End-To-End Trainable Model for Text Recognition

Lakshmi Manoharan
Stanford University
mlakshmi@stanford.edu

Anelise Fassa Golden
Stanford University
fassa@stanford.com

1. Introduction

Scene text recognition is a challenging problem in Computer Vision, that is continued to be studied even today. Text recognition is a task, similar to Optical Character Recognition (popularly known as OCR), that detects the character present in an image. Scene text recognition, on the other hand, helps detect entire words from a given image. Successful implementation of a reasonably accurate model can aid in building assistive technology for the blind among many other useful applications in robotics and autonomous driving.

Our project aims to explore the problem of text recognition by treating it as a classical example of the broader problem of image-based sequence recognition [6]. Traditional Deep Convolutional Neural Networks (DCNN) cannot be applied in isolation, as in OCR solutions, as DCNNs generally require the input image to be of fixed dimensions. However, text in English can have varying number of characters; for example, consider the word *'at'* which has only 2 characters, whereas the word *respiratory* has 11 characters. Clearly, text recognition calls for more than just vanilla DCNNs.

Intuitively, it makes sense to address the problem of text recognition by viewing it as a sequence of characters, rather than as a single image. Given the performance of Recurrent Neural Networks in the Natural Language Processing (NLP) space, it is reasonable to explore RNNs as a viable solution to the problem of text recognition, which is just character-sequence recognition. In this project, we study the application of a combination of Convolutional and Recurrent Neural Networks (CRNNs) to the problem of text recognition.

2. Problem Statement

We define text recognition as the task of identifying the word present in a given image. Unlike scene text recognition, which involves *text localization* (identifying the location of text in the image) and *text recognition* (identifying the word present in the image localized by a bounding box), our project focuses on the latter, i.e. identifying the word present in a 'cropped word instance', as illustrated in Figure 1.

Euphony
'Euphony'
Input
Case-sensitive Output

Figure 1. A typical example of input and output of our model

In this project, we train and test with words obtained using the English alphabet. We propose to use the following synthetic datasets for training the model: (i) MJSynth dataset[3] (ii) SynthText dataset[2]. MJSynth contains 9 million synthetically generated word instances covering 90k English words. SynthText contains 8 million synthetic word instances obtained from 800k images.

We propose to use test sets of the following datasets for performance evaluation: (i) ICDAR 2003 (**IC03**)[5] (ii) ICDAR 2013 (**IC13**) [4] (iii) Street View Text (**SVT**) [7]. The test datasets for **IC03**, **IC13** and **SVT** contain 860, 1015 and 647 cropped word instances respectively. Also, **IC03** and **IC13** are associated with a 50k word lexicon consisting of the words in the Hunspell spell-checking dictionary, whereas each word image in the **SVT** dataset is associated with a 50 words lexicon.

The performance of the model is evaluated based on the *recognition accuracy*, defined as below:

$$\text{Recognition Accuracy} = \frac{\text{\# Words correctly recognized}}{\text{Total number of words}}$$

We also present the *edit distance(prediction, ground_truth)*, defined as the minimum number of edit operations (insert, remove, replace) to convert the prediction to the ground truth label.

3. Technical Approach

The network architecture of CRNN, illustrated in Figure 2, consists of the following components, as outlined by Shi *et al.*:

1. Convolutional Layers, which extracts the features from the image.
2. Recurrent Layers, which use the information from the feature sequence to generate a sequence of characters as intermediate output (per-frame predictions).
3. Transcription Layer, that decodes the intermediate output (per-frame predictions) to our actual prediction (label-sequence) using Connectionist Temporal Classifiers (CTC), to be explained below.

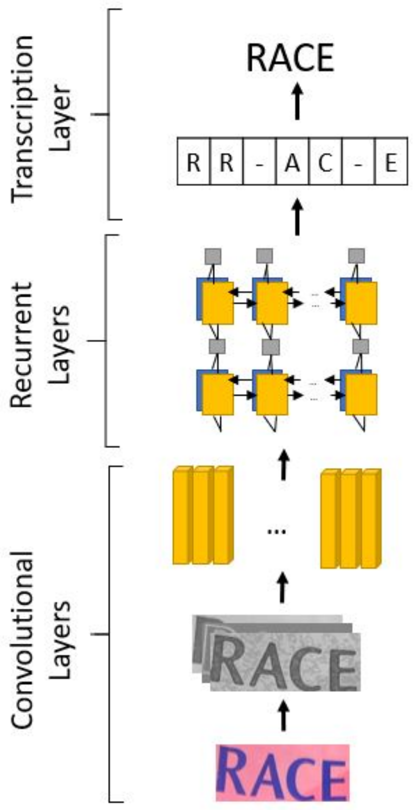

Figure 2. The CRNN Architecture consists of (1) Convolutional Layers, that extract image features (2) Recurrent Layers, that generate per-frame predictions (3) Transcription Layer, that convert the per-frame prediction to the final label sequence

3.1. Feature Extraction

The Convolutional Neural Network has an architecture similar to most standard CNN models. As illustrated in Table 1, we remove the fully-connected layers removed as we intend to extract the image features into feature sequence vectors to be fed into the Recurrent Layers in CRNN. Prior to feeding to the CNN, the input image is resized to have the same height. In our case, we have chosen to scale it to 32px as recommended in [6].

Layer	**Configurations**
Input	W x 32 Grayscale Image
Convolution	Filters: 64, k:3x3, s:1, p:1
MaxPooling	w:2x2, s:2
Convolution	Filters: 128, k:3x3, s:1, p:1
MaxPooling	w:2x2, s:2
Convolution	Filters: 256, k:3x3, s:1, p:1
Convolution	Filters: 256, k:3x3, s:1, p:1
MaxPooling	w:1x2, s:1x2
Convolution	Filters: 512, k:3x3, s:1, p:1
Batch Normalization	-
Convolution	Filters: 512, k:3x3, s:1, p:1
Batch Normalization	-
MaxPooling	w:1x2, s:1x2
Convolution	Filters: 512, k:3x3, s:1, p:0
Map-To-Sequence	-
Bidirectional LSTM	units:256
Bidirectional LSTM	units:256
Transcription	-

Table 1. CRNN Configuration ('k'-Kernel Size, 's'- Stride, 'p'-Padding, 'w'- Window Size)

We obtain the feature sequences by taking a depth slice of the activation maps with 1px width. In other words, the *i*-th feature sequence is the concatenation of the *i*-th column of all the activation maps, where each column is 1px wide.

3.2. Per-Frame Predictions using RNNs

The feature sequences obtained using the CNN is fed into a bidirectional LSTM neural network with 256 hidden units, the outputs of which are fed into another bidirectional LSTM neural network with the same configuration. Note that the two LSTM networks do not share network parameters and are trained independent of one another. As we deal only with English alphabets, we downproject the outputs of the RNN using a 53-unit affine layer (no non-linearity involved). The 53 units account for 53 classes for the per-frame prediction. The included classes are all lower-case and upper-case A-Z characters (26*2) and 1 class for the blank character.

3.3. Transcription Layer

The transcription layer converts the per-frame predictions into the final label sequence by using a Connectionist Temporal Classifier (CTC) [1]. Given the per-frame predictions, the CTC Decoder selects the path to the label sequence with highest conditional probability. This is computed by using the forward-backward algorithm described in Graves *et al*. We use the Tensorflow implementation of the CTC Beam Search Decoder to implement the transcription layer.

4. Preliminary Results

A baseline model for CRNN based on the architecture defined in Table 1 was built using Tensorflow. The baseline model was validated by first overfitting to a sample dataset of 82 cropped word images. This dataset will be henceforth referred to as Sample. The model was further validated by training on a subset of MJSynth dataset containing 600k training examples, to be hereafter referred to as MJSynth600.

4.1. Preliminary Experiments

As outlined above, the goal of the preliminary experiments was to build a baseline model that could both overfit to the Sample dataset and also learn on a larger dataset. To achieve this goal, we started by training the baseline model with Stochastic Gradient Descent on the Sample dataset.

We observed that when using SGD, the loss curve was decreasing very slowly and the accuracy improvements also showed slow progress. We decided to experiment with a more advanced optimizer, specifically we used Adadelta, which uses a per-dimension adaptive learning rate, coupled with exponential decay of the learning rate. Empirically, we found that Adadelta offered much faster convergence.

While using Adadelta, tuning the learning rate, as well as the learning rate decay, and decay step were critical to achieving good performance in the training process.

4.2. Evaluation Metrics

The baseline model validation leveraged three main evaluation metrics: loss, accuracy and edit distance. Monitoring of the learning rate throughout the experiments was also key to initial parameter tuning.

The graph in Figure 3 depicts the accuracy of about 95% which was achieved on the Sample Dataset in just about 6k iterations. Showing that the model could overfit on this small dataset helped assure us that the model was implemented correctly.

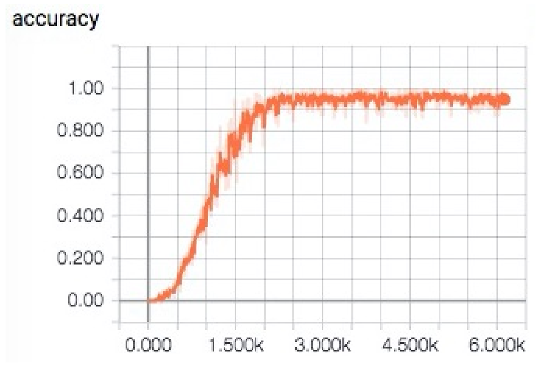

Figure 3. Model trained on Sample dataset with Adadelta optimizer

Furthermore, when training on the MJSynth600 dataset, we observed loss of around 2 (Figure 4) and accuracy around 80% (Figure 5) after 22k iterations (approximately 9 epochs, with batch size set to 256).

We also notice that the loss continues to decrease, implying that the model can be trained for longer to achieve optimal performance.

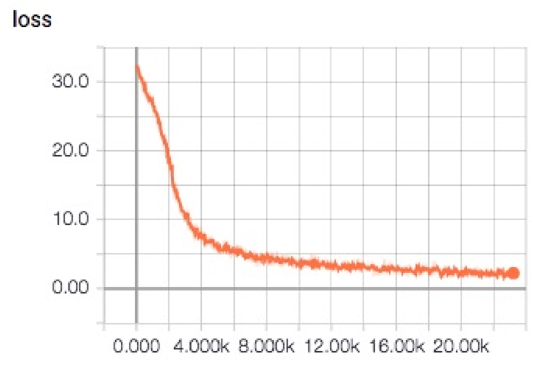

Figure 4. Loss for model trained on MJSynth600 dataset with Adadelta optimizer

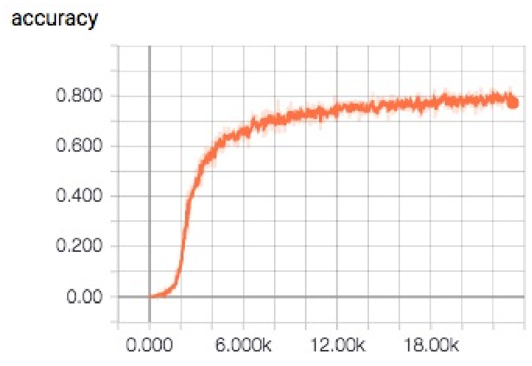

Figure 5. Accuracy for model trained on MJSynth600 dataset with Adadelta optimizer

These are encouraging preliminary results and we are optimistic that further tuning of this model will produce positive results on the task of word recognition.

As discussed in Section 2, we further plan to train the model using the Synth dataset [2], which contain word instances placed in natural scene images, taking into account the scene layout. We propose to do an exhaustive hyperparameter search for this final model, which would be evaluated using benchmark text recognition datasets mentioned in Secion 2. In light of the results from previous research in this domain, we are certain that effectively using the lexicon associated with the images can further enhance the accuracy of the predictions.

References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[2] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[4] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. De Las Heras. Icdar 2013 robust reading competition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1484–1493. IEEE, 2013.

[5] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 682–687. Citeseer, 2003.

[6] 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. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2017.

[7] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011.