International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 10 | Oct 2019 www.irjet.net p-ISSN: 2395-0072

# Automatic Data Collection from Forms using Optical Character Recognition

**Siddhesh Shinde[1], Tanmey Saraiya[2], Jayesh Jain[3] Chhaya Narvekar[4]**

*[1,2,3]Student, Information Technology, Xavier Institute of Engineering, Mumbai, 400016 (India)*
*[4]Professor, Information Technology, Xavier Institute of Engineering, Mumbai, 400016 (India)*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** *Managing data from the offline forms has been very time consuming with a lot of manpower engaged in collecting user data from the form and updating it to the database. Optical Character Recognition (OCR) seems a very viable option for this case. However, blindly carrying out OCR won't produce any good results as there are many other elements in the form apart from the user's written data. This paper presents a structured way of scanning the paper-back form, collecting only the necessary, i.e. the user input data and update it to the database accordingly. The process involves using the scale-space technique to segment out the words from the input field and feeds them as input images to a CRNN (Convolutional Recurrent Neural Network) model to perform OCR on those words provide us with output ready to be updated into the database. This automated system cuts down the waiting time and also increases the efficiency as compared to the current system being used.*

***Key Words***: Optical Character Recognition, Deep Learning, Convolutional Neural Network, Recurrent Neural Network, Connectionist Temporal Classification

## 1. INTRODUCTION

Digitization is taking place in every field for instance in many private sectors as well as in government organizations. Similarly, Indian Railway bought digitization by making the railway ticket booking system online. But many people live in rural areas that lack any computational skills as well as they have almost negligible knowledge about the internet. Such people are left with only one option, i.e. to go personally to the ticket booking window and stand there throughout the day in a long queue waiting to book railway tickets. This has become a real concern to the common people and we feel the need that we can reduce this problem using our system. We present an automated upgrade to the existing railway ticket booking system, which will cut down the waiting time of the passengers for ticket booking with a large margin.

The method used in this system is a general method of performing OCR on human handwriting. The form filled by the user can be scanned and sent as an input to the system. We will then detect all the user input areas on the form as we aim to extract the only user-entered information. Then we perform OCR on these input fields and we will get the output as data which the user entered in the handwritten format.

## 2. PROPOSED NETWORK ARCHITECTURE

We propose a system consisting of a Word Segmentation algorithm followed by neural network architecture to perform optical character recognition on the words after segmenting them from the sentences. We take the form filled by the user as the input. From this scanned form, we then separate the input fields. These input field images are then forwarded as input to the Word Segmentation algorithm. The Word Segmentation algorithm forms a bounding box around every individual word in that sentence and returns the final image as an output. These segmented words can be then passed on to the neural network. The network architecture consists of three elements, the convolutional layers followed by the recurrent layers and finally by the transcription layer. It is the CRNN (Convolutional Recurrent Neural Network) model originally presented by Baoguang Shi, Xiang Bai, and Cong Yao [1].

Initially, the convolutional layers automatically extract a feature sequence from each input image. The output of the convolutional layers is passed on to the recurrent layers for making predictions for each frame of the feature sequence. Lastly, the transcription layer is used to translate the per-frame predictions by the recurrent layers into a label sequence. Though entire architecture is composed of different kinds of network architectures (eg. CNN and RNN), it can be jointly trained with one loss function.

### 2.1 Word Segmentation

After scanning the form and segregating all the input fields, we pass on these fields to the Word Segmentation algorithm. We want to separate every word/number from the given sentence.

This scale-space technique is proposed by R. Manmatha and N. Srimal. The method is fast and easy to implement. The algorithm takes an image of a line as input and outputs the segmented words. An anisotropic filter kernel is used which creates blobs corresponding to words. After that, a threshold is applied to the blob image and the output generated corresponds to words. A function is initially used to convert the input image to gray scale and to resize it to a fixed height. After preprocessing the image, it is then passed on to the main function which used the kernel to segment words from the sentence. The shape of the kernel is determined by various parameters that we need to pass to the function.

The overall working of the algorithm works in the following ways:
1. An image is passed as input to the function
2. The filter kernel is applied to the image.
3. A blob image is generated after thresholding.
4. Bounding boxes are generated around the words on the original image.

This algorithm gives good results on datasets with large inter-word-distances and small intra-word-distances like IAM. However, for other forms of datasets, or for testing real-world handwritten data, the results might not be very good and more complex approaches should be used instead.

## 2.2 Convolutional Neural Network

The words segmented after the input field goes through the Word Segmentation algorithm are then fed as input to CRNN model. The first component of CRNN model is CNN. So the images of words go through the CNN architecture.

A CNN is a Deep Learning algorithm which is fed with images as an input. It then uses weights and biases to extract various features from the images. The initial layers of CNN extract low-level features such as horizontal and vertical lines, curves, etc. while the CNN layers in deep extract much more complex features that may not be of any significance to human eyes. All these features learned by the Neural Network helps it to classify images. The significance of CNN as compared to a vanilla DNN (Deep Neural Network) comes into the picture when we want to work with images. The hidden features present in the images cannot be detected with plain DNN. While feeding an image as an input to a DNN, we need to flatten the pixels. This result in the deformation of the pixels from the original position and the prediction of classes would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

We can capture spatial and temporal dependencies in an image through a CNN using relevant filters. Filters (also known as kernels) are matrices used to perform convolution operation. The filter starts with the top-right corner of the image and performs a matrix multiplication operation. Then it moves to the right with a certain stride value and again it performs matrix multiplication. It keeps on doing this until it reaches the width of the image. After that, it hops down to the beginning (left) of the image and repeats the process until the entire image is traversed. In the case of an RGB image, the kernel also has a depth which is the same as that of the input image. All the results are added with the bias to give us a single depth channel Convoluted Feature Output.
We can perform the above operation in two ways, either by reducing the dimensions of the convoluted features as compared to the input images or by increasing or keeping the dimensionality constant. This is done by adjusting the level of padding layers we provide to the convoluted features.

A Pooling layer is used to reduce the spatial size of the Convolved Feature. It is done, to decrease the requirement of computational power and also to extract dominant features from the image. We generally use Max Pooling which returns the maximum value from the portion of the image covered by the kernel. It helps us reduce noise in the image which cannot be obtained from other types of pooling layers such as Average Pooling. The Convolutional Layer along with the Pooling Layer, together form one layer of a CNN. We can increase or decrease the number of such layers in our network depending upon the complexities of our dataset.

The output of the CNN model is the Convolution Features detected by our Neural Network architecture. These features help us to segregate different words according to the features native to those particular words.

## 2.3 Recurrent Neural Network (LSTM)

The Convoluted Features are then given as an input to the next element of our CRNN architecture, i.e. RNN. The feature matrix is fed to RNN which goes through LSTM cells which help us get sequential data. This results in a feature extracted matrix as output. These sequential data are very important as it helps us to predict the next element based on the information gathered from the previous elements.

Traditional Neural Networks cannot keep track of previous elements link it with the later ones and neither do Convolutional Neural Network help us in this case. This issue is addressed by Recurrent Neural Networks. They are networks with loops in them, allowing information to be passed from one step of the network to the next. A recurrent neural network can be represented as multiple copies of the same network, each network passing a message to a successor. Although RNNs are better than plain DNNs in recognizing sequential data, there is a limit to the number of information RNNs can store. When the gap between the relevant information and the point where it is needed to become very large, RNNs become unable to learn to connect the information. Fortunately, LSTM (Long Short Term Memory) cells can solve this issue.

LSTMs are a special kind of RNN, designed to avoid the long-term dependency problem and in turn remembering information for a long period. Unlike RNNs which have a single chain of repeating tanh layers, LSTMS has four repeating modules, interacting in a very special way.

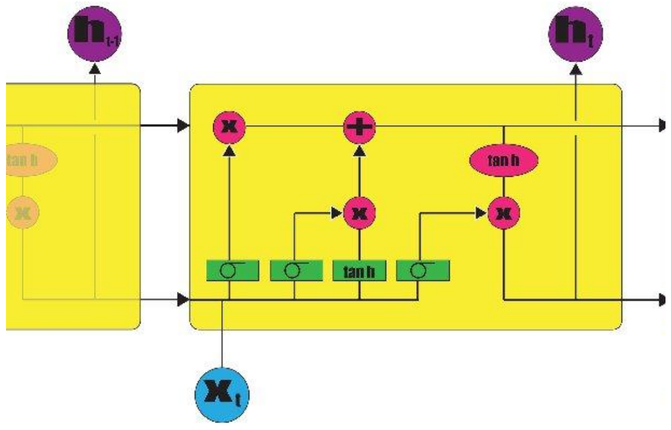

**Fig -1**: Network architecture of LSTM cell.

Figure 1 represents a basic flow of data in an LSTM cell. The key element of the LSTM cell is the cell state, which is the horizontal line at the top of a cell. It travels information from one cell to another. LSTM remove or add information to the cell state, carefully regulated by structures called gates. Gates are composed of sigmoid functions along with a pointwise multiplication operation. They decide whether to let information through them or not. The sigmoid function outputs a number between 0 and 1 indication how much of the information should be passed on. An LSTM has three such gates to control the cell state.

Each memory block in the original architecture contained an input gate and an output gate. The input gate controls the flow of input activations into the memory cell. The output gate controls the output flow of cell activations into the remaining network. Later, the forget gate was added to the memory block [2]. This addressed a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into subsequences. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. In addition, the modern LSTM architecture contains peephole connections from its internal cells to the gates in the same cell to learn precise timing of the outputs [3]. An LSTM network computes a mapping from an input sequence x = (x1,... , xT) to an output sequence y = (y1,..., yT) by calculating the network unit activations using the following equations iteratively from t = 1 to T:

$$i_t = \sigma(W_{ix} x_t + W_{im} m_{t-1} + W_{ic} c_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(W_{fx} x_t + W_{fm} m_{t-1} + W_{fc} c_{t-1} + b_f) \quad (2)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx} x_t + W_{cm} m_{t-1} + b_c) \quad (3)$$
$$o_t = \sigma(W_{ox} x_t + W_{om} m_{t-1} + W_{oc} c_t + b_o) \quad (4)$$
$$m_t = o_t \odot h(c_t) \quad (5)$$
$$y_t = \varphi(W_{ym} m_t + b_y) \quad (6)$$

where the W terms denote weight matrices, Wic , Wfc , Woc are diagonal weight matrix for peephole connections, the b terms denote bias vectors, σis the logistic sigmoid function, and i, f, o, and c are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the cell output activation vector m, ⊙ is the element-wise product of the vectors, g and h are the cell input and cell output activation functions, generally and in this paper tanh, and φ is the network output activation function, softmax in this paper [4].

## 2.4 Connectionist Temporal Classification

The output of LSTM cells is a matrix containing a score for each character at each time-step. This is the encoded text which we need to pass on to CTC as an input along with the corresponding ground-truth(GT). It tries all possible alignments of the GT text in the image and takes the sum of all scores. This way, the score of a GT text is high if the sum over the alignment-scores has a high value.

The encoding of duplicate characters in the case of CTC is resolved by pseudo-character. Pseudo-character is denoted as "-". When encoding a text, we can insert arbitrary many blanks at any position, which will be removed when decoding it. For example, the word "too" can be encoded as "---ttttttooo", or "-t-o-", or "to". The NN is trained to output an encoded text.

We then calculate the loss value for given pairs of images and GT texts. Figure 2 shows a matrix with two time-steps (t0 and t1) and three characters ("a", "b" and "-").The character-scores sum to 1 for each time-step.

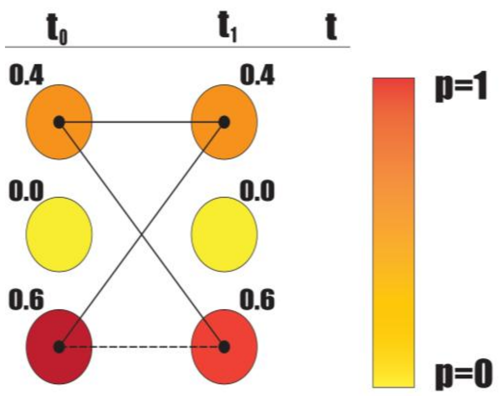

**Fig -2**: An example of a matrix containing character-score.

Thin lines are paths representing the text "a", while the thick dashed line is the only path representing the text "". The score for an alignment is then calculated by multiplying the corresponding character scores together. So for the above example, the score for the path "aa" is 0.4 x 0.4 = 0.16 while it is 0.4 x 0.6 = 0.24 for "a-" and 0.6 x 0.4 = 0.24 for "-a". We get a sum over the scores of all paths to get the score for a given GT. So for the given example, all possible paths to get "a" are: "aa", "a-" and "-a". Now summation of the scores over these paths will be 0.4 x 0.4 + 0.4 x 0.6 + 0.6 x 0.4 = 0.64. If the GT test is “”, then there is only one corresponding path ”--”, which yields an overall score of 0.6 x 0.6 = 3.6. This loss value is back-propagated to train our Neural Network and update the weights according to it.

Now to recognize text in previously unseen images, we use the best path decoding algorithm. The best path decoding algorithm consists of two steps:

1. Calculating the best path by taking the most likely character per time-step.
2. Undoing the encoding by first removing duplicate characters and then removing all blanks from the path. What remains represents the recognized text.

For example, consider there are three characters: “a”, “b” and “-” and 5 time-steps. So for time-steps t0 to t4, it will keep on selecting a character with the highest score. If the resultant text is “aaa-b”, after removing duplicate characters it becomes “a-b” and after removing the blank from the remaining path it becomes “ab”. This text is the recognized output text.

## 3. CONCLUSIONS

In this paper, we have presented a system that consists of a scale-space technique along with a neural network architecture that integrates the advantages of both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). We can take human inputs from forms and produce predictions. The above network avoids fully connected layers in the convolutional neural network which results in a much more compact and efficient model. All these properties make the network an excellent approach for image-based sequence recognition. The presented system will cut down the waiting time for scanning the tickets and feeding the data to the database. This will result in much smoother and efficient processing of the ticket booking system.

The proposed system is a general framework, thus it can be applied to other domains and problems (such as Chinese character recognition, or getting user input from some other paper-back forms as well), which involve sequence prediction in images. To further speed up the network, it can be trained on different handwriting datasets consisting of the words usually encountered while scanning a railway ticket. So further training the model may result in better performance.

## REFERENCES

[1] B. Shi, X. Bai and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 11, pp. 2298-2304, 1 Nov. 2017.

[2] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” Neural Computation, vol. 12, no. 10, pp. 2451–2471, 2000.

[3] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with LSTM recurrent networks,” Journal of Machine Learning Research, vol. 3, pp. 115–143, Mar. 2003.

[4] Haşim Sak, Andrew Senior, Françoise Beaufays, “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling,” INTERSPEECH, 14- 18 September 2014, Singapore.