

Conversion of Braille Document to Regular English Document using VGG16 BC-CNN Model

Vishwanath Venkatesh Murthy¹, Nagesh B S² and Shilpa C³

¹⁻²Associate Professor, MCA, RNSIT, Bengaluru

E-mail: vishwanathmurthy@rnsit.ac.in, nagesh.bs@rnsit.ac.in

³Assistant Professor, MCA, RNSIT, Bengaluru

Email: shilpa.c@rnsit.ac.in

Abstract—Braille metallic documents are the only medium the visually impaired people will use for reading. Due to their heavy bulky physical nature, it is cumbersome to move them manually from one area to other over the globe. Apart from distribution, it is also very tedious to maintain them as it is a natural tendency of plates to get wear and tear in different weather conditions. Hence by translating the Braille documents to natural text, will help in distributing it over the network across globe as well as it can be maintained in digital form for later reproduction into Braille form.

This research paper presents the proposed work that translate a Braille document to the corresponding English sentence. The preprocessing stage includes the removal of unwanted dots and skew detection and removal methods. Segmentation stage performs the row and column segmentation to locate the cell pattern in Braille image, the mapped cells are cropped and stored in Image array. Feature extraction stage uses the cropped Braille cell image array as input to Braille classifier model built using VGG16 CNN model. The model maps these images it to the corresponding characters and thus generate the English statement.

Index Terms— Accuracy, Braille, Cell, Gradient, Gaussian filter, Kernel, Preprocessing, Sobel operator, Skew, VGG16.

I. INTRODUCTION

A typical Braille metallic paper is exactly 11x11-1/2 inches in size. The metallic paper can have utmost 25 lines of Braille document. Each line of Braille document can have a maximum of 40-42 cells. A Cell pattern of Braille comprises of six elevated dots in 3x2 matrix form. Each row has 2 dots whereas, every column has 3 dots. A typical Braille cell is shown in figure-1(a) whereas figure 1(b) presents the physical properties of Braille cells. The physical properties are used to extract the Braille cell in feature extraction stage of research work.

A. Aim of the Research

To provide the better solution to convert Braille scanned documents into computer readable form or any other non-Braille language like English.

B. Research Objective

1. Preprocess the scanned image using techniques like contrast enhancement, histogram, Laplacian filter and Sobel filter to remove any noise or skew.

2. Segment the image using row and column segmentation to crop the Braille cells.
3. Apply VGG16 Convolution Neural network (CNN) classifier model to map the Braille cell with respective English alphabet.
4. Concatenate the classified alphabets from CNN model and translate the to the English sentence.

The research paper is ordered into 5 segments. Segment-I outline the introduction and physical properties of Braille cells, it also tells the Aim and objective of research work. Segment -II describes the methodology that described the various stages of research work. Segment-III delivers the results and discussion. Finally, section-IV concludes the research work.

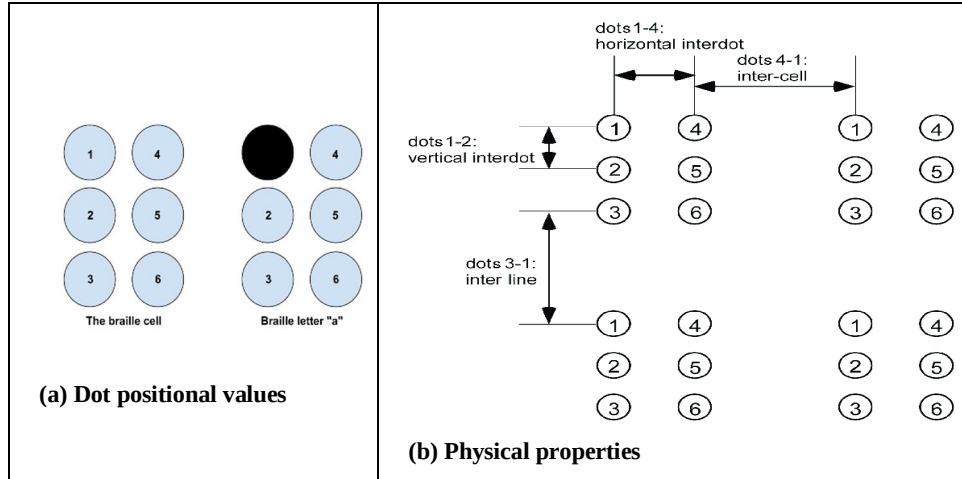


Figure-1: Positional value of dots and their physical properties

II. METHODOLOGY

A. The Proposed System Has Major 5 Stages:

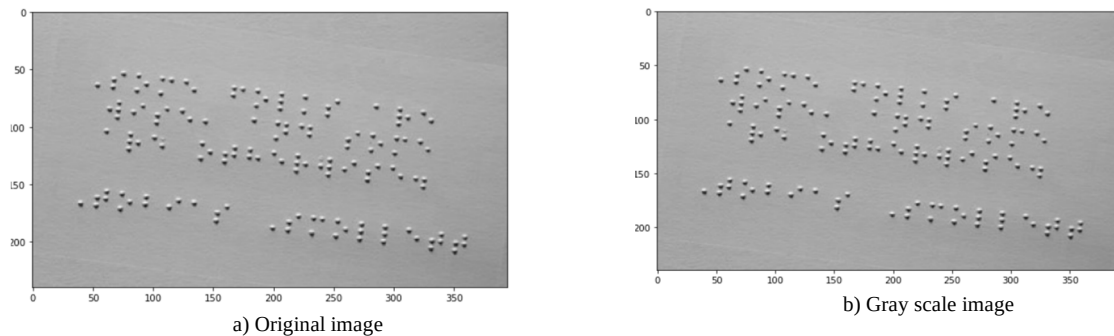
- a) Preprocessing for skew removal
- b) Dot enhancement
- c) Row and column Segmentation to map Braille cells
- d) Feature extraction using VGG16 BC-CNN model
- e) Translation to English sentence

Preprocessing

The preprocessing stage is performed again in four steps. In the first step input color image is converted to Gray Level. In the second step, Corner points are detected using corner function of MATLAB as shown in equation (1).

$$\text{corner}(\text{img}, 100) \quad (1)$$

parameter 100 indicates the total corner plots that will be recognized. In step 3, Edge detection is performed using Sobel edge detector. Finally in step-4, using Gaussian filter the tilt angle is determined and the skew is removed. The figure-2 shows all the output of various preprocessing steps.



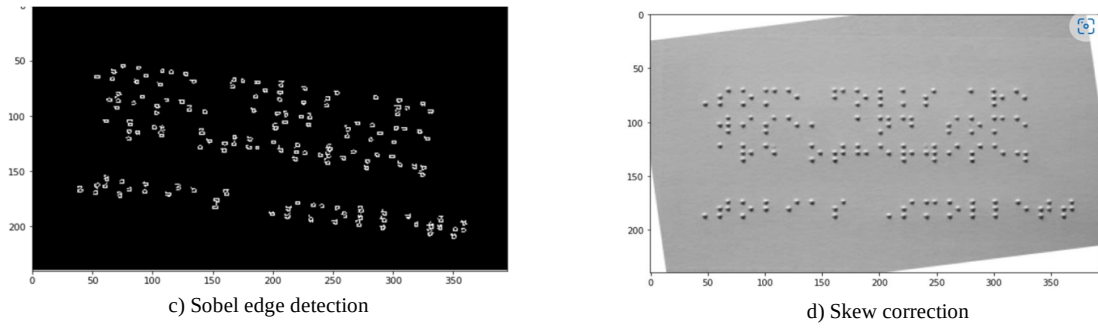


Figure-2: Output of Sobel edge detector and skew correction

Sobel operator basically is used to detect the edges in vertical as well as horizontal directions. If any unwanted dots appear in the processed image, they can be removed with the help of canny edge detector or erosion method.

Dot enhancement

After the removal of skew the image is processed for obtaining the binary gradient using morphological operations. The erosion operation is specified by $(g = f \ominus S)$ operation, where 'f' specifies binary image and 'S' is the structuring element. Dilation highlights the portions of the objects. It fills the open holes. Dilation is denoted by $(g = f \oplus S)$ operation. The results of the enhanced dots after erosion and dilation are shown in figure 3.



Figure-3: Enhanced Image after Skew removal

Row and Column Segmentation

In the preprocessing, first all the edges of the cells are marked. The segmentation process performs the row and column segmentation to map the Braille cells. Mapped cells are cropped and stored in Image array which will be the input to VGG16 model.

Row Segmentation: The row segmentation is performed for drawing the Bounding Box around the Braille content as shown in figure-4, identify the top left corner points (xMin, yMin) and bottom right corner point (xMax, yMax) which can be used for cell mapping in column segmentation.

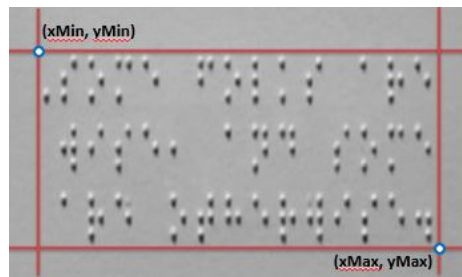


Figure-4: Boundary mapping by row segmentation

Column Segmentation: After row segmentation is successfully performed, in character segmentation, the i th character can be mapped using the following algorithm.

Algorithm-1:

Find the coordinates of any (cell- i) using loop
 $c = w + d1, r = h + d2$

```

for i = 1 to NumOfRows
  for j = 1 to NumOfCols
    Xi-min = xMin + (j - 1) * c
    Yi-min = yMin + (j - 1) * c
    Xi-max = Xi-Min + w
    Yi-max = Yi-Min + h
    Image_array = crop (Xi-Min, Yi-Min, Xi-Max, Yi-Max)
  end
  xMin = xMin + r
  yMin = yMin + r
end

```

In the algorithm, w = width of Braille cell, h = height of Braille cell, $d1$ = Horizontal gap between two adjacent Braille cells in a row and $d2$ = vertical gap between two adjacent Braille cells in a column as shown in the figure-5. Figure-6 shows the sample cropped images using Algorithm-1.

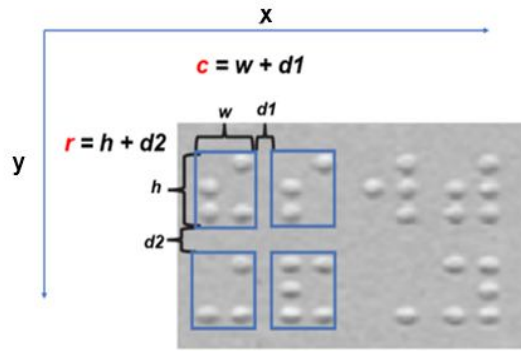


Figure-5: Braille character Segmentation for Sample two line

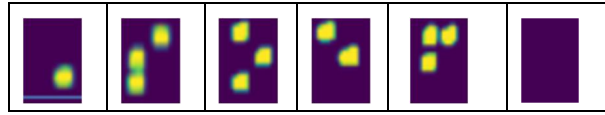


Figure-6: Sample cropped images

The next cell is mapped by increasing the column index by (3 + intercell distance). Intercell distance is the distance from dot4 of first cell to dot1 of second cell using physical properties and cropped images are shown in figure-6. Cropped images are stored in an image array and fed as input to the proposed deep learning-based CNN model.

d) Feature Extraction and Classification Using CNN Model

In Feature extraction and classification, the deep learning-based VGG16 CNN model is designed to classify the cropped images in segmentation. The model recognizes the cropped images and translates it into Braille text.

The Braille classification model using VGG16 CNN model was built using TensorFlow version '2.9.1' and Python version of '3.9.12' using Jupyter Notebook.

Total 1560 images are input as a learning model. The Braille images are downloaded from Kaggle website and stored into local directory. Local directory was split into subdirectories from 'A' to 'Z', each contains 60 images of same class. All images are preprocessed and fed to the proposed model. The augmentation is applied using 'ImageDataGenerator' library to produce addition dataset for same image as shown in figure7.

The summary of the model can be displayed using Keras model.summary() library. The summary of the Braille classification model is displayed in figure 8.

Total images considered for the model are $26 \times 60 + 30 = 1590$ images. The said model was run on 20 epochs. The output values of the model are shown in figure-9.

The figure 9 shows the validation accuracy curve. It is proved by graph that the training and validation accuracy increases by increasing the no of epochs on the model while the training and validation loss decreases to great extent. The accuracy of the model is 95%. The model is developed using python program on TensorFlow and Keras environment.

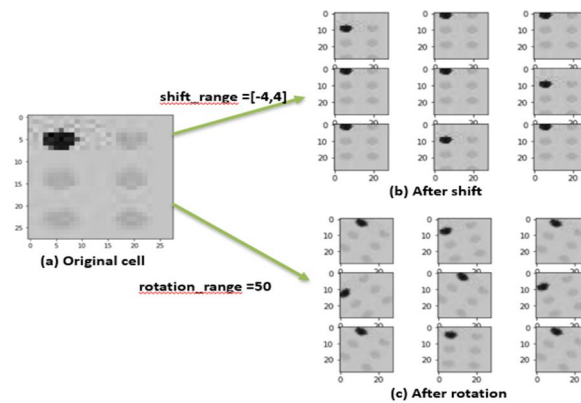


Figure-7: Image augmentation

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 128)	802944
dense_1 (Dense)	(None, 26)	3354

=====
Total params: 816,442
Trainable params: 816,442
Non-trainable params: 0
=====

Figure-8: Summary of the model

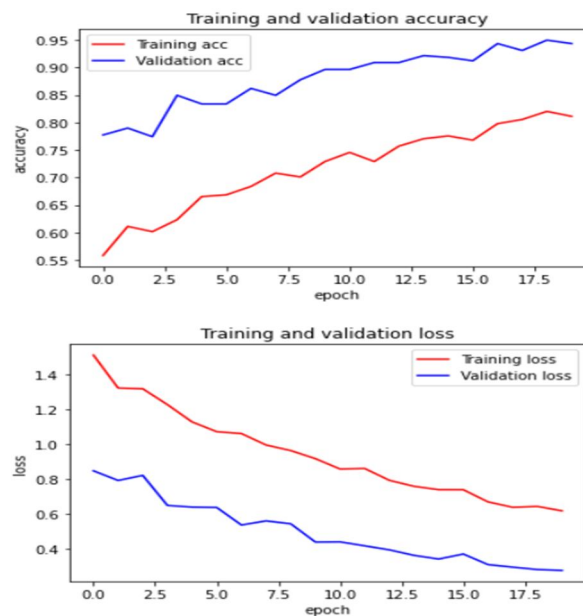


Figure-9: Validation accuracy of model

Translating Braille to English Sentence

Each cropped image from the image array is passed as input to the Braille classifier model. The model performs the preprocessing on each of this image. Apply the convolution and pooling operations to extract the image information and finally produces the class labels that represent the English alphabets.

III. RESULTS AND DISCUSSION

The proposed algorithm is evaluated using various metrics as denoted in eqn-2 to eqn-6.

$$Accuracy = \frac{y}{x} \times 100 \quad 1)$$

$$Error = \frac{|x-y|}{x} \times 100 \quad 2)$$

In the equation-2 and equation-3; the term 'x' indicates total number of cells supplied to algorithm. The symbol 'y' indicates number of correctly recognized cells.

$$Precision = \frac{TP}{TP+FP} \quad 3)$$

$$Sensitivity/Recall = \frac{TP}{TP+FN} \quad 4)$$

$$F1 \text{ score} = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad 5)$$

In the equation-4 to equation-6; the term 'TP' represents the class 'recognized correctly'. 'FP' represents the class 'wrongly recognized' and 'FN' represents the class "Present but Not recognized" from the background. Initially the preprocessing is performed on MATLAB. Later the whole algorithm was implemented in Python using the Jupyter notebook of Anaconda navigator-3 software. The proposed method was processed on the sample of 5 Braille documents and results of TP, FP, and FN are noted in table-I.

Table I: OUTCOME OF SAMPLE BRAILLE DOCUMENTS

# Braille Doc	Input cells (x)	TP	FP	FN
1	545	525	5	20
2	861	821	12	40
3	996	931	22	65
4	327	320	3	7
5	734	702	25	32

The output of Table-I is evaluated to obtain Precision, Recall as well as F1-score for sample 5 documents is shown in the table-II.

TABLE II: RESULTS OF PRECISION, RECALL AND F1 SCORE

# Braille Doc	Input cells (x)	Precision	Recall	F1
1	545	0.99	0.96	0.97
2	861	0.98	0.95	0.96
3	996	0.97	0.93	0.94
4	327	0.99	0.97	0.97
5	734	0.96	0.95	0.96

IV. CONCLUSION AND FUTURE WORK

The proposed algorithm was implemented in Python using various existing libraries for preprocessing and feature extraction that leads to segmentation. The Accuracy and error rate for the sample 5 documents is shown in table-III.

TABLE-III: RESULTS OF ACCURACY AND ERROR RATE

Doc No	Input cells (x)	Correctly recognized (y)	Accuracy%	Error %
1	545	525	96.33	3.67
2	861	801	93.03	6.97
3	996	910	91.37	8.63
4	327	320	97.86	2.14
5	734	702	95.64	4.36

Accuracy rate for the given Braille documents observed from table-III is between 91% to 99% while error rate is below 8%.

The proposed work is successfully applied on single side grade-1 and grade-2 Braille documents. In future work the algorithm can be modified and expanded to translate two side embossed Braille documents.

REFERENCES

- [1] A. Al-Salma, Y. Al Ohali, M. Al Kanhal, and A. AlRajih, "An Arabic optical braille recognition system," in *Proceedings of the 1st International Conference in Information and Communication Technology and Accessibility, ICTA, Hammamet, Tunisia, January 2007*
- [2] Bipasha Biswas, "Single-Sided Braille into Corresponding Text", *International Journal of Recent Technology and Engineering (IJRTE)*, ISSN: 2277-3878, Volume-8 Issue-6, March 2020
- [3] C. N. R. Kumar and S. Srinath, "A Novel and Efficient Algorithm to Recognize Any Universally Accepted Braille Characters: A Case with Kannada Language," doi: 10.1109/ICSIP.2014.52.
- [4] https://en.wikipedia.org/wiki/Braille_ASCII
- [5] <https://pixabay.com/photos/braille-writing-keys-metal-plate-52554/>
- [6] Kirill Smelyakov, Anastasiya Chupryna, Dmytro Yeremenko, Anton Sakhon, & Vitalii Polezhai (2018). "Braille Character Recognition Based on Neural Networks". 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), 509-513.
- [7] S. Iqbal, A. Ali, M. Younus, M. Huzaifa, and Z. Abbas, "Braille Instant Translator", pp. 327-328, National University of Computer and Emerging Sciences, Islamabad, Pakistan, 2017
- [8] Yuko Shimomura, Hiroyuki Kawabe, Hidetaka Nambo, & Shuichi Seto. "The Translation System from Japanese into Braille by Using MeCab". *Proceedings of the Twelfth International Conference on Management Science and Engineering Management*.
- [9] Hanumanthappa, M., & Murthy, V. V. Optical Braille recognition and its correspondence in the conversion of Braille script to text-A literature review: *International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*: 2016. In *Page s* (pp. 297-301).
- [10] V. V. Murthy and M. Hanumanthappa, "Improving Optical Braille Recognition in Pre-processing Stage," 2018 *International Conference on Soft-computing and Network Security (ICSNS)*, Coimbatore, India, 2018, pp. 1-3, doi: 10.1109/ICSNS.2018.8573664.