Limitations and Improvements of CRNN for OCR in Diverse Visual Environments

Sumi Hwang
Department of Computer Science
Cranberry-Lemon University
hwangsm614@gmail.com

June 5, 2025

Abstract

This study aims to improve the performance of Optical Character Recognition (OCR) for recognizing text within images. For this purpose, a pipeline was constructed by combining a Keras Text Detection model and a custom-implemented CRNN (Convolutional Recurrent Neural Network) Text Recognition model. The model was trained using the MJSynth dataset, and stable learning was induced through Adadelta optimizer and learning rate adjustments. The performance of the proposed CRNN recognizer was relatively evaluated using edit distance, assuming the Keras recognizer's results as the ground truth. While an edit distance of 0.94 was confirmed in basic dataset tests, performance degraded to an edit distance of 0.57 in diverse font environments. Furthermore, difficulties in recognition were identified in special environments such as vertical text and design posters. To address these issues, this paper proposes the necessity of securing dataset diversity, utilizing precise character region extraction based on segmentation, and modifying the architecture or adding data for vertical text processing. This research holds significance in experimentally verifying the practical applicability and limitations of CRNN-based OCR models and presenting concrete directions for future performance improvements.

1 Introduction

Optical Character Recognition (OCR) is a technology that recognizes characters within images and is utilized in various industrial and real-life applications, such as automatic license plate recognition and optical credit card recognition. With recent advancements in deep learning technology, OCR has achieved remarkable performance improvements. However, challenges persist in accurately recognizing text in demanding environments characterized by diverse fonts, complex backgrounds, unconventional text layouts, and low image resolution, often leading to a decline in recognition accuracy. This study aims to construct an

OCR system by combining a Keras text detection model and a CRNN-based text recognition model, and proposes methods to improve recognition errors that occur in diverse font and vertical text environments.

2 Method

2.1 OCR

OCR can be broadly divided into Text Detection and Text Recognition. Text Detection is the task of finding characters in an image, primarily using object detection or segmentation techniques. Text Recognition is the task of identifying what characters are in the detected regions, with CRNN being a representative method. CRNN (Convolutional Recurrent Neural Network) combines CNN and RNN. The CNN acts as a feature extractor, extracting features from unsegmented data. The RNN processes these extracted features, converted into a sequence format via a Map-To-Sequence layer, as input. CRNN utilizes Connectionist Temporal Classification (CTC) for processing unsegmented data. Figure 1

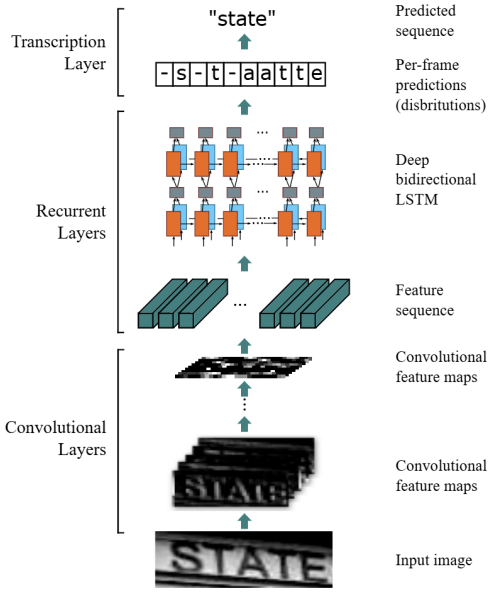

Figure 1: The network architecture. [2]

2.2 Model

An OCR pipeline was constructed by combining a pre-trained text detection model provided by the Keras library and a custom-implemented CRNN-based text recognition model. The Keras text detection model used was keras-ocr,

which utilizes the segmentation-based CRAFT [1]. Target characters were limited to a total of 36 characters, including uppercase alphabets (A-Z) and numbers (0-9), for training and recognition. When the text detection model detects character regions in an input image and outputs bounding boxes, these are used to crop the original image to only the parts containing text, and text recognition is performed using these cropped images. The CRNN model, as shown in Figure 2, uses 7 convolutional layers and 2 bidirectional LSTMs.

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 $\times$ 2, s:1, p:0
MaxPooling	Window:1 $\times$ 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 $\times$ 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 $\times$ 3, s:1, p:1
MaxPooling	Window:1 $\times$ 2, s:2
Convolution	#maps:256, k:3 $\times$ 3, s:1, p:1
Convolution	#maps:256, k:3 $\times$ 3, s:1, p:1
MaxPooling	Window:2 $\times$ 2, s:2
Convolution	#maps:128, k:3 $\times$ 3, s:1, p:1
MaxPooling	Window:2 $\times$ 2, s:2
Convolution	#maps:64, k:3 $\times$ 3, s:1, p:1
Input	$W \times 32$ gray-scale image

Figure 2: The network architecture. [2]

The model was trained on the MJSynth dataset, a synthetic dataset containing 90,000 images and 9 million words. An MJSynth dataset class for Keras model training was implemented using LMDB. The batch size was set to 128, input image size to (100, 32), and max text length to 22. To enable model learning, each character was encoded by converting it to a class index. Encoding and decoding were performed using a blank character ('-') and the target characters consisting of uppercase alphabets and numbers. The model's loss was calculated as CTC Loss using `K.ctc_batch_cost()` provided by Keras. The Adadelta optimizer was used for the model, with a learning rate of 0.1, clipnorm of 5, and 20 epochs. Performance was evaluated by calculating the edit distance (Levenshtein distance) between the prediction results of the Keras recognizer and the CRNN recognizer. Edit distance is a metric representing the difference between two strings, meaning the minimum number of insertion, deletion, or substitution operations. Manually labeling ground truth for new images during experiments requires significant time and effort, and it can be difficult for humans to recognize characters depending on the font or image resolution. Therefore, in this study, the prediction results of the keras-ocr recognizer, which is pre-trained

on a large-scale dataset, were assumed as the ground truth. While the Keras recognizer may not be 100% accurate, it is known to be robust, allowing for a relative performance evaluation against the CRNN recognizer.

3 Experiment

3.1 Hyperparameter Adjustment

3.1.1 Optimizer Change

When using the Adam optimizer (learning rate=0.1), the loss reached 226, and training was terminated by early stopping at the 4th epoch. This could be due to Adam's initial learning rate being set too high or its adaptive learning rate characteristic causing unstable learning in certain situations. The abnormally high loss value suggests a gradient exploding phenomenon, where gradients become excessively large, causing weight updates to diverge. In this experiment, Adam showed unstable behavior in the CTC-based model. This indicates that although Adam is an effective optimizer widely used in various deep learning models, it may not guarantee optimal performance depending on the specific model architecture, dataset, or initial hyperparameter settings. Using Adadelta resulted in stable convergence during training. Adadelta does not require manual setting of the learning rate and automatically adjusts it based on the magnitude of the gradients, which is interpreted as showing robustness against the gradient exploding problem.

3.1.2 Learning Rate Change

When the learning rate of the Adadelta optimizer was set to 0.1, the train loss continued to decrease after 4 epochs, but the validation loss no longer decreased and stagnated, suggesting overfitting. Therefore, by lowering the learning rate to 0.01 and conducting training for about 20 epochs, both train and validation loss showed a stable decreasing trend. This is judged to have contributed to mitigating overfitting and improving the model's generalization performance. This confirmed that there is room for performance improvement through additional epoch training. Due to time constraints, further training was not conducted, but it is expected that extending training to 50-100 epochs would yield better performance.

3.2 Performance Evaluation

3.2.1 Overall Performance

Using the model trained with a learning rate of 0.1 and Adadelta, testing on MJSynth's test dataset showed an edit distance of approximately 0.94. As an example, when outputting results, 9 out of 10 images were correctly recognized, with the remaining one misrecognizing a double 'l' as a single 'l', but otherwise

showing high recognition rates. Experiments were conducted with this model, which was suspected of overfitting, as additional training could not be performed.

3.2.2 Analysis of Failure Cases

Diverse Fonts

When experimenting with various fonts, the Keras recognizer generally performed well, whereas the CRNN recognizer showed an edit distance of 0.57. This significantly lower recognition rate seems to be because the CRNN model, using a learning rate of 0.1, experienced validation loss stagnation after 4 epochs, potentially leading to overfitting to specific fonts or styles. It could also be due to insufficient diversity in the fonts included in the training dataset. It appears necessary to add a dataset with diverse fonts for further training. Furthermore, using a segmentation method instead of an object detection method in the text detection process is expected to yield better performance. In this study, for text recognition, only the cropped image of the bounding box detected is used. Since a bounding box represents the text area as a rectangle, it can include background information or omit parts of characters, especially with slanted or complex fonts, or densely packed text. In contrast, segmentation can accurately extract text regions at the pixel level, potentially improving recognition performance by providing the recognizer with more refined text images. When characters have clear lines and distinct color contrast with the background, performing segmentation to remove the background and separate characters using line and color differences, then recognizing this like a handwriting prediction task, would significantly improve performance.

Vertical Text

Since poor performance on fonts was confirmed above, this experiment was conducted with a basic font. Neither the Keras recognizer nor the CRNN recognizer could properly recognize vertically written text. There are two types of vertical text: the common method where characters are arranged vertically, and a method where characters are arranged horizontally but rotated 90 degrees to appear vertical, and another where characters are upright but arranged vertically. The RNN layer in the CRNN model is generally designed and trained to process sequences from left to right. Therefore, for vertical text, the sequence order is completely different from existing training data, making it difficult for the RNN to learn meaningful patterns. For text rotated 90 degrees, the text detection part drew bounding boxes correctly, suggesting that this could be resolved by adding a preprocessing step to detect character orientation and rotate it. For upright characters arranged vertically, bounding boxes are drawn correctly, but recognition fails completely. Adding a vertical text dataset seems necessary, and methods for the computer to distinguish between these two types should be devised. It might also be necessary to recognize each character individually and then reassemble them, or to modify the CRNN architecture to handle vertical sequences as well as horizontal ones.

Design Posters

Similar to previous experiments, text detection was acceptable, but both Keras and CRNN models performed poorly in recognition. In posters with many design elements, unlike typical document text, colors may change mid-character, and text is arranged at various angles and styles, deviating from horizontal or vertical writing. Characters may also overlap, or there might be gradients, color changes, or texture variations within characters. Occasionally, text might be blurred, or the background might be very complex, making distinction difficult. In such cases, bounding box-based detection struggles to accurately separate individual characters. Instance segmentation, by separating each character at the pixel level, can effectively handle overlapping characters and recognize characters with internal visual variations, thus an improvement in recognition rate can be expected.

4 Conclusion

In this study, an OCR pipeline was constructed by combining Keras-OCR's CRAFT Text Detection model and a custom-implemented CRNN Text Recognition model, and its performance in various environments was experimentally evaluated. Through hyperparameter tuning, it was confirmed that the Adadelta optimizer is more stable for training CTC-based CRNN models than the Adam optimizer, and that adjusting the learning rate can mitigate overfitting and enhance the model's generalization capabilities.

In the basic performance evaluation using the MJSynth test dataset, the proposed CRNN model showed an edit distance of 0.94. However, in diverse font environments, the performance degraded to an edit distance of 0.57. This is analyzed to be due to the lack of font diversity in the training dataset and the potential overfitting of the current model. Furthermore, for vertical text and poster images with many design elements, neither the Keras recognizer nor the proposed CRNN recognizer achieved satisfactory recognition results, suggesting that the existing CRNN architecture is specialized for horizontal sequence processing and is vulnerable to complex visual elements and layout changes.

Additionally, there was an issue where the digit '9' repeatedly appeared at the end of character recognition inference results. This was caused by padding, used to make the RNN's input size uniform, being set to -1, which decoded to the last index of the target characters (the -1 index). This error was resolved by not reflecting the index designated as -1 during decoding. However, the current post-processing method of ignoring padding indices is a temporary solution. For a substantial fix, model-level improvements are needed, such as adjusting logits values so that predictions corresponding to padding do not map to any character during model output. In conclusion, this study presented the possibilities and improvement measures for CRNN-based OCR systems by analyzing their construction process and performance in various environments. The proposed improvement measures are expected to contribute to the development of more

robust and accurate OCR systems in the future.

References

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4715–4723, 2019.

[2] Baoguang Shi, Xiang Bai, and Cong 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, 2016.