ISSN: 0974-5823 Vol. 7 No. 1 January, 2022

International Journal of Mechanical Engineering

Information Retrieval and Linguistic Analysis on Cursive Characters

Dr. A.V.Sriharsha

Professor, CSE

Sree Vidyanikethan Engineering College (Autonomous)m Tirupati, India

Abstract

Optical Character Recognition (OCR) has many promising applications in the current world. Various types of character recognition problems can be solved with considerate efforts; however building an efficient OCR application is a challenging task. Many frameworks of CNN works on character recognition involved with slicing, sampling and image normalization which are part of activities in image preprocessing. Algorithms of Image Processing, Image Recognition combined with CNN and OCR API could converge and develop an efficient application for recognizing characters with cursive styles. The primary research of the task is applying adaptive classification methods to classify and rank the cursives of the characters, combine with a language recognizer and interpret the characters as in their original forms. Tesseract is an effective OCR library contains rich API that could be incorporated into ANN and CNN in development of OCR frameworks.

Introduction

OCR is best applied for English like languages than languages that have characters with cursive syllables and notes. Often, characters are consonants, vowels and compound characters which are combination of vowel and consonants. In the context of languages with cursive characters application of OCR is cumbersome, when compared with English like languages. Many recent works are done with development of OCR systems available for the languages with cursive characters [4]. Characters of Arabic, Urdu, Telugu, Tamil, Devanagari, Bengali, Odiya and Kannada belonging to Indian sub-continent are amongst the languages with cursive character set. An OCR engine, that can eliminate the constraints on input formats, quality of real world documents need to be developed for a very common problem of character recognition with natural language parser[2]. While digitizing the old documents of languages with cursive characters, one must consider the noise in the graphical formats of scanned sources.

Pepper noise exists in the scanned sources of images with cursive language text, where it has be removed for escaping mis-classification of characters. Error margin methods are very sensitive for extra noise as the image consists of cursive language text in different forms and (fonts) handwriting styles [3]. The text in cursive languages has characters connected into words, words connected with spaces or special characters as into sentences. Extra noise in image is a noise that can be located at random locations that are distributed throughout the text, which often influences the connectedness in text, thus 'violating chain code' and dimensions of characters [3][4]. While parsing the text content in the images scanned containing the cursive languages characters, as aspect called "character image variation" is observed which is always symmetric and not random, such images are divided into segments.

An elementary approach in preprocessing the corpus of the images is classification. A k-NN classification [8][9][10][13] is employed best for the k-best images of the same letter and implemented in a convolutional neural network. Further based on the features of the characters like thrust or thickness, curvy nature, orientation of character in the images are optimally selected using a Harmony-Search algorithm [6][7][11][12].

Problem Statement

Consider the set of images that contains cursive language text, the goal of the problem is to find the set of output variables from the set of input images with an accuracy as high as possible. Employ a 'recognize and search method' for any type of cursive characters. An OCR algorithm that can read any language characters, with provided background knowledge of sampled cursive character images. Propose a suitable OCR engine that can apply techniques of sampling, classification, recognition of cursive characters. A multi-language OCR tool that can understand and parse the cursive characters and draw various inferences to understand the scanned content of text. Enabling the extraction of keywords or key-characters of the document useful for classification and indexing.

Methodology

Recognition of cursive language characters shall start elementarily at the recognition of individual characters. Segmentation of the images allows in recognition of isolated characters of the scanned image of cursive language text content. Level-wise segmentation becomes mandatory in order to immensely parse the text content of the cursive languages. Level-wise segmentation helps in identifying and isolating characters from the scanned image of cursive language text content.

The 'begin' and 'end' of the cursive character partially or fully is first observed by tracing the white pixel columns of the isolated character. While the image of the isolated character is scanned column-wise, the presence of first black pixel in the column is identified as the 'begin' and as scan continues subsequently presence of a white pixel is the 'end' of character. An image of the cursive language character is thus preserved as image in-between two white columns. Several such small images of individual characters are represented as the image repository of the curse language. Each isolated character in separate image is void in four sides. The chain of code on sampled characters is observed with start of white pixel count, similar pattern of chain of code is observed in the cursive language text representing one pixel left on each side of margins in the isolated image. After determining the class of character, a chain code is calculated and matched with the sampled characters, this aids in improving the reliability and efficiency in identifying the characters. The calculated chain of code is matched with each column reading column by column with every sampled character's chain of code amongst the images of the determined class. All the columns of the sampled images shall be matched with the images of the determined class, with an extent of error margin. As same character from the different image sources is represented in different forms with little difference of properties, a threshold of error margin will be preset based on the complexity of the characters in the source images, to ease out the comparison of characters. If any sampled image is found to be prone as error, the chain of code is subsumed in the matching method in order to recognize the correct and exact character. If error count of the sampled images with the candidate images mismatches for columns calculated, then width of image is noted as different. Sampled images are equalized with the widths of the candidate images by adding the empty columns in either kind of images. After the width is adjusted for all the categories of images, height of the columns is justified; an empty row is added to the columns. After the rows and columns are balanced in the candidate images and sample images, the character shall be identified with least error count, which is also less than error margin.

Given a set of input images $\boldsymbol{I} = \{\boldsymbol{i}_0, \boldsymbol{i}_1, \boldsymbol{i}_2, \ldots, \boldsymbol{i}_n\}$, and a set of expected output $\boldsymbol{V} = \{V_0, V_1, V_2, \ldots V_n\}$, each of the V is a set of output variables. $\boldsymbol{i}$ is an image containing lines of cursive language text and V_i is the information extracted from the image $\boldsymbol{i}$. An image processing function $\boldsymbol{X}$ is deployed on images that can read the information from the images and assumed to be accurate for 100%, where $\boldsymbol{X}$: $\boldsymbol{I} \rightarrow \boldsymbol{V}$. As A is the accuracy of the X, then $\boldsymbol{A}(\boldsymbol{X})$ depends on the three steps in detection and recognition of cursive characters in three steps preprocessing, OCR and post-processing.

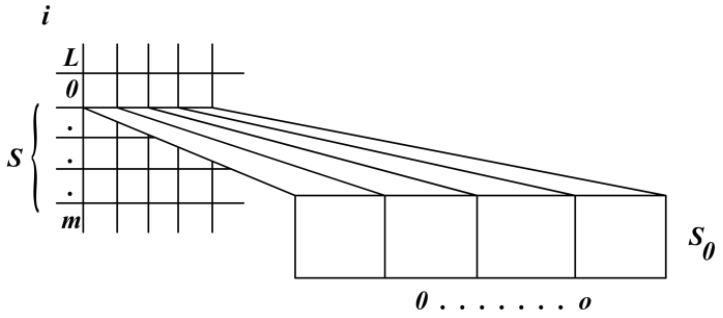

Fig. 1: Composition of symbols and lines in images.

Optical Character Recognition

In a set of lines scanned from the image, where each line comprises of symbol images, the process of OCR should output the list of lines, where each line is a list of symbols. If $\boldsymbol{S}'$ is the output symbol, then $\boldsymbol{S}' = \{\boldsymbol{L}'_0, \boldsymbol{L}'_1, \boldsymbol{L}'_2 \ldots \boldsymbol{L}'_n\}$, and each i^{th} line in the output $\boldsymbol{L}'_i = \{\boldsymbol{s}'_0, \boldsymbol{s}'_1, \boldsymbol{s}'_2 \ldots \boldsymbol{s}'_n\}$. Difference between the symbol and symbol image shall be noted in output and input to the OCR engine. Thus, the function of OCR operates on input $\boldsymbol{S}$ and output $\boldsymbol{S}'$.

Noise Filters

Many filters are available and applied to images to filter noise, in order to make the OCR algorithm read characters efficiently. Most commonly, a linear filter which uses a convolutional approach is applied to eliminate the fundamental visible noise from the images.

Tesseract

Around 1985 and 2005, Hewlett-Packard (HP) has developed Tesseract, as a open source OCR engine. Latest version of this open source engine is 4.0.0, employs a neural network based for training, testing recognition of specific characters.

Images containing lines of cursive language text are grayscaled and binarized, further by using luminance method and decolorize methods, to enable the correct parts of the crucial curves of characters. Tesseract OCR API can read such characters, which are not readable from human perceptions, while they look like complex scripts. While, Tesseract failed to recognize 10-20% of characters.

Convolutional Neural Network

A convolutional layer $\boldsymbol{l}$, following with M x M neuron nodes with an N x N filter, produces a $\boldsymbol{k}$-feature map. A Convolutional layer extracts corners, edges and endpoints, thus acts as a feature extractor. The non-linear properties of the decision function are improvised by the application of ReLU, which shall not decrease the performance of the receptive fields in the convolutional layers.

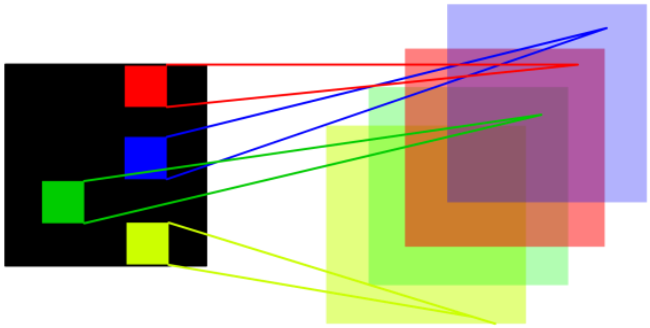

Fig. 2: k-feature map of a convolutional layer.

As some characters are ambiguous with similar strokes, tiny difference in strokes brings challenges in similar pair of characters. Some characters have ambiguous cursive strokes which seem to be same but are different, leads to wrong classification.

The detection and recognition of cursive characters in the languages is completed influence by the preprocessing of the input data. Selection, scaling, applying filters and classification are the preprocessing tasks that vary based on source data.

Table 1. The comparative Error Rates in Misclassification on the Training and Testing Data on 6000 images of cursive language text.

	Without Preprocessing	After Binarization	After Sharpening	After applying Linear filters
Tesseract OCR API	97.90%	97.90%	97.90%	97.90%
CNN	94.49%	94.49%	94.49%	94.49%

The results of classification are much meticulously studied based on the error rate of misclassification, less the error rate is the efficient the classification method, the CNN employs a better way of identifying the symbols in the line of text compared to traditional Tesseract OCR engine. The Tesseract OCR engine needs to be trained for the cursive characters where a additional algorithms shall be added to the engine to recognize cursive characters of all languages in general, where CNN can detect with the help of a minimal OCR and with kernel generated during the flow in the convolutional layers with most accuracy.

Experimentation

A. Data Collection

The benchmark data set for the experimentation is collected from MNIST. The MNIST database of handwritten fixed-size images containing a training and test data sets of 60,000 and 10,000 examples respectively, developed by Corinna Cotes et. al [5].

B. Methodology

Scanned images with cursive characters are preprocessed by converting to common size and grayscaling. All images are sized equally with cropping, chunking into single characters. Further categorization and noise elimination are performed on each character image. As large amounts of data sets are required for CNN to train the algorithm, all the images of cursive character text are broken into several character images with single cursive characters. Altogether, in the datasets of MNIST cursive digit images, 70,000 images of characters are drawn for the development of 28 x 28 sized characters. After preprocessing all the images are scaled into 24 x 24 sized single character images into grayscale thus avoiding so much of noisy data.

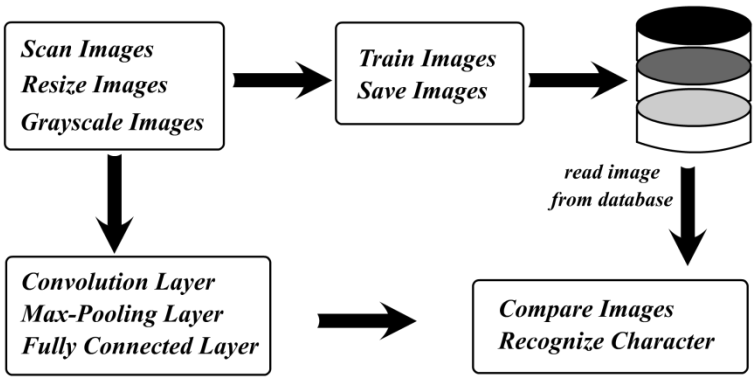

Fig.1 Methodology of CNN Implementation

The proposed model comprises of four convolutional layers, max-pooling layers each, to enhance learning of the images, where the preprocessed images are the input through the convolutional, max-pool layers, which are fully-connected layers. A ***2 x 2***

maximization is performed in the two hidden layers of the model and ReLU is used after each hidden layer. The following figure explains the process of enhancement in the layers on the images.

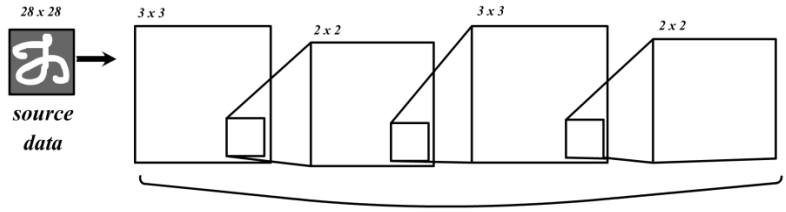

Fig. Convolutional and Pooling layers

C. Training and Testing

In above mentioned model almost 80% of data is used in training and 20% for testing. Each character after scaling and resizing has been drawn into a partition of 200 images where a collection of 50 images used in testing the dataset. A random partition attempt tests the accuracy for each attempt. The maximum is computed from the average accuracy from the attempts of the experiment and hence maximum is considered as accuracy.

Results and Discussions

A classification table is drawn illustrating the predicted number of successes compared with the number of successes actually observed and the predicted number of failures compared with the number of failures actually observed. The possible outcomes in the classification table are (TP) true positives, (TN) true negatives, (FP) false positives, (FN) false negatives.

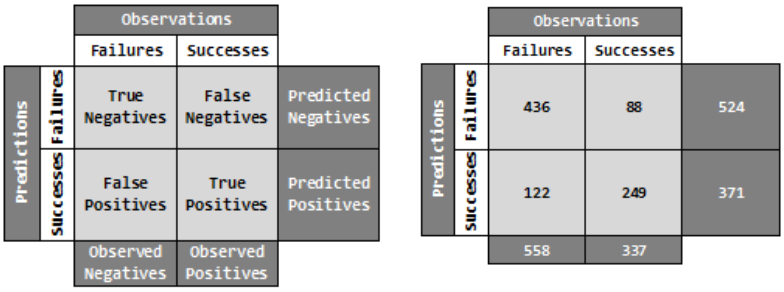

Fig. Classification Table / Confusion Matrix for MNIST database of handwritten images

The relative statistics for the above values is as follows:

True Positive Rate (Sensitivity)	0.738872
True Negative Rate (Specificity)	0.781362
Accuracy	0.765363
False Positive Rate	0.218638
Positive Predictive Value	0.671159
Negative Predictive Value	0.832061

The following data is computed on the samples of 100 images collected from the MNIST database, representing the number of elements recognized correctly and incorrectly.

Table 1: Worked table for computing the ROC on recognized characaters as correct or incorrect.

	Recognized		***Cumulative***				
No. of Random Samples of Images	*Correct*	*Incorrect*	*Correct*	*Incorrect*	*FPR*	*TPR*	*AUC*
			0	*0*	*1*	*1*	*0.082437*
50	46	3	46	3	0.9175627	0.9910979	*0.133212*
100	75	7	121	10	0.7831541	0.9703264	*0.168677*
150	97	11	218	21	0.609319	0.9376855	*0.171405*
200	102	25	320	46	0.4265233	0.8635015	*0.179509*

250	116	42	436	88	0.218638	0.7388724	*0.128442*
300	97	65	533	153	0.0448029	0.5459941	*0.010763*
350	11	82	544	235	0.0250896	0.3026706	*0.004882*
400	9	48	553	283	0.0089606	0.1602374	*0.001436*
450	5	36	558	319	0	0.0534125	*0*
500	0	18	558	337	0	0	*0*
	558	337					

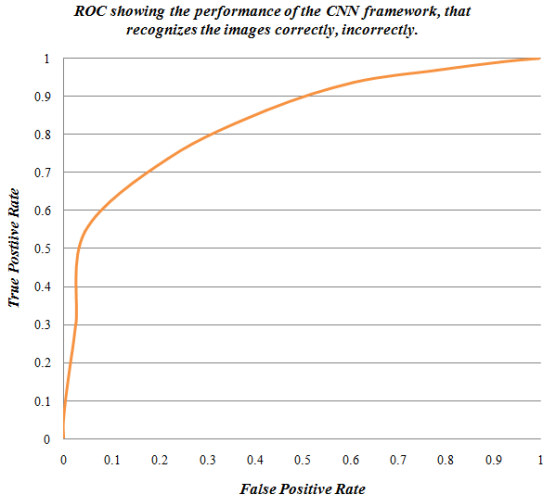

Fig.: ROC Curve showing the AUC

The peak points in the ROC curve represents the better fit of the proposed CNN framework. The AUC can be inferred that, closer to 1 is maximum value i.e., better fit. The AUC value which is closed to 0.5 states that the proposed model has ability of discriminating between correct and incorrect recognition of images as characters of cursive language, which is a due chance.

The classification is employed with an incremental characteristic analysis, where the cursive handwritten characters are evaluated. The cursive handwritten characters are borrowed selectively from MNIST database. A specialized sample based selection is considered while computing the accuracy level in the activity of the recognition, a five character classes are considered as the choice for experimentation. The character classes are progressively increased in multiples of five (5) and observed the level of accuracy is decreased, such that a median number of the character classes is considered based on the complexity of the class character image representation. The successive observations of the CNN prove that the higher the size of the character classes the lower the accuracy achieved from the experiment, hence the median number is chosen for the size of a character class, for the definition of the input through the CNN.

Conclusion

The differences between traditional Tesseract OCR Engine and the CNN for image processing have been discussed. Preprocessing the source images before actual implementation of character recognition is a mandatory, which will improve the efficiency of detection and recognition. Modern OCR implementations certainly have advanced features of recognizing the characters of multiple languages. As far as OCR engine developed in the laboratory is concerned that they can recognize a particular language which is defined in the corpus as background knowledge, the attempt in this paper have been made that any language with cursive characters can be recognized with the better corpus provided with all the possibilities of representing the parts of the characters. Images of various parts cursive characters in the languages have been studied and presented the basic approach of identification of cursive characters and the differences between a traditional Tesseract OCR engine and CNN for image processing.

References

[1] J. Mariyathas, V. Shanmuganathan and B. Kuhaneswaran, "Sinhala Handwritten Character Recognition using Convolutional Neural Network," 2020 5th International Conference on Information Technology Research (ICITR), 2020, pp. 1-6, doi: 10.1109/ICITR51448.2020.9310914.

[2] El-Sawy, Ahmed and M. Benha. "Characters Recognition using Convolutional Neural Network." (2017).

[3] Wei, Tan, Usman Ullah Sheikh and Ab Al-Hadi Ab Rahman. "Improved optical character recognition with deep neural network." 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA) (2018): 245-249.

[4] Anil, R., K. Manjusha, S. Sachin Kumar and K. P. Soman. "Convolutional Neural Networks for the Recognition of Malayalam Characters." FICTA (2014).

[5] Cortez, Corinna; Burges, Christopher C.J.; LeCun, Yann, "The MNIST Handwritten Digit Database". Yann LeCun's Website yann.lecun.com. Retrieved 30 April 2020.

[6] Sriharsha, A. V., and A. Rama Mohan Reddy. "Ripple Effect in Software Architectures: en route avoidance." International Journal of Recent Trends in Engineering 1, no. 2 (2009): 131.

[7] Katyayani, J., A. V. Sriharsha, and B. Sudhir. "Text mining: Finding hot topics TF* PDF vs. LSI." In 2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 526-529. IEEE, 2009.

[8] Sowjanya, K., and A. V. Sriharsha. "Improvisation Of k-NN Classifier On Semantically Secure Encrypted Relational Data." International Journal of Computer Engineering & Technology (IJCET) 8, no. 3 (2017): 1-10.

[9] Prema, K., and A. V. Sriharsha. "Differential privacy in big data analytics for haptic applications." Technology 8, no. 3 (2017): 11.

[10] Sriharsha, A. V., and C. Parthasarathy. "Multi-level and multi-key trust in PPDM." In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), pp. 1-5. IEEE, 2013.

[11] Sriharsha, A., and A. Rama Mohan Reddy. "Music inspired HS algorithm for determining software design patterns." Issues 1 (2014): 230-238.

[12] Sriharsha, A. V. "A Novel Approach For Identifying Optimal Design Pattern for user specified Requirements using formal modeling.",2018

[13] Sriharsha, A. V. "Multi level multi key trust in PPDM using syntactic anonymity on sensitive data.",2017