LaTeX Guidelines for Author Response

A. 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.

B. 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.

C. Method

C.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. Example of caption. It is set in Roman so that mathematics (always set in Roman: $B \sin A = A \sin B$) may be included without an ugly clash.

C.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.

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$_batch_cost()$ $provided by Keras. The Adadelta optimizer was used for ocr recognizer, which is pre-trained on a large-scale dataset, were assumed as the ground truth. While the Keras recogni$

D. Experiment

D.1. Hyperparameter Adjustment

References