

Article

Braille Recognition for Reducing Asymmetric Communication between the Blind and Non-Blind

Bi-Min Hsu

Department of Industrial Engineering & Management, Cheng Shiu University, Kaohsiung 83347, Taiwan; bmhsu@csu.edu.tw

Received: 3 June 2020; Accepted: 27 June 2020; Published: 30 June 2020



Abstract: Assistive braille technology has existed for many years with the purpose of aiding the blind in performing common tasks such as reading, writing, and communicating with others. Such technologies are aimed towards helping those who are visually impaired to better adapt to the visual world. However, an obvious gap exists in current technology when it comes to symmetric two-way communication between the blind and non-blind, as little technology allows non-blind individuals to understand the braille system. This research presents a novel approach to convert images of braille into English text by employing a convolutional neural network (CNN) model and a ratio character segmentation algorithm (RCSA). Further, a new dataset was constructed, containing a total of 26,724 labeled braille images, which consists of 37 braille symbols that correspond to 71 different English characters, including the alphabet, punctuation, and numbers. The performance of the CNN model yielded a prediction accuracy of 98.73% on the test set. The functionality performance of this artificial intelligence (AI) based recognition system could be tested through accessible user interfaces in the future.

Keywords: asymmetric communication; optical braille recognition (OBR); artificial intelligence (AI); machine learning; convolutional neural network (CNN)

1. Introduction

The World Health Organization (WHO) estimates that 38 million people in the world are blind and 217 million are moderately to severely vision 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 can fill up to six raised dots [2]. The locations for the six raised dots are numbered one to six and arranged in two columns as shown in Figure 1. According to the National Federation of the Blind [3], 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 to be lengthy and cumbersome [4]. As a consequence, braille literacy has begun to decline as more people are beginning to turn to technology and audio recordings as a substitution for braille [5]. However, assistive technologies and audio recordings are not suitable for everything that a vision impaired individual encounters on a daily basis. More importantly, those who are visually-impaired may require the help of others to aid them in interpreting braille. Moreover, individuals who work with the blind and are not familiar with braille may also require resources for two-way interaction via braille because of the complexity of the system.

The Braille Cell

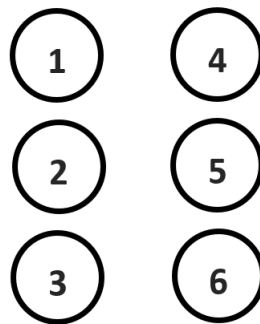


Figure 1. Illustration of the basic structure of the braille cell.

Asymmetric two-way communication was a term first defined by Grunig and Hunt to describe an imbalance in communication between individuals participating in public relations where information is often lost or altered [6]. Although the problem presented in this paper may not necessarily be related to public relations, it is indeed an example of asymmetric communication (Figure 2), as there are fewer communication methods to allow non-visually impaired individuals to understand braille, which would most likely lead to a depletion in braille literacy and thus hinder the blind in effectively making contributions to society. As such, technology that helps to balance this asymmetry in communication is needed to promote information flow and communication between blind and non-blind individuals.

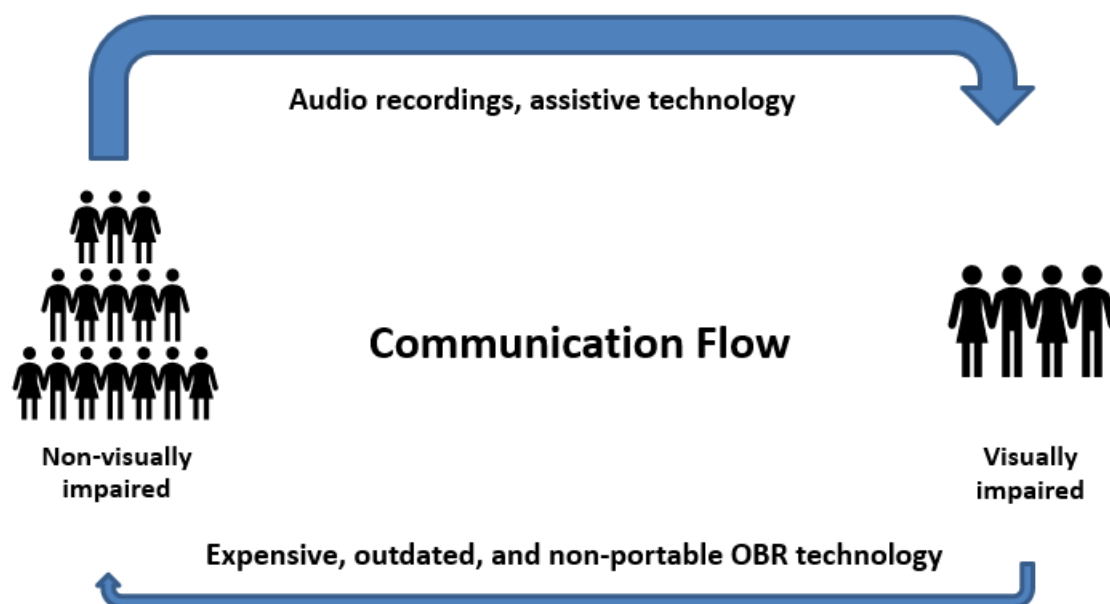


Figure 2. Illustration of asymmetric communication between visually impaired and non-visually impaired individuals.

Optical character recognition (OCR) is the process of capturing and processing images of handwritten, printed, or typed text into natural language characters. Although there exist many modern OCR devices, such devices cannot be effectively used to recognize braille characters since braille consists of individual dots rather than continuous strokes [7]. Optical braille recognition (OBR)

technologies, such as the braille translation software offered by Neovision (<http://www.eyeway.org/?q=optical-braille-recognition-software>), have been created to convert braille images into text and to account for the differences between normal characters and braille symbols, but they can be expensive, outdated, and non-portable. Moreover, according to Isayed and Tahboub, there is a lack of research on the use of portable devices, such as mobile and camera devices, for the acquisition of braille images in the field of OBR [8]. Thus, the aim of this research is to create a free, modern, and user-friendly technology that is readily available and accessible.

This research introduces a new braille dataset consisting of 37 characters corresponding to 71 character classes and provides a novel method which couples a ratio character segmentation algorithm (RCSA) and a convolutional neural network model (CNN) to convert a line of braille into its English counterpart. This simple RCSA algorithm replaces the need for a much more complex image segmentation algorithm that is needed for most braille recognition systems.

1.1. Motivation

While many blind individuals use braille to better adapt to the visual world, by mapping braille symbols to natural language characters, there are not as many non-blind individuals who can understand the braille system. Hence, braille documents and labels that are constructed and used by the visually impaired cannot be understood by the majority of the population, thus creating a segregation in society and an imbalance in communication flow. As someone who has had close interactions with the blind, this study was conceived by considering any difficulties that a blind individual or their close acquaintances may encounter on a daily basis. For instance, while caring for a blind individual who suffered from a recurring medical condition that required prompt medication, it became obvious how important it was to understand the braille system. All of the individual's medicine bottles were labeled in braille, so the process of finding the correct medication was a laborious task. With the lack of accessible braille-to-text conversion technology, I used old braille-to-English dictionaries to convert every label into English text, one letter at a time. While this may be just one example where fast and efficient OBR technologies may be beneficial to both the blind and non-blind, there are many other instances where the blind rely on non-visually impaired individuals to help them navigate their everyday activities. Parents of visually impaired children that struggle with understanding their children's braille system for reading and writing may have trouble communicating with and teaching their children and would likely benefit from having easily accessible OBR technology. Moreover, with the help of portable OBR systems, staff working at institutes for the visually impaired will have a more effortless time understanding the resources used by the visually impaired. As such, this braille system has the potential to be useful in decreasing asymmetric communication that any non-blind individual may face when trying to assist a blind individual.

In addition, research has revealed a noticeable gap in the reproduction and preservation of braille documents constructed by the blind [9], which can be remedied by leveraging efficient OBR systems. Many have disregarded braille documents simply because they were unable to interpret their contents. This negatively impacts the blind community as it hinders their ability to communicate their ideas to society through print. Hence, the main purpose of this recognition system is to aid non-blind individuals in understanding braille; it is a tool to assist non-blind individuals in "reading" braille in order to better communicate with and to aid the blind. It also serves the purpose of preserving and reproducing the contents of braille documents that may otherwise be uninterpretable by the general population.

This paper is organized as follows: Section 2 gives an overview of the techniques used in this study as well as previous approaches to OBR. Section 3 gives an introduction to convolutional neural networks, a machine learning approach to OBR which has not been done previously. Sections 4–6 outline the setup of the study, the proposed approach, and the results respectively. Finally, Section 7 provides a discussion of the results, limitations of the study, future directions, as well as a conclusion on the implications of this study.

2. Literature Reviews

OBR has been an active field of research for many years. The act of converting braille images into text can be split into three main parts, image acquisition, cell segmentation, and braille-to-text translation. Researchers in the past have used scanners to acquire braille images; however this technique was incapable of handling distortions such as stains or defects in the paper [10–13]. Murray and Dias demonstrated an attempt to overlook paper quality by scanning only small portions of a braille document at once, but this method proved to be quite slow and laborious [14]. Furthermore, scanners are not practical image acquisition devices since they only work on thin and flat surfaces such as paper; however, daily occurrences of braille may not always fall into this category. As such, it was decided that portable devices such as cellphones and cameras should be considered for image acquisition.

Once a braille image is captured, the individual dots in the image need to be separated from the background. Edge detection is a common image processing technique that finds the edges of objects within an image by detecting discontinuities in image brightness [15]. Edge detection is however not effective with images acquired from cameras since these images may be subject to glare and distortions caused by camera angles and lighting. Thus, in this study, a method that does not rely on differences in image brightness is required for dot recognition.

Due to the uniformity in the placement of braille dots, past research efforts focused on the use of grids and matrices for braille-to-text translation [10–12,16]. By plotting the dots in each character and noting the presence or absence of each of the six dots, they were able to map each braille symbol to its corresponding text character [8]. The mapping of braille symbols to text is often performed using binary codes for each braille cell, which are then converted to decimal codes using the following formula [8,10,16]:

$$D = \sum_{k=1}^6 2^{b_k} l_k \quad (1)$$

where l_k is the dot location as numbered in Figure 1, b_k is either 0 if there is no dot or 1 if there is a dot at location k , and D is the decimal code. Although this method of recognizing braille symbols is rather straightforward, it may not be effective when there is too much noise in the image as image defects could be mistaken for braille dots. In order to address this issue, image pre-processing techniques can be employed to de-noise the background of the braille image. Some researchers have used gray scale image conversion and noise filtering techniques such as increasing the brightness of images [10]. An example of an RGB to grayscale conversion formula used by Li and Yan is as follows [13]:

$$\text{grayscale} = 0.299r + 0.599g + 0.112b \quad (2)$$

where r , g , and b are the values for the red, green, and blue color components, respectively.

An alternative approach that omits the need to de-noise the data is to build a machine learning based recognition system that learns noisy data and improves its performance overtime. Machine learning algorithms have been successfully applied in the past to convert braille symbols into their natural language counterpart. Li and Yan implemented a support vector machine model [13], Morgavi and Morando used an MLP model [11], and Wong et al. applied a probabilistic neural network [17]; however, none so far have used convolutional neural networks to perform braille-to-text conversions.

3. Introduction to Convolutional Neural Networks

An artificial neural network (ANN) is a machine learning method consisting of connected neurons, inspired by the biological nervous systems. In Figure 3, each node represents a neuron while each arrow represents a connection from the output of one node to the input of another node. Each connection is assigned a weight that conveys its importance. In all ANN models, there is one input layer and

one output layer, while the number of hidden layers can be tuned. The process of training a neural network model with multiple layers is known as “deep learning” [18].

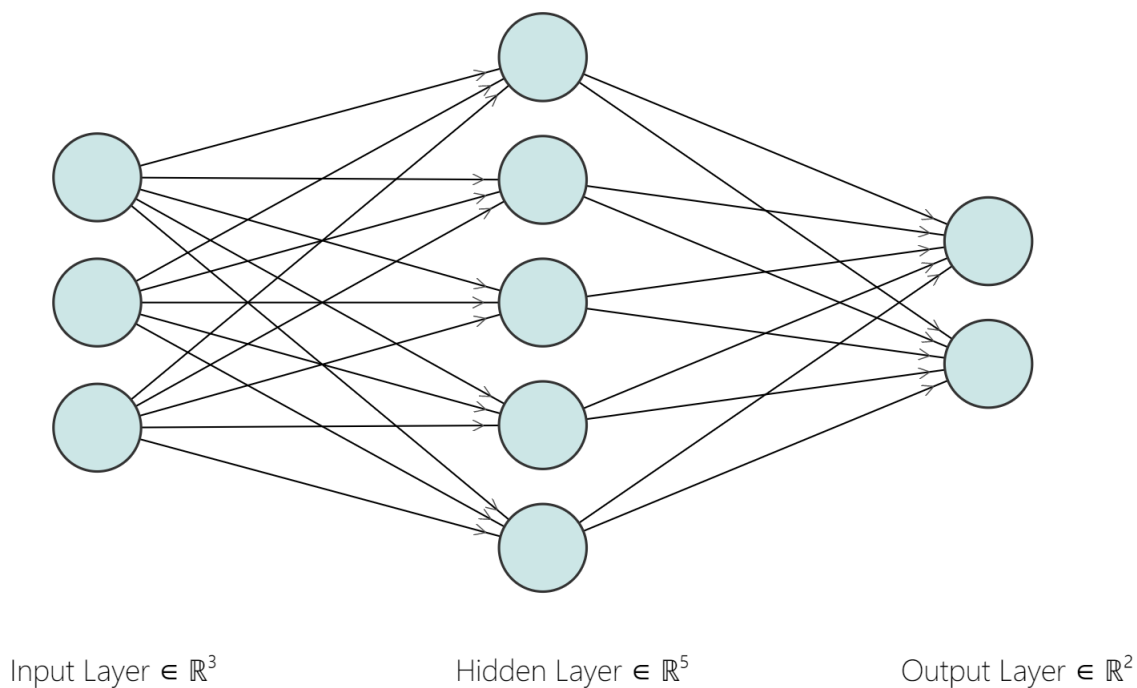


Figure 3. Diagram of a simple artificial neural network (ANN) model consisting of an input layer (three input nodes), a hidden layer (five hidden nodes), and an output layer (two output nodes).

An image consists of a rectangular matrix of pixels specified as width $\times$ height, where width and height are both measured in the number of pixels. Furthermore, there is a third parameter, the number of channels, which specifies whether an image is black and white (number of channels = 1) or colored (number of channels = 3, representing RGB). Thus, an image with dimension $28 \times 28 \times 3$ is an RGB image of size 28 pixels by 28 pixels.

The convolutional neural network (CNN) is one form of ANN that is mostly applied to image or pattern recognition tasks [19]. By using kernels (filters) with a specified size and stride, CNNs are able to maintain the spatial and temporal information of the images, unlike other types of neural networks, such as the multilayer perceptron (MLP), which squeezes image matrices into one dimensional vectors [20]. Most CNN models consist of convolutional layers, pooling layers, and fully-connected layers [21]. Rectified linear unit (ReLU) is a non-linear activation function that rounds negative values to zero and is often used for deep learning in computer vision [22]. These layers consist of some or all of the following tunable hyperparameters: stride, kernel size, and padding. A larger kernel size or stride can overlook features and skip essential details in the images whereas a smaller-sized kernel could provide more information that may not necessarily be relevant [23]. Since pixels in the middle of the image are used more often than those on the border, padding can be added to the border of an image to increase the amount of information preserved on the borders of images [24]. If the image size is $n \times n$, the filter size is $f \times f$, the padding size is p , and the stride is of length s , then the dimension of the output image after each convolutional or pooling layers can be calculated as follows:

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \quad (3)$$

In Figure 4, the input image (blue) has size 5×5 , kernel of size 3×3 , padding (white) of size (1, 1), and stride of (2, 2). The output image (green) is then of size

$$\left\lfloor \frac{5+2-3}{2} + 1 \right\rfloor \times \left\lfloor \frac{5+2-3}{2} + 1 \right\rfloor = 3 \times 3 \quad (4)$$

Each pixel in the output image is a result of the corresponding convolution (grey).

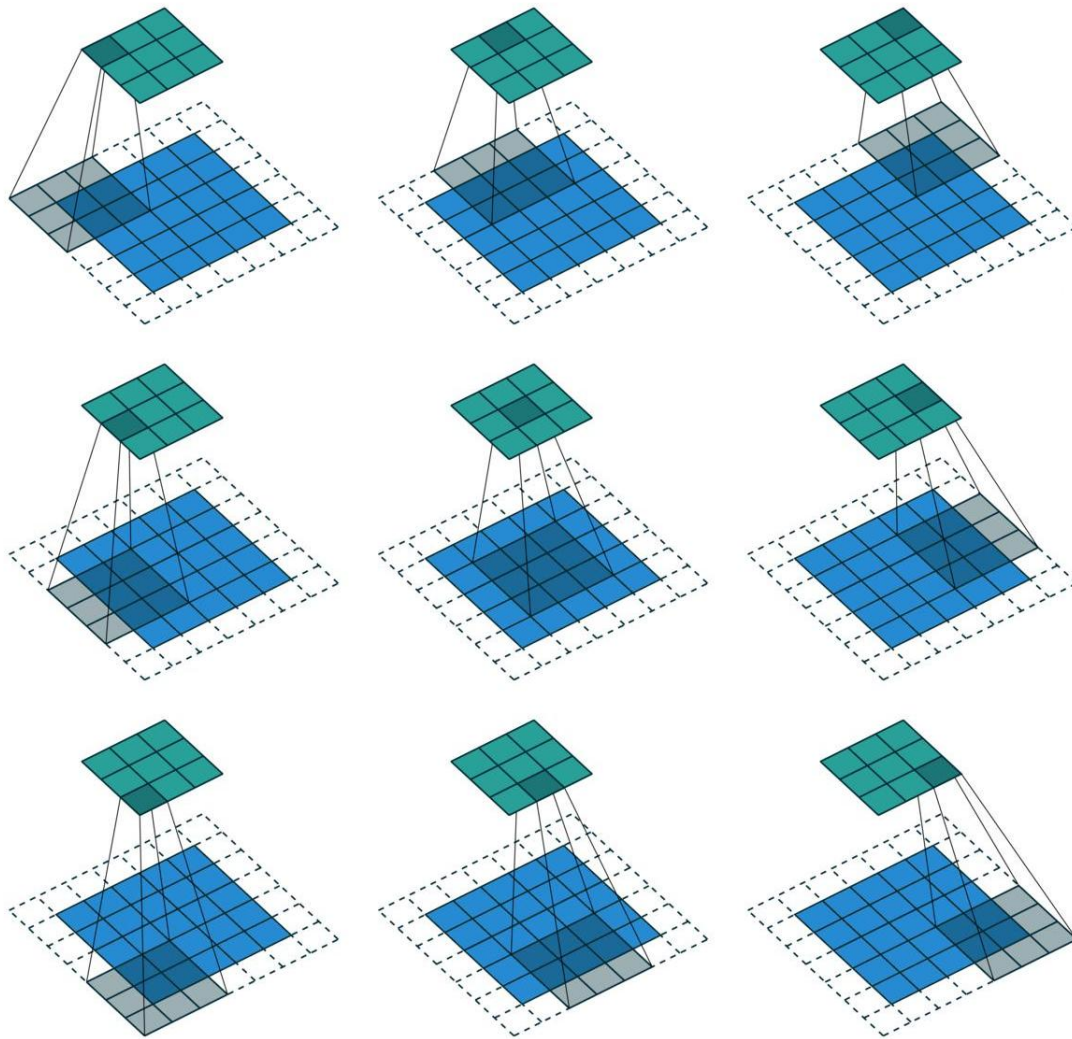
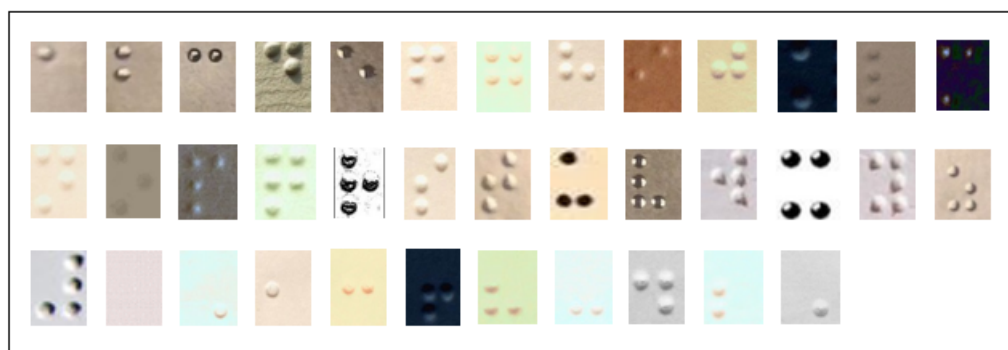


Figure 4. Illustration showing how convolutions work on an image with a single channel for simplicity (adapted from https://github.com/vdumoulin/conv_arithmetic/blob/master/gif/padding_strides.gif).

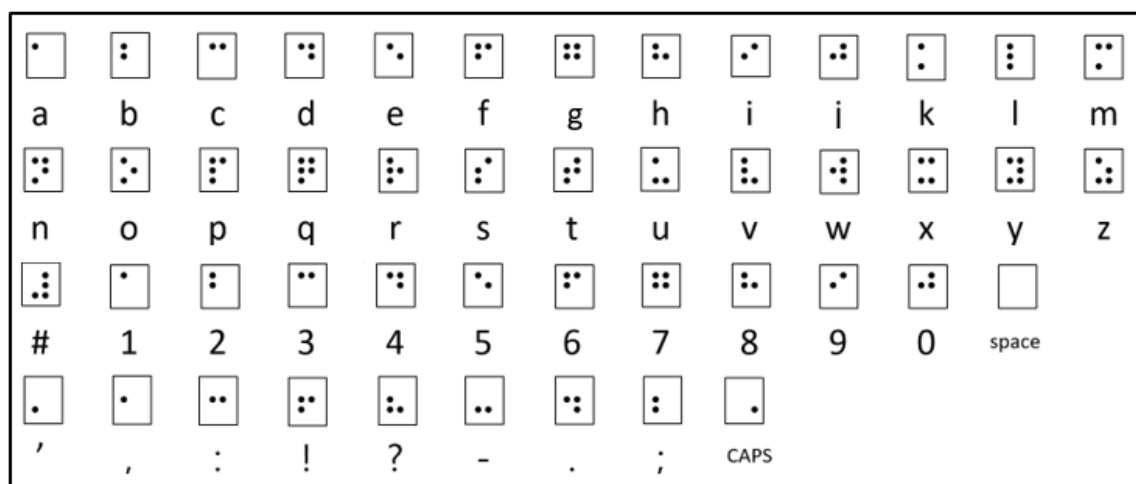
4. Dataset and Setup

The dataset consists of English braille symbols, specifically English Grade 1 braille which is un-contracted, meaning there is a specific braille symbol for each letter in the English alphabet as well as for punctuation. Just as there are insufficient datasets for speech-to-text recognition [25], multiple online searches showed a lack of braille character datasets that fit the needs of this study. The few that do exist are either in foreign languages, have broken links, are not publicly available, or only contain printed images of braille, which are not suitable for training a recognition system that can learn imperfect and blurry images. As such, a new dataset was constructed. In consideration of the fact that braille may not always look embossed or printed in a photograph or may have distortions due to camera angles or lighting, it was recognized that a substantial dataset of characters would be required to train the model. Images of braille were gathered from various sources, including rare book

collections, online images, signs, and labels. Images were then cropped into single characters and the characters were then labeled individually. To ensure that the training, test, and validation sets did not overlap in terms of source images, only a few randomly chosen braille characters were cropped out of each source image and included in the dataset. The data was exported as a csv file, containing the labels and source directories of the images, to be used for model training. Noise and filters including changing sharpness, resolution, contrast, brightness, and color of the images (Figure 5a), were added not only to increase the dataset size, but also to better train the model by allowing it to recognize and predict imperfect images of braille. Since the dataset was created by cropping various images of braille, the images were not uniformly sized. To address this issue, all the images in the dataset were re-sized to $28 \times 28 \times 3$ pixels. These dimensions were then permuted to $3 \times 28 \times 28$ pixels prior to being fed into the model.



(a)



(b)

Figure 5. (a) Sample cropped images from the braille dataset. (b) The 37 different symbols in the braille dataset, which makes up 71 braille character classes. Note: The first 10 letters of the alphabet have the same braille symbols as the 10 numerical digits, so the “#” symbol preceding the digit character is used to denote a numerical symbol. Similarly, the “CAPS” symbol precedes a letter symbol to denote a capital letter.

The dataset consists of approximately 700 copies of each of the 37 different braille symbols, as shown in Figure 5b, which adds up to a total of 26,724 labeled braille character images. The 37 characters are comprised of 26 braille symbols representing letters in the English alphabet, 10 numerical digits and 11 other braille symbols, 8 of which are English punctuation characters. Since there are not enough symbols in the braille system to encompass all the possible letters in the English language, some

symbols are repeated and require an additional symbol preceding them in order to be differentiated. A number character precedes a braille symbol to signify a numerical value rather than a letter of the alphabet, since the first 10 letters of the alphabet have the same braille symbols as the 10 numerical digits, and a capitalization character comes before a letter of the alphabet to represent a capital letter [2]. A space character was also incorporated into the dataset to allow for the recognition of individual words amongst a string of braille. Considering all the different English characters including punctuation, numbers, the space symbol, and lowercase and capital letters, a total of 71 character classes are contained in the dataset.

5. Proposed Approach

A diagram illustrating the general methodology is shown in Figure 6. After a braille image is acquired, it is first subject to pre-processing before it is submitted to the recognition system. The two main steps involved in converting the image into English text are segmentation of the image by the ratio character segmentation algorithm (RCSA) and model training and predicting as shown in Figure 7. Each braille character that was segmented out using the RCSA was individually fed into the model; their outputs were then concatenated to produce the English conversion.

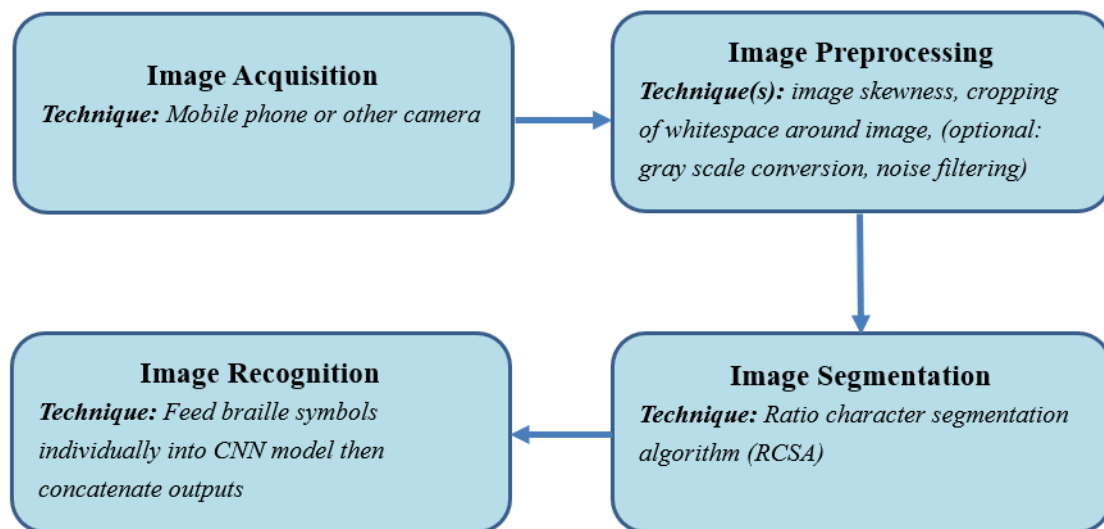


Figure 6. Overview of the steps involved in optical braille recognition (OBR).

5.1. Image Acquisition and Pre-Processing

Braille images can be acquired using devices such as mobile phones and cameras that allow for the easy submission of the image onto the recognition system. Prior to feeding the image into the RCSA and CNN model, certain properties such as image skewness need to be fixed. A simple pre-processing technique for solving the skewing problem is to use linear regression to find the line-of-best-fit through the braille dots to find and correct the angle of skewness [26]. Using the general linear regression formula:

$$Y = bX + a \quad (5)$$

b can be derived by:

$$b = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \quad (6)$$

where n is the number of pixels and x_i and y_i are the coordinates of the image.

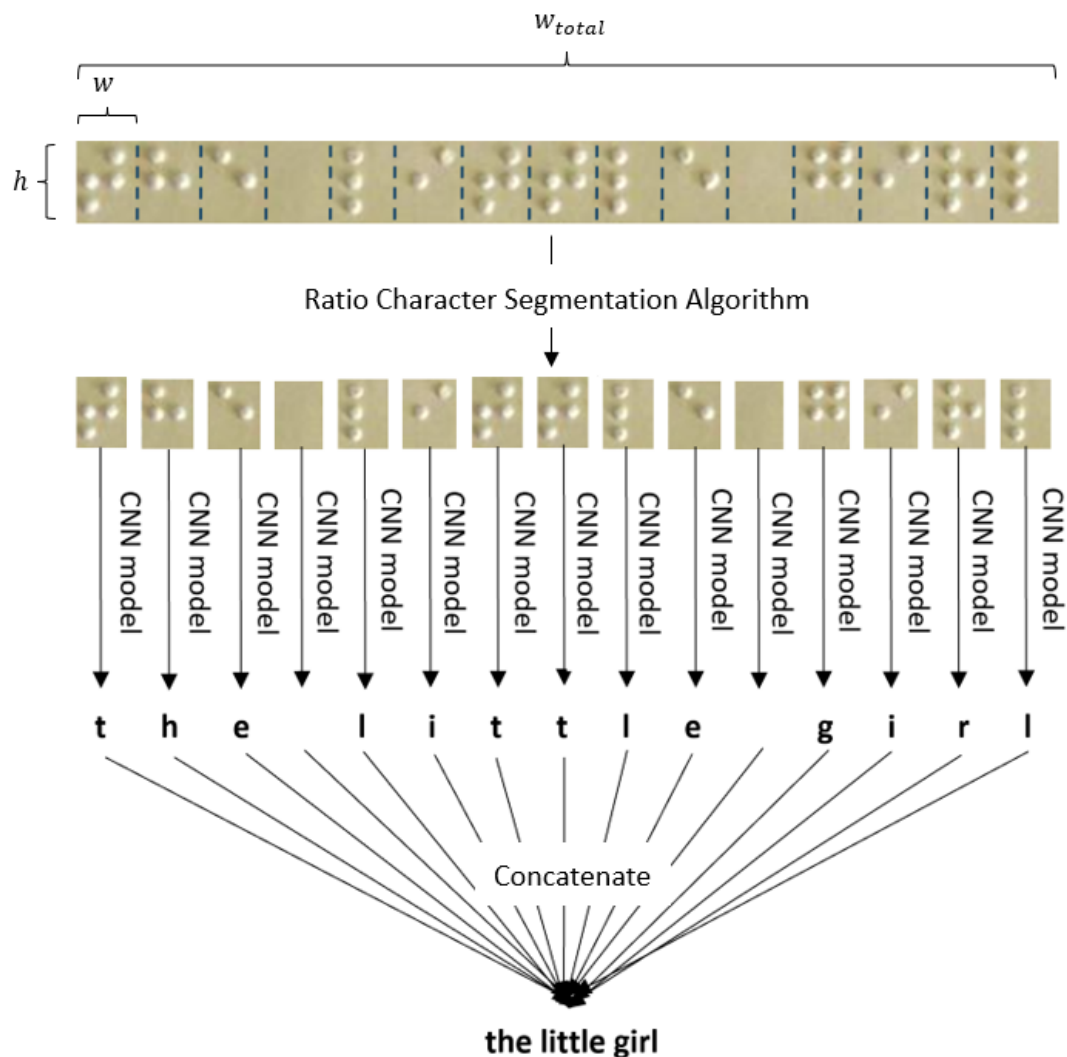


Figure 7. Diagram of the proposed approach for the ratio character segmentation algorithm (RCSA) and model training and prediction.

Another method for addressing image skewness is to simply rotate the image until parallel lines that run vertically and horizontally across the image intersect the dots exactly [27]. Other pre-processing steps described in Section 2, such as de-noising and grayscale conversion are not required in this study since the braille dataset already contains noisy and filtered samples to allow the model to learn imperfect braille images.

An additional pre-processing technique that is required for this recognition system is the cropping of white-space around the braille image to allow for optimal image segmentation in the proceeding step.

5.2. Ratio Character Segmentation Algorithm

Many popular OCR libraries utilize deep learning with image segmentation to distinguish single characters within an image [28]; however, since the spacing around each braille character is uniform due to the three-by-two matrix structure of the braille cell, as seen in Figure 1, a novel ratio character segmentation algorithm was created for partitioning an image of braille into individual

braille characters. This algorithm took the height of the braille image and multiplied it by 0.78 to derive the width of a single braille character within the image. This function was able to be extended to a full line of consecutive braille characters due to the uniformity of the dimensions of each braille cell. The number of characters within a braille image was obtained by dividing the total width of the braille image by the width of a single braille character within the image. Since the ratio segmentation algorithm depended upon a certain height-to-width ratio for each braille character, the expected input of the algorithm was a single-lined braille image that was cropped till the padding around the entire image was approximately one-third of the distance between each braille character, as seen in the upper portion of Figure 7. An in-depth step-by-step procedure for implementing the RCSA is as follows:

1. Crop a single line of braille till the padding around the entire image is approximately one-third of the distance between each braille character.
2. Measure the height (h) of the braille image.
3. The width (w) of the braille characters is $h * 0.78$.
4. Measure the total width (w_{total}) of the line of braille.
5. The number of characters in the braille image (n_{chars}) is thus w_{total} / w .
6. Segment the image into n_{chars} equal segments width-wise. Each character will then be fed individually into the CNN model.

5.3. Convolutional Neural Network Model

A convolutional neural network (CNN) was used to train on the newly constructed dataset due to its wide scale success in image classification and recognition [29]. The CNN model includes two convolutional 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 as seen in Figure 8.

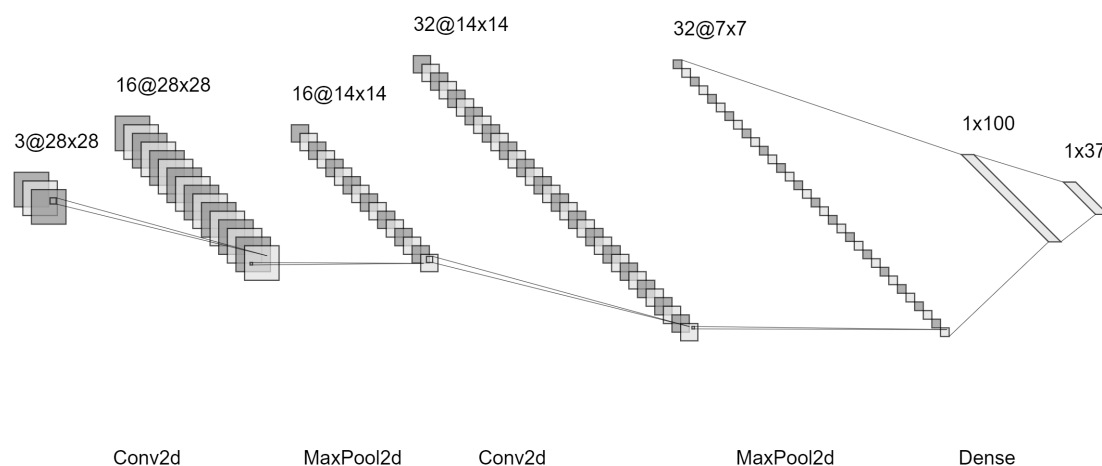


Figure 8. Diagram of the architecture of the convolutional neural network (CNN) model.

A total of 80% of the dataset was allocated to the training set and the remaining 20% was split evenly between the test and validation sets. A training set was used to train the CNN model, while a test set was used to estimate how well the model had been trained to predict and classify different braille images. An additional validation step was included for hyperparameter selection in order to ensure that the parameters did not overfit the training set after thorough training, or in other words, to ensure that the model did not learn the data and noise in the training set too well that it failed to fit additional data and predict future data reliably. However, the test set served as the "true" measure of performance since nothing was optimized for performance on the test set and it was possible for hyperparameter search to overfit the validation set by trying out too many hyperparameters. A batch size of 30 was used for each of the three subsets. Specifications such as the batch size, a learning rate of

1e-4, cross entropy loss as the loss function, and the Adam optimization algorithm as the deep learning optimizer, were determined based on multiple tests on the training performance of the model.

The CNN model with the best hyperparameters is saved then used in the OBR algorithm. Pseudocode for the proposed implementation of OBR is shown in Algorithm 1. Lines 5 to 9 correspond to the RCSA for one input braille image, while lines 11 to 24 detail how the recognition system outputs the corresponding English character based on the model prediction.

Algorithm 1 OBR algorithm.

```

1: model = CNN()
2: model.load()
3: for each input image im do
4:   resize and permute image to  $3 \times 28 \times 28$  pixels
5:    $h \leftarrow$  height of im
6:    $w \leftarrow h * 0.78$ 
7:    $w_{total} \leftarrow$  total width of im
8:    $n_{chars} \leftarrow w_{total} / w$ 
9:   divide image into  $n_{chars}$  equal segments width-wise
10:  translation  $\leftarrow$  ""
11:  for each braille symbol sym in the image do
12:    prediction  $\leftarrow$  model.predict(sym)
13:    if prediction == "#" or "CAPS" then
14:      continue
15:    end if
16:    if preceding symbol prediction is "#" then
17:      translation.append(numerical prediction corresponding to prediction)
18:    else if preceding symbol prediction is "CAPS" then
19:      translation.append(uppercase prediction corresponding to prediction)
20:    else
21:      translation.append(character corresponding to prediction)
22:    end if
23:  end for
24:  return translation
25: return translation
26: end for

```

6. Results

For evaluating the proposed CNN model, 80% of the dataset was allocated for the training set and the remaining 20% was split evenly among the test and validation sets. After thorough training, the CNN model was able to predict and classify images with an accuracy of 98.73% on the test set. A learning curve was plotted after thorough training and testing of the model to check for overfitting as shown in Figure 9.

Since both the training and validation losses slowly plateau overtime, and the validation loss does not suddenly peak above the training loss, this indicates that the model was learning rather than memorizing the training subset. To experiment on the functionality of this AI system, the model and the ratio segmentation algorithm were incorporated into two user interfaces, a Flask web application and an Android application. The web application contains a page that allows users to easily upload and submit a pre-cropped image of braille and receive its English counterpart, while the Android application employs the uCrop API functionality provided by Yalantis (<https://github.com/Yalantis/uCrop>) to allow users to take a picture or select an image of braille from their phone gallery, crop the image, and submit the image into the AI system to receive an instant English conversion.

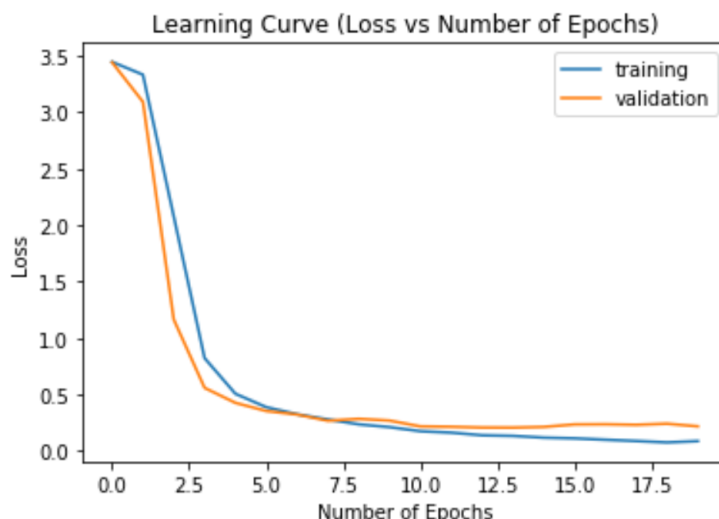


Figure 9. Plot showing the training and validation loss as a function of the number of training epochs.

7. Discussion and Conclusions

There are many methods for acquiring images of braille including the use of scanners and document cameras; however, these devices are not as commonly used as mobile phones, which are portable and can easily capture one's day-to-day observations. The proposed method of braille image acquisition makes use of portable phones or camera devices that can capture images of braille and allow users to receive instant braille-to-text conversions. Although this approach may be more prone to noise due to the innate nature of camera-captured images, this method can be employed seamlessly throughout real-world encounters of braille, which are not limited to printed braille. Thus, by constructing the dataset from a diverse collection of braille images, the feat of recognizing and predicting blurry, unfocused, and distorted images was able to be accomplished.

The initial approach to character segregation was to utilize fewer layers in the model and instead incorporate edge detection to distinguish braille character dots from their background. Edge detection is a common machine learning technique used to find boundaries of objects within an image, which would enable the individual braille dots to be extracted from the image, as suggested by Ng et al. [15] who separated embossed pages of braille into distinct dots and background sections. However, it was decided that this technique would not be used as it relies on the detection of discontinuities in brightness which is not as effective when the data contains background noise.

In addition, although the standard approach to OBR is to divide each braille cell into 3 by 2 grids and account for the presence or absence of a dot in a particular grid [28], it was recognized that this method may not be as suitable for images of braille that could be highly affected by camera angles and lighting. A new approach was taken with regards to character segmentation followed by model training on the individual images. This approach allows the recognition system to recognize the white-space around characters rather than the presence or absence of individual dots in each braille cell and proved to be equally efficient at segregating and recognizing individual braille characters. However, this character segmentation approach relies on a ratio segmentation cropping algorithm that is very sensitive to an excess or lack of padding, thus requiring an additional step of cropping the image for optimal translations.

The AI system has been implemented to recognize and translate the characters and words of a single line of braille into English text with an accuracy of 98.73%. This was achieved by linking the CNN model to two recognition techniques: character and word recognition. Single characters were able to be recognized through the use of the RCSA while individual word recognition was achieved by adding a space character as one of the 37 symbols in the braille dataset. A diagram illustrating this progress is presented in Figure 10.

This approach presents some limitations that may be sources of future investigation. Line-by-line conversions of braille may not be the most efficient method for converting large lengths of braille into English text. In the future, image segmentation should be incorporated into this AI system, which will provide an additional function of partitioning an image of a page of braille into multiple lines of braille that can then be interpreted and converted into English text by the current system. An alternative approach is to extend the ratio character segmentation technique into line ratio segmentation, so that the segmentation of characters and lines of braille can be performed simultaneously.

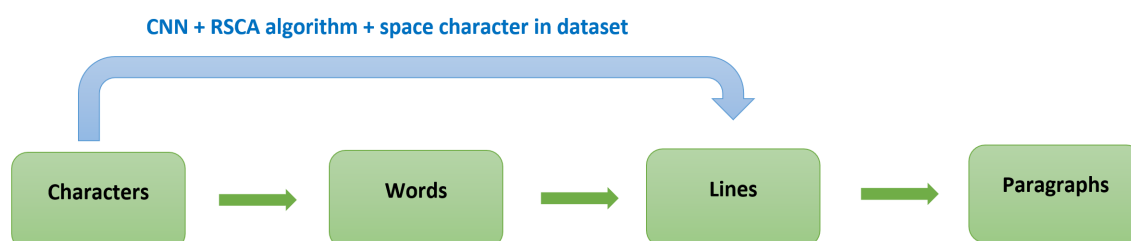


Figure 10. Flow chart showing the typical steps needed to predict paragraphs of braille (in green). The combination of the CNN model and the RSCA algorithm with the addition of a space character in the dataset (in blue) enables the jump from predicting braille characters to braille lines.

Future directions could also involve adding additional characters to the braille dataset in order to encompass the entire braille system and to further improve the accuracy of braille-to-text conversions. At the moment, it is difficult to make comparisons with other research works related to OBR since the datasets used amongst these works are inconsistent in terms of language and acquisition sources. However, it is worthy to note that other OBR studies have also yielded high prediction performance. Morgavi and Morando achieved 98% accuracy by using a scanner and an MLP model on printed braille documents [11]. Li and Yan’s approach yielded 95% by also using a scanner and printed braille documents, except with a support vector machine model [13]. The consistent achievements of high prediction accuracy amongst different OBR approaches is most likely related to the uniformity in the placement of the braille dots.

Additionally, both user interfaces are currently undergoing alpha testing to identify any potential issues that may arise when in use and are intended to be made public once testing is complete. This will allow for error-free real-world conversions of braille images to English text.

Braille may not be a system that is familiar to many, but it is certainly important to those with impaired vision or those who are close to individuals who rely on the braille system. By leveraging a phone camera capturing method, a simple ratio character segmentation algorithm, and a neural network approach to OBR, this technology has the potential to deeply impact the blind community as it provides a solution to asymmetric communication flow between the blind and non-blind by allowing the braille system to be universally interpreted and understood.

Funding: This work was partially supported by the Ministry of Science and Technology of Taiwan under grant number MOST 106-2410-H-230-001-MY2.

Conflicts of Interest: The author declares no conflict of interest.

References

- Liste, S. *Blindness and Vision Impairment*; World Health Organization: Geneva, Switzerland, 2018.
- Millar, S. *Reading by Touch*; Routledge: London, UK, 1997.
- of the Blind, N.F. *Facing the Truth, Reversing the Trend, Empowering the Blind*; Technical Report; Jernigan Institute: Baltimore, Maryland, 2009.
- Wiazowski, J. Can Braille Be Revived? A Possible Impact of High-End Braille and Mainstream Technology on the Revival of Tactile Literacy Medium. *Assitive Technol.* **2014**, *26*, 227–230.
- Reed, B.T.M.W.R. *Assistive Technologies in the Library*; American Library Association: Chicago, IL, USA, 2011.

6. Grunig, J.E.; Hunt, T. *Managing Public Relations*; Holt, Rinehart & Winston: New York, NY, USA, 1984.
7. Srinath, S.; Kumar, C. An Insight into Optical Braille Character Recognition. *Int. J. Comput. Appl.* **2011**, *33*, 1–4.
8. Isayed, S.; Tahboub, R. A review of optical Braille recognition. In Proceedings of the 2015 2nd World Symposium on Web Applications and Networking (WSWAN), Sousse, Tunisia, 21–23 March 2015, pp. 1–6.
9. Sisco, L.J. Braille preservation: recognising and respecting archival materials produced by and for the blind. *Arch. Manuscr.* **2015**, *43*, 18–28.
10. Kumar, C.N.R.; Srinath, S. A Novel and Efficient Algorithm to Recognize Any Universally Accepted Braille Characters: A Case with Kannada Language. In Proceedings of the 2014 Fifth International Conference on Signal and Image Processing, Bangalore, India, 8–10 January 2014, pp. 292–296.
11. Morgavi, G.; Morando, M. A neural network hybrid model for an optical Braille recognition. In Proceedings of the International Conference on Signal, Speech and Image Processing 2002 (ICOSSIP), Orlando, FL, USA, 13–17 May 2002.
12. Al-Shamma, S.D.; Fathi, S. Arabic braille recognition and transcription into text and voice. In Proceedings of the 2010 5th Cairo International Biomedical Engineering Conference (CIBEC), Cairo, Egypt, 16–18 December 2010, pp. 227–231.
13. Li, J.; Yan, X. Optical braille character recognition with support-vector machine classifier. In Proceedings of the International Conference Computer Application and System Modeling, Taiyuan, China, 22–24 October 2010; Volume 12, pp. V12–V219.
14. Murray, I.; Dias, T. A portable device for optically recognizing Braille. In Proceedings of the The Seventh Australian and New Zealand Intelligent Information Systems Conference, Perth, Western Australia, 18–21 November 2001, pp. 302–306.
15. Ng, C.M.; Ng, V.; Lau, Y. Regular feature extraction for recognition of Braille. In Proceedings of the Third International Conference on Computational Intelligence and Multimedia Applications: ICCIMA'99, New Delhi, India, 23–26 September 2002, pp. 302–306.
16. Zhang, S.; Yoshino, K. A braille recognition system by themobile phone with embedded camera. In Proceedings of the Second International Conference on Innovative Computing, Information and Control, ICICIC 2007, Kumamoto, Japan, 5–7 September 2007, p. 223.
17. Wong, L.; Abdulla, W.; Hussmann, S. A software algorithm prototype for optical recognition of embossed Braille. In Proceedings of the 17th International Conference on Pattern Recognition (ICPR), Cambridge, UK, 23–26 August 2004, Volume 2, pp. 586–589.
18. Jain, A.K.; Mao, J.; Mohiuddin, K.M. Artificial neural networks: A tutorial. *Computer* **1996**, *29*, 31–44.
19. O'Shea, K.; Nash, R. An Introduction to Convolutional Neural Networks. *ArXiv* **2015**, arXiv:1511.08458.
20. Wu, J. *Introduction to Convolutional Neural Networks*; National Key Lab for Novel Software Technology Nanjing University: Nanjing, China, 2017.
21. CS231n Convolutional Neural Networks for Visual Recognition. Available online: <https://cs231n.github.io/convolutional-networks/> (accessed on 29 May 2020).
22. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). *ArXiv* **2018**, ArXiv:1803.08375.
23. Xu, L.; Ren, J.S.J.; Liu, C.; Jia, J. Deep Convolutional Neural Network for Image Deconvolution. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: New York City, NY, USA, 2014; pp. 1790–1798.
24. Prusa, J.D.; Khoshgoftaar, T.M. Improving deep neural network design with new text data representations. *J. Big Data* **2017**, *4*.
25. Polat, H.; Oyucu, S. Building a Speech and Text Corpus of Turkish: Large Corpus Collection with Initial Speech Recognition Results. *Symmetry* **2020**, *12*, 290.
26. Babadi, Y.; Jafari, S. Novel grid-based optical braille conversion: from scanning to wording. *Int. J. Electron.* **2011**, *98*, 1659–1671.
27. Al-Salman, A.; El-Zaart, A.; Al-Suhaibani, Y.; Al-Hokail, K.; Al-Qabbany, A.A. An efficient braille cells recognition. In Proceedings of the 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), Chengdu City, China, 23–25 September 2010, pp. 1–4.

28. Mehul, G.; Ankita, P.; Namrata, D.; Rahul, G.; Sheth, S. Text-based Image Segmentation Methodology. *Procedia Technol.* **2014**, *14*, 465–472.
29. Khan, S.; Rahmani, H.; Shah, S.A.A.; Bennamoun, M.; Medioni, G.; Dickinson, S. *A Guide to Convolutional Neural Networks for Computer Vision*; Morgan & Claypool Publishers: San Rafael, PH, USA, 2018.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).