

Introduction to the Problem Statement and Approach

Scene text recognition comprises of two tasks - (i) Text detection (ii) Text Recognition. The current problem statement deals with ***text recognition specifically in images consisting of names of streets, roads and other places in India written in Latin script, similar to what we see on direction boards on the roads.***

Although there are freely available Optical Character Recognition (OCR) APIs online, these resources generally use the character-by-character recognition paradigm in which individual characters of the text are predicted by DCNN models trained using labelled character images. However, this requires training a very strong and accurate character detector. Moreover, this method does not take advantage of the fact that words are not a random assortment of characters and that the contextual relation between a sequence of characters in a word could be used to accurately predict the next character of the word. There also exist methods that treat text recognition as an image classification task but these methods lead to less generalised text recognition models which are restricted to the lexicon of the dataset.

To deal with the drawbacks of the above mentioned methods, recent research in this field has focused on devising methods that treat text recognition as a sequence recognition problem thereby taking advantage of the underlying semantics. Broadly, such methods involve a preprocessing step in which the image is converted into a set of feature vectors which are then fed to a Recurrent Neural Network which outputs the predicted sequence of characters. Therefore, to complete the task at hand, an ***End-to-End trainable CRNN model [1] was trained using synthesised data to recognise names on the direction boards on the Indian roads.*** In this the image features are extracted using a Convolutional Neural Network in the pre-processing step and then fed to a Bidirectional Recurrent Neural Network, hence the name CRNN.

Details about the data synthesis and training of model are discussed in the following paragraphs.

Data Synthesis

Since there are no existent text recognition datasets for street names in India, there was a need to generate such images to train the model. The dictionary of text for generating the dataset was extracted from the GeoNames database [2] which covers all countries and contains over eleven million names of places. Filtering out only the places in India resulted in a lexicon of ~300k names.

In order to limit the complexity of the model due to computational constraints, the character set was limited to only the upper-case letters (A-Z) and the apostrophe character ('). Therefore, the images of the text generated are all in upper-case letters only. The dataset was synthesised using the TextRecognitionDataGenerator [3]. Grayscale images of height 32 pixels and variable width (depending on the length of the word) were generated to train the model. Various font styles and distortions (gaussian blurring and skewing) were used to generate the text images. A few samples are shown below:

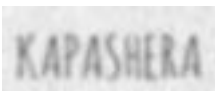


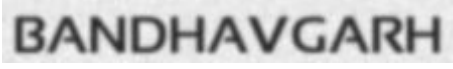
Image	<label>_<id>.jpg
	KAPASHERA_59575.jpg
	AMROLI_22025.jpg
	MEHDIKHERA_165467.jpg
	BANDHAVGARH_31189.jpg

Fig1: Sample of synthesised text images.

Model and Training Details

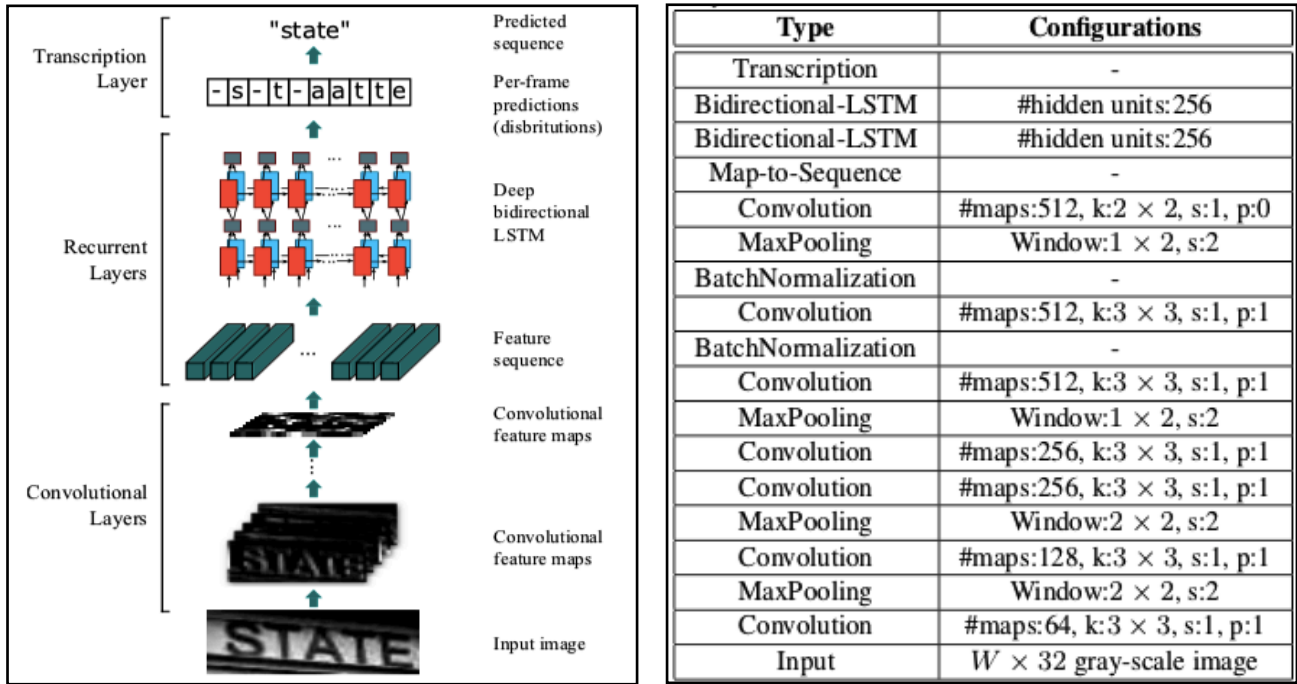


Fig 2: (a) Network Architecture (b) Network Configuration (k: kernel size, s: stride, p: padding).

The architecture consists of three parts:

1. **Convolutional layers**, that extract a feature sequence from the input image. Seven convolutional layers are present in the network architecture being used.
2. **Recurrent layers**, that predict a label distribution for each frame. Here, a bidirectional LSTM is used since in text images it is useful to incorporate context from both the forward and backward directions.
3. **Transcription layer**, that translates the per-frame predictions into the final label sequence.

The exact configuration of this architecture is shown in Fig 1(b). The width (W) of the input images to the CNN was fixed to 100 pixels. While images with $W > 100$ were resized to $W = 100$, images with $W < 100$ were padded with zeros.

Since the CNN and RNN architectures in the model are jointly trained using a single loss function, it is an end-to-end trainable model. The loss function used with CRNN models is the Connectionist Temporal Classification (CTC) loss function. The network was trained on 80% of the synthesised data using the Adam optimiser for 10 epochs with a batch size of 64. This took ~36 hours. The accuracy obtained by the trained model on test images which have been generated and are not obtained from real-life traffic boards was 99.30%.

Outputs obtained from the trained model on a few real-life images as well as generated images are shown below:

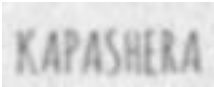

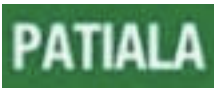
Image	Output
 (synthesised)	KAPASHERA
 (non-synthesised)	NIZAMUDDIN
 (non-synthesised)	PATIALA




Image	Output
 (synthesised)	PALAM
 (non-synthesised)	PALIAIVI
 (non-synthesised)	GHAZIABAD

Fig 3: Outputs on a few text images

Observations

The model seems to be giving excellent results on all of the synthesised images. However, in case of text cropped out from real sign board images (i.e, non-synthesised images) found online, the model does not seem to perform as well. This could be due to the fact that the fonts used in synthesising data are not exactly the same as those found on sign boards. Another reason could be distortions in the image. While during synthesis of data, noise and distortions are randomly and more uniformly added to the entire image, real images captured from cameras may have uneven distortions due to different angle at which light strikes the sign boards. This could lead to features which are very different from that present in the training dataset.

References

- [1] [An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition](#)
- [2] [The GeoNames geographical database](#)
- [3] [TRDG : TextRecognitionDataGenerator](#)