A Study on CRNN-CTC Based Single-Word OCR Using Synthetic LMDB Dataset

Bohyeuk Yim

dlaqhgur4911@gmail.com

Abstract

This paper analyzes the behavior of a Convolutional Recurrent Neural Network (CRNN)-based optical character recognition (OCR) model trained on the synthetic MJSynth dataset. The model combines a CNN encoder with a bidirectional LSTM and a Connectionist Temporal Classification (CTC) decoder, achieving a word accuracy of 88.80% and a character error rate (CER) of 3.31% on a held-out synthetic test set. To better understand its limitations, we examine errors with respect to sequence length and character topology. The results show a noticeable decline in accuracy for words longer than ten characters, which appears to be associated with the fixed-width input normalization and the relative scarcity of long-word samples in the training distribution. These factors, together with spatial downsampling, contribute to frequent deletions of thin-stroke characters such as `I`*,* `l`*, and* `t`*. We also evaluate the model on real scanned documents (FUNSD) and observe a substantial performance gap. An analysis using ground-truth bounding boxes indicates that this gap is largely linked to recognition difficulties in the presence of fragmented strokes and punctuation not included in the training vocabulary. The study highlights conditions under which fixed-width CRNN models underperform and provides observations that may inform future improvements to OCR robustness.*

1. Introduction

Optical character recognition (OCR) is a fundamental component in document understanding, automated data extraction, and human–computer interaction systems. While recent OCR pipelines increasingly incorporate large-scale transformer-based architectures, lightweight sequence-based models remain widely used in resource-constrained environments due to their efficiency and relatively simple deployment. Among these models, the Convolutional Recurrent Neural Network (CRNN) [3] combined with Connectionist Temporal Classification (CTC) [2] has become a common baseline for word-level text recognition.

In the CRNN–CTC framework, a convolutional encoder extracts visual features from an input word image, which are then reshaped into a one-dimensional sequence and processed by a recurrent module to capture contextual dependencies across characters. The CTC decoder provides alignment-free transcription, enabling training without character-level bounding box annotations. Due to its compact architecture, the CRNN–CTC paradigm remains a practical choice in scenarios requiring high-speed inference and moderate robustness to distortions.

Despite its advantages, several architectural characteristics of CRNN–CTC models continue to influence their behavior in nontrivial ways. Prior work such as Baek et al. [1] has discussed how fixed-size input normalization can introduce aspect-ratio distortion and reduce feature fidelity. Building on these observations, we examine how the fixed temporal resolution of the encoder–decoder pipeline interacts with word length. In our experiments, recognition accuracy decreases noticeably for words longer than approximately ten characters, suggesting that a limited output sequence length ($T = 35$ in our setting) may restrict the effective alignment capacity of the CTC decoder.

We also explore error patterns that emerge under these conditions. Deletion errors—particularly among thin-stroke characters such as `I`, `l`, and `t`—occur frequently, indicating that spatial downsampling may reduce the visibility of fine structural details. In addition, although synthetic datasets such as MJSynth provide large quantities of training data, their visual statistics differ from those of real scanned documents, which often contain fragmented strokes, background noise, and punctuation not present in the training vocabulary. Understanding how these factors affect recognition performance can provide useful insight into the limitations of fixed-width CRNN models.

In this study, we analyze a CRNN–CTC model trained on a synthetic dataset and evaluate its behavior across both synthetic and real-world conditions. Our work focuses on the following questions:

- How does the fixed-resolution constraint influence recognition performance across different word lengths?
- How are deletion errors related to stroke width and continuity in downsampled representations?

- To what extent does recognition performance change on real scanned documents (FUNSD) when detection-related factors are controlled?

Our contributions are summarized as follows:

- We analyze the behavior of a standard CRNN–CTC architecture and report a consistent decline in recognition accuracy for words longer than ten characters, which may be associated with the fixed-width normalization and limited sequence length.
- We examine character-level errors and find that deletion errors dominate, especially for thin-stroke characters, suggesting sensitivity to downsampling-induced feature loss.
- We quantify the performance gap between synthetic and real-world documents using an Oracle Test on FUNSD and describe factors that appear to contribute to recognition difficulty in real scanned environments.

Through these observations, we aim to provide a clearer understanding of the operational boundaries of CRNN–CTC models and highlight considerations that may guide future improvements in OCR robustness.

2. Methodology

This section describes the CRNN–CTC architecture used in our experiments, the datasets for training and evaluation, and the implementation details including preprocessing and optimization strategies.

2.1. Network Architecture

Our text recognition model follows a customized CRNN design tailored for fixed-size input images. The network processes an input tensor of shape $(B, 3, 32, 100)$ through three stages: convolutional feature extraction, sequence modeling, and transcription. The overall architecture is illustrated in Figure 1, and a detailed layer configuration is summarized in Table 1.

2.1.1. Feature Extraction (CNN)

We adopt a VGG-style convolutional encoder consisting of seven convolutional layers:

- **Convolutional Layers.** All convolutional layers use 3×3 kernels, except for the final layer, which uses a 2×2 kernel. Batch Normalization is applied to the fifth and sixth layers to stabilize training.
- **Pooling Strategy.** The pooling configuration combines 2×2 max-pooling (for early layers) with later 1×2 pooling operations that compress the width more aggressively than the height. This design maintains some vertical resolution, leading to a final feature map of size $(512, 7, 5)$.
- **Map-to-Sequence.** Instead of collapsing the height dimension to one as in common CRNN variants, we flatten both spatial dimensions (7×5) to produce a sequence of $T = 35$ feature vectors. This retains vertical information, but constrains the available temporal resolution during CTC decoding—a property analyzed in Section 3.3.

2.1.2. Sequence Modeling (RNN)

The flattened feature sequence $(B, 35, 512)$ is processed by two Bi-LSTM layers. Each layer has a hidden size of 256, producing a bidirectional output of 512 units per timestep. This recurrent structure models contextual relationships across the 35 feature frames.

2.1.3. Transcription (CTC)

A linear projection maps the LSTM outputs to 37 character classes (26 uppercase letters, 10 digits, and a blank token). A Log-Softmax activation produces per-frame log probabilities required for the CTC loss.

Table 1. Detailed configuration of the CRNN architecture.

Layer Type	**Kernel / Stride**	**Output Shape** (C, H, W)
Input	-	$(3, 32, 100)$
Conv1 + ReLU	$3 \times 3, 1$	$(64, 32, 100)$
MaxPool1	$2 \times 2, 2$	$(64, 16, 50)$
Conv2 + ReLU	$3 \times 3, 1$	$(128, 16, 50)$
MaxPool2	$2 \times 2, 2$	$(128, 8, 25)$
Conv3, 4 + ReLU	$3 \times 3, 1$	$(256, 8, 25)$
MaxPool3	$1 \times 2, (1, 2)$	$(256, 8, 12)$
Conv5, 6 + BN	$3 \times 3, 1$	$(512, 8, 12)$
MaxPool4	$1 \times 2, (1, 2)$	$(512, 8, 6)$
Conv7 + ReLU	$2 \times 2, 1$	$(512, 7, 5)$
Map-to-Seq	Flatten	$(T = 35, D = 512)$
Bi-LSTM ×2	Hidden=256	$(35, 512)$
Linear (CTC)	-	$(35, 37)$

2.2. Datasets

We evaluate the model using two datasets with differing characteristics:

- **MJSynth (Synthetic).** The model is trained on the MJSynth dataset, which contains roughly 7.2 million synthetically rendered word images. A held-out subset of 25,000 samples is used for internal evaluation.
- **FUNSD (Real-world).** The FUNSD dataset consists of real scanned forms containing noise, irregular stroke patterns, and diverse layout structures. This dataset enables examination of model behavior under visually challenging, real-world text conditions.

2.3. Implementation Details

All input images are resized to a fixed resolution of 100×32:

$$W_{\text{target}} = 100, \quad H_{\text{target}} = 32. \tag{1}$$

While this standardization simplifies batching, it also constrains the spatial detail available for long word sequences. After convolutional downsampling, the sequence length is fixed at $T = 35$.

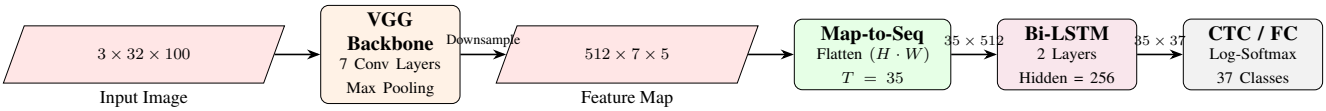

Figure 1. **Overview of the CRNN architecture.** The convolutional encoder downsamples the input image while preserving both spatial dimensions. The final feature map is flattened to a sequence of length $T = 35$, processed by a Bi-LSTM, and decoded using CTC.

The model is trained with the Adadelta optimizer (learning rate 1.0) and batch size 128. Mild data augmentation, including random rotation and color jittering, is applied. No additional regularization techniques are used, allowing the evaluation to reflect the intrinsic behavior of the base CRNN architecture.

3. Experiments

In this section, we report the quantitative results of the CRNN–CTC model and analyze its behavior with respect to sequence length, character-level errors, and domain shift between synthetic and real-world data.

3.1. Experimental Setup

We evaluate the model using three metrics: word accuracy (Acc), character error rate (CER), and normalized edit distance (NED). A prediction is counted as correct only when the predicted string exactly matches the ground truth (case-insensitive).

3.2. Performance on Synthetic Data

The model trained on MJSynth achieves a word accuracy of **88.80%** and a CER of **3.31%** on the held-out test set (25,000 samples). The 95% confidence interval for accuracy is $[88.41\%, 89.19\%]$. These results are in line with typical performance of lightweight CRNN-based recognizers. However, aggregate metrics do not reveal how recognition varies across word lengths or character types, so we further decompose the performance.

3.3. Length-Dependent Behavior

To examine how sequence length affects recognition, we measure accuracy as a function of word length and compare this with the length distribution of the training data.

Table 2 summarizes the accuracy grouped by word length.

- **Short to medium words (1–10 chars):** The model maintains relatively high accuracy (above 86%, and above 89% for lengths up to 9).
- **Long words (>10 chars):** Accuracy decreases for longer sequences, dropping to 76.2% for lengths 11–14 and to 45.8% for length ≥ 15.

To put this in context, Table 3 reports the length distribution of the MJSynth training data. Most samples fall in the 6–10 character range, while very long words constitute a small fraction of the corpus.

Table 2. Recognition accuracy by word length on the MJSynth test set.

Word Length	Count	Accuracy
1–5	2,522	**93.7%**
6–10	8,655	**89.5%**
11–14	4,500	**76.2%**
≥ 15	165	**45.8%**

Note: Longer words are relatively infrequent in the training set (e.g., length 17 is about 0.10% of the training data).

Table 3. **Distribution of word lengths in the MJSynth training data.** The dataset exhibits a pronounced long-tail distribution: sequences with length ≥ 15 are relatively rare.

Category	Length	Count	Percentage
Short	1–5	1,030,870	14.25%
Medium (Majority)	**6–10**	**4,971,285**	**68.80%**
Long	11–14	1,156,114	16.00%
Very Long	**15**	**41,093**	**0.57%**
	16	**15,724**	**0.22%**
	17	**7,147**	**0.10%**
	≥ 18	3,353	0.05%
Total	All	7,225,586	100.00%

Taken together, these results suggest that performance on long words is influenced by two factors: (i) the fixed sequence length ($T = 35$), which limits the number of timesteps available for CTC alignment as word length grows, and (ii) the relative scarcity of very long words in the training data, which may reduce the model's exposure to such patterns during optimization.

We also attempted a naive architectural modification in which the input width was doubled (from 100×32 to 200×32), resulting in a sequence length of $T = 70$ instead of $T = 35$. Under the same training setup, this change led to a substantial degradation in overall accuracy. This suggests that simply increasing the sequence length does not trivially resolve the long-sequence degradation; instead, it introduces additional optimization challenges and expands the CTC alignment space, which can negatively affect convergence. Designing architectures that provide greater sequence capacity while remaining trainable therefore appears to be a non-trivial problem and is left for future work.

3.4. Error Topology Analysis

To better understand the visual limitations of the model, we analyze character-level error statistics. Overall, **character deletion** is the most frequent error type.

- **Thin-stroke deletions.** The characters most frequently deleted include `I`, `l`, `t`, and `E`. This pattern is consistent with the idea that downsampling from a width of 100 pixels can make thin vertical strokes less distinguishable from the background or cause them to be suppressed during max-pooling.
- **Morphological confusion.** Substitution errors are often observed between shape-similar pairs such as `A` $\leftrightarrow$ `O` and `I` $\leftrightarrow$ `L`. This suggests that resizing and feature compression may reduce the separability of visually similar glyphs.

Although we do not include feature map visualizations here, the character-level statistics indicate that the deletion rate for thin characters (e.g., `I`, `l`, `t`) is substantially higher (approximately four times) than for wider characters such as `M` and `W`. This difference is consistent with the limited spatial resolution of the final feature map (7×5), which may be insufficient to reliably represent very narrow strokes.

3.5. Real-World Domain Gap Analysis

To examine the behavior of the model under real scanned document conditions, we evaluate recognition performance on a subset of FUNSD images. Each image contains approximately 100 annotated word-level bounding boxes, and we measure the model's accuracy and average edit distance for each image independently. Unlike the synthetic test set, these samples exhibit substantial noise, stroke fragmentation, and layout variability.

Table 4 summarizes the results across ten representative images.

Table 4. Recognition performance on selected FUNSD images. Each row corresponds to all word-level boxes in a single image (typically 100 words).

Image	Words	Correct	Accuracy	Avg. ED
0000971160.png	100	59	59.00%	0.61
0000989556.png	100	57	57.00%	0.73
0000990274.png	100	62	62.00%	0.54
0000999294.png	100	22	22.00%	2.15
0001118259.png	90	55	61.11%	0.60
0001123541.png	100	56	56.00%	0.82
0001129658.png	100	32	32.00%	1.62
0001209043.png	100	52	52.00%	0.88
0001239897.png	100	31	31.00%	1.35
0001438955.png	100	40	40.00%	1.08

Overall accuracy across these samples ranges from 22% to 62%, with an average edit distance between 0.54 and 2.15. These results indicate a substantial performance reduction compared to the synthetic test set (88.80%). Because the evaluation uses ground-truth bounding boxes, the observed degradation reflects recognition difficulty rather than detection error.

3.6. Position-Dependent Effects of OOV Characters

During the evaluation on FUNSD, we further examined how Out-of-Vocabulary (OOV) characters affect CTC decoding. We observed that the impact of OOV tokens is strongly position-dependent. When an OOV character appears at the end of a word, the resulting error is often localized: the final character is mismatched, while the preceding characters remain correctly aligned. In contrast, when an OOV character occurs at the beginning or in the middle of the word, the CTC alignment frequently collapses, causing errors that propagate to most or all subsequent characters.

This behavior is consistent with the monotonic alignment constraint of CTC and the fixed sequence length ($T = 35$). Once the model is forced to emit an incorrect class at an interior position, the alignment path must be adjusted for the remainder of the sequence, leaving little margin to recover. These observations suggest that CTC-based OCR models are not only sensitive to the presence of OOV tokens, but also to their position within the sequence, and that this limitation is architectural rather than something that can be fully addressed by additional training alone.

Additional qualitative examples, including the corresponding word-level predictions and visual artifacts observed in the FUNSD images, are provided in the Appendix for reference.

Several factors may contribute to this performance gap:

- **Stroke fragmentation and noise.** Many FUNSD words exhibit broken or faint strokes, particularly in thin characters. This interacts with the deletion patterns analyzed in Section 3.4, where narrow structures are already sensitive to downsampling.
- **Punctuation and OOV symbols.** The model is trained exclusively on alphanumeric characters, while FUNSD frequently includes punctuation such as colons, periods, and commas. These out-of-vocabulary tokens contribute directly to word-level mismatches and elevate edit distances.
- **Visual variability across pages.** Differences in scan quality, lighting, and form templates lead to image-dependent performance variations, as reflected by the spread of accuracies in Table 4.

Although the evaluation is limited to a subset of images, the results consistently show that the model faces difficulties with real handwritten and machine-printed text containing degraded strokes and non-alphanumeric content. These observations suggest that improving robustness to real-world noise and expanding the output vocabulary are likely necessary for deploying CRNN-based systems on scanned forms.

1. **Visual domain shift.** Real scanned documents often contain fragmented strokes, ink bleed, and background

noise. These artifacts can exacerbate the deletion patterns discussed earlier, particularly for thin-stroke characters.

2. **Vocabulary mismatch (OOV).** The model is trained exclusively on alphanumeric characters (0–9, A–Z), whereas FUNSD frequently includes punctuation (e.g., '.', ':', ',') in fields such as dates and prices. Since these symbols are out-of-vocabulary, they cannot be produced by the model and therefore contribute directly to word-level errors.

It is also worth noting that FUNSD annotations are often field-level rather than strictly word-level, which can disadvantage a single-word recognizer when multiple tokens are present within a single bounding box. Even with this limitation in mind, qualitative inspection indicates that many errors are associated with visual degradation and OOV punctuation, suggesting that both visual and lexical factors contribute to the synthetic-to-real performance gap.

4. Conclusion

This study examined the behavior of a CRNN–CTC model trained on synthetic data and evaluated its performance across different word lengths, character structures, and real scanned documents. While the model reaches 88.80% accuracy on MJSynth, a more detailed analysis reveals patterns that are not captured by aggregate metrics alone.

Recognition accuracy decreases for words longer than ten characters, and the results indicate that architectural constraints, together with the long-tail length distribution of the training data, may jointly influence performance on extended sequences. Character-level error statistics further show that thin-stroke letters are particularly susceptible to deletion under downsampling.

Evaluation on real scanned documents highlights additional challenges: fragmented strokes and out-of-vocabulary punctuation frequently lead to recognition errors, contributing to a noticeable gap relative to synthetic data. These observations suggest that future improvements may be obtained by exploring variable-width input strategies, better representation of long sequences during training, and expanded vocabularies for text commonly found in scanned forms.

We also observed position-dependent failure patterns for OOV characters, where interior OOV tokens can destabilize the entire CTC alignment, highlighting an additional structural limitation of the CRNN–CTC pipeline.

Overall, the analysis provides a clearer sense of the operating conditions under which CRNN–CTC models tend to struggle and may help guide future work toward more robust OCR systems.

A. Qualitative Examples on FUNSD

Figure 2 presents the scanned images used in our real-world evaluation, along with word-level predictions for all annotated bounding boxes. The examples illustrate common error patterns such as stroke fragmentation, background noise, and out-of-vocabulary punctuation.

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. 1

[2] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen 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, 2006. 1

[3] Gil Keren and Björn Schuller. Convolutional rnn: an enhanced model for extracting features from sequential data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3412–3419. IEEE, 2016. 1

Figure 2. Qualitative examples from the FUNSD evaluation set. Each image contains approximately 100 annotated word boxes.