Descry: Optical Character Recognition for Low-Resource Writing Systems

Daria Kryvosheieva

MIT
daria_k@mit.edu

Abstract

We present Descry, a web application for the recognition of text written in low-resource writing systems. This report outlines the neural networks driving the app, our methods for generating training data, and the process of model training.

Introduction

Modern state-of-the-art OCR engines can recognize text in over 100 languages written in over 30 writing systems (Tesseract 2016; Google 2022). However, there still exist languages and scripts for which OCR engines have not been developed yet due to their rarity and the lack of digital data. We set out on a journey towards bringing OCR technology to those underrepresented scripts by developing a text detection and recognition pipeline for two new alphabets: Adlam, used to write the Fulani language spoken in West Africa, and Kayah Li, used for dialects of the Kayah continuum in Myanmar. Given an input image, we first detect bounding boxes of words with CRAFT (Baek et al. 2019) and then recognize words in these bounding boxes with a CRNN (Shi, Bai, and Yao 2017). We address the problem of data scarcity by generating synthetic datasets and follow the existing open-source OCR tool EasyOCR (JaidedAI 2020) in the implementation and training of the models.

Datasets

So far, we constrain the task to the recognition of document-type text, characterized by clear background and straight rectangular text orientation. At the same time, we consider all characters in the alphabets' Unicode blocks, including digits, punctuation, characters with diacritics, and special characters occurring only in loanwords. We also strive to include a wide variety of fonts and account for bold and italic styles.

Some prior text detection and recognition models have been trained on synthetic datasets like SynthText (Gupta, Vedaldi, and Zisserman 2016). We embrace the idea of training on artificial images as a solution to the problem of limited real-world data.

Recognition

For each alphabet, we generated 8,000,000 train images and 100,000 test images, sized at 200x64 pixels, using the Python `Pillow` library. Every image contained a string between 1 and 10 symbols long, with symbols sampled randomly, uniformly, and independently from the Unicode block corresponding to the alphabet. The font was randomly chosen from a directory of available fonts (15 options for Adlam and 9 for Kayah Li), while the font size was fixed at 30pt. With a probability of 0.8, the images depicted a black string on white background, and with probability 0.2, a white string on black background. The images were labeled with the strings they depicted. Examples are shown in Figure 1.

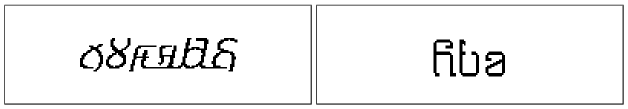

Figure 1: Sample recognition dataset images (Adlam on the left, Kayah Li on the right).

Detection

We placed 1 to 15 strings generated as described above at random positions on a common 512x512 canvas, making sure their bounding boxes do not overlap. To enhance the model's ability to detect big and small text, we varied the strings' font sizes between 25 and 35 points. Each image was annotated with the list of bounding boxes of all strings it contained. The size of the train set was 125,000 and the size of the test set was 100,000. See Figure 2 for examples.

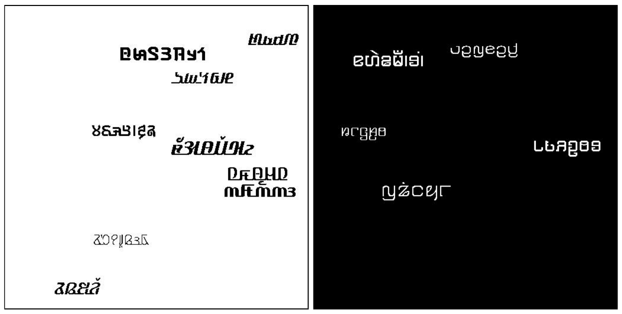

Figure 2: Sample detection dataset images.

Models

Detection

The detector is a CRAFT model whose architecture is summarized in Table 1. We used our detection dataset to fine-tune a pretrained model downloaded from EasyOCR GitHub (`github.com/JaidedAI/EasyOCR`), which also utilizes a pretrained VGG16-bn feature extractor imported from the `torchvision.models` library. This system accepts input images of arbitrary width and height but requires 3 color channels (RGB).

CRAFT produces a tuple of the image's character region score map, which represents every pixel's probability of being the center of a character, and affinity score map, which indicates the probability of being the center of a space between two characters. There score maps are converted into an array of word-level bounding boxes using the *QuadBox* procedure. It involves setting thresholds for both the character region and affinity scores, searching for connected components in the image where both scores exceed their respective thresholds, and calculating the minimum-area rectangles that enclose every component. These rectangles are the desired bounding boxes.

We trained a dedicated detector for each of the two alphabets using the MSE loss function and the Adam optimizer. The batch size and number of iterations were set to 25,000 and 5 respectively, which equates to one pass through the traning set. Computation was distributed across 8 NVIDIA A100 GPUs on MIT's OpenMind computing cluster. Each model took approximately 5 hours to complete training.

Input: h x w x 3
VGG16-bn
UpSample
Convolution BatchNormalization ReLU Convolution BatchNormalization ReLU
UpSample
UpSample
UpSample
Convolution + ReLU Convolution + ReLU Convolution + ReLU Convolution + ReLU Convolution

Table 1: CRAFT architecture.

Recognition

The recognizer is a CRNN designed to process 200x64 binary black-and-white images. The CRNN architecture, illustrated in Table 2, combines a CNN with two bidirectional LSTM layers. The inclusion of bidirectional LSTMs allows the network to effectively process text written in both left-to-right and right-to-left alphabets.

The raw output of the CRNN is a probability distribution over characters at each LSTM time step. It is then converted into a readable Unicode string using the CTC greedy decoding algorithm (Graves et al. 2006): select the highest-probability characters at all time steps and merge adjacent time steps that predict the same character. A special ”blank” token is added to the alphabet to match image regions with no characters and act as a separator in cases where the ground-truth sequence has consecutive repeating characters. This ”blank” token is deleted from the string after merging.

Both alphabets' recognizers were trained with the CTC loss funtion, the Adadelta optimizer, a batch size of 32, and 250,000 iterations. Training took 4.5 hours per alphabet on OpenMind's 8-GPU machine.

Input: 200 x 64 x 1
Convolution + ReLU maxpool Convolution + ReLU maxpool Convolution + ReLU Convolution + ReLU maxpool Convolution BatchNormalization + ReLU Convolution BatchNormalization + ReLU maxpool Convolution + ReLU
Bidirectional-LSTM Bidirectional-LSTM
Linear

Table 2: Recognition model architecture.

Results

For the detectors, we computed precision, recall, and the F1 score (harmonic mean of precision and recall) on the test set. We provide these scores in Table 3.

	Precision	**Recall**	**F1**
Adlam	0.994	0.860	0.922
Kayah Li	0.982	0.862	0.918

Table 3: Precision, recall, and F1 scores on the test set of our detection models.

For the recognizers, we measured accuracy on the test set as the fraction of images for which the predicted sequence exactly matches the ground-truth sequence. On this metric, the Adlam model scored **0.960** and the Kayah Li model scored **0.843**.

Next Steps

In the future, we would like to:

- Add more writing systems;
- Add support for the recognition of irregularly-shaped text and text in natural scenes;
- Contact developers of established open-source OCR engines regarding the integration of our work with theirs.

References

Baek, Y.; Lee, B.; Han, D.; Yun, S.; and Lee, H. 2019. Character Region Awareness for Text Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Google. 2022. Document AI release notes. `https://cloud.google.com/document-ai/docs/release-notes`. Accessed: 2024-06-01.

Graves, A.; Fernández, S.; Gomez, F.; and Schmidhuber, J. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd international conference on Machine learning (ICML)*.

Gupta, A.; Vedaldi, A.; and Zisserman, A. 2016. Synthetic Data for Text Localisation in Natural Images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

JaidedAI. 2020. EasyOCR. `https://github.com/JaidedAI/EasyOCR/tree/master`. Accessed: 2024-08-09.

Shi, B.; Bai, X.; and Yao, C. 2017. 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*.

Tesseract. 2016. Tesseract User Manual. `https://tesseract-ocr.github.io/tessdoc/`. Accessed: 2024-06-01.