Image-Based Recognition of Braille Using Neural Networks on Mobile Devices

Christopher Baumgärtner, Thorsten Schwarz, and Rainer Stiefelhagen

Study Centre for the Visually Impaired, Karlsruhe Institute of Technology, Engesserstr. 4, 76131 Karlsruhe, Germany

thorsten.schwarz@kit.edu
https://www.szs.kit.edu

Abstract. Braille documents are part of the collaboration with blind people. To overcome the problem of learning Braille as a sighted person, a technical solution for reading Braille would be beneficial. Thus, a mobile and easy-to-use system is needed for every day situations. Since it should be a mobile system, the environment cannot be controlled, which requires modern computer vision algorithms. Therefore, we present a mobile Optical Braille Recognition system using state-of-the-art deep learning implemented as an app and server application.

Keywords: Optical Braille Recognition · Deep learning · Mobile devices

1 Introduction

The Convention on the Rights of Persons with Disabilities aims to ensure that all people with disabilities are fully included in all areas of society. This also implies that people with blindness can communicate with people without blindness without obstacles. Since Braille is the printed script of people who are blind, knowledge of Braille would improve cooperation, e.g. for teachers at regular schools or lecturers at universities who teach blind students. However, learning Braille is very difficult due to different Braille systems such as Braille with six or eight dots, contractions, shorthand, Braille for various languages, Braille for music, math and other fields, which makes it even harder to learn. Mobile approaches on a modern smartphone or a tablet that allow scanning with an on-board camera and translating Braille into text could solve this problem, without using a special hardware. A system like that needs to overcome different challenges. First, it must detect and recognize embossed Braille using a camera. When a mobile system is used, the environment in which the data is collected cannot be controlled, which in turn is a challenge for the computer vision algorithms used. A second challenge arises from the documents themselves. Braille documents can be created using various techniques and therefore the output looks very different, which makes it difficult to recognize Braille. Another problem is the conversion
of recognized Braille into text. The transliteration is not trivial. Braille characters are ambiguous and depend on the system used. The identification of the system and contractions must also be addressed.

In this paper, we present the development of a mobile Optical Braille Recognition (OBR) system using state-of-the-art deep learning techniques.

2 Related Work

There are many approaches dealing with Optical Braille Recognition. An overview of seven different optical Braille recognition algorithms are given by Isayed and Tahboub (2015) [7]. They derive a common pipeline used in most publications. First, data is captured, mostly using flatbed scanners. Then the images are converted to gray-scale. A rotation correction is done using projections of the image onto the x- and y-axis. This method is also used for extracting the different lines and columns Braille is arranged in. The Braille dots are usually detected using static or dynamic threshold algorithms on a global or local level. To improve the results and to remove unwanted noise, morphological operations are utilized. After the dots are segmented, a grouping is performed. Therefore the lines and columns are used to divide the text in single character cells. The characters are recognized by splitting every cell in its 6- or 8-dot structure. Every possible dot position is checked for existence of a dot. The translation is usually done via translation tables.

Besides the papers discussed in [7], there are some other interesting approaches not using flatbed scanners. Hentzschel et al. (1995) introduce the twin-shadow-approach [5]. Their system uses a fixed camera, directly over the document, and two light sources. The lights are placed on the right and left side of the document at an angle. The system takes two pictures, using one light individually. The two pictures show the same scene but different lighting and shadows. The two pictures are used to calculate the difference picture. By doing this, nearly everything but the different shadows can be removed.

Schwarz et al. [13] presented a system based on [5]. They mounted a camera in a black box, and for lighting, several LED-stripes are placed on each side. By taking the brightness gradient into account, this system can remove the print from objects. The result only displays the Braille embossing. They also utilize the dedicated library, Liblouis* [9], for reverse translation from Braille to text. It uses translation tables for numerous different Braille languages and variants to convert text to Braille. Based on these tables a reverse translation is implemented. The drawback of the system is that it is far from being mobile.

Considering current approaches of optical character recognition (OCR), the state of the art is based on neural networks. These models haven’t been used in OBR yet, but show great results in general OCR. One of these approach was published by Shi et al. [14] in 2016. They use a combination of Convolutional Neural Networks (CNN) and Long Short-Term Memories (LSTM) for OCR. Image features are extracted using the CNN creating multiple feature maps. The columns of these maps are used as input for a bidirectional LSTM successively.
The LSTM creates a per-frame prediction distribution over all possible characters. A transcription layer generates the final text output based on the predictions. Building upon the architecture of Shi et al., Deng et al. introduce another model using CNNs and LSTMs. In addition to the former architecture, Deng et al. used an attention mechanism to take spatial information into account. Their system is used to recognize mathematical equations from images. The attention mechanism yields better performance for this problem. This leads back to the spatial arrangement of symbols being critical for mathematical equations.

3 The Mobile Braille Scanner

In our approach, we use different smartphones, like a Huawei P20 Lite and a P20 Pro as a mobile Braille scanner combined with a server for computationally intensive processes. For this purpose, we develop a mobile application for the smartphone with the aim of capturing images of Braille documents and translating Braille into standard text using modern image processing. By outsourcing the majority of the image processing to an external server, the requirements for the smartphone are low, which means that the App requires Android 8 (API level 26 [4]) or higher.

3.1 Design Challenges

A great challenge for a mobile system is that images may vary widely in background, lighting and overall quality of the image. Former approaches use controlled lighting to make the dots more visible. Inspired by that, we use a flashlight to create shadows for each Braille dot. We observed that the flashlight increases the visibility of the Braille by creating bright spots and shadows. But in contrast to a scanned document, the page is very uneven lit. Moreover, depending on the ambient lighting, the effect of the flashlight may vary. Those images can not be easily processed using simple threshold algorithms and morphological operations. Thus, we use modern deep learning architectures to detect and recognize Braille.

3.2 Processing Pipeline

The processing pipeline is shown in Fig. 1. First, the images are cropped. After converting to gray-scale, a line detection is performed. Therefore a Faster R-CNN [12] is used. Faster R-CNNs are neural networks based on a CNN and Region-Proposal Network, which are used for object detection. The model is pre-trained on the COCO data set [10] and fine-tuned with our own data set.

After line detection, the cropped lines are used as an input for character recognition. A histogram equalization is applied to increase the contrast. We use a model based on the architecture of Deng et al. [2] with a standard attention mechanism. To train the model from scratch, we use a second data set which is discussed in more detail in the subsequent Sect. 3.3.
3.3 Generation of the Training and Evaluation Corpus

We used three different types of corpora to cover different scanning conditions in our training and evaluation corpora: (i) images of pages of a book printed with three different embossers\(^1\) to obtain a variation of different embossing techniques, (ii) generated images on the basis of the first corpus and sentences from the dataset “German Vocabulary” of Leipzig University [11], where the lighting conditions and background are varied to simulate shots from different angles with different background (iii) images of documents that contain both, normal print and Braille.

![Processing pipeline including preprocessing with line detection and character recognition](image)

Neural networks are usually trained on annotated data. Therefore, we needed segmented data for our training regime, containing bounding boxes of lines for line detection and line images for OBR in Braille and normal print, as well as a mapping for Braille to normal print. We captured images of our first corpus without additional light source because it was pure Braille without normal print. The corpus was semi-automatically annotated and manually corrected. Annotating, however, is time-consuming and laborious and large amounts of data are required to develop robust models for neural networks. Therefore, we automatically generated the second corpus together with the annotations for lines, characters and mappings in order to significantly enlarge the training corpus. The third corpus contained images of documents either embossed with Emfuse

| Table 1. Data set for Optical Braille Recognition. Number of lines for training, validation and test |
|---------------------------------------------------------------|
| **Images of** | **Pure Braille** | **Braille+ print** | **Overall** |
| **Embosed** | **Generated** | **Embosed** | **Generated** | **Embosed** | **Generated** | **Overall** |
| Training | 9.000 | 50.000 | 13.795 | 72.795 |
| Validation | 1.250 | 11.100 | 1.050 | 13.500 |
| Test | 1.000 | 10.000 | 1.737 | 12.737 |

\(^1\) Emfuse, EmBraille from Viewplus and Everest from Index Braille.
or came from pharmaceutical packages. With this corpus our models had to learn to distinguish between normal print and Braille. Table 1 gives an overview of the data sets used for OBR. The data set for line detection was created from the segmented lines of the previous data set. It contains 1305 images of Braille documents and 31,005 objects. The data set is split into 24,989 training samples, 2,978 validation samples and 3,038 test samples.

### 3.4 Implementation

We implemented an Android App for image capturing, pre-processing and result output (Fig. 2). The two deep learning models for line detection and for OBR are running on a server using tensorflow-serving [3] and docker. The app sends the cropped image to the server for line detection. The resulting Braille lines are sent to the server one by one for Braille recognition. The process of scanning a document using our system is shown in Fig. 2.

![Fig. 2. Screenshots taken from the Android App.](image)

### 4 Evaluation

We evaluated our approach twofold: (i) The deep learning models are validated using the test data from our data sets, and (ii) The usability of the application is evaluated by carrying out a user study.

#### 4.1 Line Detection Model

The model for line detection is evaluated using the COCO object detection metric [10]. This metric is based on the intersection over union (IoU), which measures the overlap between two bounding boxes. For different IoU-thresholds the average precision (AP) is calculated. The higher the IoU-threshold is chosen, the better the bounding boxes have to align to be considered a successful detection.
Our model reaches a AP@IoU(0.5) of 0.91 and a AP@IoU(0.5;0.95;0.05) of 0.59. The scores for each IoU-threshold are shown in Fig. 3. In general the results are satisfying but it can be seen, that the results for higher IoU-thresholds get worse. Considering the detection being the foundation for character recognition, better results for higher thresholds would be desirable. Therefore mainly a larger data set needs to be created.

![Fig. 3. Evaluation of line detection using the COCO metric.](image)

### 4.2 Optical Braille Recognition Model

Optical Braille Recognition is evaluated using the test data split into documents with and without additional print without any generated images. We use the Levenshtein distance/edit distance (ed) [8] to compare the ground truth to the prediction of our model. It counts the insertions, deletions or substitutions needed to change string $s_1$ to $s_2$. Based on this distance we calculate the “Character-Error-Rate (cer)” which is defined as $\text{cer}(s_1, s_2) = \frac{\text{ed}(s_1, s_2)}{\max(|s_1|, |s_2|)}$. We calculate the ed and cer for every line length independently. The evaluation with documents of Braille with additional print is shown in Fig. 4a. The model works well for short and long lines. On average, the models achieve a cer of 0.02 and an ed of 0.362 operations. Figure 4b shows the model’s performance on non printed documents without synthetic data. The results deviate greatly from the printed documents. The model reaches a cer of 0.11. The biggest difference between these two types of data is the use of a flashlight. It was only used with printed documents because the print has much more contrast than embossing at ambient light. To check, if it is part of the worse results, a few images of the same documents were taken using the additional light. The results with these images are shown in Fig. 4c. The model reaches much better scores on this data. This observation confirms that the correct illumination is crucial.

### 4.3 User Study

To evaluate the usability of the app, we performed a user study with participants which don’t have regular contact to blind persons. This study involved 7
persons with an average age of 40 (29–61, 4 female and 3 male). The task of each participant was to scan a document and get results from our system. There were three types of documents available: without print, printed with matching text and a pharmaceutical packaging. An additional light source was available. First, the participant were introduced to the task. Then the smartphone was handed over with the app already running.

Every participant was able to complete the task without help. To evaluate usability, the participants were asked to fill out a questionnaire consisting of general questions, User-Experience-Questionnaire (UEQ) [6] and System Usability Scale (SUS) [1]. We received a SUS score of 90% and overall very good UEQ ratings. The results show that the system is well usable for an average person. The questionnaire provided some valuable feedback to further improve the application. The participants suggested that it should be possible to adjust the results from the line detection, to include automatic orientation recognition, or a functionality to save and load results.

5 Conclusions

We present a new approach to Optical Braille Recognition using a mobile device and state-of-the-art computer vision algorithms. Our system is based on a Faster R-CNN architecture for Braille line detection. For character recognition, a combination of a CNN and LSTM is used. To capture the important spatial information of Braille, an attention mechanism is included. The models are trained
with custom data sets. An Android application is developed which is used to capture the images and show the results. Both deep learning models are hosted using TensorFlow-serving and docker. This architecture allows for easy updates of both neural networks without updating the app itself.

In the future, some improvements to the application like automatic orientation detection of Braille writing are considered. To further improve the performance of both deep learning models, more annotated data has to be collected. Therefore, a special data collection application can be created to get interested contributors to participate. To make the system accessible for blind or visually impaired users, a fixture could be used to hold the smartphone and document in a fixed position. This would also allow for some controlled lighting to further improve performance.

References

1. Brooke, J., et al.: SUS: a quick and dirty usability scale. In: Usability Evaluation in Industry, pp. 189–194 (1996)
2. Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-markup generation with coarse-to-fine attention. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 980–989. JMLR. org (2017)
3. Google: Tensorflow (2019). https://www.tensorflow.org/. Accessed 5 Jan 2020
4. Google: Android sdk (2020). https://developer.android.com/studio/releases. Accessed 14 Apr 2020
5. Hentzschel, T., Blenkhorn, P.: An optical reading system for embossed Braille characters using a twin shadows approach. J. Microcomput. Appl. 18(4), 341–354 (1995)
6. Hinderks, A., Schrepp, M., Thomaschewski, J.: User experience questionnaire (2020). https://www.ueq-online.org/. Accessed 18 Mar 2020
7. Isayed, S., Tahboub, R.: A review of optical braille recognition. In: 2015 2nd World Symposium on Web Applications and Networking (WSWAN), pp. 1–6. IEEE (2015)
8. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet physics doklady 10, 707–710 (1966)
9. Liblouis* (2019). http://liblouis.org/. Accessed 4 Dec 2019
10. Lin, T.-Y.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
11. Quasthoff, U., Richter, M.: Projekt Der Deutsche Wortschatz (1998)
12. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497 (2015). http://arxiv.org/abs/1506.01497
13. Schwarz, T., Dolp, R., Stiefelhagen, R.: Optical braille recognition. In: Miesenberger, K., Kouroupetroglou, G. (eds.) ICCHP 2018. LNCS, vol. 10896, pp. 122–130. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94277-3_22
14. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. 39(11), 2298–2304 (2016)