6.819 Advances in Computer Vision

Final Project

A Optical Character Recognition System with Pattern Editing Functionalities

Po-Han Lin, Daniel Wang

Abstract

We present an Optical Character Recognition (OCR) system that allows text segmentation, text editing (duplication, replacement, and removal), and text recognition in a scanned document containing either printed or handwritten texts. The system utilizes Python OpenCV2 module for text segmentation, Pytorch neural network module for text recognition, EMNIST dataset for network training, and achieves an overall 89% accuracy rate.

Introduction

Optical character recognition (OCR) is the conversion from images of typed, handwritten or printed texts into machine-encoded texts, whether from a scanned document or a photo. A typical OCR system has two main components: text segmentation and text recognition. Each character in the text is extracted from the image, individually mapped to a machine-encoded symbol, and later assembled into words, lines, and paragraphs in a output text file.

Current implementations of OCR are well-designed with high accuracy. However, the incapability of editing texts in place limits their usage. There are times when we not only want to identify texts in an image and generate the output text file, but also editing specific text segments in the image, whether it be removing certain words, copying certain characters, or to move a paragraph to another location in the scanned document. Some other applications include inserting signatures to a scanned document and removing watermarks in a picture. There is clearly a high demand for these aforementioned functionalities, and therefore we decided to build a OCR system that is capable of pattern editing as well as text segmentation and text recognition.

In this project, we successfully demonstrated the three core functions of editing in scanned texts. The backfill of deleted patterns was relatively well performed. Using self-trained convolutional neural network, we also demonstrated a functional text recognition system.

Related work

There are currently various projects that implement typical OCR functionalities available. Several modules have also been developed for such purpose, including the EAST text detector for text segmentation (reference 1) and the Tesseract module for text recognition (reference 2). However, none of these modules allow users to move specific patterns around a scanned document. While several picture editing software allows pattern editing, none of them are specialized for texts and need to be performed manually (e.g. eraser function in Microsoft Paint or background removal in Adobe Photoshop).

Approach

Text Segmentation

The image of interest is first converted into a binary image, then the findContours function in OvenCV2 is utilized to separate text from the background. After obtaining the list of contours, we first discard the outermost contour that contains the whole image by filtering out the contour with the largest area. The remaining contours were iterated through each pair in order to generate the hierarchy. A contour is considered "inside" another contour if all of its coordinates are within the region enclosed by another contour. This can be done by examining the bounding rectangle of each contour. At the end of this process, we have a list of parent contours that contains the outline of each character, a list of children contours that contains are inside each character, and a hash table detailing the parent-children relationships.

Text Editing

In order to select the texts that the user is interested in editing, we utilized the click module to interactively gather user inputs. The first two double clicks defines the region of interest (the top left and the bottom right corner), and the third double clicks denote the new coordinates to place the top left corner of the region of interest.

The characters within the region of interest are first identified from the list of out contours. Using the parent-children relationship table, we can then obtain the "inter-region" of each character. The inter-region is defined as a list of coordinates where the text actually occupies (i.e. within the outline of each character and excluding the hollow parts). This can be done by creating masks for the parent contour and all of its children contours where the value in the enclosed region are set to one and using the XOR logical operation.

In order to enable the desired three text editing functionalities, we implement two sub-functions "remove text" and "add text", each of which can be enabled or disabled individually. Replacement of texts can be done by activating both sub-functions; duplication of texts can be done by disabling "remove text"; removal of texts can be done by disabling "add text".

The process of removing and adding texts are fairly straightforward. The former can be done by setting the color of inter-region pixels to the background (details in next section), and the latter can be done by shifting the original texts to the new location and overwriting the pixel values.

Background Fill

A challenging part after removing texts is to cleverly fill in the missing pixels in the original location. We use a nearest neighbor algorithm to recursively fill in the missing pixels. For each missing pixel, we look in all four cardinal directions for the closest pixel that is not a missing pixel and keep record of its color and distance. We then choose one of these four pixels with probability directly proportional to the inverse of the squared distance and copy its color. The pixel is then considered "filled" and removed from the mask. This process is performed recursively until all pixels are filled.

Text Recognition

A convolutional neural network was built using Pytorch module with 2 convolutional layers, 1 max-pool layer, and 3 fully connected layers. We chose stochastic gradient descent as the optimizer, cross entropy loss as the criterion, and ReLU as the activation function. The training was run for various epochs with learning rate set to 0.001 and momentum set to 0.9. Both the training and test data sets were generated from the EMNIST Database using the unbalanced class (reference 3).

Experimental results

Text Segmentation

The segmentation procedure was tested on various images and two selected results were shown in Figure 1. The procedure correctly identified most images with different font sizes and font styles (both printed and handwritten) with various backgrounds. The procedure is effective in identifying all characters, including overlapping patterns, punctuation marks, and handwritten signatures. However, the character "i" is treated as two different bounding boxes due to the disconnection of its two parts. This issue can potentially be mitigated by constructing a machine learning model that detects lines and checking whether the dot is towards the top half of the line.

Figure 1. Identification of individual characters. Each read box is a out contour found by the text segmentation procedure and defines a character.

Text Editing

Our editing procedure also showed significant results. The duplication of text was nearly perfect, and the randomized background filling algorithm is a fairly good approximation for the original background (Figure 2 & 3). The randomization and the inference from nearest pixels generate smoother gradients, with the exception of a few black and white pixels scattering. This is most likely because of the rounding error in the contour detection algorithm and missing a few pixels.

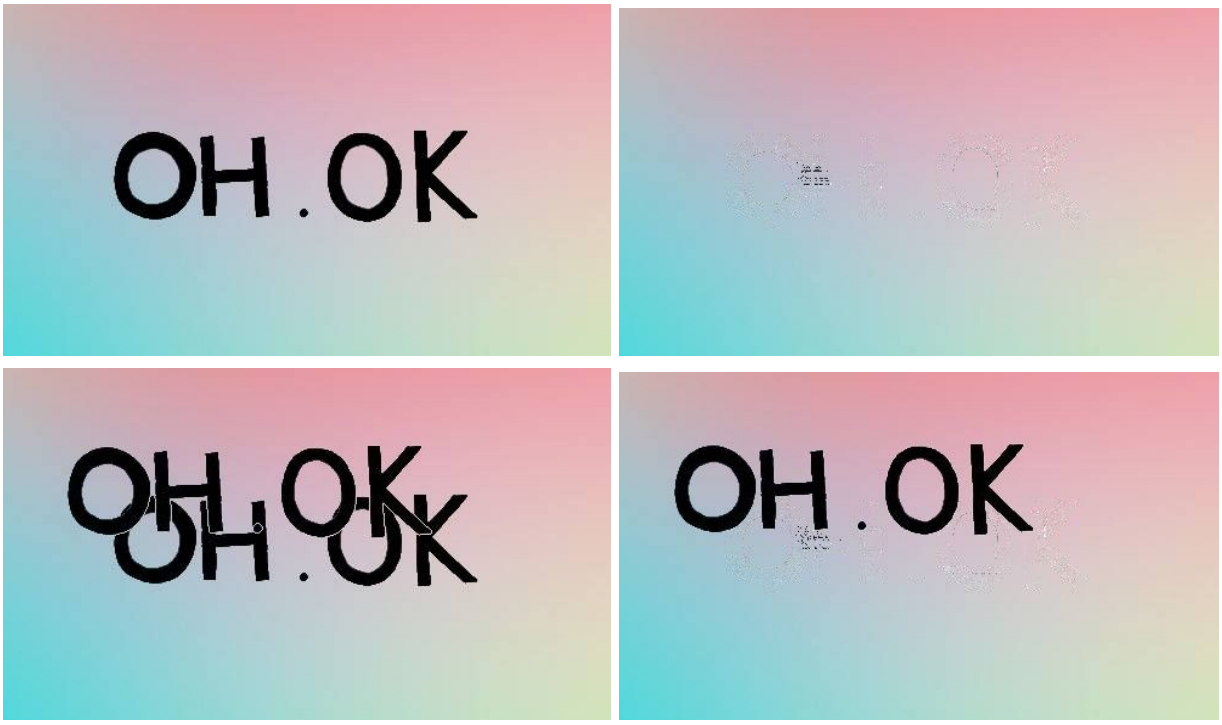

Figure 2. Different editing functionalities implemented in the OCR system. Topleft: the original image, topright: text removal , bottomleft: text duplication, bottomright: text replacement

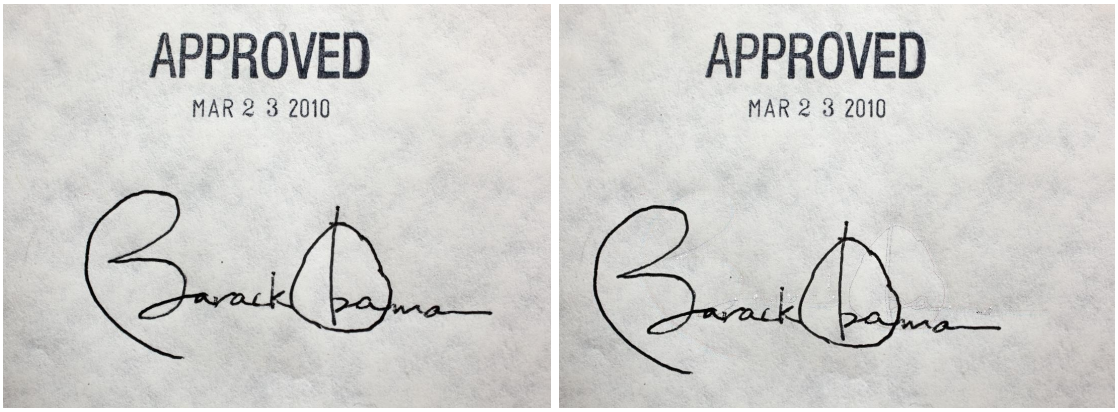

Figure 3. As a demonstration, we moved Barack Obama's signature to the left. Notice that the background remains relatively unchanged.

Text Recognition

The accuracy of the trained neural network is relatively well for printed and handwritten characters. The accuracy analysis was tested against the unbalanced EMNIST test dataset (reference 3). A table of different number of epochs is shown below, and the maximum accuracy is almost 90% at 25 training epochs.

Table 1. The accuracy of text recognition using unbalanced EMNIST test dataset

# of Epochs	2	5	10	20	25
Test Accuracy	80.45	85.03	87.58	88.80	89.06

Conclusion/Further Steps

We successfully achieved the goal of implementing a OCR system with text editing functionalities. However, several improvements and enhancements can be added and some other potential applications can be considered. Instead of the randomized nearest-neighbor algorithm, the background filling can be done using texture synthesis as we did for the class. Instead of replacing the original text, we can consider specifying the font size, font color, and even font styles for the relocated texts since we convert them to machine-encoded characters. With proper set up, the duplication of texts can create a showing effect on the original texts. Overlaying specific masks to the scanned document can also be utilized for aesthetics purposes with some creativity. One example is shown below in Figure 4.

Figure 4. Bolding specific parts of a text document in order to create the effect of overlaying a background image.

References

1. Xinyu Zhou et al. (2017). EAST: An Efficient and Accurate Scene Text Detector. arXiv:1704.03155 [cs.CV]
2. Ray Smith (2007). An overview of the Tesseract OCR Engine. Proc. 9th IEEE ICDAR.
3. Gregory Cohen et al. (2017). EMNIST: an extension of MNIST to handwritten letters. arXiv:1702.05373 [cs.CV]
4. Adrian Rosebrock (2018). OpenCV Text Detection (EAST text detector). https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/
5. Soumith Chintala. Deep Learning with PyTorch: A 60 Minute Blitz. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Individual Contribution

This project is jointly completed by Po-Han Lin and Daniel Wang. Po-Han and Daniel both conducted the initial research, participated in the the idea generation stage, and collectively finished this report. Po-Han is responsible for building, training, and testing the neural network, writing the base code for the text segmentation and text editing algorithm, and generating example graphs. Daniel is responsible for the clicking module input in the text editing algorithm, the implementation of the backfilling algorithm, handling out-of-bound exceptions, and the integration of separate parts.

Source Code

The complete source code of this project can be accessed at the Github repository with URL https://github.com/danielwang5/ocr2.git A readme is included in the directory with the command line options explained.