KanOCR: Conversion of printed Kannada documents to editable form using Convolutional Neural Networks

Pradyumna Mukunda, Niraj S Prasad, Santhosh DM
PES Institute of Technology, Bangalore

Dr.Mamatha HR
Professor, Department of Computer Science and Engineering, PES University, Bangalore

Abstract - **'Optical Character Recognition', or OCR, is basically converting an image containing text to an editable text format. This paper concentrates on OCR for Kannada language text. Unlike English, Kannada Language has a greater number of characters since it includes kaagunithas, vattaksharas, etc. This makes recognition of the characters much more complex.**
The characters are first extracted from the images using various segmentation methods. Then they are fed as inputs to Convolutional Neural Networks (CNNs) for recognition. It is a classification problem. The recognized characters are next translated into Unicode and then printed.
KanOCR's use of Supervised Learning in the form of CNNs for character recognition provides it a higher accuracy than OCR systems that use image processing techniques for recognition. This system has been tested with varying fonts. A total number of 37 sample documents is used for experimentation. The system has been developed for only printed Kannada Text as of now.

Index Terms - Base-line Identification, CNN, Kannada, Neural Network, Optical Character Recognition, Pre-processing, Python, Segmentation.

I. INTRODUCTION

Optical Character Recognition is basically an electronic or mechanical conversion of image text into editable text format. The image could be hand-written, a scanned document, a photograph with some content, or any printed text. Digitizing printed text is essential in editing and storing valuable information on a software platform.
Demand for a stable regional language OCR system has grown over the past few years. With the advance in technology, a smartphone can now read any text on an image captured.
Using OCR systems, organisations can now edit documents more efficiently. Digital text data can be stored and saved without much risk of loss, unlike a physical document. Traffic police now accept digitally scanned documents. In case of a change in information, scanning and editing a document is currently practiced in a lot of places. Scanning and editing a file is one of the best approaches one can find.

There is a high demand for storing information on to a digital storage device from the data available in printed or handwritten documents or images to later re-utilize this information by means of computers. And one way to store the information to a computer system from these printed documents could be to scan the documents or files. But to re-utilize this information, it would be very challenging to read or modify text or other information from these image files. Therefore, a method to automatically retrieve and store information, in the form of text, from image files is needed. This is where Optical Character Recognition comes into play.

The paper is organized as follows. Section II is a background study on OCR systems. Section III describes the methodology and implementation of the project. Section IV shows the results and analysis. Finally Section V is the conclusion and references

II. BACKGROUND STUDY

Types of OCR Systems:
There are different types of OCR systems, such as:

- OCR which targets typed text one character at a time.
- OWR or Optical Word Recognition which focuses on typed text one word at a time. This is possible only for languages that use a word-divider or a space.
- ICR or Intelligent Character Recognition that involves Machine Learning, ICR targets one sentence at a time.
- IWR or Intelligent Word Recognition that involves Machine Learning. IWR works for handwritten print scripts really well. Cursive text and those glyphs that are not separated in cursive works really well with IWR.

OCR usually analyses a static document; hence it is labelled 'offline.' Handwriting moving analysis can be used for hand-written text to generate the output in real-time. This process is called 'Dynamic Character Recognition' or 'ICR', or 'Real-time character Recognition.'

Techniques used in OCR:

Pre-processing:

OCR systems might have to make some changes in the image before running the algorithm in order to get much higher accuracy and hit rate. To get a successful output, the following techniques might have to be considered –

1. *De-Skew*: The image might be tilted, or the text image might not be aligned perfectly as the system wants to it be. So some de-skewing needs to be done before OCR takes place. The document may be tilted either clockwise or counter-clockwise to align it perfectly.
2. *De-speckle*: The system smoothens the image and removes the negative and positive spots of the picture. This ensures greater accuracy.
3. *Binarisation*: In case the input is a color image, the system might not be able to differentiate between different colors. So is converts the image to greyscale and then performs OCR.
4. *Line and Word Detection*: It separates the words if necessary. It also establishes a baseline for words.
5. *Script Recognition*: In the same document, there might be different scripts. The system must be able to realize which script it is working with.
6. *Normalizing* the scale and the aspect-ratio.
7. *Character isolation and segmentation*: this is useful in Character OCR. Words are broken down into many characters, hence making reading smooth and accurate.

Character Recognition:

There are currently two types of the core OCR algorithm.
Matrix Matching is a technique wherein a stored glyph is compared with the image. It is also called pattern recognition and image correlation. This technique works best with typewritten text. It does not work well when a different font is encountered.
Feature Matching breaks down the glyphs into lines, lops, line-intersections, and line directions. These features are compared with the abstract vector-like representation of the character. Feature detection in Computer Vision is applicable to this type of OCR System. This method uses Machine Learning and is the most modern OCR Software. In this paper, we have used an entirely different technique in the form of CNN.
Specific software such as Tesseract and Coneiform uses a two pass approach to character recognition. This is useful when the font is distorted, blurred, or faded.

Post Processing:

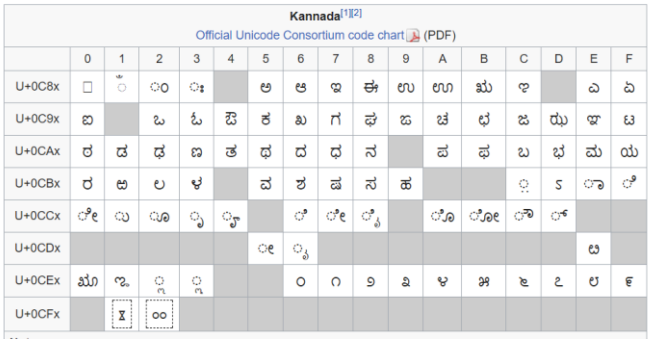
A dictionary can be added to check the accuracy of the word, and it can be processed with maximum efficiency possible. The output stream can be a plain text file of characters. Sophisticated OCR systems also preserve the layout and font perfectly. In order to further optimize results from an OCR API in post-processing, the Levenshtein Distance algorithm can be used.

Convolutional Neural Networks:

In a fully connected neural network, every neuron takes inputs from each and every neuron in the previous layer and feeds its output to each and every neuron in the next layer. In other words, every possible connection between two neurons in two different layers is made. This architecture of neural network does not scale well for images. Firstly, the images would require a lot of manually engineered pre-processing if we want to obtain an accurate result. Secondly, the size of the input is very large.
The use of Convolutional Neural Networks for images reduces the complexity of the problem because they are constructed such that each neuron processes data from only a restricted area of the image known as the neuron's receptive field, and not the entire image. Using CNNs also greatly reduces the amount of manual preprocessing needed for the image. In fact, since we use trainable filters we can say that the CNN learns the preprocessing itself.
CNNs are classified as deep learning networks because they usually contain a large number of hidden layers.

Unicode:

Unicode is defined by Wikipedia as "a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems."
Unicode uses 16 bits (specifically UTF-16 uses 16 bits), and can hold 65,536 characters, which is way more than enough to represent characters in all of the world's living languages, as well as historic scripts such as Brahmi.
UTF-16 assigns each of its characters with a unique 16-bit identification number known as a code point, and leaves the rendering of the character to the software. The code points for Kannada characters are in the range of 0x0C82 to 0x0CF2. This range of code points is reserved exclusively for Kannada characters.



Kannada[1][2]
Official Unicode Consortium code chart (PDF)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0C8x	ಀ	ಁ	ಂ	ಃ		ಅ	ಆ	ಇ	ಈ	ಉ	ಊ	ಋ	ಌ		ಎ	ಏ
U+0C9x	ಐ		ಒ	ಓ	ಔ	ಕ	ಖ	ಗ	ಘ	ಙ	ಚ	ಛ	ಜ	ಝ	ಞ	ಟ
U+0CAx	ಠ	ಡ	ಢ	ಣ	ತ	ಥ	ದ	ಧ	ನ		ಪ	ಫ	ಬ	ಭ	ಮ	ಯ
U+0CBx	ರ	ಱ	ಲ	ಳ		ವ	ಶ	ಷ	ಸ	ಹ			಼	ಽ	ಾ	ಿ
U+0CCx	ೀ	ು	ೂ	ೃ	ೄ		ೆ	ೇ	ೈ		ೊ	ೋ	ೌ	್		
U+0CDx						ೕ	ೖ								ೞ	
U+0CEx	ೠ	ೡ	ೢ	ೣ			೦	೧	೨	೩	೪	೫	೬	೭	೮	೯
U+0CFx		ೱ	ೲ													

Figure 2.1: Unicodes for Kannada characters (Source: Wikipedia)

Applications of OCR:

- Data Entry for passports, invoices, and other business documents.
- Automatic Number Plate Recognition.
- Key information extraction from insurance documents.
- Making electronic images of printed documents searchable.
- Giving instructions to computers by writing.
- Assistive technology for the blind and visually impaired users.
- Convert printed text-books into editable e-Books easily.

III. LITERATURE SURVEY

A significant amount of research has been done on Kannada Character Recognition:

A method of identifying Kannada characters by feature extraction using Curvelet transform followed by a KNN classifier is described in [1].

Different methods of segmenting lines in a paragraph of Kannada text, such as HPP based segmentation, Morphology based segmentation and Bounding Box based segmentation, are discussed in [2].

Segmentation of overlapping lines of text in printed Indian scripts is discussed in [3]. The overlapping of lines of text causes a lot of difficulty in implementing OCR.

A method of identifying characters by zoning, followed by feature extraction by dividing segments into tracks and sectors (like in a compact disc), then an SVM classifier, is described in [4].

A number of methods for feature extraction for the purpose of character recognition are discussed in [5].

The paper [6] discusses and compares different methods of segmentation, feature extraction and classification with respect to the accuracy of the OCR obtained.

A method of identifying Kannada characters by zoning, feature extraction using Hu's moments, HPP and VPP, followed by a PNN classifier is described in [7].

Existing OCR Systems for the Kannada Language:
There are only a few notable OCR systems for the Kannada Language. KanScan [14] is an app that converts a Kannada text image into editable text format. It contains a lot of flaws and gives a lot of errors when it runs. Another system is i2OCR [15] that offers an accuracy of around 60% with a run-time of almost 1 whole minute for a 200-word article.

Hence, it can be seen that the need for a top-notch Kannada OCR system is essential for the masses. Use of CNN gets us the highest possible accuracy in this system.

IV. METHODOLOGY AND IMPLEMENTATION

A. System Architecture:

The sample image is the printed Kannada text given as the input to the system. The Pre-Processing block helps with the segmentation of words and characters. Size-based classification too occurs. 4 CNN's are trained with varied input sizes of the image. The dataset consists of twenty fonts overall. A UID table is then constructed to ensure the Unicode is matched. The output is then compared by the system with the UID table to get the editable text file.

B. Pre Processing

The pre-processing block consists of segmentation of characters. The algorithm goes as follows:

1. First trim out the extra image pixels around the word.
2. Next identify the position of the base line, which will be used later on in the process.
3. Then perform thresholding to differentiate foreground pixels and background pixels.
4. Next dilate the image just enough so that every character becomes a single connected component
5. Then label each of the connected components from left to right.
6. Using the labels as reference, crop out each character individually.
7. Identify each character as a regular character or vattakshara character using the position of the base line.
8. Finally feed the character to the appropriate set of CNNs for identification.

A demonstration of the segmentation is shown below:

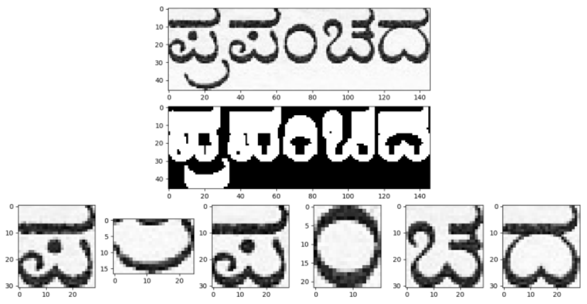


Figure 1. Sample of Segmentation

C. Baseline Identification

Baseline is the imaginary horizontal line on which the non-subscript text "rests" upon. To find the position of this line, the following method is used:

1. First apply thresholding to the input image of the word
2. This is followed by Sobel edge detection to get all the **lower edges** in the image.
3. Then take the HPP (Horizontal Point Projection) of the edge detection result.
4. When the HPP is plotted on a graph two maxima, upper and lower, are obtained. The position of the lower maxima is taken as the position of the base line.

This method is demonstrated for the image below. The base line has been identified to be at position 29 on the H (height) axis.

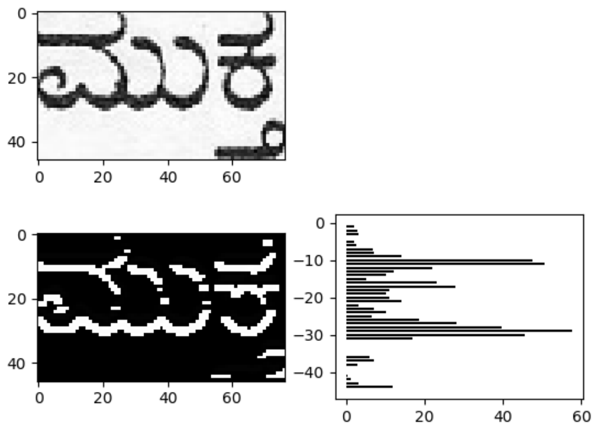
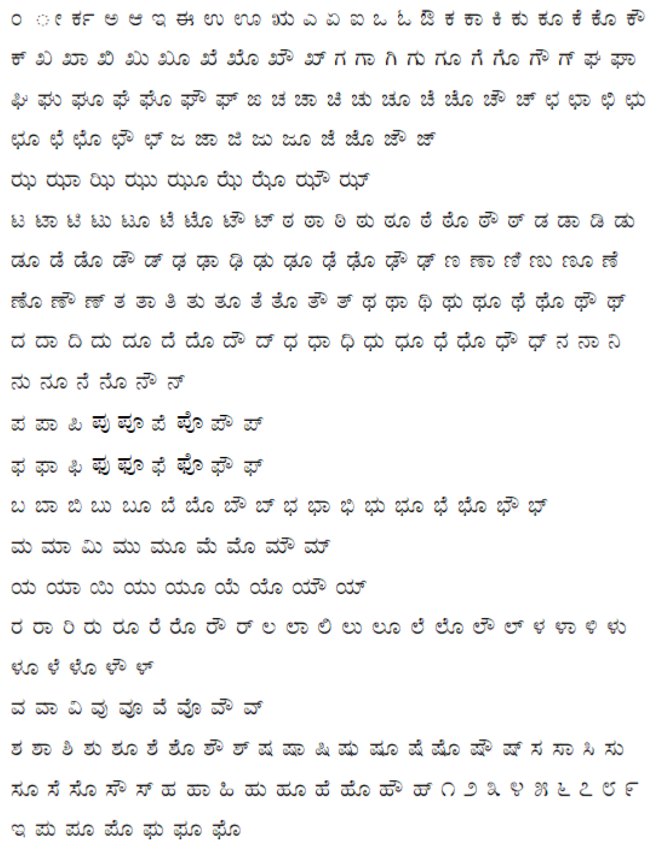


Figure 2. Base-line detection

This line is used to differentiate between regular characters and subscript characters.

D. Dataset

To create the dataset the first step is to list out all possible Kannada glyphs in a Microsoft Word document, as shown in figure 3.



ಂ ◌ೃ ರ್ಕ ಅ ಆ ಇ ಈ ಉ ಊ ಋ ಎ ಏ ಐ ಒ ಓ ಔ ಕ ಕಾ ಕಿ ಕು ಕೂ ಕೆ ಕೊ ಕೌ
ಕ್ ಖ ಖಾ ಖಿ ಖು ಖೂ ಖೆ ಖೊ ಖೌ ಖ್ ಗ ಗಾ ಗಿ ಗು ಗೂ ಗೆ ಗೊ ಗೌ ಗ್ ಘ ಘಾ
ಘಿ ಘು ಘೂ ಘೆ ಘೊ ಘೌ ಘ್ ಙ ಚ ಚಾ ಚಿ ಚು ಚೂ ಚೆ ಚೊ ಚೌ ಚ್ ಛ ಛಾ ಛಿ ಛು
ಛೂ ಛೆ ಛೊ ಛೌ ಛ್ ಜ ಜಾ ಜಿ ಜು ಜೂ ಜೆ ಜೊ ಜೌ ಜ್
ಝ ಝಾ ಝಿ ಝು ಝೂ ಝೆ ಝೊ ಝೌ ಝ್
ಟ ಟಾ ಟಿ ಟು ಟೂ ಟೆ ಟೊ ಟೌ ಟ್ ಠ ಠಾ ಠಿ ಠು ಠೂ ಠೆ ಠೊ ಠೌ ಠ್ ಡ ಡಾ ಡಿ ಡು
ಡೂ ಡೆ ಡೊ ಡೌ ಡ್ ಢ ಢಾ ಢಿ ಢು ಢೂ ಢೆ ಢೊ ಢೌ ಢ್ ಣ ಣಾ ಣಿ ಣು ಣೂ ಣೆ
ಣೊ ಣೌ ಣ್ ತ ತಾ ತಿ ತು ತೂ ತೆ ತೊ ತೌ ತ್ ಥ ಥಾ ಥಿ ಥು ಥೂ ಥೆ ಥೊ ಥೌ ಥ್
ದ ದಾ ದಿ ದು ದೂ ದೆ ದೊ ದೌ ದ್ ಧ ಧಾ ಧಿ ಧು ಧೂ ಧೆ ಧೊ ಧೌ ಧ್ ನ ನಾ ನಿ
ನು ನೂ ನೆ ನೊ ನೌ ನ್
ಪ ಪಾ ಪಿ ಪು ಪೂ ಪೆ ಪೊ ಪೌ ಪ್
ಫ ಫಾ ಫಿ ಫು ಫೂ ಫೆ ಫೊ ಫೌ ಫ್
ಬ ಬಾ ಬಿ ಬು ಬೂ ಬೆ ಬೊ ಬೌ ಬ್ ಭ ಭಾ ಭಿ ಭು ಭೂ ಭೆ ಭೊ ಭೌ ಭ್
ಮ ಮಾ ಮಿ ಮು ಮೂ ಮೆ ಮೊ ಮೌ ಮ್
ಯ ಯಾ ಯಿ ಯು ಯೂ ಯೆ ಯೊ ಯೌ ಯ್
ರ ರಾ ರಿ ರು ರೂ ರೆ ರೊ ರೌ ರ್ ಲ ಲಾ ಲಿ ಲು ಲೂ ಲೆ ಲೊ ಲೌ ಲ್ ಳ ಳಾ ಳಿ ಳು
ಳೂ ಳೆ ಳೊ ಳೌ ಳ್
ವ ವಾ ವಿ ವು ವೂ ವೆ ವೊ ವೌ ವ್
ಶ ಶಾ ಶಿ ಶು ಶೂ ಶೆ ಶೊ ಶೌ ಶ್ ಷ ಷಾ ಷಿ ಷು ಷೂ ಷೆ ಷೊ ಷೌ ಷ್ ಸ ಸಾ ಸಿ ಸು
ಸೂ ಸೆ ಸೊ ಸೌ ಸ್ ಹ ಹಾ ಹಿ ಹು ಹೂ ಹೆ ಹೊ ಹೌ ಹ್ ೧ ೨ ೩ ೪ ೫ ೬ ೭ ೮ ೯
ಇ ಪು ಪೂ ಪೊ ಫು ಫೂ ಫೊ

Figure 3. Dataset

This is done for 10 Kannada fonts, in normal text and in bold text for each font. The fonts used are given below:

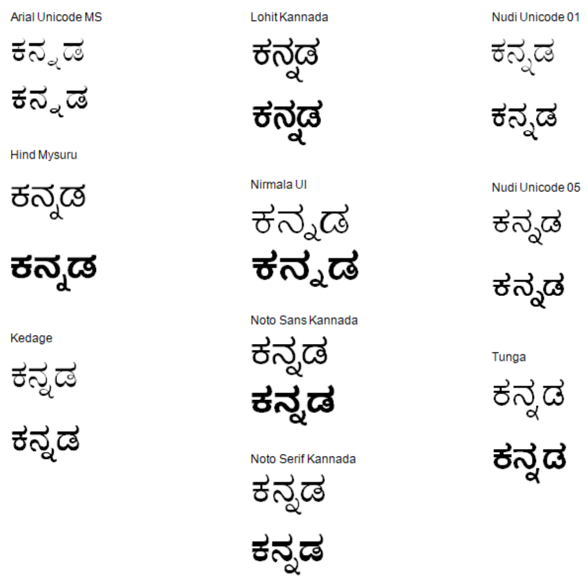


Figure 4. Characters depicting different fonts

A CNN takes a constant input size, however the sizes and aspect ratios of the characters are variable. All characters must be resized to the constant input size before being input to the CNN. But using a single CNN for all the characters ignoring the difference in aspect ratios would give very inaccurate results. The solution is to divide the dataset into 4 classes based on the aspect ratios, then use these datasets to train 4 CNNs of different input sizes. The sizes used are 15x20, 25x20, 30x20 and 40x20. In this way the original aspect ratio of the characters is roughly preserved so they can be identified properly.

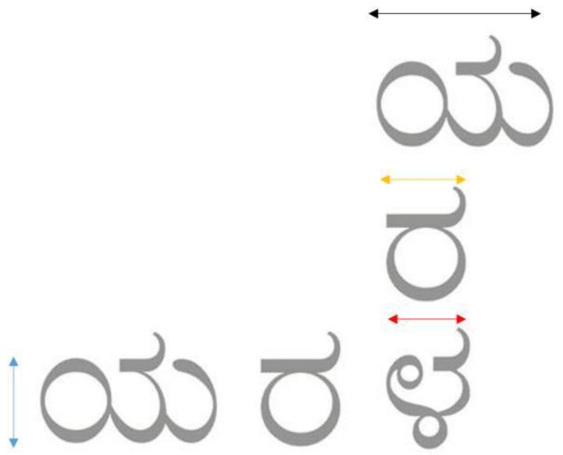


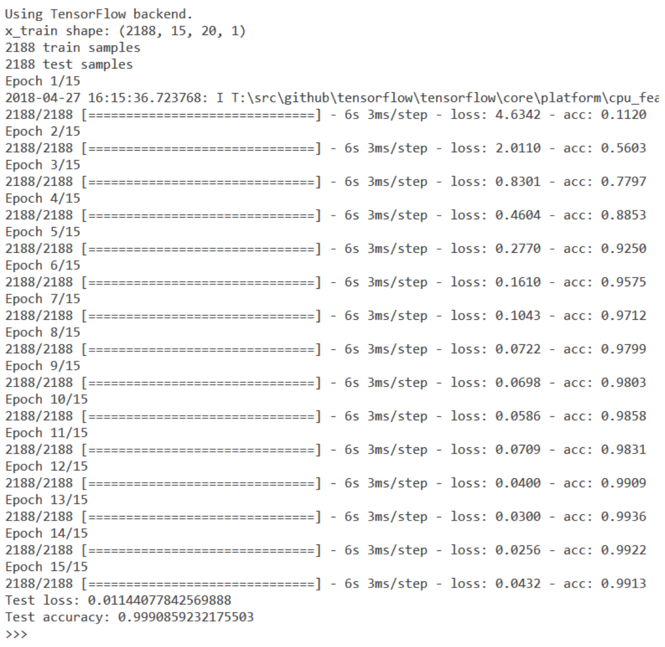
Figure 5. Characters with varying aspect ratios

E. Design and training of CNNs

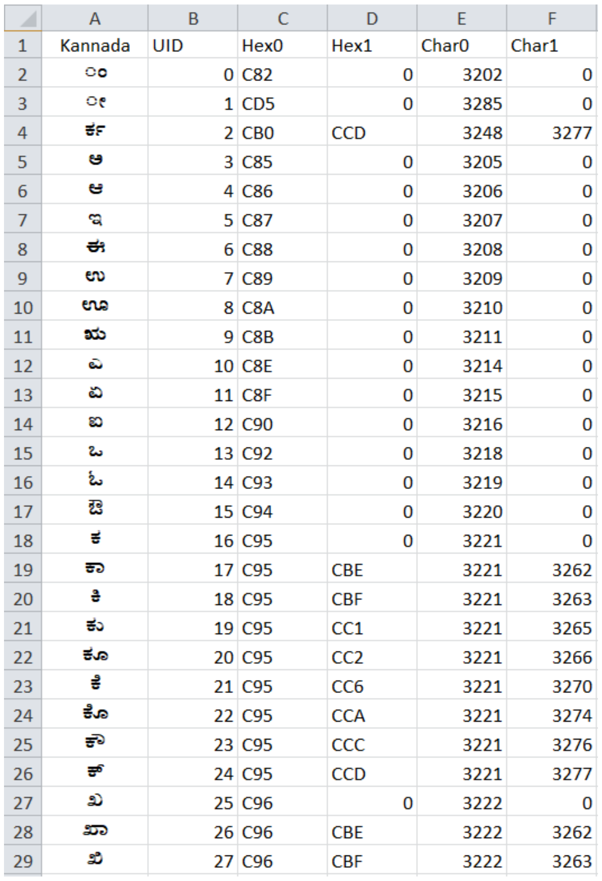
Keras and TensorFlow were used for implementing the CNNs, and OpenCV was used for all image processing operations. The project was implemented in Python.

The CNN implemented is a Keras Sequential model beginning with one convolutional layer of kernel size 5x5 with 64 channels, followed by a ReLU activation function layer, followed by a 2x2 max-pooling layer. Next all values are flattened into a single dimension. This becomes an input for a fully connected layer of 1000 hidden neurons with ReLU activation function and finally a fully connected output layer of 340 neurons with softmax activation function.

The loss function used for training is categorical cross-entropy and the optimization algorithm used is the Adam optimizer, explained in detail in the paper *Adam: A Method for Stochastic Optimization*. [12][13]


```
Using TensorFlow backend.
x_train shape: (2188, 15, 20, 1)
2188 train samples
2188 test samples
Epoch 1/15
2018-04-27 16:15:36.723768: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_fea
2188/2188 [==============================] - 6s 3ms/step - loss: 4.6342 - acc: 0.1120
Epoch 2/15
2188/2188 [==============================] - 6s 3ms/step - loss: 2.0110 - acc: 0.5603
Epoch 3/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.8301 - acc: 0.7797
Epoch 4/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.4604 - acc: 0.8853
Epoch 5/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.2770 - acc: 0.9250
Epoch 6/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.1610 - acc: 0.9575
Epoch 7/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.1043 - acc: 0.9712
Epoch 8/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0722 - acc: 0.9799
Epoch 9/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0698 - acc: 0.9803
Epoch 10/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0586 - acc: 0.9858
Epoch 11/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0709 - acc: 0.9831
Epoch 12/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0400 - acc: 0.9909
Epoch 13/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0300 - acc: 0.9936
Epoch 14/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0256 - acc: 0.9922
Epoch 15/15
2188/2188 [==============================] - 6s 3ms/step - loss: 0.0432 - acc: 0.9913
Test loss: 0.01144077842569888
Test accuracy: 0.9990859232175503
>>>
```
Figure 6. Demonstration of CNN training

After using the CNN to identify the character and obtaining its UID the UID is queried into a Lookup Table to obtain the Unicode encoding of the respective character for printing.



	A	B	C	D	E	F
1	Kannada	UID	Hex0	Hex1	Char0	Char1
2	◌ಂ	0	C82	0	3202	0
3	◌ೕ	1	CD5	0	3285	0
4	ರ್	2	CB0	CCD	3248	3277
5	ಅ	3	C85	0	3205	0
6	ಆ	4	C86	0	3206	0
7	ಇ	5	C87	0	3207	0
8	ಈ	6	C88	0	3208	0
9	ಉ	7	C89	0	3209	0
10	ಊ	8	C8A	0	3210	0
11	ಋ	9	C8B	0	3211	0
12	ಎ	10	C8E	0	3214	0
13	ಏ	11	C8F	0	3215	0
14	ಐ	12	C90	0	3216	0
15	ಒ	13	C92	0	3218	0
16	ಓ	14	C93	0	3219	0
17	ಔ	15	C94	0	3220	0
18	ಕ	16	C95	0	3221	0
19	ಕಾ	17	C95	CBE	3221	3262
20	ಕಿ	18	C95	CBF	3221	3263
21	ಕು	19	C95	CC1	3221	3265
22	ಕೂ	20	C95	CC2	3221	3266
23	ಕೆ	21	C95	CC6	3221	3270
24	ಕೊ	22	C95	CCA	3221	3274
25	ಕೌ	23	C95	CCC	3221	3276
26	ಕ್	24	C95	CCD	3221	3277
27	ಖ	25	C96	0	3222	0
28	ಖಾ	26	C96	CBE	3222	3262
29	ಖಿ	27	C96	CBF	3222	3263

Figure 7. Lookup Table

F. Vattakshara characters

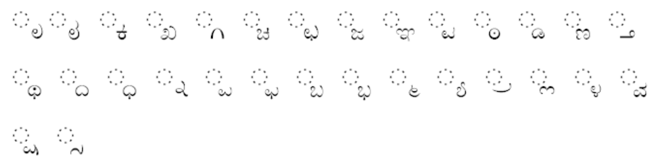
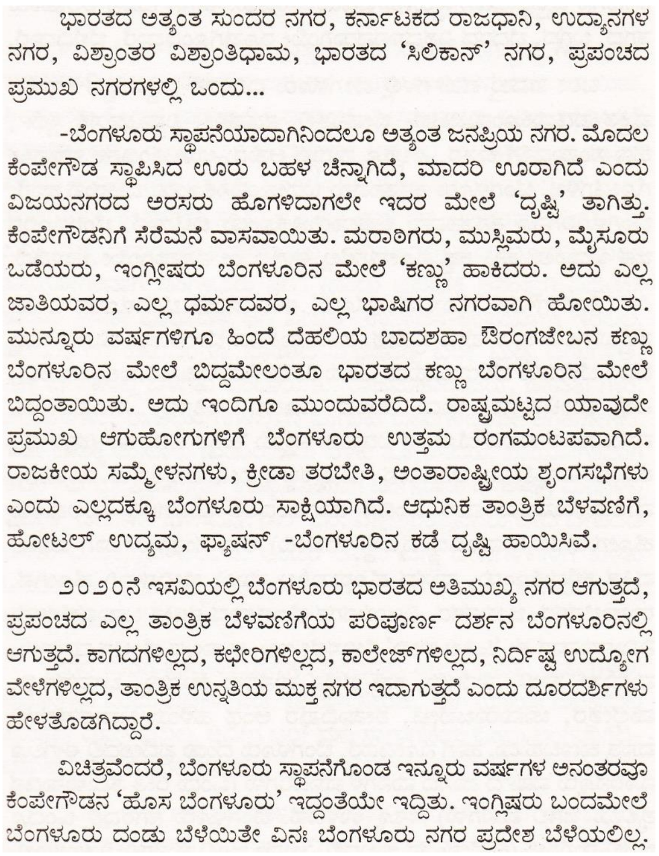


Figure 8. Vattaksharas

The vattakshara i.e. subscript characters are excluded from the main dataset, because most of them look very similar to their regular character counterparts and therefore we do not want to confuse the CNNs and hence reduce the accuracy of the system.
Instead a separate set of CNNs and a separate Unicode lookup table is used just for the identification of vattakshara characters. The entire process of creating the dataset, training the CNNs and constructing the lookup table is repeated. In this case, we use only 3 CNNs, with input sizes 15x15, 20x15 and 30x15. A character is identified as a regular character or vattakshara character based on the position of the character with respect to the base line.

V. RESULTS AND ANALYSIS

A sample image containing Kannada text was taken from an e-paper for checking the accuracy of the system.



ಭಾರತದ ಅತ್ಯಂತ ಸುಂದರ ನಗರ, ಕರ್ನಾಟಕದ ರಾಜಧಾನಿ, ಉದ್ಯಾನಗಳ ನಗರ, ವಿಶ್ರಾಂತರ ವಿಶ್ರಾಂತಿಧಾಮ, ಭಾರತದ 'ಸಿಲಿಕಾನ್' ನಗರ, ಪ್ರಪಂಚದ ಪ್ರಮುಖ ನಗರಗಳಲ್ಲಿ ಒಂದು...

-ಬೆಂಗಳೂರು ಸ್ಥಾಪನೆಯಾದಾಗಿನಿಂದಲೂ ಅತ್ಯಂತ ಜನಪ್ರಿಯ ನಗರ. ಮೊದಲ ಕೆಂಪೇಗೌಡ ಸ್ಥಾಪಿಸಿದ ಊರು ಬಹಳ ಚೆನ್ನಾಗಿದೆ, ಮಾದರಿ ಊರಾಗಿದೆ ಎಂದು ವಿಜಯನಗರದ ಅರಸರು ಹೊಗಳಿದಾಗಲೇ ಇದರ ಮೇಲೆ 'ದೃಷ್ಟಿ' ತಾಗಿತ್ತು. ಕೆಂಪೇಗೌಡನಿಗೆ ಸೆರೆಮನೆ ವಾಸವಾಯಿತು. ಮರಾಠಿಗರು, ಮುಸ್ಲಿಮರು, ಮೈಸೂರು ಒಡೆಯರು, ಇಂಗ್ಲೀಷರು ಬೆಂಗಳೂರಿನ ಮೇಲೆ 'ಕಣ್ಣು' ಹಾಕಿದರು. ಅದು ಎಲ್ಲ ಜಾತಿಯವರ, ಎಲ್ಲ ಧರ್ಮದವರ, ಎಲ್ಲ ಭಾಷಿಗರ ನಗರವಾಗಿ ಹೋಯಿತು. ಮುನ್ನೂರು ವರ್ಷಗಳಿಗೂ ಹಿಂದೆ ದೆಹಲಿಯ ಬಾದಶಹಾ ಔರಂಗಜೇಬನ ಕಣ್ಣು ಬೆಂಗಳೂರಿನ ಮೇಲೆ ಬಿದ್ದಮೇಲಂತೂ ಭಾರತದ ಕಣ್ಣು ಬೆಂಗಳೂರಿನ ಮೇಲೆ ಬಿದ್ದಂತಾಯಿತು. ಅದು ಇಂದಿಗೂ ಮುಂದುವರೆದಿದೆ. ರಾಷ್ಟ್ರಮಟ್ಟದ ಯಾವುದೇ ಪ್ರಮುಖ ಆಗುಹೋಗುಗಳಿಗೆ ಬೆಂಗಳೂರು ಉತ್ತಮ ರಂಗಮಂಟಪವಾಗಿದೆ. ರಾಜಕೀಯ ಸಮ್ಮೇಳನಗಳು, ಕ್ರೀಡಾ ತರಬೇತಿ, ಅಂತಾರಾಷ್ಟ್ರೀಯ ಶೃಂಗಸಭೆಗಳು ಎಂದು ಎಲ್ಲದಕ್ಕೂ ಬೆಂಗಳೂರು ಸಾಕ್ಷಿಯಾಗಿದೆ. ಆಧುನಿಕ ತಾಂತ್ರಿಕ ಬೆಳವಣಿಗೆ, ಹೋಟಲ್ ಉದ್ಯಮ, ಫ್ಯಾಷನ್ -ಬೆಂಗಳೂರಿನ ಕಡೆ ದೃಷ್ಟಿ ಹಾಯಿಸಿವೆ.

೨೦೨೦ನೆ ಇಸವಿಯಲ್ಲಿ ಬೆಂಗಳೂರು ಭಾರತದ ಅತಿಮುಖ್ಯ ನಗರ ಆಗುತ್ತದೆ, ಪ್ರಪಂಚದ ಎಲ್ಲ ತಾಂತ್ರಿಕ ಬೆಳವಣಿಗೆಯ ಪರಿಪೂರ್ಣ ದರ್ಶನ ಬೆಂಗಳೂರಿನಲ್ಲಿ ಆಗುತ್ತದೆ. ಕಾಗದಗಳಿಲ್ಲದ, ಕಛೇರಿಗಳಿಲ್ಲದ, ಕಾಲೇಜ್‌ಗಳಿಲ್ಲದ, ನಿರ್ದಿಷ್ಟ ಉದ್ಯೋಗ ವೇಳೆಗಳಿಲ್ಲದ, ತಾಂತ್ರಿಕ ಉನ್ನತಿಯ ಮುಕ್ತ ನಗರ ಇದಾಗುತ್ತದೆ ಎಂದು ದೂರದರ್ಶಿಗಳು ಹೇಳತೊಡಗಿದ್ದಾರೆ.

ವಿಚಿತ್ರವೆಂದರೆ, ಬೆಂಗಳೂರು ಸ್ಥಾಪನೆಗೊಂಡ ಇನ್ನೂರು ವರ್ಷಗಳ ಅನಂತರವೂ ಕೆಂಪೇಗೌಡನ 'ಹೊಸ ಬೆಂಗಳೂರು' ಇದ್ದಂತೆಯೇ ಇದ್ದಿತು. ಇಂಗ್ಲಿಷರು ಬಂದಮೇಲೆ ಬೆಂಗಳೂರು ದಂಡು ಬೆಳೆಯಿತೇ ವಿನಃ ಬೆಂಗಳೂರು ನಗರ ಪ್ರದೇಶ ಬೆಳೆಯಲಿಲ್ಲ.

The Editable Output Text is as shown below. The errors have been highlighted manually.

KanOCR Result:

ಭಾರತದ ಅತ್ಯಂತ ಸುಂದರ ನಗರ , ಕರ್ನಾಟಕದ ರಾಜಧಾನಿ, ಉದ್ಯಾನಗಳ ನಗರ, ವಿಶ್ರಾಂತರ ವಿಶ್ರಾಂತಿಧಾಮ, ಭಾರತದ ಳಸಿಲಿಕಾನ್, ನಗರ, ಪ್ರಪಂಚದ ಪ್ರಮುಖ ನಗರಗಳಲ್ಲಿ ಒಂದು.

ಬೆಂಗಳೂರು ಸ್ಥಾಪನೆಯಾದಾಗಿನಿಂದಲೂ ಅತ್ಯಂತ ಜನಪ್ರಿಯ ನಗರ. ಮೊದಲ ಕೆಂಪೇಗೌಡ ಸ್ಥಾಪಿಸಿದ ಊರು ಬಹಳ ಚೆನ್ನಾಗಿದೆ, ಮಾದರಿ ಊರಾಗಿದೆ ಎಂದು ವಿಜಯನಗರದ ಅರಸರು ಹೊಗಳಿದಾಗಲೇ ಇದರ ಮೇಲೆ **ದಷ್ಟಿ** , ತಾಗಿತ್ತು. ಒಡೆಯರು, ಇಂಗ್ಲೀಷರು ಬೆಂಗಳೂರಿನ ಮೇಲೆ **ಹೆಕಣ್ಣು**, ಹಾಕಿದರು. ಅದು ಎಲ್ಲ ಜಾತಿಯವರ, ಎಲ್ಲ ಧರ್ಮದವರ, ಎಲ್ಲ ಭಾಷಿಗರ ನಗರವಾಗಿ ಹೋಯಿತು. **ಮುವ್ನೂರು** ವರ್ಷಗಳಿಗೂ ಹಿಂದೆ ದೆಹಲಿಯ ಬಾದಶಹಾ ಔರಂಗಜೇಬನ ಕಣ್ಣು ಬೆಂಗಳೂರಿನ ಮೇಲೆ ಬಿದ್ವಮೇಲಂತೂ ಭಾರತದ ಕಣ್ಣು ಬೆಂಗಳೂರಿನ ಮೇಲೆ ಬಿದ್ವಂತಾಯಿತು ಅದು ಇಂದಿಗೂ ಮುಂದುವರೆದಿದೆ **ರಾಷ್ವಮಟ್ವದ** ಯಾವುದೇ ಪ್ರಮುಖ ಆಗುಹೋಗುಗಳಿಗೆ ಬೆಂಗಳೂರು ಉತ್ತಮ ರಂಗಮಂಟಪವಾಗಿದೆ ರಾಜಕೀಯ ಸಮ್ಮೇಳನಗಳು, ಕ್ರೀಡಾ ತರಬೇತಿ, ಅಂತಾರಾಷ್ಟ್ರೀಯ ಶೃಂಗಸಭೆಗಳು ಎಂದು ಎಲ್ಲದಕ್ಕೂ ಬೆಂಗಳೂರು ಸಾಕ್ಷಿಯಾಗಿದೆ ಆಧುನಿಕ ತಾಂತ್ರಿಕ ಬೆಳವಣಿಗೆ, ಹೋಟಲ್ ಉದ್ಯಮ, **ಪ್ಯಷನ್** -ಬೆಂಗಳೂರಿನ ಕಡೆ ದೃಷ್ಟಿ ಹಾಯಿಸಿವೆ
೨◌೦ ೨◌೦ನೆ ಇಸವಿಯಲ್ಲಿ ಬೆಂಗಳೂರು ಭಾರತದ ಅತಿಮುಖ್ಯ ನಗರ ಆಗುತ್ತದೆ, **ಪಪಂದ** ಎಲ್ಲ ತಾಂತ್ರಿಕ ಬೆಳವಣಿಗೆಯ ಪರಿಪೂರ್ಣ ದರ್ಶನ ಬೆಂಗಳೂರಿನಲ್ಲಿ ಆಗುತ್ತದೆ ಕಾಗದಗಳಿಲ್ಲದ, ಕಛೇರಿಗಳಿಲ್ಲದ, ಕಾಲೇಜ್ಗಳಿಲ್ಲದ, **ನಿರ್ದಿ** ಉದ್ಯೋಗ ವೇಳೆಗಳಿಲ್ಲದ, ತಾಂತ್ರಿಕ ಉನ್ನತಿಯ ಮುಕ್ತ ನಗರ ಇದಾಗುತ್ತದೆ ಎಂದು ದೂರದರ್ಶಿಗಳು ಹೇಳತೊಡಗಿದಾರೆ

ವಿಚಿತವೆಂದರೆ, ಬೆಂಗಳೂರು ಸ್ಥಾಪನೆಗೊಂಡ ಇನ್ನೂರು ವರ್ಷಗಳ ಅನಂತರವೂ ಕೆಂಪೇಗೌಡನ **ಳಹೊಸ** ಬೆಂಗಳೂರು, ಇದ್ವಂತೆಯೇ ಇದ್ವಿತು ಇಂಗ್ಲಿಷರು ಬಂದಮೇಲೆ ಕೆಂಪೇಗೌಡನಿಗೆ ಸೆರೆಮನೆ ವಾಸವಾಯಿತು. ಮರಾಠಿಗರು, ಮುಸ್ಲಿಮರು, ಮೈಸೂರು **◌ಂಗಳೂರು** ದಂಡು ಬೆಳೆಯಿತೇ ವಿನಃ ಬೆಂಗಳೂರು ನಗರ ಪ್ರದೇಶ ಬೆಳೆಯಲಿಲ್ಲ

OCR Result on i2OCR for same sample:

ಭಾರತದ ಅತ್ಯಂತ **ಸುಲದರ** ನಗರ, **ಕನಾಣುಕದ** ರಾಜಧಾನಿ, **ಉಧ್ಯಾನಗಳ** ನಗರ, **ಏತ್ರಾಲತರ ಏಶ್ರಾರಿತಿಧಾಮ**, ಭಾರತದ "ಸಿಲಿಕಾನ್ ನಗರ, **ಪ್ತಪಲಚದ** ಪ್ತಮುಖ ನಗರಗಳಲ್ಲಿ ಒಂದು...

=ಬೆಂಗಳೂರು ಸ್ಥಾಪನೆಯಾದಾಗಿನಿಂದಲೂ **ಅತ್ಯೆರಿತ ಜನಪ್ತಿಯ** ನಗರ. ಮೊದಲ **ಕೆಂವೇಗೌಡ** ಸ್ಥಾಪಿಸಿದ ಊರು ಬಹಳ ಚೆನ್ನಾಗಿದೆ, ಮಾದರಿ **ಲೂರಾಗಿದೆ ಎರಿದು** ವಿಜಯನಗರದ ಅರಸರು **ಡೂಗಳಿದಾಗಲೆಆ** ಇದರ **ಮೆಆಲೆ** "**ದ್ನಷ್ಟ** ತಾಗಿತ್ತು. **ಕೆರಿಪೆಆಗೌಡನಿಗೆ** ಸೆರೆಮನೆ ವಾಸವಾಯಿತು. ಮರಾಠಿಗರು, ಮುಸ್ಲಿಮರು, ಮೈಸೂರು ಒಡೆಯರು, **ಇಲಗ್ಲೀಷರು ಬೆರಿಗಳವಿರಿನ ಮೆಅಲೆ** "**ಕಣ್ಣಾ** ಹಾಕಿದರು. ಅದು **ಎಲಸ್ವೂ** ಜಾತಿಯೆವರ, ಎಲ್ಲ ಧರ್ಮದವರ, ಎಲ್ಲ ಭಾಷಿಗರ ನಗರವಾಗಿ **ಹೊಅಯಿತು**. ಮುನ್ನೂರು ವರ್ಷಗಳಿಗೂ **ಹಿಲದೆ** ದೆಹಲಿಯ ಬಾದಶಹಾ ಔರಂಗಜೇಬನ ಕಣ್ಣು **ಬೆಲಗಳುಶಿರಿನ ಮೆಆಲೆ ಬಿದ್ವಮೆಆಲಂತೂ** ಭಾರತದ ಕಣ್ಣು **ಬೆರಿಗಳವಿರಿನ ಮೆಆಲೆ ಬಿದ್ವಲತಾಯಿತು**. ಅದು **ಇರಿದಿಗೊ ವಬಂದುವರೆದಿದೆ**. ರಾಷ್ಟ್ರಮಟ್ಟದ ಯಾವುದೇ ಪ್ತಮುಖ **ಆಗುಹುಠಿರಿಗುಗಳಿಗೆ** ಬೆಂಗಳೂರು ಉತ್ತಮ ರಂಗಮಂಟಪವಾಗಿದೆ.
ರಾಜಕೀಯ **ಸಮೆಗ್ರ್ಯಳನಗಳಎ**, ಕ್ರೀಡಾ ತರಬೇತಿ, **ಅರಿತಾರಾಷ್ಟಿಔಯೆ ಸ್ಪಂಗಸಭೆಗಳಎ ಎಲದು** ಎಲ್ಲದಕ್ಕೂ **ಬೆಲಗಳಣುರು** ಸಾಕ್ಷಿಯಾಗಿದೆ. ಆಧುನಿಕ **ತಾರಿತ್ತಿಕ** ಬೆಳವಣಿಗೆ, **ಹೊಳಟಲ್** ಉದ್ಯಮ, **ಫಾಶೆಷನ್** =ಬೆಂಗಳೂರಿನ ಕಡೆ **ವೃಷ್ಟಿ** ಹಾಯಿಸಿವೆ.

೨೦೨೦ನೆ **ಇಸಏಯಲ್ಲಿ** 'ಬೆಂಗಳೂರು ಭಾರತದ ಅತಿಮುಖ್ಯ ನಗರ ಆಗುತ್ತೆದೆ, **ಪ್ತಪರಿಚದ** ಎಲ್ಲ **ತಾಲತ್ತಿಕ** ಬೆಳವಣಿಗೆಯ **ಪರಿಪುರ್ನಿರ್ನಿ** ದರ್ಶನ **ದೆರಿಗಳಪುರಿನಲ್ಲಿ** ಆಗುತ್ತದೆ. ಕಾಗದಗಳಿಲ್ಲದ, ಕಛೇರಿಗಳಿಲ್ಲದ,

ಕಾಲೆಳಜ್ಞಳಿಲ್ಲದ, **ನಿರ್ದೀಷ್ಣ ಉದೊಶೆಳಗ ವೇಳೆಗಳಿಲ್ಲದ**, **ತಾರಿತ್ತಿಕ** ಉನ್ನತಿಯ ಮುಕ್ತ ನಗರ **ಇದಾಗುತ್ತಂ ಎಲದು** ದೂರದರ್ಶಿಗಳು **ಹೆಆಳೆತೊಡಗಿದ್ದಾರೆ**.

ಏಚಿತ್ತವೆಂದರೆ, **ಬೆಂಗಳೂರು ಸ್ಥಾಪನೆಗೆನಿಂಡ** ಇನ್ನೂರು ವರ್ಷಗಳ **ಅನಲತರವೊ** ಕೆಂಪೇಗೌಡನ "ಹೊಸ **ಬೆಲಗಳೊರು' ಇದ್ದಂತೆಯಆ** ಇದ್ದಿತು. ಇಂಗ್ಲಿಷರು **ಬಲದಮೆಆಲೆ** ಬೆಂಗಳೂರು ದಂಡು ಬೆಳೆಯಿತೇ **ಎನ**: **ಬೆಲಗಳುಎರು** ನಗರ ಪ್ರದೇಶ ಬೆಳೆಯಲಿಲ್ಲ.

The words were classified as Easy, Medium, and Hard.
Easy words are simple words consisting of only aksharas, without kaagunithas or vattaksharas (like ನಗರ)
Medium words include Kaagunithas but not vattaksharas. (like ರಾಜಧಾನಿ).
The hard words consisted of everything, aksharas, kagunithas and vattaksharas (like ವಿಶ್ರಾಂತಿಧಾಮ)

The accuracy was then calculated for each category, and then as a whole:

Table 1. Accuracy for each category of words

Category	No. of Words	Incorrect	Accuracy
Easy	17	1	94.11 %
Medium	64	2	96.87 %
Hard	44	6	86.36 %
-	-	Overall	*92.45 %*

The same sample was used to check the accuracy of some existing systems. The results and comparison are as shown below:

Table 2. Comparison of accuracy of different systems

System	Accuracy
Mobile App – *KanScan*	**-no clear output-**
Website – i2OCR	**60 % approx.**
KanOCR	**92.45 %**

Three other random samples from the internet were used to further compare the accuracy of the system. The results are provided below:

Table 3. Comparison of accuracy of different systems for different samples

Sample Number	Number of Words	i2OCR Accuracy	KanOCR Accuracy
1	23	14/23 = **60.9%**	23/23 = **100%**
2	38	18/38 = **47.36%**	37/38 = **97.36%**
3	55	35/55 = **63.63%**	53/55 = **96.36%**

Overall Accuracy	**57.3 %**	**97.9%**

There is an increase in accuracy of around **40** percent from the already existing popular OCR system to our Kannada OCR System.

Accuracy for 33 other samples was carried out. The samples were taken from the internet and they had different sizes and fonts. The accuracy for all the samples was found out to be **82.01** percent.

Table 4: Accuracy calculation for each sample

Sl. No	Word count	Accuracy
1.	14/18	77.77
2.	17/19	89.5
3.	18/19	94.75
4.	20/22	90.1
5.	17/31	55
6.	210/250	84
7.	257/304	84.5
8.	248/296	83.7
9.	274/326	84.04
10.	175/236	74.1
11.	78/92	84.8
12.	218/236	92.3
13.	288/316	91.1
14.	270/312	86.5
15.	30/38	78.9
16.	197/248	79.43
17.	226/279	81
18.	212/251	84.5
19.	267/306	87.25
20.	133/173	76.87
21.	214/271	79
22.	112/144	77.77
23.	198/240	82.5

24.	246/270	91.1
25.	177/215	82.3
26.	213/266	80
27.	199/275	72.4
28.	134/172	78
29.	64/77	83.11
30.	58/70	82.8
31.	119/154	77.72
32.	188/229	82.1
33.	102/130	78.5

VI. CONCLUSION AND FUTURE WORK

Using Convolutional Neural Networks in OCR systems is found to be a very reliable method of converting an image to a text document. We were able to construct an OCR system for the Kannada language with an accuracy of over 80 %. The runtime for an article of a minimum of 100 words was found to be less than two seconds, which is majorly lesser than the existing systems available in the market.

However the system has the following drawbacks:

1. It is designed to work with printed documents, it cannot identify handwritten characters.
2. The characters must be crisp and at least 20 pixels in height otherwise the system does not work properly.
3. The system can be calibrated only for one Kannada font and size at a time, if a single sample contains text in multiple fonts or sizes the system cannot correctly identify all of the text in one run.
4. It has a lot of trouble recognizing punctuation.
5. Certain characters such as double vattaksharas have been excluded from the dataset entirely because of difficulties.
6. The accuracy can be improved by reworking algorithms or using a dictionary to constrain the output of the system

All of these challenges can be addressed in future work.

REFERENCES

[1] HR Mamatha, S Sucharitha, Srikanta Murthy, “Multi-font and Multi-size Kannada Character Recognition based on the Curvelets and Standard Deviation”, International Journal of Computer Applications, Foundation of Computer Science, New York, USA, 2011.

[2] R Prajna, VR Ramya, HR Mamatha “A study of different text line extraction techniques for multi-font and multi-size printed kannada documents”, International Journal of Computer Applications, Foundation of Computer Science, 2015.

[3] M.K Jindal, R. K. Sharma & G.S. Lehal, "Segmentation of Horizontally Overlapping Lines in Printed Indian Scripts", International Journal of Computational Intelligence Research. ISSN 0973-1873 Vol.3, No.4 (2007), pp. 277–286

[4] Ashwin T.V and P.S Sastry, “A font and size independent OCR system for printed Kannada using SVM”, Sadhana, vol. 27, Part 1, February 2002, pp. 35–58.

[5] Anil. K. Jain, “Feature Extraction methods for Character Recognition – A survey”

[6] K. Indira, S. Sethu Selvi, “Kannada Character Recognition System: A Review”

[7] Netravati Belagali, Shanmukhappa A. Angadi, “OCR for Handwritten Kannada Language Script”

[8] http://cs231n.github.io/convolutional-networks/

[9] https://en.wikipedia.org/wiki/Convolutional_neural_network

[10] http://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/

[11] https://en.wikipedia.org/wiki/Kannada_alphabet

[12] https://keras.io/losses

[13] https://keras.io/optimizers

[14]https://play.google.com/store/apps/details?id=com.kaleidosoftware.kanscan.free

[15]www.i2ocr.com/free-online-kannada-ocr