secure member area of Border & Custom Checkpoints to prevent crime. It can be used at Highway Toll Collection by knowing the vehicle type & to charge toll tax accordingly. It can be used to control the traffic flow management and Red-Light Violation enforcement.

Literature survey

The paper[1] discuss the traffic recognition system based on Convolutional Neural Network (CNN). Selecting sub-candidate windows, such as dividing the candidate areas, is part of this method's unsupervised partition technique; use CNN to do the computation. This model is used to recognize logos of vehicle. SVM is used for logo classification. Initially shallow CNN is used to retain all the features and then deep CNN is used for recognition. The main drawback is the system failed for images having illumination changes and noise and the system fails if two of more plates in the image. In[2] the algorithm uses an integration layer to reduce the amount of data processing, further graying and normalization are used to enhance the image. A 2-step convolution layer is used to calculate the performance which is based on ResNet. This algorithm is checked only on CCPD (Chinese City Parking Dataset). In[3] the number plate is recognized from the image using a prepared deep learning model. Different types of vehicles that contain number plate of different shapes and sizes are prepared. 80% images are used for training and 20% data is used in testing. A machine learning model is used for labelling and used to classify features from a data set. The labelled data set is applied in CNN. The license plate with a different background color cannot be detected by this technology.[4]

Discuss about the single shot Detector (SSD) neural network used for plate detection and VGG-16 used for feature extraction, RESNet-18 is applied for segmentation and recognition. This paper uses AMQP protocol and RabbitMQ software vendor is used to increase interoperability modules. The main draw back is that it involves huge computation cost and resources. It is not suitable for noisy and illuminated data set.[5] With weight transferred from highly trained networks, the software includes two YOLO-based tiny and fast networks that operate in cascade mode. The algorithm uses a second network for character acquisition, with a focus on Brazilian LPs, to gain excellent memory and accuracy of 2: 2ms performance with a good GPU. It uses straight edge features to make LP local. This algorithm uses vertical edge features to localize the LP and CNN network to segment and recognize the numbers. The main draw back is that if there were many vehicles in the frame the accuracy would reduce. [6] In this paper Road Wolf ANPR car camera and video frame scanner is used. The line and column indexes of the plate area are obtained by analysing the connected part. Connected Component Labelling (CCL) is used where each character is labelled separately and identified in the next step. The recognition is obtained through CNN. The draw back is it is tested with less data set of 25 images and fails when the speed of the vehicles is high and not tested for skew angles and inconsistent plates. [7] The license plate extracts the vertical edges of the input plate based on a 2D wavelet transformation. The maximum density of the vertical edge is first calculated to determine the possible locations of the license plate. Then the CNN is used to confirm whether or not these possible locations are indeed car plates. Using a straightforward technique based on the vacant space between the letters, the characters are separated when the license plate is acquired. Ultimately, training for a different CNN section kept these candidates apart.[8]

Employs Cascaded CNN which keeps high precision R-CNN. The detection network is divided in to P-Net for number plate acquisition. R-Net to train the candidate and uses NMS system to clean broken windows. O-Net

https://doi.org/10.56294/sctconf2024698

acquires the four-spot of the license plate. CRNN and CTC methods were employed to retrieve the characters without separate them. [9] In this paper, author solves the ALPR task by two-level deep neural networks (DNN) to train. Object recognition CNN using YOLO detects license plates. One engine uses multi-image convolution recurrent neural networks for free detection and classification. The second engine works on shared acquisition. It takes more computation time for processing.[13] two-step segmentation and training were suggested in this work. An SSD was used to distinguish three rows of license plates, while Inception-v3 and MobileNets were used to train the system. The characters of the numbers are divided and trained in the same ways.[14] MD-YOLO method with Back propagation, leaky and identity function is used as a methodology in this paper. To acquire a car license plate, each input image is separated into standard S × S grid cells. The cell containing the vehicle plate center is used for this purpose.[16] discussed that layered ANN was used consisting of input, hidden, and output layers. A genetic algorithm will be used to calculate the number of hidden layers. The input layer has 109 neurons (108 input for pixels and 1 for bias). The output layer has 36(A-Z 0-9). The learning rate, momentum rate, and number of hidden neurons may all be concurrently optimized by a genetic algorithm. Because there are more hidden layers, there is a disadvantage: slower feed forward times. The literature survey summarizes that the techniques used such as

CNN, RNN, deep CNN etc. will give greater accuracy but requires more resources and time for computation.

Proposed System

The proposed system aims to overcome problems in the existing system like the cost of computation, low accuracy, complex images, etc., by incorporating SNN (Spiking Neural Network) which is a 3rd generation neural network model using surrogate leaky integrate fire model. SNN mimics human neurons, which increases the speed and efficiency of plate recognition.

METHODOLOGY

SNN processes the data in form of spikes where the single-bit line toggles between 0 and 1 as shown in figure 1,[11] the transmission of spikes from one neuron to another takes place only when the threshold or membrane potential is reached, thus saving computation, and making the model more efficient. The input layer of the model converts the images of segmented characters to spikes trains, which are passed on to the hidden layers also called synapses. The hidden layer modulates the spikes with the help of weights and then sends them to the output layer. The output layer will process the spikes with the help of the activation function and then generate the output spikes.

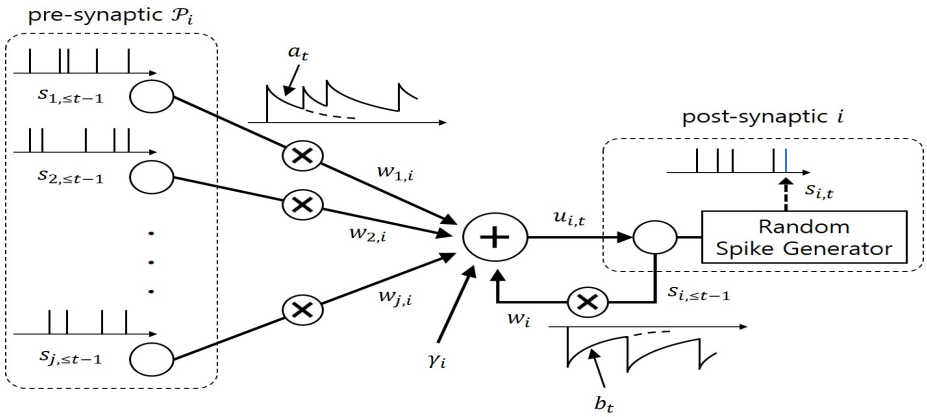

Figure 1. SNN membrane potential model[11]

The input image will be taken from the given dataset, Then, the number plate will be cleaned to remove noise from the plate. Using linked components analysis, the characters on the license plate will be divided into segments. Then, by using a segmented plate the character will be recognised. We are using three different algorithms namely CNN, SNN and Tesseract OCR for character recognition. At Final stage, we will display the output of the recognised number with the accuracy percentage of each algorithm.

We have used OpenCV (Computer Vision Library), Jupyter Notebook, Python to implement the System and used real time data sets to train and test. The users can upload the plate image using the interface designed using Django, the interface supports uploading of single/multiple plate images. The uploaded image will be stored in the media and passed on to segmentation, which will be an internal process without any user communication. The segmented characters will be sent to all three models: CNN, SNN and tesseract and the accuracy will be calculated based on the number of characters recognized correctly. After successful recognition all the three results and accuracy respectively from CNN, SNN and tesseract will be displayed to the user along with the plate with segmented boundaries in a card view.

https://doi.org/10.56294/sctconf2024698

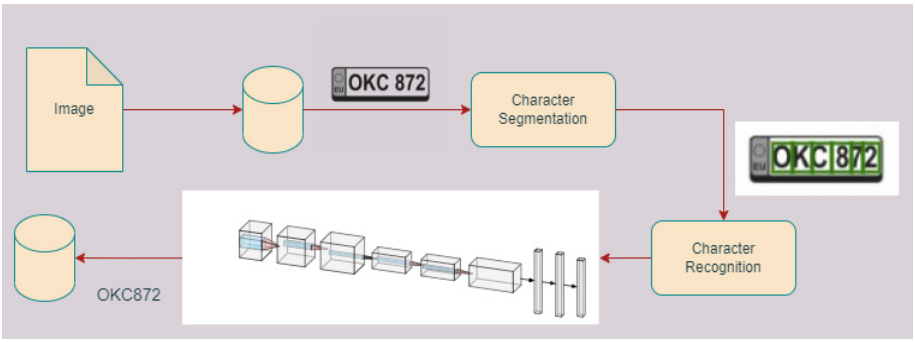

Figure 2. Block diagram for ANPR

Implementation

Input image

We used Django framework to create the user interface to upload the data sets. This module is developed to provide a simple frontend for the application. The accuracy defined to get the accuracy of given images by using all the algorithms provided such as CNN, SNN and Tesseract OCR. Album is defined to request the URL page to be displayed by getting a request. get_file_name is defined to get the file to get it uploaded there. get_segmented_plate is defined to get the segmented image of the number plate as well in the output. image_ upload is defined to upload the images from devices and show the final output to the display.

Character Segmentation

Trying to decompose a single image comprising a sequence of characters into smaller images of each individual character is known as character segmentation. We have used localized plate images as input for our model. Threshold is applied to the plate to reduce the noise. The characters are divided using Connected Component Analysis. If the area is greater than the threshold and the ratio of height and width is less than the threshold, a bounding box is drawn around the character. The character is resized and stored in a list.

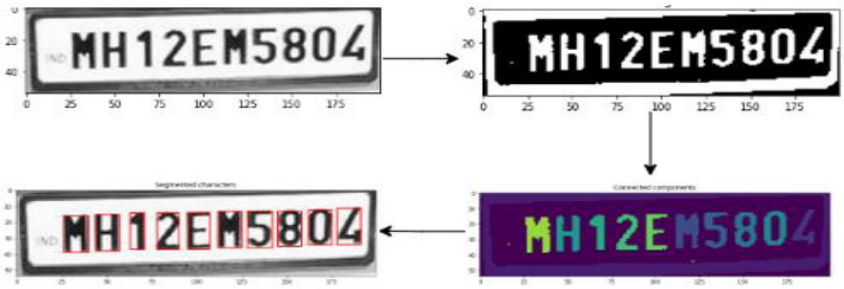

Figure 3. Character Segmentation

```
def find_contours(dimensions, img, is_plot, filename):
    cntrs, _ = cv2.findContours(
        img.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # Retrieve potential dimensions
    lower_width = dimensions[0]
    upper_width = dimensions[1]
    lower_height = dimensions[2]
    upper_height = dimensions[3]

    # Check largest 5 or  15 contours for license plate or character respectively
    cntrs = sorted(cntrs, key=cv2.contourArea, reverse=True)[:15]

    ii = cv2.imread('contour.jpg')

    x_cntr_list = []
    img_res = []
    for cntr in cntrs:
        # detects contour in binary image and returns the coordinates of rectangle enclosing it
        intX, intY, intWidth, intHeight = cv2.boundingRect(cntr)

        # checking the dimensions of the contour to filter out the characters by contour's size
        if lower_width < intWidth < upper_width and lower_height < intHeight < upper_height:
            # stores the x coordinate of the character's contour, to used later for indexing the contours
            x_cntr_list.append(intX)

            char_copy = np.zeros((44, 24))
            # extracting each character using the enclosing rectangle's coordinates.
            char = img[intY:intY + intHeight, intX:intX + intWidth]
            char = cv2.resize(char, (20, 40))
```

Figure 4. Segmentation code

https://doi.org/10.56294/sctconf2024698

Find_countours is defined to draw the bounding box on the number plate images. First, it detects the contours in the binary image and send the coordinates of the rectangles enclosing it. Then it will check the length of the contours to filter out the characters by contour's size. Then the image will be segmented by using pre-processed cropped license plate image.

Character recognition

We used a 28x28 pixel image to train. In CNN Image Data Generator is used to augment when your model is still training, images in real-time. In SNN method we use 3 layers of neurons to recognize the character namely:

a) Input layer (28*28 neurons)
b) Hidden layer (1000 neurons)
c) Output layer (36 neurons)

CNN

We used six hidden layers, one output layer, and one input layer. MAX Pooling Layer and Flatten Layer make up the hidden layer. By choosing the largest element from the feature map region that the filter covers, maximum pooling is a kind of pooling operation. The most crucial characteristics of the preceding layer will thus be included in the output that follows the maximum pooling layer. The flattening layer is used to make a multidimensional input one-dimensional. This is commonly used when moving from a convolution layer to a fully connected layer

```
def create_model():
    K.clear_session()
    model = Sequential()
    model.add(Conv2D(16, (22, 22), input_shape=(
        28, 28, 3), activation='relu', padding='same'))
    model.add(Conv2D(32, (16, 16), input_shape=(
        28, 28, 3), activation='relu', padding='same'))
    model.add(Conv2D(64, (8, 8), input_shape=(28, 28, 3),
                     activation='relu', padding='same'))
    model.add(Conv2D(64, (4, 4), input_shape=(28, 28, 3),
                     activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(4, 4)))
    model.add(Dropout(0.4))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(36, activation='softmax'))

    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer=optimizers.Adam(lr=0.0001), metrics=[custom_f1score])

    # model.summary()
    return model
```

Figure 5. CNN code

Tesseract

Runteserract() function is defined which uses in built function to run the tesseract algorithm. In this, the image will be read, then it will be gray scaled and at last the character will be extracted from the image plate in the form of strings.

```
def runteserract(global_path):
    img = cv2.imread(global_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    text = pytesseract.image_to_string(gray)
    # print(text)
    char = "".join(re.split("[^a-zA-Z0-9]*", text))
    return str(char).upper()
```

Figure 6. Tesseract Code

https://doi.org/10.56294/sctconf2024698

SNN

Leaky surrogate neurons are used to train the model on a digital dataset with minimal loss, which can be achieved using a backpropagation algorithm. Where **Θ** is step function

$$S[t] = \Theta(U[t] - U_{\text{thr}})$$

Equation 1. Step Function

If we consider a single isolated time step, the derivative of the step function evaluates everywhere except when Uthr= **θ** where it goes to infinity. Which means no learning takes place, thus resulting in dead neuron problems. This is overcome by using a surrogate model where the step function remains as it is during the forward pass but during back propagation the derivative of S will be replaced by S. If S does not spike, then spike gradient is 0, if S spikes then spike gradient is 1.

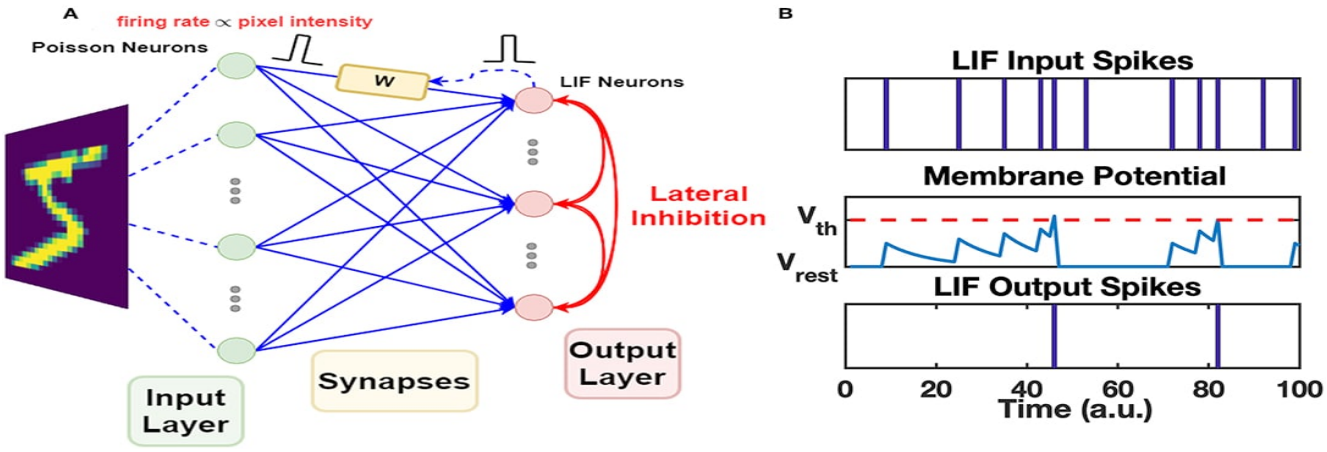

Figure 7. SNN Model[12]

```python
class Net(nn.Module):

    def __init__(self):
        super().__init__()

        # Initialize layers
        self.fc1 = nn.Linear(num_inputs, num_hidden)
        self.lif1 = snn.Leaky(beta=beta)
        self.fc2 = nn.Linear(num_hidden, num_outputs)
        self.lif2 = snn.Leaky(beta=beta)

    def forward(self, x):
        # Initialize hidden states at t=0
        mem1 = self.lif1.init_leaky()
        mem2 = self.lif2.init_leaky()

        # Record the final layer
        spk2_rec = []
        mem2_rec = []

        for step in range(num_steps):
            cur1 = self.fc1(x)
            spk1, mem1 = self.lif1(cur1, mem1)
            cur2 = self.fc2(spk1)
            spk2, mem2 = self.lif2(cur2, mem2)
            spk2_rec.append(spk2)
            mem2_rec.append(mem2)

        return torch.stack(spk2_rec, dim=0), torch.stack(mem2_rec, dim=0)
```

Figure 8. SNN sample code

https://doi.org/10.56294/sctconf2024698

Figure 9. Sample Data sets

We have a dataset of different varieties of number plates like Blurred number plate, Normal number plate, Illuminated number plate and Tilted number plate.

RESULTS AND DISCUSSION

Table 1. Performance comparison of 3 different models

Sl. No	Techniques / Types of dataset	CNN (Accuracy %)	SNN (Accuracy %)	Tesseract (Accuracy %)	Images used
1	Normal	78	78	85,3	158
2	Blurred	44,8	40,3	57,7	14
3	Illuminated	57,3	57,2	64,7	44
4	Tilted	18,5	16,5	19,8	32
5	Shaded	48,5	38,7	67,1	45
6	Yellow	64	36	44	6

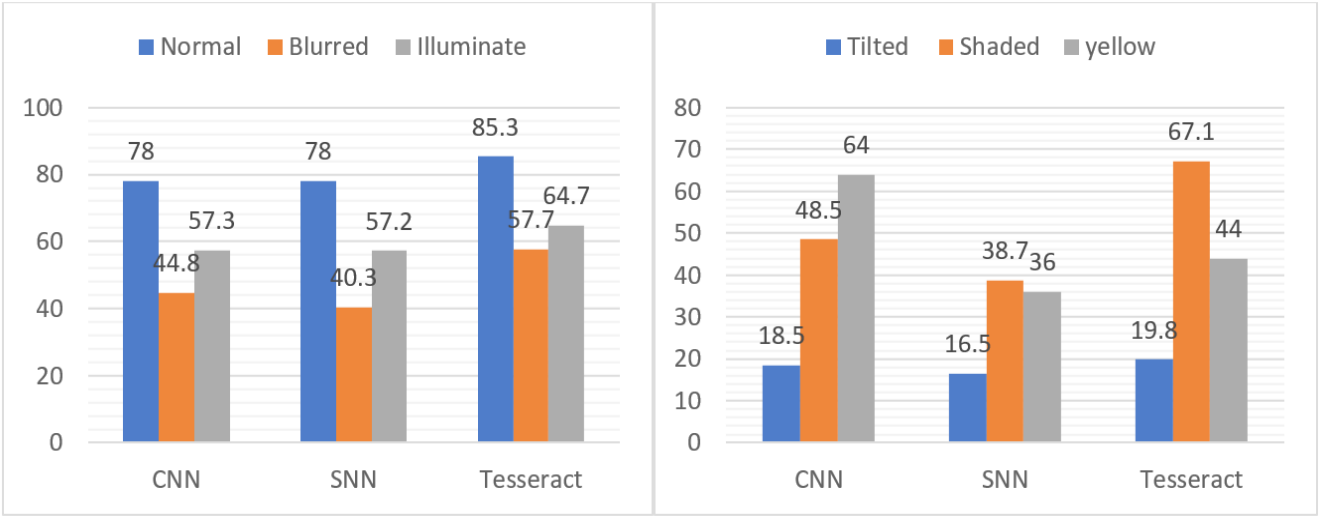

Figure 10. Performance analysis of Different Types of data set

https://doi.org/10.56294/sctconf2024698

In this study, Automatic Number Plate Recognition (ANPR) using Neural Network, uses two Deep Learning techniques, CNN and SNN, as well as an OCR technology called Tesseract. The number plate characters are displayed after the system receives a plate image, processes it using a variety of Computer Vision techniques (such as grayscale, binary thresholding, etc.), segments the characters using Connected Component Analysis, and then applies the Deep Learning algorithms, including OCR.

With the planning and execution of each of its several stages, the system has accomplished its goals. The stages are Image acquisition, image segmentation, and character recognition. The primary testing conducted has produced an overall success rate for OCR of 85 %, CNN of 78 % SNN of 70 % for normal images. In the complementing trials that have also been conducted for various sorts of images, such as slanted, illuminated, etc the ANPR system will need to be improved in several areas to perform well enough. Using time series (neuromorphic) datasets for SNN model training can enhance SNN performance. SNN becomes stable over time because it is still in the research stage and might need lot of fine-tuning adjustments.

REFERENCES

1. Ma L, and Zhang Y. Research on vehicle license plate recognition technology based on deep convolutional neural networks. Microprocessors and Microsystems, 82, pp. 103932. https://doi.org/10.1016/j.micpro.2021.103932.

2. Wang Z, Jiang Y, Liu J, Gong S, Yao J, and Jiang F, et al. Research and implementation of fast-LPRNet algorithm for license plate recognition. Journal of Electrical and Computer Engineering, pp. 1-11. https://doi.org/10.1155/2021/8592216.

3. Vinay KV, Balaobaiah IO, Mahiboob MS, and Nagnath SD, AUTOMATIC NUMBER PLATE RECOGNITION FOR DIFFERENT FONTS AND NON-ROMAN SCRIPT. 5(11), pp. 314-317.

4. Rusakov KD. Automatic modular license plate recognition system using fast convolutional neural networks. In 13th International Conference" Management of large-scale system development"(MLSD), pp. 1-4. DOI: 10.1109/MLSD49919.2020.9247817.

5. Montazzolli S, and Jung C. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In 30th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI), pp. 55-62. DOI: 10.1109/SIBGRAPI.2017.14.

6. Slimani I, Zaarane A, Al Okaishi W, Atouf I, and Hamdoun A, et al. An automated license plate detection and recognition system based on wavelet decomposition and CNN. Array, 8, pp. 100040. https://doi.org/10.1016/j.array.2020.100040.

7. Wang W, Yang J, Chen M, and Wang P, et al. A light CNN for end-to-end car license plates detection and recognition. IEEE Access, 7, pp. 173875-173883. DOI: 10.1109/ACCESS.2019.2956357.

8. Kessentini Y, Besbes MD, Ammar S, and Chabbouh A, et al. A two-stage deep neural network for multi-norm license plate detection and recognition. Expert systems with applications, 136, pp. 159-170. https://doi.org/10.1016/j.eswa.2019.06.036.

9. Zhang J, Li Y, Li T, Xun L, and Shan C, et al. License plate localization in unconstrained scenes using a two-stage CNN-RNN. IEEE Sensors Journal, 19(13), pp. 5256-5265. DOI: 10.1109/JSEN.2019.2900257.

10. Jang H, Simeone O, Gardner B, and Gruning A, et al. An introduction to probabilistic spiking neural networks: Probabilistic models, learning rules, and applications. IEEE Signal Processing Magazine, 36(6), pp. 64-77. DOI: 10.1109/MSP.2019.2935234.

11. Guo Y, Wu H, Gao B, and Qian H, et al. Unsupervised learning on resistive memory array based spiking neural networks. Frontiers in neuroscience, 13, pp. 1-16. https://doi.org/10.3389/fnins.2019.00812.

12. Puarungroj W, and Boonsirisumpun N. Thai license plate recognition based on deep learning. Procedia Computer Science, 135, pp. 214-221. https://doi.org/10.1016/j.procs.2018.08.168.

13. Xie L, Ahmad T, Jin L, Liu Y, and Zhang S, et al. A new CNN-based method for multi-directional car license plate detection. IEEE Transactions on Intelligent Transportation Systems, 19(2), pp. 507-517. DOI: 10.1109/TITS.2017.2784093.

14. Mahalakshmi S, and Sendhil Kumar R. Smart toll collection using automatic license plate recognition techniques. In Computing, Analytics and Networks: First International Conference, pp. 34-41. https://doi.org/10.1007/978-981-13-0755-3_3.

15. Tarigan J, Diedan R, and Suryana Y, et al. Plate recognition using backpropagation neural network and genetic algorithm. Procedia Computer Science, 116, pp. 365-372. https://doi.org/10.1016/j.procs.2017.10.068.

16. Krishnakumar P. Automatic Number Plate Recognition (ANPR) through smart phones using image processing techniques. IOSR Journal of VLSI and Signal Processing, 4(4), pp. 19-23. DOI:10.9790/4200-04431923.

FINANCING

The authors did not receive financing for the development of this research.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

AUTHORSHIP CONTRIBUTION

Conceptualization: Dheeba J.
Data curation: Mahalakshmi S.
Formal analysis: Dheeba J.
Research: Mahalakshmi S.
Methodology: Mahalakshmi S.
Drafting - original draft: Dheeba J.
Writing - proofreading and editing: Dheeba J.