An Optical Braille Recognition System for Enhancing Braille Literacy and Communication between The Blind and Non-blind

Helen Gezahegn, Ting-Yi Su, Wan-Chun Su, Marin Ito

1 Introduction

The World Health Organization (WHO) estimates that there are 36 million people in the world who are blind and 217 million that are moderately to severely visually impaired [1]. Individuals who are blind or have impaired vision are unable to read printed text; instead, some use braille - a system of raised dots that can be read by touch. Braille symbols are formed within braille cells: units of space which when full would consist of six raised dots. The six raised dots are numbered from one to six and arranged in two columns as shown in Figure 1.

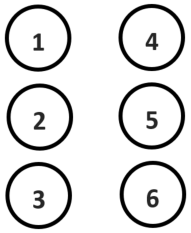

Figure 1: The Braille Cell.

According to the National Federation of the Blind [2], fewer than 10 percent of the 1.3 million that are legally blind in the United States use braille to read. These statistics stem from the fact that some visually impaired individuals find the process of learning and using braille lengthy and cumbersome. As a consequence, braille literacy has slowly begun to decline as people begin to turn to technology and audio recordings as a substitution for braille [3]. However, assistive technologies or audio recordings are limited and do not account for

everything that a vision impaired individual encounters on a daily basis. More importantly, the visually-impaired may require the help of those that are not blind to aid them in interpreting braille. People who work with the blind and are not familiar with the braille system may also require resources for two-way interaction via braille.

Optical braille Recognition (OBR) technologies were created to counteract this problem, but they are often expensive, outdated, and non-portable. With this in mind, we created a free, modern, and user-friendly technology that is readily available and accessible so that the braille system can be universally interpreted and understood. In this work, we identify an important problem with the lack of technology and resources available to assist visual individuals in communicating with the visually impaired. We took imperative steps in providing resources, such as with our new dataset Braille37, to promote braille literacy and learning.

2 Approach

2.1 Braille37 Dataset

At the start of our project, we immediately noticed the lack of available braille character datasets online and concluded that we would have to generate our own. We constructed our dataset using Labelbox - a data labeling service that allowed us to create, label, and manage our data[1]. We gathered images of braille from rare book collections and online sources, cropped each image to individual characters, and labeled them one-by-one. To add noise to the dataset, we used filters and XNConvert, a free cross-platform batch image processor[2].

Currently, we have 26,724 labelled images of 37 Braille characters comprised of 26 Braille symbols representing the English alphabet and 10 numerical digits; and 11 other braille symbols, 8 of which are English punctuation characters as seen in Figure 2. We also decided to incorporate a space character into our dataset to allow for the recognition of individual words amongst a string of text.

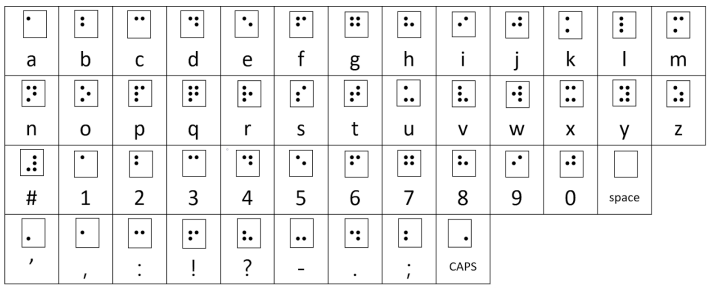

Figure 2: The 37 Classes.

[1]Labelbox can be found at https://www.labelbox.com/
[2]XNConvert can be found at https://www.xnview.com/en/xnconvert/

2.2 The Model

Our convolutional neural network was written in Pytorch with two convolution blocks each consisting of a convolution layer, max pooling layer, and leaky ReLU, followed by a linear layer, a leaky ReLU, and a final linear layer. These specifics were determined based on multiple tests on the training performance of our model. Moreover, our CNN model performed well without many layers and by the end of three weeks, it was able to classify images with an accuracy of 99.333% on the validation dataset. Another approach was to use image segmentation; however, we recognized that this process would be difficult to implement in a timely manner. We were able to overcome this time constraint and discover another method, as illustrated in Figure 3 that would allow us to segment words and sentences by making one assumption - that the spacing of each braille character would be uniform. We were able to deduce this by understanding that each embossed braille character consists of the same three-by-two matrix structure, making the spacing between each character the same, or approximately close.

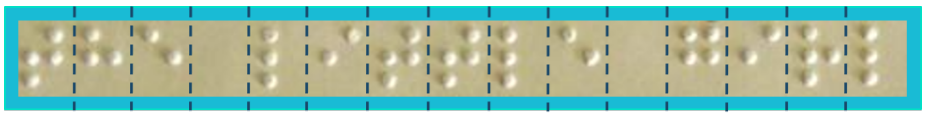

Figure 3: A sentence of embossed Braille demonstrating the uniform spacing and our method of character segmentation.

With this assumption, we were able to take an image and divide it into individual braille characters. To divide the image into equal parts, we considered the size of the image, resized it to be 28 by 28, and then used a variable containing the standard size of one braille cell to determine the number of braille characters in the image and where each character starts and ends. Once the image was properly divided, the braille characters were individually cut out and fed into our model. Their outputs were then concatenated to produce the English conversion.

3 User Interfaces

To test the functionality of our product, we created two user interfaces, a Flask web application and an Android application. Our web application has a page that is able to upload a pre-cropped image of Braille and output its English counterpart, while our Android application uses the uCrop API functionality provided by Yalantis to allow users to take a picture or select an image of Braille from their phone gallery, crop the image, and receive the English conversion.[3] Images of our Android application can be found in Figure 4. While our Android application is currently undergoing alpha testing, our website is fully functional [4].

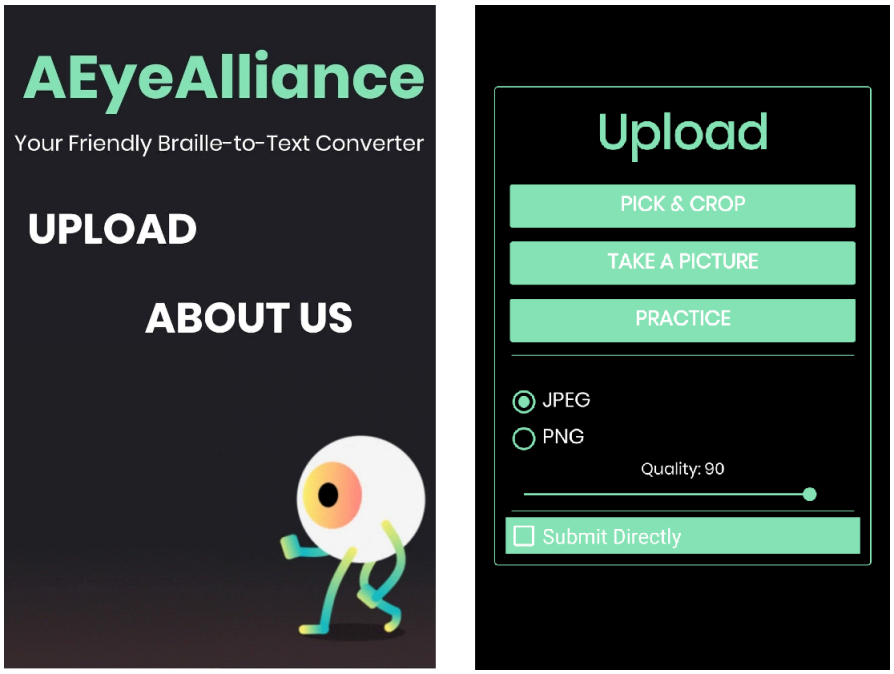

Figure 4: Illustration of the home screen (left) and upload function (right) of our Android Application.

[3]The Yalantis uCrop API can be found at https://github.com/Yalantis/uCrop
[4]Our web application can be found at https://github.com/AEyeAlliance/web-application/

4 Conclusion and Future Goals

The most significant accomplishments of this work are the novel dataset Braille37, the high accuracy performance of our model, as well as the ability of our ratio segmentation cropping function to accurately distinguish each Braille character without the additional use of edge detection or image segmentation. As a result of our own frustrations with finding readily available Braille datasets, we wish to release our dataset as open source for others to deploy to their own projects.

For future work, we plan to expand our dataset to $100,000$ Braille images by adding more noise and classes, facilitating improved conversions and accuracy. Furthermore, we aim to employ image segmentation to enable users to receive Braille-to-text conversions from images of not only lines of Braille, but also of a paragraph or an entire page of Braille-text. Additionally, to improve accessibility, we intend to use Google's Translation API[6] to acquire conversions of Braille-text into other languages as well.

In terms of our user interfaces, we aim to finalize testing of our Android app, and deploy and host our web application as a user friendly and accessible website. Our ultimate goal is to allow the public to visually explore the world of Braille and promote braille literacy among both visually-impaired and non-visually-impaired individuals.

References

[1] W. H. Organization, "Blindness and visual impairment." http://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment, 2017.

[2] N. F. of the Blind, "The braille literacy crisis in america: Facing the truth, reversing the trend, empowering the blind," 2009.

[3] Hdavies, "Making libraries accessible: Adaptive design and assistive technology," Jul 2017.

[6] Google's Translation API can be found at https://cloud.google.com/translate/