Descry: Optical Character Recognition for Low-Resource Writing Systems

Daria Kryvosheieva

MIT
daria_k@mit.edu

Abstract

We present Descry, a web app for the recognition of characters from a selection of low-resource writing systems (currently Adlam, N'Ko, and Kayah Li). In this report, we describe the CNNs powering the app as well as the training datasets and the procedures for data collection and 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 writing systems for which OCR engines have not been developed yet due to their rarity and the lack of training data available on the Internet. We set out on a journey towards bringing OCR technology to those low-resource scripts by training character classifiers for three new alphabets. We follow approaches standard for other image classification problems, such as MNIST (LeCun et al. 1998), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), or CIFAR-10 (Krizhevsky 2009). Finally, we inspect filters learned by our classifier models and attempt to interpret their purpose.

Datasets

Currently, the project focuses on the recognition of individual printed characters, excluding digits, diacritics, and supplemental characters used only in loanwords (if present in the writing system).

For each writing system, we collected 1200-1400 28x28 character images from the Internet and labeled them. Within a writing system, all characters were equally represented. If the writing system makes a case distinction (Adlam), upper and lower case variants of every character were also equally represented. Wherever possible, the datasets accounted for italic and non-italic characters, various degrees of bolding, various relative sizes and positions of the character with respect to the image frame, differences in character shapes due to different fonts, and 'light' versus 'dark' themes (dark character on light background or vice versa). To extract characters from raw images and convert them into a 28x28 format, we used the web service ImageResizeOnline.com. Example images are shown in Figure 1.

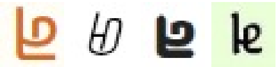

Figure 1: Variants of the same symbol (lowercase *b*) from the Adlam dataset.

Making the datasets suitable for training required additional preprocessing steps. First, because the color of the character or background is irrelevant to the classification task, colored images were converted into grayscale. Next, each dataset was augmented by a factor of 51 by adding 50 altered versions of every image. Alterations were combinations of rotation by an angle between -10° and 10°, translation by at most 2 pixels up, down, left, or right, and scaling by a fraction of the original size between 0.93 and 1.07. Finally, each dataset was randomly shuffled, and 80% was used for training while the remaining 20% was used for evaluation. An example of an altered image is shown in Figure 2. Dataset statistics for each writing system are presented in Table 1.

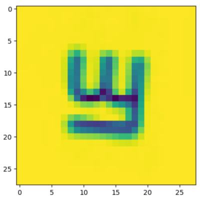

Figure 2: A sample symbol (uppercase *S*) from the augmented Adlam dataset.

Models

We trained three CNNs (one per alphabet) sharing the same architecture inspired by EnsNet (Hirata and Takahashi 2023). Compared to EnsNet, our models are simplified (most importantly lacking subnetworks) but have more output classes (corresponding to the number of letters in the alphabet). The structure of our models is presented in Figure 3; note that all convolutional layers as well as the first fully connected layer use the ReLU activation function. All models were trained with the Adam optimizer and with the batch size and number of epochs set to 64 and 10 respectively.

	Images before augmentation *total (per character)*	Images after augmentation *total (per character)*	Images in train dataset	Images in test dataset
Adlam	1,288 (46)	65,688 (2,346)	52,550	13,138
N’Ko	1,242 (46)	63,342 (2,346)	50,673	12,669
Kayah Li	1,320 (40)	67,320 (2,040)	53,856	13,464

Table 1: Sizes of image datasets for each writing system.

Input: 28x28 character image
Conv3-64 BatchNormalization Dropout(0.35) Conv3-128 BatchNormalization Dropout(0.35) Conv3-256 BatchNormalization
maxpool(2x2)
Dropout(0.35) Conv3-512 BatchNormalization Dropout(0.35) Conv3-1024 BatchNormalization
maxpool(2x2)
Dropout(0.35)
FC-512 BatchNormalization Dropout(0.35)
Flatten
FC-<num_classes> + softmax

Figure 3: The structure of the CNNs. Convolutional layers are denoted as Conv<kernel size>-<number of filters>, and fully connected layers are denoted as FC-<number of nodes>.

Results

All models achieve accuracy scores of over 95% on the respective test sets: accuracy is 99.81% for Adlam, 96.47% for N’Ko, and 98.70% for Kayah Li.

Filter Visualizations

We used gradient ascent (Chollet 2020) to generate artificial images that maximize filter activations. We examined these images to understand what kinds of features were learned by our CNNs.

Within each convolutional layer, meaningless filters (such as the one in Figure 4) are interspersed between filters that detect nontrivial features. The latter ones take the form of rim detectors (Figure 5) succeeded by diagonal (Figure 6) or vertical (Figure 7) line detectors, which are in turn succeeded by complex patterns (Figure 8).

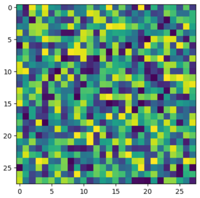

Figure 4: A typical noisy filter (42 in layer Conv3-64 of the N’Ko CNN).

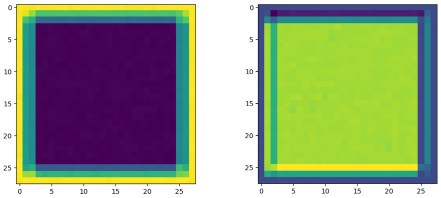

Figure 5: In layer Conv3-128 of the Kayah Li CNN, filter 26 detects light rims and filter 47 detects dark rims.

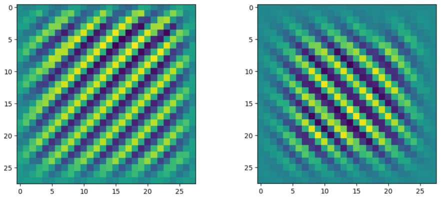

Figure 6: Diagonal line detectors are common in the Kayah Li CNN and especially abundant in the Adlam CNN due to the abundance of diagonal lines in the shapes of the characters. In layer Conv3-512 of the Adlam CNN, filter 244 detects bottom-left to top-right diagonals, and filter 497 detects top-left to bottom-right diagonals.

Next Steps

In the future, we would like to:

- Add support for more writing systems;
- Learn to recognize multi-character words and phrases with a more advanced model like CRAFT (Baek et al. 2019) that detects the bounding boxes of words and characters in the image.

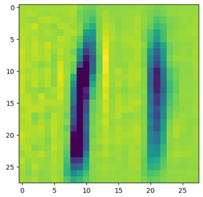

Figure 7: The N'Ko CNN has many vertical line detectors, like filter 772 in layer Conv3-1024 shown here, because N'Ko characters contain a lot of vertical lines.

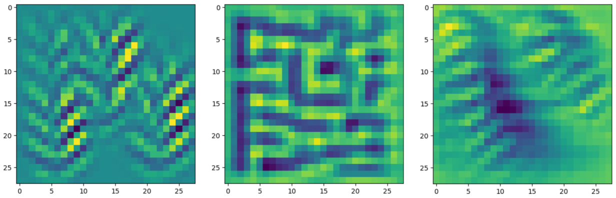

Figure 8: Selected complex pattern detectors—(left to right) filter 462 in layer Conv3-1024 of the Adlam CNN, filter 117 in layer Conv3-128 of the N'Ko CNN, filter 677 in layer Conv3-1024 of the Kayah Li CNN.

References

Baek, Y.; Lee, B.; Han, D.; Yun, S.; and Lee, H. 2019. Character Region Awareness for Text Detection. arXiv:1904.01941.

Chollet, F. 2020. Visualizing what convnets learn. https://keras.io/examples/vision/visualizing_what_convnets_learn/. Accessed: 2024-06-01.

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

Hirata, D.; and Takahashi, N. 2023. Ensemble Learning in CNN Augmented with Fully Connected Subnetworks. *IEICE Transactions on Information and Systems*, E106-D(7): 1258–1261.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*.

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

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.