university of groningen / faculty of science and engineering

Handwriting Recognition on the Dead Sea Scrolls and IAM Datasets

Group 20

C. M. Nampouri, s4721810, c.m.nampouri@student.rug.nl

March 15, 2024

1 Discussion

In this study, we worked on two different handwriting recognition systems. The first one is based on the Dead Sea Scrolls (DSS) collection and aims at the character-level transcription of these handwritten texts. The second uses the IAM database for English handwriting and aims at writer-independent handwritten text recognition.

Our methodology for the DSS system is more handcrafted. As input, we have totally unlabeled text images and only single characters are given as labeled data; the desired output should be a text document containing the digitized Hebrew characters that appear in each such image. The proposed pipeline consists of three major processes, i.e., image-level pre-processing, character segmentation, and character classification.

Good segmentation is the key to accurate recognition. Thus, our focus was on making the characters of an entire text image well-separated and distinguishable. According to this, pre-processing involved morphological operations that would detach thicker, connected characters as well as enlarge erased and cropped ones. Segmentation was performed by using a contour-based approach and managing extreme cases. Finally, classification of the segmented character images was performed using a trained convolutional neural network (CNN).

Our proposed pre-processing step was performed at the image level and in some cases it did boost the result of the segmentation, as we saw in Figure 3.6. However, there are many limitations that can be improved. Firstly, we used only one threshold for categorizing images to *thicker* or *erased* ones based on the average intensity value; instead, we could have considered more thresholds and experimented with the number of times an operator is applied in each case. More operators could also have been involved, like skeletons, or reconstruction operators for the erased images that still under-perform by far. In addition, what we did not consider in our implementation is normalizing the average intensity value. In fact, the different text images, have different numbers of lines and this is something that we should have taken into account. Finally, even when pre-processing improved the segmentation part, it reasonably made the recognition part harder having deteriorated the characters. A suggestion would be to apply the reverse transformation directly to the characters after the segmentation step.

Regarding the segmentation part, we detected rectangular contours of characters. We considered the case of very small contours that correspond to noise and not characters, but also we tried to handle the larger contours that correspond to characters still attached after pre-processing. On most occasions, the segmentation produces robust results as we can see in Figure 3.11. The strongest limitation derives from the managing of the larger contours. Specifically, we considered only the case of at most two characters under the same contour, thus ignoring many characters in the recognition, i.e., one contour results in one predicted character. This approach can be extended for more characters attached, e.g., three characters, thus splitting the larger contour into three smaller ones. In addition, an alternative approach would be not to restrict to contours of only rectangular shape, but also to experiment with other types-shapes of contours like active contours that adapt to the specific shape of objects and generate more accurate outlines.

Lastly, we have the classification part which can be considered the most standard component of the system. Assuming that the segmentation result could not be totally perfect for handwritten texts,

the least we could expect was to not incorrectly classify the well-separated characters. However, we also found difficulties in this part.

As the character samples of our training data have better quality than those of the DSS collection after segmentation, data augmentation was the most crucial step. Data augmentation can help in two ways. Firstly, to generate training samples similar to the testing ones. Our test images were subjected to dilation, opening, and erosion through the pre-processing step as well as cropping through the segmentation. Similar operators were also chosen to augment the training images. However, more augmentation could have been applied or combinations of augmentation techniques. Secondly, with augmentation, we can handle an unbalanced training dataset which happens in our case, but is not considered at all. In fact, most of the training characters have 300 samples, but there are few with around 10-20. However, our augmentation preserves the initial ratio of characters which makes it difficult for the classifier to recognize the more sparse characters. In the end, the characters with relatively fewer samples are lost since the model learns and gives emphasis on the most frequent ones. A potential good approach in order to balance the training data, but also give emphasis on some characters more, would be to consider the rough frequencies of characters and then, augment them proportionally (since we are given the n-gram frequencies).

The training and validation learning curves are shown in Figures 3.9 and 3.10. Although the result is satisfying on the training data, with little overfitting, in the DSS data we do not have similar performance. In general, we should have even more augmentation, a more complex network architecture, and more training epochs. Finally, although we implement a bigram model, we do not use it for our predictions. The reason is that since our CNN does not perform well enough, then the bigram would propagate the error in the entire text recognition. With a better classifier, we could have used bigrams and even further, higher-order n-grams.

Overall, our entire methodology was mostly oriented toward improving the segmentation results. Pre-processing and final segmentation are strictly connected with each other and aim at managing as many cases as possible. Since we have a good baseline in terms of our methods, our future work will be to enrich those techniques, considering more possible cases and operators and ensuring an accurate trained model for the prediction of at least the well-segmented characters.

On the other hand, the methodology for the IAM recognition system relies on a more end-to-end implementation. IAM database differs in both format and quality from the DSS collection. In this case, as input, we are given already segmented lines of text along with their full transcript labels. These lines are directly fed to a complex deep learning architecture and trained using the Connectionist Temporal Classification (CTC) loss and their corresponding ground-truth. In fact, no complex pre-or post-processing or segmentation steps are needed for at least a basic running model, since this approach can work directly at both word-level and line level.

However, even with this approach, data augmentation is proven to be valuable, especially considering Figure 4.3. To be more precise, the occurrences of characters greatly vary. As in the DSS system, characters that appear less often are not recognized very well (e.g., numbers), justifying the relatively low performance of the trained model on some occasions. As a result, for future work, there is the need for more balanced data, ensuring that all characters have a similar frequency. Moreover, the generation of synthetic line images, apart from standard augmentation techniques, can also help in improving the robustness of the model. Other promising alternatives would be to employ pre-trained models on synthetic data and fine-tune them on IAM, as well as to include a statistical language model since this approach inherently takes advantage of the sequences of words and characters.

Finally, both recognition systems share some common limitations; both require a fixed input size for the model and can only predict characters that are contained in the training data. Apart from that, they follow two completely different directions; the DSS is focusing on the low-level processing of images, while the IAM on the more high-level deep learning approach. It is all about the available data and the level of annotation to choose the most appropriate methodology.