DOI: http://dx.doi.org/10.26483/ijarcs.v13i2.6808

ISSN No. 0976-5697

Volume 13, No. 2, March-April 2022

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

DETECTION AND RECOGNITION OF HINDI TEXT FROM NATURAL SCENES AND ITS TRANSLITERATION TO ENGLISH

Kumar Shwait
Dept. of Computer Science and Engineering,
Sant Longowal Institute of Engineering and Technology
Longowal, Punjab, India

Preetpal Kaur Buttar
Dept. of Computer Science and Engineering,
Sant Longowal Institute of Engineering and Technology
Longowal, Punjab, India

Rahul Gautam
Dept. of Computer Science and Engineering,
Sant Longowal Institute of Engineering and Technology
Longowal, Punjab, India

Abstract: India is a country with many cultures and if you travel from one place to another, you might find yourself in totally different culture. This also means the languages change from place to place in India and it gets very difficult to read signboards, shop names and even many other common things written in local languages. This can create problems for not only the travelers travelling from other countries but also the people who move withing the country from different regions. But most of the signboards, shop names or other landmarks mostly use English or Hindi in most of the regions. Here we propose a complete text detection & recognition as well as transliteration system that will help travelers read text written in Hindi on any signboards or shops and then transliterate that detected text into English. The proposed system is capable of detecting text written in Hindi language in natural environment using Progressive Scale Expansion algorithm and then transliterating the detected text into English language. Our proposed system can detect text in tough scenarios, and it can even detect curved text from natural images. Our system after detecting text region, extracts the text from the detected area using PyTesseract OCR engine and then the extracted text is further transliterated into English text with the help of seq2seq MultiRNN LSTM model which gives us accurate transliterations without losing the actual pronunciation of the original Hindi words. We use a synthetic dataset for Hindi Text images containing approx. 100000 for Text Detection and FIRE2013 dataset for transliteration. The overall system is evaluated using BLEU score.

Keywords: Text detection; text recognition; transliteration; scene text; LSTM; neural machine translation

I. INTRODUCTION

Travelling in India has been an attraction for many tourists since many years. The country is full of different cultures from North to South and from East to West. The diversity of languages and culture makes the country an attraction for many tourists from other countries as well as within India. This makes travelling very tough in India for many travelers, not just coming from other countries, but also for those who travel from different parts of the country. As an average traveler, it's easy to get confused by various signboards written in unfamiliar languages. Here we put our focus on Hindi language as it is the most dominant language used in India. We propose a system to detect text from signboards with local language and transliterate them to English, word by word, so they don't lose the actual pronunciation after getting converted into English. The task at hand can be divided into three major divisions:

- Text Detection
- Recognition of Individual Characters
- Natural Language Processing (NLP) for transliteration.

Text detection is the process of detecting text from natural images like Signboards or Shop Boards. For any given image, the system should be able to detect the text and cut out the detected text region for further processing. There are many systems and features handcrafted for this task. For characters having features like stroke width, uniform size and chromatic consistency, there are techniques like Stroke Width Transform (SWT) [1] and Maximally Stable External Regions (MSER) [2], [3]. These features are used for Connected Component Analysis (CCA) to extract and separate out each text components. Other algorithms like Sliding Window Technique [4] are used where windows of different sizes are slid on image to detect if ant text is present. Then graph based grouping algorithm is applied to combine them into chunks texts to group them as words. With Deep Learning, Text Detection problem is mainly considered as an Object Detection problem. There are two main ways to handle the problem, the first approach is detecting word by word and then processing to get the characters in each word. The second approach is to detect each character first and then group them into words. Another Deep Learning approach is to classify pixel wise if it belongs to a text or not and then use k-nearest approach to instance segment each character [5].

After detecting text from the image, it is very important to extract that information for further processing. There are mainly two ways of text recognition, Character recognition and Word Recognition. Character recognition methods divide the text in the image into multiple cut-outs of single characters. In past few decades, Optical Character Recognition (OCR) has gotten very better. There have been many algorithms based on deep learning method. The most common methods are

Convolutional Neural Network (CNN) [6] and Recurrent neural Network (RNN). Other methods are based on combination of these methods [7]. Word recognition uses character recognition outputs along with language models or lexicons to recognize words from text image. When there is limited number of word possibilities in input images, word recognition is better approach than character recognition.

There are also many open-source OCR engines which have very good accuracy. They can recognize different languages with good accuracy and their accuracy can further be improved with good image pre-processing steps. The accuracy of PyTesseract is fairly high out of the box which can be further improved significantly with the help of a well-designed Tesseract image pre-processing pipeline. Tesseract OCR engine has been used for text recognition for many researches in past decade because of its high accuracy [8]–[10]. Here in this research, we also use Tesseract OCR engine.

For text transliteration, traditional Natural Language Processing (NLP) based solutions were used for a long time. But if we need to handle longer text, the quality degrades. Hence for complicated data and to obtain higher accuracy, other methods like Deep Learning Sequential models like RNN has gained thrust in the research filed. One of the earliest works is by treating it as object detection problem [11]. For transliteration, Deep Belief networks were used [12]. Currently, most of the research work is focused on Deep Sequence to Sequential Models, attentional mechanisms are also used to improve quality [13], [14].

II. Related Work

Work in text detection and recognition has been going on since decades. However, the best results were observed in the Deep Learning era. An approach [15] to text recognition in natural scene images which was unlike any already existed work was proposed which assume that texts are horizontal and frontal parallel to the image plane and was able to recognize perspective texts of arbitrary orientations. It was proposed that features computed at octave-spaced scale intervals are sufficient to approximate features on a finely sampled pyramid which helped a lot in scene text detection [16]. Researchers also presented a residual learning framework [17] to ease the training of networks which are substantially deeper than those used previously. In their work, layers were explicitly reformulated as learning residual functions with reference to the layer inputs instead of learning unreferenced functions. An end-to-end trainable fast scene text detector was proposed which was called TextBoxes [18] and it can detect text in natural images in a single network forward pass, thus it does not involve any post-process except for a standard non-maximum suppression. Their detector was very fast, and it took only 0.09s per image. Researchers also proposed a pipeline that yields fast and accurate text detection in natural scenes [5]. The pipeline directly predicts text in any shape and quadrilateral shapes in full images with a single neural network. The simplicity of proposed pipeline allows concentrating efforts on designing loss functions and neural network architecture. In 2018, researchers proposed a method named sliding line point regression (SLPR) [19] to detect arbitrary-shape text in natural scene. SLPR regresses multiple points on the edge of text line and then sketches the outline of text using these points. For transliteration, it was observed that NMT requires very less amount of data size for training and thus exhibits satisfactory results [20].

III. Datasets Used

For Indian languages, one of the limitations is availability of databases for both detection as well as transliteration. Our work includes different databases for text detection as well as text transliteration. Both databases are publicly available databases.

For text detection, we use a synthetic dataset which includes approximately 1,00,000 images with annotations provided separately for each image. For transliteration, we have used FIRE dataset. Fire dataset is useful for both translation as well as transliteration tasks for several languages. Here we have used FIRE 2013 Eng-Hin Dataset. It contains total of 30,823 words transliterations from English to Hindi. We use these word-to-word transliterations to train our transliteration model.

IV. Proposed Method

The overall pipeline consists of mainly two parts, i.e., Text Detection & Recognition, and Transliteration. The Hindi text in the image is first detected using Progressive Scale Expansion Network (PSENet) [21] and then the detected text is extracted from the image. After extraction, hindi text is passed to the transliteration system which uses encoder and decoder and transliterates the input Hindi text to English text. The overall pipeline can be easily understood with the help of Figure 1.

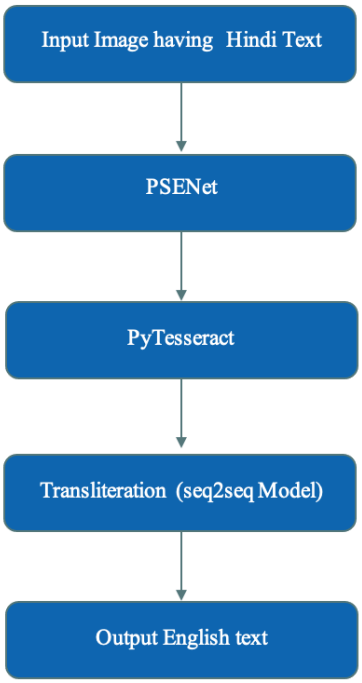

Figure 1. Overall pipeline of the proposed method.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. Text Detection and Recognition

Text Detection is the one of the most important tasks in any OCR system. For text detection, we are using PSENet which is

a segmentation-based network that can detect text very well. This method achieves state of art results for text detection on International Conference on Document Analysis and Recognition (ICDAR) 2015 and ICDAR 2017 datasets. This method is already exceptional in detecting English Text and here we fine tune it so that it can detect Hindi text very well. The reasons we use this method is because:

- It can precisely detect text instances with arbitrary shapes.
- It is able to accurately separate the text instances standing closely to each other.

PSENet is inspired by Feature Pyramids Network (FPN) [22]. The overall pipeline is described thoroughly in [21]. PSENet is combination of FPN and Progressive Scale Expansion algorithm (PSE). In PSENet, low-level feature maps are concatenated with high-level feature maps which results in having overall 4 concatenated feature maps. These obtained feature maps are then fused in F to encode information with various receptive views which is very likely to facilitate the generations of the kernels with various number of scales. After that the feature map F is projected into n branches which produces multiple segmentation results S_1, $S_{2,\ldots,}S_n$ where each S_i would be one segmentation mask for all the text instances at a certain scale. Among these produced masks, S_1 gives the segmentation result for the text instances with smallest scales (i.e., the minimal kernels) and S_n denotes for the original segmentation mask (i.e., the maximal kernels). After obtaining these segmentation masks, PSE algorithm is used to gradually expand all the instances' kernels in S_1, to their complete shapes in S_n. The final detection result is obtained as R as shown in Figure 2.

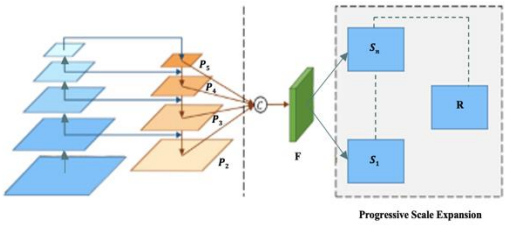

Figure 2. Architecture of PSENet.

It is very hard for segmentation-based method to separate closely packed text instances. To solve this issue, PSENet uses PSE algorithm. PSE is based on Breadth First Search algorithm (BFS) which starts from the pixels of multiple kernels and iteratively merges the adjacent text pixels. There may be some conflicted pixels during expansion which are dealt by merging the confusing pixel by one single kernel on the basis of first come first served. Due to the progressive expansion procedure, these boundary conflicts do not affect the final detections and the performance. The summary of scale expansion algorithm is given in Algorithm 1 to understand how it works is given below.

Algorithm 1: Progressive Scale Expansion

Require: $Kernels$: C, Segmentation Result: S_i
Ensure: $Scale\ Expanded\ Kernels$: E
function $EXPANSION(C, S_i)$
  $T \leftarrow \emptyset$; $P \leftarrow \emptyset$; $Q \leftarrow \emptyset$
  for each $c_i \in C$ **do**
    $T \leftarrow T \cup \{(p, label)|(p,\ label) \in c_i\}$; $P \leftarrow P \cup \{p|(p, label) \in c_i\}$
    $\boldsymbol{Enqueue}(Q, c_i)$
  end for
  while $Q \neq \emptyset$ **do**
    $(p, label) \leftarrow \boldsymbol{Dequeue}(Q)$
    if $\exists q \in \boldsymbol{Neighbor}(p)$ and $q \notin P$ and $S_i[q] = True$ **then**
      $T \leftarrow T \cup \{(q, label)\}$; $P \leftarrow P \cup \{q\}$
      $\boldsymbol{Enqueue}(Q, (q, label))$
    end if
  end while
  $E = \boldsymbol{GroupByLabel}(T)$
  return E
end function

In the pseudocode, T, P are the intermediate results and Q is a queue. $Neighbor\ (.)$ represents the neighbor pixels of p. $GroupByLabel\ (.)$ is the function of grouping the intermediate result by label. "$S_i[q] = True$" means that the predicted value of pixel q in S_i belongs to the text part.

The label generation is done using polygon shrinking where ground truth labels are conducted by shrinking the original text instance. If we consider the scale ratio as r_i, the margin d_i between p_n and p_i can be calculated as:

$$d_i = \frac{\text{Area}(p_n) \times (1 - r_i^2)}{\text{Perimeter}(p_n)} \quad (1)$$

where $Area\ (.)$ is the function of computing the polygon area and $Perimeter\ (.)$is the function of computing the polygon perimeter. Further, the scale ratio r_i for ground truth map G_i can be defined as:

$$r_i = 1 - \frac{(1 - m) \times (n - i)}{n - 1} \quad (2)$$

where m is the minimal scale ratio whose value lies in (0,1]. The values of scale ratios (i.e., $r_1, r_{2,\ldots,} r_n$) are decided by two hyper-parameters n and m, which increase linearly from m to 1. The loss function can for PSENet is given by:

$$L = \lambda L_c + (1 - \lambda) L_s \quad (3)$$

where L_c and L_s represent the losses for the complete text instances and the shrunk ones respectively. λ balances the importance between L_c and L_s. Dice coefficient is also adopted using the information from [23] to deal with network predictions being biased to the non-text region. Online Hard Example Mining (OHEM) [24] is also adopted during training to better distinguish between patterns like text strokes, such as fences, lattices, etc.

In our experiments, we implemented the PSENet using the FPN [22]. The feature maps are obtained in same way as described in [21] and the obtained feature maps are fused with F. The function used for fusion is $C\ (.)$ which is described as:

$$F = C(P_2, P_3, P_4, P_5) = P_2 \parallel Up \times 2(P_3) \parallel Up \times 4(P_4) \parallel Up \times 8(P_5)$$

where " ‖ " refers to the concatenation and $Up \times 2(.)$, $Up \times 4(.)$, $Up \times 8(.)$ refer to 2, 4, 8 times up sampling. Then, feature map F is passed to Conv (3, 3)-BN-ReLU layers. After that, it is passed through multiple Conv (1, 1)-Up-Sigmoid layers which produces n segmentation results S_1, $S_{2,\ldots\ldots}$ S_n. Here, Conv refers to convolution [25], BN refers to batch normalization [26], ReLU refers to rectified linear units [27] and Up refer to upsampling.

The value of n is set to 6 and value of m is set to 0.5 for label generation which gives the scales {0.5, 0.6, 0.7, 0.8, 0.9, 1.0}. The value of m lies between 0.1 to 0.9. The performance of system drops when m is either too large or too small. When m is too large, it gets difficult for network to separate the text instances standing closely to each other and when m is too small, network often splits a whole text line into different parts incorrectly and the training cannot converge very well. This is why we decided to set m = 0.5 which gives the best results. The λ of loss balance value is set to 0.7. We calculate the minimal area rectangle to extract the bounding boxes as final predictions.

In our experiments, ResNet [28] is used which was pretrained on ImageNet Dataset [29] and the weights initialization is adopted from [30]. 36000 synthetic training images and 4000 synthetic validation images were used to train the model with the help of GPU (NVIDIA GTX 1080 Ti). The batch size, epochs and decayed learning rate is adopted from [21].

After training is complete, we used PyTesseract for extracting the text from detected text region. PyTesseract is an open-source python library which is based on CNN-LSTM base model and is very fast method to extract the detected text and it is also very accurate. The extracted text is stored in separate text files which we finally use in our next step where we transliterate it to English language.

B. Transliteration

For transliteration, we used seq2seq based architecture which uses RNN with Long Short Term Memory (LSTM) cells, Gated Recurrent Unit (GRU) cells and MultiRNN LSTM cells. RNNs are a special type of neural network with loops that allow information to persist throughout different steps in a network. The loop enables the neural network to go back and check what happened in all of the previous words before deciding what the current word actually means. In simple language, a RNN can be thought of as copy-pasting the same network over and over again, with each new copy-paste adding a bit more information than the previous one. The input of RNN can be fixed to seq, seq to fix or seq to seq. We are using the seq to seq in which we provide the Hindi text and get the text of transliterated in English. A seq-to-seq network is generally represented as seq2seq network.

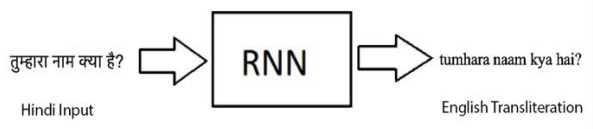

Figure 3. seq2seq RNN model.

Our model for machine transliteration uses encoder and decoder architecture [31]. Encoder–decoders are two separate recurrent neural networks (RNN). The first network is used to encode the input sequence of the source language into a sequence vector which used by another network to decode that vector into an output sequence of the target language. This sequence vector is a hidden state of the network which uses recurrent activation function. Encoder and decoder consist of the memory cell (LSTM/GRU) that helps the network to remember long sequences. A GRU cell has one less gate than an LSTM i.e., it just has a reset and update gates instead of the forget, input, and output gate of LSTM. To understand working of our models, let's say for transliterating a source sentence x = { x_1, $x_{2,\ldots\ldots}$ x_n} into a target sentence y = { y_1, $y_{2,\ldots\ldots}$ y_n}, two RNNs are used, i.e., an encoder and a decoder. These encoder and decoder are trained together to generate output sequence y from the input sequence x, where x_t and y_t are a token of sequence at any given time t and n is the total number of tokens in a given sequence. The transliteration system we built is a character based RNN encoder and uses characters as tokens. RNN encoder encodes the input into a hidden sequence h = { h_1, $h_{2,\ldots\ldots}$ h_i}, where h_t is a hidden state of the encoder at time step t and i is a dimension of the hidden vector described in the equation below:

$$h_t = \sigma(Ux_t + Vh_{t-1}) \quad (5)$$

In the above equation, U and V are weight matrices that connect the input sentences and recurrent output, respectively. h_t is the hidden state at time step t which represents input at the same time step x_t, modified by a weight vector U added to a previous hidden state h_{t-1} multiplied by its own hidden state matric V. A logistic sigmoid activation function is used to squash the sum of weight input and hidden state. The encoder encodes the input sequence into fixed dimension vector representation which helps to compute the new internal state at time step t using the history of sequence vector h_{t-1}.

The decoder RNN takes the output h_t from the encoder and target output o_t as input at time step t along with its decoder previous state h_{t-1} and the output sequence is generated token by token at each time step t using the SoftMax activation function as shown in the equation below:

$$d_{o_t} = softmax\left(W\left(\sigma(Io_t + Jh_t + Kh_t + b)\right)\right) \quad (6)$$

Here, d_{o_t} is a decoder output, I, J and K are decoder weight vectors, b is a decoder bias vector, h_{t-1} is a decoder previous hidden state, h_t is an encoder hidden state.

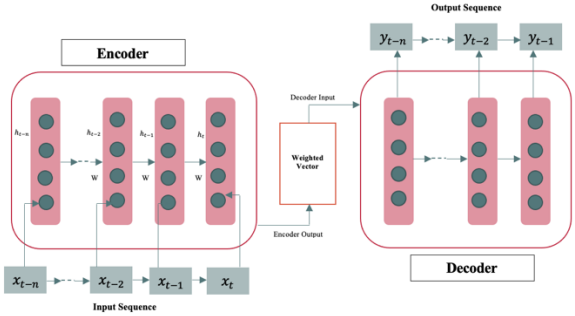

Figure 4. Detailed architecture of seq2seq model.

Thus, RNN will try to predict the next token based on the history of input tokens. For a series of letters, a RNN will use the first character to help determine its perception of the second character. For our seq2seq model, we experimented with GRU Cells, LSTM cells and MultiRNN LSTM cells and used the implementation with the embedding layer provided by TensorFlow. We used FIRE2013 dataset which contains total number of 30823 transliteration samples and 90% of the samples from the dataset were used for the training and remaining 10% for the validation. As for loss function, we used Weighted SoftMax Cross Entropy function. We used learning rate of 0.0001. The batch size used is 30 and the size of the cells used is 128. The training was done on Google Colab using GPU runtime for 20 epochs.

V. RESULTS AND COMPARISON

For Text detection and Recognition, we performed our experiments on Synthetic Train Dataset. We trained and validated on 36000 samples of images and 4000 samples of images respectively. To test the performance, we used test images from the synthetic dataset and obtained output images with text detected inside bounding boxes alongside a text file containing extracted text which we need to transliterate into English. The output of the model can be seen below:

Figure 5. Output images containing detected text.

The results for different methods applied on FIRE2013 Eng-Hin dataset are given in the table below:

Table I. Results for seq2seq Model on FIRE2013 dataset

Method	*BLEU Score*
GRU based RNN	0.76
LSTM based RNN	0.79
MultiRNN LSTM	0.82

As we can see our seq2seq model with MultiRNN LSTM clearly outperforms the other methods and gives accurate results. We observed that MultiRNN LSTM cells are better approach than RNN with GRU cells and LSTM cells which outperforms RNN with GRU cells by 0.6 and RNN with LSTM cells by 0.3.

We further compared the proposed method with Google Translate which is a very commonly used app for Translation as well as transliteration for tasks like reading Signboards with Hindi language or reading shop names. We compared for few words to show how our transliteration method is better than Google Translate which fails to give accurate transliterations and just returns translations instead. The comparison is shown in Table 2.

Table II. Comparison Between Google Translate & Proposed Method

Input Hindi Word	*Expected Output*	*Google Translate*	*Proposed Method*
□□□□□ □□□	Sunder Nagar	Beautiful City	sundar nagar
□□□□□ □□□	Vikas Nagar	Vikas Nagar	Vikaas nagar
□□□ □□□□	Chai Wala	Tea Maker	Chaay waala
□□□□	Guru	Master	guru
□□□□□ □□□□	Chashme Wala	Spectacled	Chashme waala
□□□□ □□□□□	Khada Patthar	Standing Stone	khada patthar

As we can see from the Table 2, Google Translate is able to transliterate Hindi from English and it gave accurate transliteration for "□□□□□ □□□", however, it fails to transliterate most of the Hindi words and instead just gives translations for them whereas our seq2seq model provides accurate English transliterations for Hindi words.

VI. CONCLUSIONS AND FUTURE SCOPE

We proposed a system for text detection & recognition using PSENet and PyTesseract and further connected it to a seq2seq encoder-decoder model to transliterate the output given by it into English language. The system is capable of detecting text instances in natural images. Our system is very robust to shapes and is capable to detect closely packed text instances in natural images by progressively expanding the detected regions from small kernels to large and complete instances via multiple segmentation maps. We further use PyTesseract to extract the text from the detected region. Our system also performed very well for the transliteration problem even though the dataset available was very small. We further observed that using a smaller number of layers and thus keeping our model less complex yielded better results. The reason for this could be the simplicity of LSTM models which is advantageous in our case where the dataset is fairly small and not very complex.

So far, we only focused on Hindi language as it is spoken more as compared to other languages in India but there are many different regional languages spoken here and we can work in future on this project to make a transliteration service which works for many other languages. For future researchers, there is also scope for work to create a bi-directional transliteration system where they can detect more than one language in natural images and transliterate them in any direction. That is, they can easily convert English to their regional language or regional language to English. As we

know machine learning yields better results with more data, more data can be used to train to make the system even more accurate. The dataset for Hindi to English transliteration is very limited, therefore, in future more work can be done on gathering more data for Hindi to English transliteration which can provide better results.

VII. REFERENCES

[1] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 2963–2970, 2010, doi: 10.1109/CVPR.2010.5540041.

[2] S. Bhargava and E. Yablonovitch, "Lowering HAMR near-field transducer temperature via inverse electromagnetic design," IEEE Trans. Magn., vol. 51, no. 4, 2015, doi: 10.1109/TMAG.2014.2355215.

[3] S. Karim, A. A. Laghari, A. Halepoto, A. Manzoor, N. Hussain Phulpoto, and A. Ali, "Vehicle detection in Satellite Imagery using Maximally Stable Extremal Regions," IJCSNS Int. J. Comput. Sci. Netw. Secur., vol. 18, no. 4, pp. 75–78, 2018.

[4] I. Ahmad and G. A. Fink, "Handwritten Arabic text recognition using multi-stage sub-core-shape HMMs," Int. J. Doc. Anal. Recognit., vol. 22, no. 3, pp. 329–349, 2019, doi: 10.1007/s10032-019-00339-8.

[5] X. Zhou et al., "East: An efficient and accurate scene text detector," arXiv, pp. 5551–5560, 2017.

[6] J. Wang and X. Hu, "Gated Recurrent Convolution Neural Network for OCR," no. Nips, 2017.

[7] P. Shivakumara, D. Tang, M. Asadzadehkaljahi, T. Lu, U. Pal, and M. H. Anisi, "CNN-RNN based method for license plate recognition," CAAI Trans. Intell. Technol., vol. 3, no. 3, pp. 169–175, 2018, doi: 10.1049/trit.2018.1015.

[8] L. Giridhar, A. Dharani, and V. Guruviah, "A novel approach to OCR using image recognition based classification for ancient tamil inscriptions in temples," arXiv, pp. 1–8, 2019.

[9] S. Prajapati, S. R. Joshi, A. Maharjan, and B. Balami, "Evaluating Performance of Nepali Script OCR using Tesseract and Artificial Neural Network," Proc. 2018 IEEE 3rd Int. Conf. Comput. Commun. Secur. ICCCS 2018, pp. 104–107, 2018, doi: 10.1109/CCCS.2018.8586808.

[10] A. S., J. Yankey, and E. O., "An Automatic Number Plate Recognition System using OpenCV and Tesseract OCR Engine," Int. J. Comput. Appl., vol. 180, no. 43, pp. 1–5, 2018, doi: 10.5120/ijca2018917150.

[11] P. Duygulu, K. Barnard, J. F. G. de Freitas, and D. A. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2353, pp. 97–112, 2002, doi: 10.1007/3-540-47979-1_7.

[12] T. Deselaers, S. Hasan, O. Bender, and H. Ney, "A deep learning approach to machine transliteration," no. March, p. 233, 2009, doi: 10.3115/1626431.1626476.

[13] M. Alam and S. ul Hussain, "Sequence to sequence networks for roman-Urdu to Urdu transliteration," arXiv, pp. 1–7, 2017.

[14] Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv e-prints, p. arXiv:1609.08144, 2016, [Online]. Available: http://arxiv.org/abs/1609.08144.

[15] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan, "Recognizing text with perspective distortion in natural scenes," Proc. IEEE Int. Conf. Comput. Vis., pp. 569–576, 2013, doi: 10.1109/ICCV.2013.76.

[16] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 8, pp. 1532–1545, 2014, doi: 10.1109/TPAMI.2014.2300479.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016, doi: 10.1109/CVPR.2016.90.

[18] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A fast text detector with a single deep neural network," 31st AAAI Conf. Artif. Intell. AAAI 2017, pp. 4161–4167, 2017.

[19] Y. Zhu and J. Du, "Sliding line point regression for shape robust scene text detection," arXiv, pp. 3735–3740, 2018.

[20] S. R. Laskar, A. Dutta, P. Pakray, and S. Bandyopadhyay, "Neural machine translation: English to hindi," 2019 IEEE Conf. Inf. Commun. Technol. CICT 2019, pp. 25–30, 2019, doi: 10.1109/CICT48419.2019.9066238.

[21] W. Wang et al., "Shape robust text detection with progressive scale expansion network," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, no. c, pp. 9328–9337, 2019, doi: 10.1109/CVPR.2019.00956.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," Proc. - 2019 IEEE Intl Conf Parallel Distrib. Process. with Appl. Big Data Cloud Comput. Sustain. Comput. Commun. Soc. Comput. Networking, ISPA/BDCloud/SustainCom/SocialCom 2019, pp. 1500–1504, Dec. 2016, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217.

[23] F. Milletari, N. Navab, and S. A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," Proc. - 2016 4th Int. Conf. 3D Vision, 3DV 2016, pp. 565–571, 2016, doi: 10.1109/3DV.2016.79.

[24] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 761–769, 2016, doi: 10.1109/CVPR.2016.89.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 32nd Int. Conf. Mach. Learn. ICML 2015, vol. 1, pp. 448–456, 2015.

[27] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," J. Mach. Learn. Res., vol. 15, pp. 315–323, 2011.

[28] B. Leibe, J. Matas, N. Sebe, and M. Welling, "Preface," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9906

LNCS, pp. VII–IX, 2016, doi: 10.1007/978-3-319-46493-0.

[29] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," pp. 248–255, 2009, doi: 10.1109/cvprw.2009.5206848.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034, doi: 10.1109/ICCV.2015.123.

[31] A. Khan and A. Sarfaraz, "RNN-LSTM-GRU based language transformation," Soft Comput., vol. 23, no. 24, pp. 13007–13024, 2019, doi: 10.1007/s00500-019-04281-z.

