Chinese OCR through a CRNN Architecture with Attention

Hao Qi, Ziye Chen, Xi Chen

haoqi@bu.edu, ziyechen@bu.edu, xich0108@bu.edu

Github Link

Abstract—*This project presents a Convolutional Recurrent Neural Network (CRNN) with an attention mechanism tailored for Chinese scene text recognition. It overcomes challenges such as variable illumination and text distortions. Enhanced by traditional computer graphics methods and deep learning, the model we built from scratch can precisely recognize single-line modern Chinese text, offering a foundation for multilingual text recognition assistant systems.*

1. Introduction

Optical Character Recognition (OCR) is integral to artificial intelligence, transforming text from images into digital data. This technology is crucial in geolocation, autonomous driving, and real-time translation applications. Imagine how helpful it would be for a foreigner or someone with visual impairments to understand various languages using an OCR-based assistive tool smoothly. However, natural scenes pose significant challenges with their textual irregularities and diverse conditions, especially when the text is skewed, curved, or distorted. Chinese text recognition in such environments is particularly challenging due to the language's character complexity and various contextual forms [7]. Environmental factors such as motion blur, inconsistent lighting, and variability in font and color compound these difficulties.

In response, this project has developed a Convolutional Recurrent Neural Network (CRNN) architecture that leverages the feature extraction prowess of CNNs with the sequential handling ability of RNNs, further enhanced with an attention mechanism. We adopt spatial transformation techniques to preprocess input images, creating a more uniform input that aids recognition. Also we establish a user interface for convenience. This report details the model's architecture and our experiment results. We further discuss improvements and applications in the end.

2. Related Work

Generally, OCR technology can be divided into two parts: text detection and text recognition.

2.1. Text Detection

Today's OCR technology is always associated with deep learning and neural networks, and the text recognition process always includes text region detection, feature extraction, etc. CTPN, introduced by [4], mainly consists of three parts: convolution layer, Bi-LSTM layer, and fully connected layer. Liao [6] provided DBNet, a text detection algorithm based on segmentation. Among all kinds of text detection algorithms, the segmentation-based detection algorithm can deal with irregular shape text, such as bending better, so it can often achieve better detection results.

2.2. Text Recognition

Text recognition usually follows the detection part in the pipeline of an OCR task. Shi introduced one of the most typical models, CRNN (Convolutional Recurrent Neural Network)[3], which introduces bidirectional LSTM(Long Short-Term Memory)[2] to enhance context modeling. Experiments show that a bidirectional LSTM module can effectively extract context information from images. Finally, the output feature sequence is directly decoded in the CTC module.

Since traditional encoder-decoder frameworks can only encode input sequences into a fixed-length vector representation, the introduction of the Attention mechanism[5] outputs a sequence composed of vectors of indefinite length, assigns more weight to the target data and related data, makes the decoder's "attention" focus on the target data, obtains more details, and can learn a reasonable vector representation of a longer input sequence.

3. Preparation

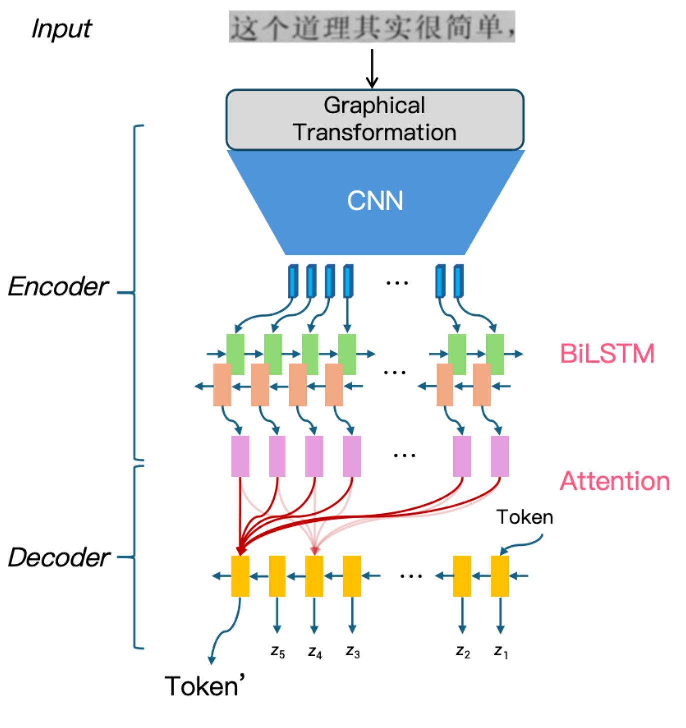


Figure 1. *Workflow*

Inspired by Shi's work [3], we implement a workflow illustrated in Figure 1. An input sequence is encoded using a CNN and a Bi-LSTM network. The decoder then generates an output sequence token by token. We apply an attention mechanism in the decoder, which allows the model to focus dynamically on different parts of the input sequence when generating each output token of various lengths.

3.1. Dataset

The training dataset was generated to mimic the complexities of real-world text as encountered in natural scenes. The dataset incorporates diverse text characteristics by utilizing a comprehensive Chinese corpus combining news articles and literature. Each image features text with variations in font styles, sizes, and degrees of grayscale, which are essential for training robust OCR systems. Additionally, the text images include artificial distortions such as blur, perspective shifts, and stretching to simulate challenging conditions.

A specialized dictionary comprising 5,990 characters—including Chinese characters, punctuation marks, English letters, and numerals—was utilized. These characters were selected based on frequency statistics from the corpus, considering both full-width and half-width forms. Each generated sample contains a fixed sequence of 10 characters randomly selected from sentence fragments in the corpus, ensuring a varied and representative text dataset. The images were standardized to 280x32 pixels to maintain consistency in input dimensions.

In total, approximately 3.6 million images were generated. These were split into training and testing sets in a 90:10 ratio, resulting in about 60,000 images for each subset. This division allows for comprehensive training and practical evaluation of the model's performance across various text recognition scenarios.

3.2. Data Pre-Processing

We notice that images in the selected dataset were augmented. Consequently, in our pre-processing workflow for text recognition, we primarily focus on enhancing OCR accuracy through several graphical transforms. Initially, we remove noise and unnecessary details from the image margins, ensuring a cleaner and distraction-free periphery. This step is crucial for reducing potential errors during text recognition. Following this, we concentrate on detecting and correcting any rotational misalignments in the text, taking multiple trials to standardize text orientation across all images.

4. Model

To address the challenges of Chinese scene text recognition, we have developed a Convolutional Recurrent Neural Network (CRNN) model as an encoder and an attention-based seq-to-seq model as a decoder.

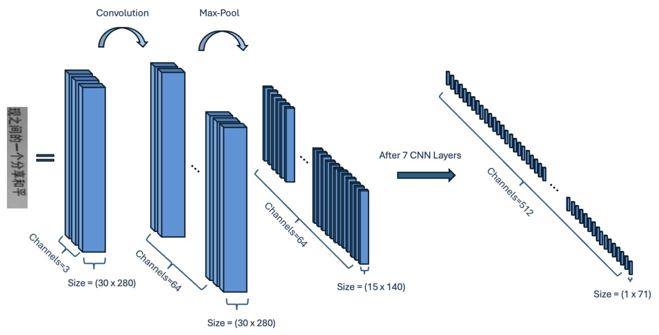


Figure 2. *CNN architecture in the encoder*

4.1. CRNN Encoder

The CRNN model effectively combines the perceptive capabilities of Convolutional Neural Networks (CNNs) with the sequential data fluency of Recurrent Neural Networks (RNNs) to excel in text recognition. In this hybrid model, the CNN acts as a feature extractor, transforming input images from size [3, 32, 280] to a feature-rich map of [512, 1, 71], as is shown in Figure 2

This transformation is accomplished through an engineered seven-layer CNN architecture. Starting with a convolutional layer that applies 64 filters followed by a ReLU activation and a MaxPooling operation, the model progressively increases in depth, enhancing feature detection. Batch Normalization is employed after layer depth increases to 256 and 512 filters, ensuring stable learning by normalizing activations. Specialized MaxPooling configurations help maintain text aspect ratios in deeper layers, preparing a detailed feature map for effective sequence decoding by the RNN.

```
class CNN(nn.Module):
    def __init__(self, channel_size):
        super(CNN, self).__init__()
        self.cnn = nn.Sequential(
                              nn.Conv2d(channel_size, 64, 3, 1, 1), nn.ReLU(True), nn.MaxPool2d(2, 2),
                              nn.Conv2d(64, 128, 3, 1, 1), nn.ReLU(True), nn.MaxPool2d(2, 2),
                              nn.Conv2d(128, 256, 3, 1, 1), nn.BatchNorm2d(256), nn.ReLU(True),
                              nn.Conv2d(256, 256, 3, 1, 1), nn.ReLU(True), nn.MaxPool2d((2,2), (2,1), (0,1)),
                              nn.Conv2d(256, 512, 3, 1, 1), nn.BatchNorm2d(512), nn.ReLU(True),
                              nn.Conv2d(512, 512, 3, 1, 1), nn.ReLU(True), nn.MaxPool2d((2,2), (2,1), (0,1)),
                              nn.Conv2d(512, 512, 2, 1, 0), nn.BatchNorm2d(512), nn.ReLU(True))
        nn.Conv2d()

    def forward(self, input):
        # [n, channel_size, 32, 280] -> [n, 512, 1, 71]
        conv = self.cnn(input)
        return conv
```

Code 1. *CNNs architecture*

These CNN layers are critical for reducing dimensionality and highlighting text features, setting the stage for the RNN's sequential text interpretation.

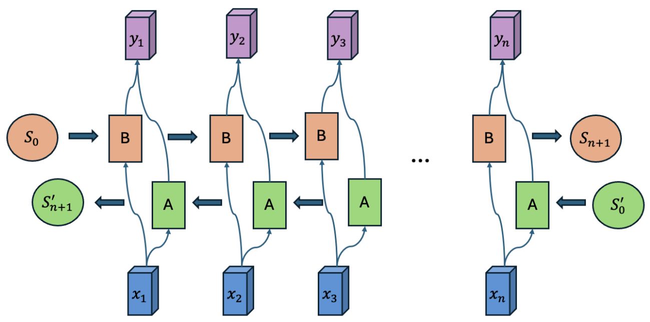


Figure 3. *BiLSTM architecture in the encoder*

4.2. BiLSTM architecture in the encoder

After the CNN layers extract and condense the features from the input images into a compact, the LSTM layers in Figure 3 interpret these sequential features. The module consists of two LSTMs: one processes the data in the forward direction and the other in the backward direction, allowing the network to capture dependencies from past and future contexts effectively. The input to the BiLSTM is the permuted output from the CNN, consisting of a sequence of 71 steps, each containing a 512-dimensional feature vector for each example in the batch. It then processes these sequences and outputs the encoded sequence, where each element is enriched with contextual information from both directions.

4.3. Seq-to-seq Decoder

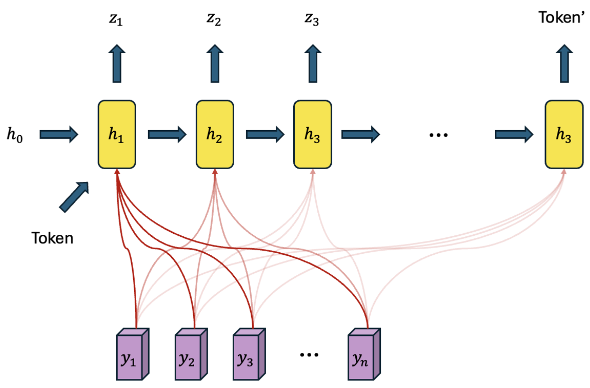


Figure 4. *Attention Mechanism*

Inspired by machine translation techniques, the attention mechanism is integral to this architecture, deftly woven into the sequence-to-sequence model shown in Figure 4. It acts as a spotlight, selectively illuminating the segments of the image sequence that are most pertinent for character recognition. This targeted focus is particularly pivotal when confronting the challenge of irregular text alignments—such as those found in skewed or curved lines—and compensates for the visual impairments caused by blurs or distortions. We also apply the teacher forcing technique with decreasing probabilities.

```
class AttnDecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size,
     dropout_p=0.1, max_length=71):
        super(AttnDecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.dropout_p = dropout_p
        self.max_length = max_length
        self.embedding = nn.Embedding(self.
    output_size, self.hidden_size)
        self.attn = nn.Linear(self.hidden_size *
     2, self.max_length)
        self.attn_combine = nn.Linear(self.
    hidden_size * 2, self.hidden_size)
        self.dropout = nn.Dropout(self.dropout_p
    )
        self.gru = nn.GRU(self.hidden_size, self
    .hidden_size)
        self.out = nn.Linear(self.hidden_size,
    self.output_size)
```

Code 2. *RNN with Attention*

Training of the CRNN model is conducted end-to-end, streamlining the learning process by feeding the model directly with image data and training it to output the corresponding textual information. Such an integrated training approach simplifies the model's learning curve and bolsters its capacity to generalize. Each epoch of the training process takes around 15 mintues with a NVIDIA V100 GPU and a batch size of 3,200.

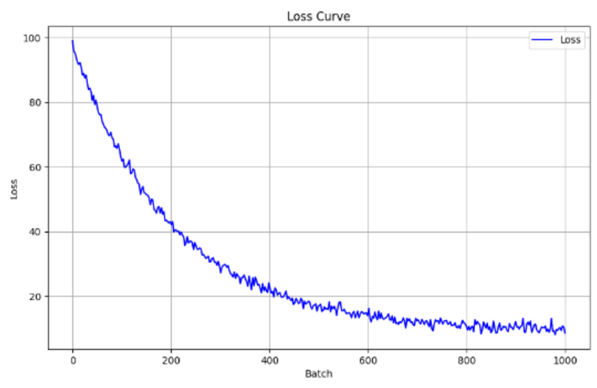
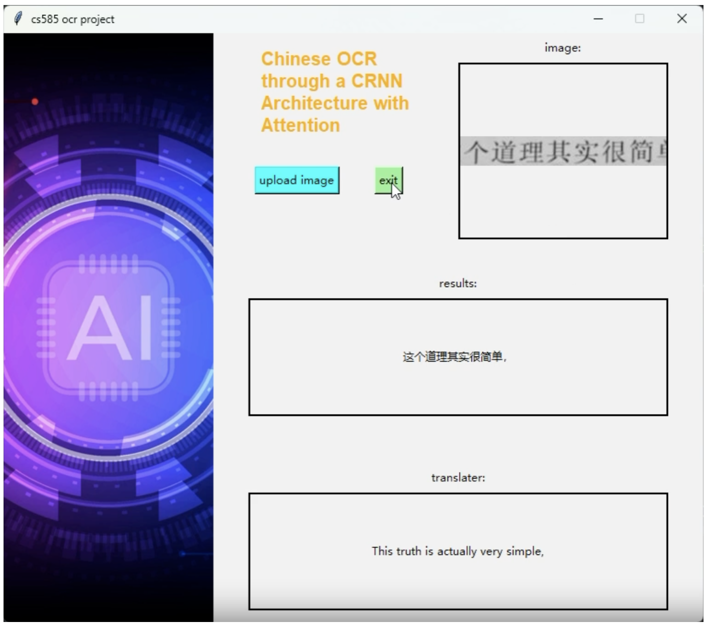


Figure 5. *Loss Curve*

5. Results

We choose Negative Log-Likelihood(NLL) as the loss function. After 10 epochs, the model converges generally well on the selected dataset, the loss curve of the final epoch is shown in Figure 5. The loss on the validation set is also close to the minimum.

We created a demo to prove the completeness of our project. In our experiment, the Attention mechanism performs well in analyzing context information, and it might be an improvement on the original CRNN model.



这个道理其实很简单，

This truth is actually very simple,

Figure 6. *Demo*

6. Discussion

Restricted to the simple dataset, we do not have to fine-tune the hyper-parameters many times and have reached a good result. However, this kind of model might have the problem of weak generality. To better fit the model, several skills for data pre-processing are always necessary, such as picture cropping, image rotation, and image blur. This may result in a relatively large amount of work, and the data may often still not fit the model. Moreover, since many of the datasets are based on the utf-8 encoding system, it is difficult for our model to reach a high accuracy when recognizing ancient Chinese characters (since they usually differ from modern typefaces in shape and structure). A set of new technologies may help ameliorate the problem.

With the development of computing power, several large language models(LLM), like gpt4 and Llama, have shown their amazing ability to summarize and process data. In future work, combining current structures with LLM, especially in the data-processing component, maybe a latent solution.

At length, we believe that this Chinese character structure has a good prospect in the future, especially combined with the camera mouse technology [1] we had learned in class. Combining these two technologies will contribute to helping foreigners and disabled people when reading literature.

References

[1] M. Betke, J. Gips, and P. Fleming, "The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1–10, 2002. DOI: 10.1109/TNSRE.2002.1021581.

[2] A. Graves and A. Graves, "Long short-term memory", *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[3] 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*, vol. 39, no. 11, pp. 2298–2304, 2016.

[4] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network", in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, Springer, 2016, pp. 56–72.

[5] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[6] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 11 474–11 481.

[7] K. Wang, Y. Yi, Z. Tang, and J. Peng, "Multi-scene ancient chinese text recognition with deep coupled alignments", *Applied Soft Computing*, vol. 108, p. 107 475, 2021.