SETSCI conference proceedings
SETSCI Conference Proceedings
9, 431-436, 2019

4th International Symposium on Innovative Approaches in Engineering and Natural Sciences
November 22-24, 2019, Samsun, Turkey

https://doi.org/10.36287/setsci.4.6.111
2687-5527/ © 2019 The Authors. Published by SETSCI
ISAS WINTER - 2019

Recognition of Handwritten Source Code Characters With Deep Neural Networks

BarışKılıçlar[1+], Metehan Makinacı[2*]

[1]*The Graduate School of Natural and Applied Sciences, Dokuz Eylul University, İzmir, Turkey*
[2]*Department of Electrical and Electronics Engineering, Dokuz Eylul University, İzmir, Turkey*
[*]*Corresponding author: metehan.makinaci@deu.edu.tr*
[+]*Speaker: bariskiliclar@gmail.com*
Presentation/Paper Type: Sözlü / Full Paper

Abstract **–**In this paper we present an application of deep learning techniques to recognize handwritten source code characters. Although there are many works on the handwritten character recognition (HCR) problem, very few have been done about the offline handwritten source code character recognition. The problem includes the recognition of source code specific characters. We designed and implemented an application, performing preprocessing, histogram based segmentation and normalization on the scanned documents of exam papers which include codes that were written in C programming language. Constructed dataset includes 7093 source code character samples. We enriched this dataset with character samples from the CROHME database by transforming them to offline samples. With resulting 95 classes of 17748 samples, we trained and tested several models of convolutional neural networks (CNN). CNN is a deep learning architecture which is shown to produce state-of-the-art performance rates for handwritten character recognition tasks as well as for various other computer vision applications. Experimental evaluations gave performance rates between 95.43% and 97.49%. We conclude that CNN based classifiers are powerful tools for handwritten source code character recognition task.

Keywords *–Handwritten Character Recognition, Deep Neural Networks, Convolutional Neural Networks, Deep Learning, Handwritten Source Code Classification*

I. Introduction

Handwriting is still being the one of the most used, natural and convenient way for collecting, storing and transmitting the information. Automating the translation of handwriting document into the digital environment obviously has many useful applications. They include automatizing processes that involves handwritten document analysis, such as postal code handling, bank check reading, and digitally storing historical documents.

The term handwriting character recognition (HCR) is used for the progression from the document image analysis to content analysis. Analyzed document which contains handwritten source code can be used as input for computers. This method can have some advantages and convenience for certain circumstances over the techniques solely digital, like keyboard input.

Although extensive amount of research has been made at recent years, handwritten character recognition field still needs to be improved [1]. It is much challenging because there are so many factors affecting the performance.

Handwriting recognition field history dates back to 1960s [2]. For recognition, various classifiers are proposed in the basic application of pattern recognition since that time such as Support Vector Machine (SVM) [3][4], Multilayer Perceptron (MLP) [5], K Nearest Neighbor (KNN) [4], Self Organizing Map (SOM) [6] and Hidden Markov Model (HMM) [7][8]. Amongst others, Deep Learning Techniques gained popularity because Convolutional Neural Networks (CNN) presented outstanding success rates recently and as a result, CNNs are agreed to be the state of the art for HCR applications.

We studied on the handwritten character recognition of scanned exam papers answered by the students of the Department of Electrical and Electronics Engineering at Dokuz Eylül University. Exam papers contain source codes which are written in C programming language. Documents are partially constrained since the exam papers are outlined by red guide lines. Although the name field has a grid of guide lines, answer field has only line guides. In this field, handwritten source code in C programming language is written line by line. Guidelines make segmentation methods to give more accurate results.

There are many works done in the field of HCR [9] [10], but very few progress has been made regarding the *handwritten source code character* recognition. To the best of our knowledge, only few papers have been published applying Convolutional Neural Networks (CNNs) to the task of *handwritten source code character recognition* [2]. We extract character images and construct a dataset for this task and obtain results with different CNN models for comparison. Performance comparisons propound CNN based classifiers are powerful tools for *handwritten source code character recognition* task.

II. Materials and Method

Examining the previous works done in the field of HCR, we can see that the processes can be grouped in phases as:

preprocessing, segmentation, representation, training and recognition and postprocessing. Not all works implement all of these phases, some of them might be merged, omitted and some other phases might be added.

In training and recognition phase, mostly feature extraction and classification is used. Feature extraction is used to get a prominent representation of data to make classification performance high. Usually feature extraction process done manually because of the data variety between different problems. After getting fairly promising representation, classification process takes place which uses a different method than the feature extraction. Lately a different type of approach is increasingly used which integrates feature extraction and recognition phases into one process automatically. This is called as deep learning technique and developed based on neural networks [11].

A. Convolutional Neural Networks

Deep learning algorithms are proven to have good performance on the learning of the features of data which are better suit to task at hand. They are organized in a hierarchy with multiple levels each of which is associated with a different feature of data. Convolutional neural networks (CNNs) are one of the architectures implementing deep learning technique and they can extract hierarchical features from input data, automatically [11]. They are a regularized variant of multilayer perceptrons (MLPs) and developed using insights from the work of Hubel and Wiesel on cat's visual cortex [12]. Their work reveals that cat's visual cortex has complicated structure of locally sensitive cells. Each group of these cells are sensitive to different subregions of the image appeared on the input space (retina). These subregions are called as receptive field of that group of cells. Local receptive fields are best suited to utilize the local correlations in the images since the images have strong 2D local structure and neighboring pixels are highly correlated. Utilization of spatial correlation yields better feature extraction in hierarchical levels, for example edges and ovals in lower levels and eyes and ears in higher levels. Fukushima proposed the first CNN model, called Neocognitron for the task of recognition handwritten digits [13]. Reminiscent of cells in the Hubel and Wiesel work, for extracting local features of input data, Neocognitron employed locally receptive field idea such that each neuron connected only subregion of the preceding layer which consist of some neighboring neurons in that layer. LeCun et al. proposed another CNN model, LeNet-5, using gradient based back propagation learning algorithm [14]. CNN architectures generally adapt three concepts. These are local receptive fields, weight sharing and spatial subsampling which gives shift, rotation and distortion invariance at some extent. Without doubt this quality of CNNs is very beneficial and best suited for the tasks involving recognition from 2D images.

As a general design principle, almost every CNN model has the following sequential structure: convolutional layers, subsampling layers, one or more hidden layer of fully connected traditional neural networks and final output layer. Layers are the planes of organized units of neurons through which the input data is transformed, called also as *credit assignment path* (CAP), whereas the number of layers called as CAP depth. Deep learning architectures generally uses substantial amount of CAP depth, hence the name *"deep"*. The number of convolutional and subsequent subsampling layers is chosen to best fit to the actual task of the network. In these layers each unit of cells are connected only to a small subset of neighboring units in the preceding layer which constitutes local receptive field of the unit concerned. Local receptive fields give units the ability to extract features hierarchically, i.e., low level features in former levels and high level features in latter levels. All cells of units in the layer plane share the same set of weights and are constrained to perform the same operation along all the portions of the input image. Outputs of these units constitute the feature map of this plane. Multiple features are extracted for each portion of the input image because a convolutional layer utilizes more than one feature map, and these feature maps have different weights. Resulting feature maps are fed into the hidden layer which is generally fully connected layer of cells. These cells are connected to the output layer. Output layer performs final classification based on its inputs. Number of outputs of this layer is determined by the number of specific target classes of the actual task. Fig. 1 shows a diagram of CNN used in this work for handwritten character recognition task.

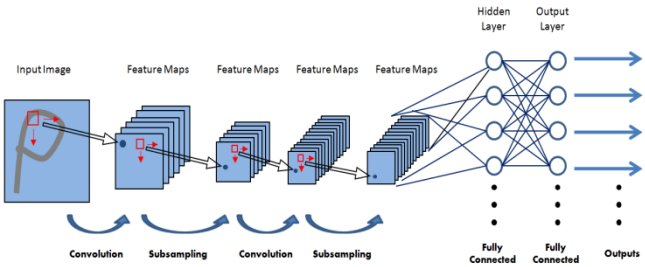


Fig. 1 Representation of CNN architecture.

B. Convolutional Layer

Convolutional layers are responsible for extracting features that corresponds to patterns to be learned and detected. In image processing, convolution is a mathematical operation and a sort of filtering which involves input matrix (image) and a filter matrix. This filter is also called as kernel in image processing terminology, and denoted as *h*, and image denoted as *f* in the following formula. The indexes of rows and columns of the resulting matrix are marked with *m* and *n* respectively.

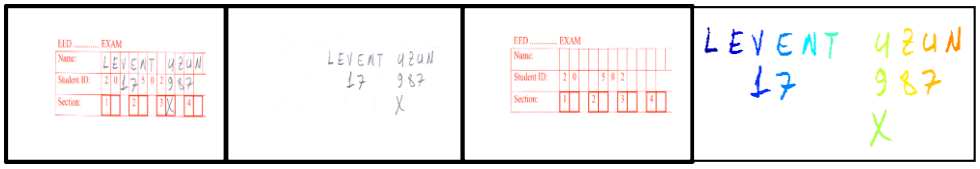
$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

C. Activation Layer

Feature maps produces by the convolutional layers are much smaller in size than the input matrix. Since the CNNs resemble NNs, they as well, adopt activation functions which are arranged in the layer called as *Activation Layer*.This layer is thought to be a successive part of the preceding convolutional layer, it is not shown explicitly in topological schemes since it is almost always used after a convolutional layer.

Activation functions determine the output of the neuron, based on the input value. Various activation functions exist for CNNs such as *linear function*, *sigmoid*, *tanh*, *binary step*

function and *ReLu*. Non-linear activation functions are the most popular ones since they have localized outputs thus reducing the complexity of the input and make it possible the network to generalize or adapt with data and they better suit with the backpropagation learning algorithms which allows the network to train itself making it capable to learn and perform more complex tasks.



a b c d

Fig. 3 Guideline removal gives image of handwriting(b) and guidelines(c). Original image is depicted in (a). Connected component labelling analysis illustrated in (d). Different colors show different labels determined.

ReLu (Rectifying Linear Unit) is the most common activation function recently, which is defined as:

$$ReLu(x) = max(x, 0)$$

It is simply performing a threshold operation on the input and can be implemented and calculated more easily than the *tanh* or *sigmoid* functions. Thresholding operation causes only few neurons to be activated at a time. These attributes of the *ReLu* function make it possible to reduce overall computational cost effectively.

D. Subsampling Layer

The next layer is subsampling layer in CNN architecture. It is also called as downsampling layer. Subsampling layer operates on the output of the activation layer. The intuition for the subsampling operation is that the exact location of a feature is not important if it has been found. The result of subsampling operation is reduction in the size of feature maps while keeping the important information. That means there would be less parameter to be calculated, thus, subsampling too, helps to reduce the computational cost. Subsampling helps also to prevent overfitting.

Max pooling is the most common used type of subsampling that supplies position invariance over larger local regions. Scherer et al. [15] states that max pooling can converge faster, yield better generalization and select superior invariant features. General feature pooling's and max pooling's theoretical analysis can be found in the work of Bourreau et al. [16].

E. Fully Connected Layer

A CNN model usually has several amounts of convolutional-activation-subsampling layers and one or more fully connected layers subsequently. Typically, they have much more parameters than the convolutional layers which is the one reason why they are generally used at the end of the architecture since the feature maps sizes get smaller there. Another reason is that convolutional layers give localized results using *local receptive fields* principle by decreased number of connections. Presumably, fully connected layers can give the same results after much more training epochs.

In the input of first fully connected layer, resulting feature maps from the previous layer are flattened, i.e., reconstructed as a vector of one dimension and fed into the fully connected layer which is essentially a traditional neural network layer. Neurons in the fully connected layers are connected all the outputs (feature maps or also called as activations) of the previous layer, as its name implies. The output of this layer is calculated as the matrix multiplication of the input by a weight matrix and adding a bias vector. Learning and recognizing of the larger patterns happens at this layer since it combines all the information learned by the previous layers. Combining all the information gives network the ability to classify the whole input image correctly for classification problems.

For the networks designed for classification problems, the output size of the last fully connected layer is equal to the number of classes of the dataset. Its outputs are probabilities for the possible labels that the image to be classified in (e.g. car, train, bus). The label corresponding to the output with the highest probability is the classification decision.

F. Methodology

Our work has two phases. (a) dataset construction, (b) training and testing CNN by enriched dataset consist of ours merged with CROHME dataset.

The first phase can be divided into following steps:

- (a1) image pre-processing including guideline removal, orientation correction, median filtering and connected component based filtering
- (a2) image segmentation
- (a3) normalization
- (a4) manual labelling
- (a5) dataset enrichment by merging with CROHME dataset.

The second phase can be divided into below steps:

- (b1) choosing/designing and implementation of different CNN topologies.
- (b2) training CNNs for character level classification
- (b3) testing/ comparison

G. Dataset Construction

As mentioned earlier, we used exam papers for C programming language course. A sample document is shown in Fig. 2.

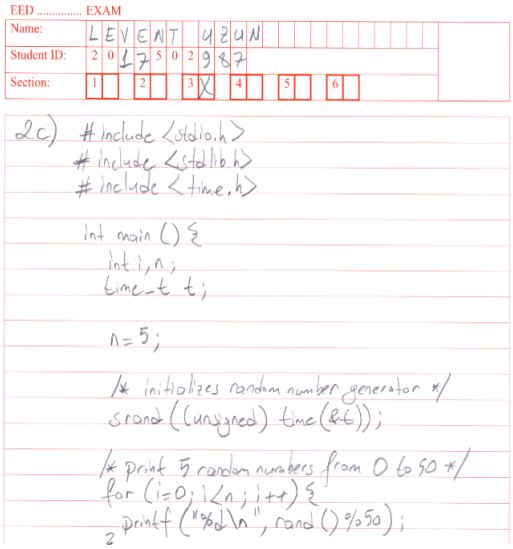


Fig. 2 A sample of document used in HCR processes.

We constructed a dataset of all handwritten characters, including source code specific ones, by using steps of dataset construction phase, i.e., preprocessing, segmentation and normalization. Sample results of preprocessing step are shown in Fig. 3.

H. Segmentation

For segmentation, we benefit from guidelines. Guidelines aid character level segmentation for the upper part where horizontal and vertical guidelines exist. At the lower part, guidelines are used for textline level segmentation since there exist only horizontal guidelines in this part. Then we used histogram projection algorithm for character level segmentation.

To make use of guidelines, we took the guidelines image extracted previously at guidelines removal step. First we found histogram projection of guidelines on both dimensions, vertical and horizontal. The histogram projection was found simply by summing all the values along the relevant dimension.

Vertical Histogram:

$$\boldsymbol{V}_i = \sum_j \boldsymbol{Q}_{i,j} \; i = 1,..,m \; ; j = 1,..,n$$

Horizontal Histogram:

$$\boldsymbol{H}_j = \sum_i \boldsymbol{Q}_{i,j} \; i = 1,..,m \; ; j = 1,..,n$$

where $\boldsymbol{V}$ is a raw vector showing vertical histogram projection, $\boldsymbol{H}$ is a column vector showing horizontal histogram projection and $\boldsymbol{Q}$ is the m x n size binary image of guidelines. Resulting projections are shown in Fig. 4 along with the guideline image.

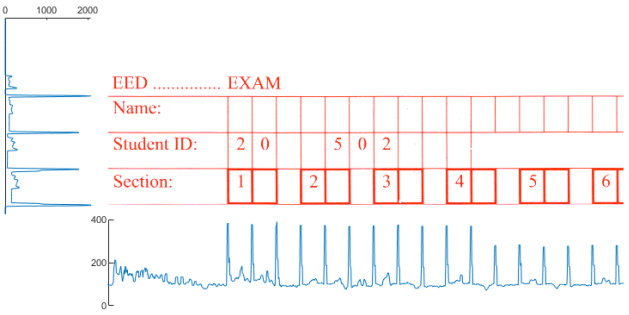


Fig. 4 Vertical and horizontal histogram projections for the upper part of the guidelines.

Then we found regions surrounded by guidelines by applying threshold on the histogram projections. Thresholds are found emprically.

$$G_i = V_i > T_v \; ; \; i = 1,..,m$$
$$G_j = H_j > T_h \; ; \; j = 1,..,n$$

where G_i is a binary raw vector showing vertical guideline locations, G_j is a binary column vector showing horizontal guideline locations, T_v and T_h are threshold values for vertical and horizontal projections, respectively.

Then we look at connected components labels at every region surrounded by the guidelines. For every region we found all the labels whose center falls in that region. Fig. 5 illustrates this process. The "}" character is assumed to be in the region which is surrounded by the guidelines since its center fall into that region, whereas the character "p"'s is not.

İ. Manual Labelling

We developed an application which implements all these steps. In the application, segmented and normalized characters are shown and asked user to label it. A screenshot of the application depicted in Fig. 6. This application employs a NN which recognizes segmented characters to assist the user in manual labelling process. This NN predicts wrong labels especially for source code specific characters since it hasn't beentrained to recognize them.

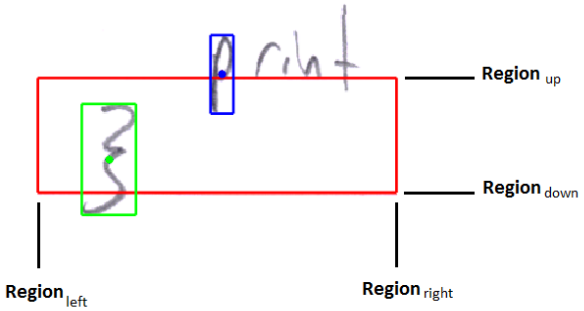


Fig. 5 Illustration for assignment of labels to regions

Formatted character images labeled by the user were saved to our database. Users discarded wrongly segmented characters here.

We processed and labeled 116 images by this application. We saved 2536 alphanumeric characters, i.e. digits and uppercase and lowercase letters, including Turkish specific ones. And we saved 4557 handwritten non-alphanumeric characters such as punctuations, parentheses, and other C programming language source code specific characters.

Alphanumeric characters:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzÇİĞÖŞÜçığöşü

Non-alphanumeric characters:

!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

Our overall database is composed of 7093 labeled characters.

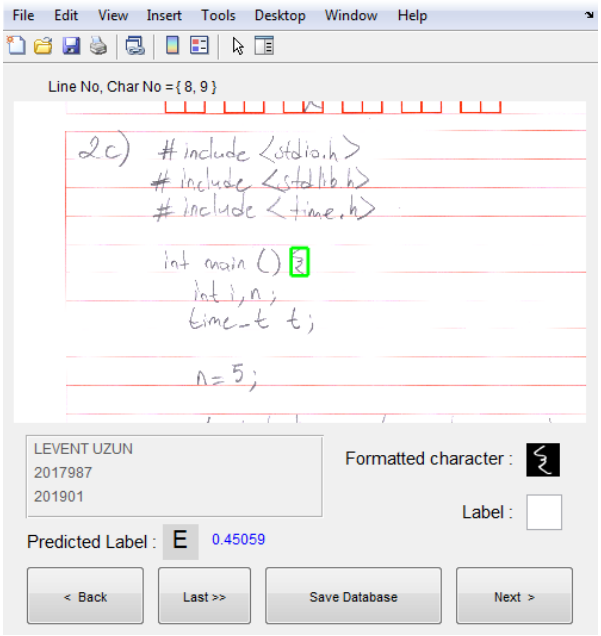


Fig. 6 Manual labelling application that implements preprocessing, segmentation and normalization.

J. *Dataset Enrichment*

To increase number of characters, we enriched our database by using CROHME dataset which is publicly available. CROHME is Competition on Recognition of Online Handwritten Mathematical Expressions, being organized since 2011. All data is freely available for research purposes [17]. Data is in InkML format.

Although CROHME dataset is for recognition of online handwritten mathematical expressions, it has much common symbols with the C programming language source code like punctuations, mathematical operators and parentheses. We converted online data in InkML format to the offline segmented and labeled characters with the same format of our dataset, i.e., 28x28 intensity images but without gray levels (black and white intensity images). We enriched our dataset by merging ones we took from CROHME dataset. We took 5281 alphanumeric and 5374 non-alphanumeric handwritten characters with their labels. After merging, dataset overall volume is 17748 offline handwritten characters with labelsin 95 classes. Resulting merged datasetwas used to train and test CNNs.

K. *Train and Test of CNNs*

Hierarchical learning capabilities of CNNs gives them advantage over other techniques for learning features automatically so that classification based on raw image data can be done. In this work we utilized this competence of the CNN architecture. In training phase, document images after preprocessing phase are fed into the segmentation stage to extract character images. Without any feature extraction, these images along with the labels are directly used for training of the CNN with deep learning algorithm in a supervised learning manner. Test phase use the same scheme for character extraction. Then extracted character images are submitted to the CNN classifier which is trained in the training phase. Character labels are used to measure the performance of the system by checking the match with the classifier's output. Training and test phases are illustrated in Fig. 7.

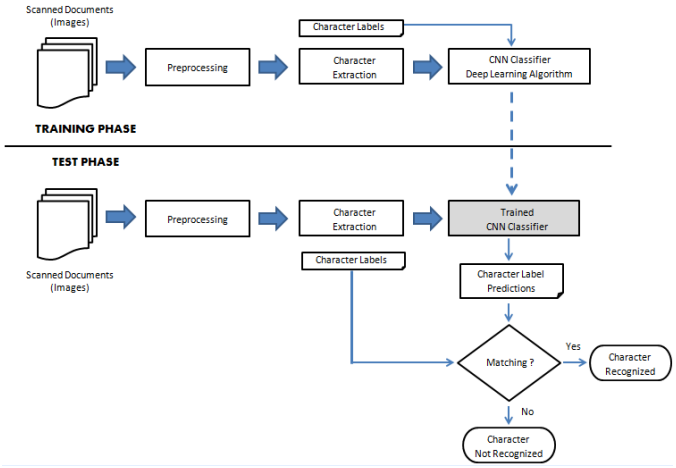


Fig. 7 Used methodology for character recognition task with a deep learning architecture: Training and test phases.

L. *CNN Topologies*

We used different CNNs with different layers and parameters for comparison. We will use the following notation to explain the topology of CNNs that we trained and tested:

C(f,n) : Convolutional Layer with the *n* number of filter (kernel) size *fxf*. Stride is 1 for all convolutional layers.

Convolutional Layers are always followed by batch normalization and ReLu layers subsequently.

M(p,s) : Maxpool with pool size = *p* x *p* and stride = *s* x *s*

FC(q) : Fully connected network layer of *q* neurons

Networks end with a softmax layer and an output classification layer.

III. RESULTS

We trained and tested the following CNNs:

CNN1: C(3,8)-M(2,2)-C(3,16)-M(2,2)-C(3,32)-FC(95)

CNN2: C(5,8)-M(2,2)-C(5,16)-M(2,2)-C(5,32)-FC(95)

CNN3: C(8,8)-M(2,2)-C(8,16)-M(2,2)-C(8,32)-FC(95)

CNN4: C(12,8)-M(2,2)-C(12,16)-M(2,2)-C(12,32)-FC(95)

CNN5: C(5,32)-M(2,2)-C(5,64)-M(2,2)-C(5,128)-FC(100)-FC(95)

CNN6: C(5,32)-M(2,2)-C(5,64)-M(2,2)-C(5,128)-FC(95)

Initial learning rate was chosen as 0.003 for training of all the CNNs. Validation and test data sizes are both chosen approximately 1/10 of the total data size. Table 1 shows validation andtestperformances of the CNNs.

Table 1. CNN performances.

System	Validation Accuracy %	Test Accuracy %
CNN1	95.94	95.72
CNN2	95.83	95.43
CNN3	96.45	95.77

CNN4	96.62	96.05
CNN5	96.28	96.56
CNN6	97.86	97.47

We did 6-fold cross-validation for CNN6 to get more accurate result. This gave **97.59%** validation accuracy and **97.49%** test accuracy as the average of the 6-fold cross-validation results.

IV. DISCUSSION

Results show that if CNNs with proper topology trained with sufficient amount of data and with correct parameters, they can succeed in automated reading of handwritten source code characters as with a human reader. Some characters are very difficult to discriminate such as "I" and "1", "U" and "u", "-" and "_", even for a human reader, without using contextual and positional information.

We think that results can be improved by including positional information of the data which is lost by normalization process in the dataset construction phase.

V. CONCLUSION

Deep neural network architectures have gained increasing popularity in machine learning applications recently because of their high performances. CNN model is one of them which has widespread application fields involving 2D pattern recognition. We think that *recognition of handwritten source code characters* is a promising task that will be useful in various cases. CNNs can successfully be utilized at this task. If publicly available source code character databases are created, that would boost the new researches at this field.

Segmentation is very important for character based recognition schemes. More effective segmentation methods are still needed to assist the recognition.

Our test results exhibit that the CNNs are the state of the art models for *recognition of handwritten source code characters* as with other various pattern recognition tasks.

REFERENCES

[1] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 2, pp. 216-233, May 2001.

[2] ZhiQiyu, Metoyer Ronald. "Recognizing Handwritten Source Code", *Proceedings of Graphics Interface* 2017,p. 163 - 170.

[3] Shruthi A., Patel M. S., "Offline Handwritten Word Recognition using Multiple Features with SVM Classifier for Holistic Approach", *International Journal of Innovative Research in Computer and Communication Engineering*., vol. 3, Issue 6, 2015.

[4] M.Jangid, K.Singh, R.Dhir, R.Rani, "Performance Comparison of Devanagari and written Numerals Recognition",*International Journal of Computer Applications*, p.0975 – 8887, Volume 22– No.1, May 2011.

[5] B. Hussain, M. Kabuka, "A Novel Feature Recognition Neural Network and its Application to Character Recognition", *IEEE Trans. Pattern and Machine Intelligence*, vol. 16, no. 1, p.99-106, 1994.

[6] C. Y. Liou, H.C. Yang, "Handprinted Character Recognition Based on Spatial Topology Distance Measurements", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.18, no.9, p.941-945, 1996.

[7] El-Yacoubi, A., Sabourin, R., Gilloux, M. et Suen, C.Y.. 1999. "Off-line handwitten word recognition using hidden markov models". In *Knowledge-based intelligent techniques in character recognition.* p. 191-230. USA : CRC Press Inc.

[8] M. Y. Chen, A. Kundu, J. Zhou, "Off-line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network", *IEEE Trans. Pattern Recognition and Machine Intelligence*, vol.16, p.481-496, 1994.

[9] Ciresan, D.C., Meier, U., &Schmidhuber, J. "Multi-column deep neural networks for image classification". *IEEE Conference on Computer Vision and Pattern Recognition,*2012, p. 3642-3649.

[10] Øivind Due Trier, Anil K. Jain, TorfinnTaxt, "Feature extraction methods for character recognition-A survey", *Pattern Recognition*, Volume 29, Issue 4, 1996, p. 641-662, ISSN 0031-3203.

[11] Darmatasia and M. I. Fanany, "Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)," *5th International Conference on Information and Communication Technology (ICoIC7)*, 2017, p. 1-6.

[12] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex."*The Journal of physiology*, 160(1), p.106, 1962

[13] Kunihiko Fukushima. "Neocognitron: A hierarchical neural network capable of visual pattern recognition."*Neural networks*, 1(2) p.119–130, 1988.

[14] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based Learning Applied to Document Recognition", *Proc. of the IEEE*, Vol. 86, Issue 11, 1998, p.2278–2324.

[15] Dominik Scherer, AdreasM¨uller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In *International Conference on Artificial Neural Networks*, 2010.p. 92-101.

[16] Y-Lan Boureau, Jean Ponce, and Yann LeCun. "A Theoretical Analysis of Feature Pooling in Visual Recognition". In *International Conference on Machine Learning*, 2010.

[17] (2019) CROHME dataset. [Online]. Available: https://www.isical.ac.in/~crohme/CROHME_data.html