



IJIRCCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 7, July 2021

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 7.542



9940 572 462



6381 907 438



ijircce@gmail.com



www.ijircce.com

Optical Character Scanning & Recognition

Swapnil Thorat¹, Harshit Mohan Srivastava², Aayush Maheshwari³

Faculty, Department of E&TC Engineering, Bharati Vidyapeeth (Deemed to be) University, College of Engineering,
Pune, Maharashtra, India ¹

Department of E&TC Engineering, Bharati Vidyapeeth (Deemed to be) University, College of Engineering, Pune,
Maharashtra, India ^{2,3}

ABSTRACT: Given the availability of handwritten documents for human trafficking, Optical Character Recognition (OCR) of texts is of limited value. Character recognition is a science that allows the translation of various types of documents or images into analytical, edible and searchable data. For the past ten years, researchers have used sophisticated reading / machine learning tools to automatically analyze handwritten and printed texts in order to convert them to electronic format. The purpose of this research paper is to highlight the character recognition method that uses machine learning skills. Other python libraries used/ are Keras, Tensorflow, OpenCV. Other algorithms used are CNN and Deep Learning.

KEYWORDS: CNN, Deep Learning, KerasTensorflow, Machine Learning, Open CV

I. INTRODUCTION

This paper, 'Character Recognition' is a software algorithm work to detect any character well on a computer capable of inserting an old optical image. Character recognition, often abbreviated in OCR alphabetical visualization, mechanical or electronic translation of typed or printed text (usually scanned by a scanner) into text edited text. It is a field of research in pattern identification, artificial intelligence and machine vision. Although research studies in this field are ongoing, the focus on character recognition has shifted to the use of proven techniques. Character recognition is a program that allows a computer to read, understand, improve and translate a written or printed character in its own language, but present the same as defined by the user. Original Character Recognition uses the image processing process to identify any computer character / typewriter. This field has been highly researched. But further refinement of OCR strategies is done based on the fact that the algorithm must have high recognition accuracy, high persistence at certain times of accurate prediction and extra time to perform. The idea is to develop appropriate techniques that include input in digital image format. It then processes the image for better comparison. After that the image used is compared to an existing set of font images. The final step provides character prediction with percentage accuracy.

II. THEORY

3.1 Artificial Neural Network

The first phase of the Neural Network was developed by Warren McCulloch and Walter Pitts in 1943 which was a computer model based on Mathematics and algorithms. This model paved the way for research that focused on the use of Neural Networks in Artificial Intelligence.

The neural implant network is basically a match for a large number of connected cells. The order of cells is such that each cell receives input and conducts the output of the next cells. Each cell can be pre-defined

The diagram below is a block diagram showing the structure and function of the Artificial Neural Network generated. Neurons are connected to each other in a sequential way. The network has many hidden layers depending on the input and data comparison decision.

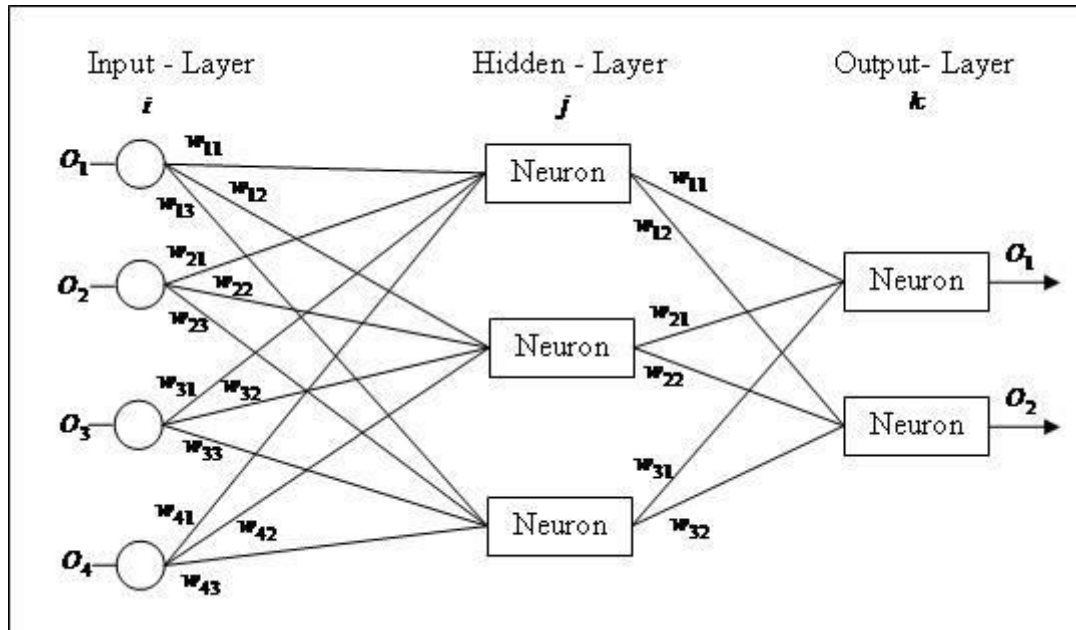


Figure1: Plotting the number of alphabets in the dataset

3.2 Network Building and Training

In case of character recognition we should create a 2D vector of characters that can be networked as a good collection of input variables. In our case there are 26 English letters in total that we should see.

3.3 Convolutional Neural Networks:

CNN stands for Convolutional Neural Networks which is used to extract image elements using specific layers of filters.

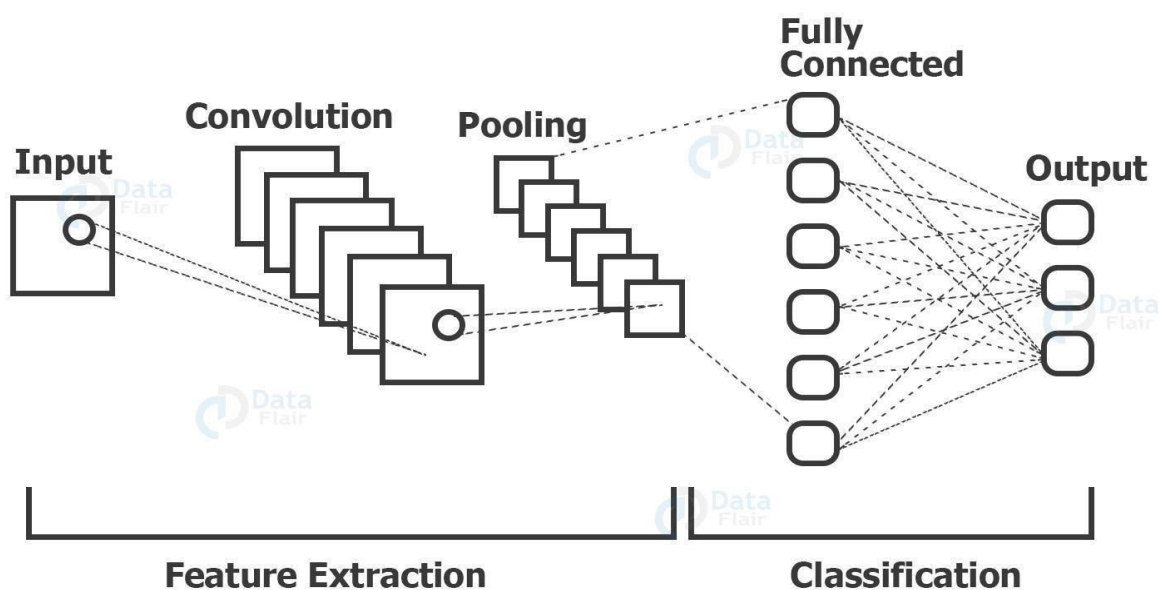


Figure 2: Plotting the number of alphabets in the dataset

Layers of convolution are usually followed by layers of maxpool used to reduce the amount of material extracted and finally maxpool extensions and layers and convolution layers are embedded in a single vector velocity and provided as a Dense layer (fully integrated network) We have a CNN model built for training model in training database.

III. SOFTWARE SPECIFICATION

Python:

Python is a translated, object-oriented, high-quality language with dynamic semantics. Its high-quality data structures, combined with strong typing and powerful binding, make it very attractive in the development of a fast-paced system, as well as being used as a scripting or paste to connect existing components together.

Python language is very easy to use and learn for beginners and newcomers. Python language is one of the most accessible languages available because it simplifies syntax and is not complicated, which gives great emphasis to natural language.

Because of its easy learning and use, python codes can be easily written and made much faster than other programming languages.

Why we use Python for our research:

Hundreds of Python libraries and frames

Thanks to the support of the company and the large supporting python community, python has excellent libraries that you can use to select and save your time and effort in the first round of development. There are also many media services that provide cross-platform support with tools such as libraries, which can be very helpful.

Some of the libraries used in our work are:

1. Keras / tensorflow:

Tensorflow is a platform for the ultimate open source learning machine. It has a complete ecosystem of tools, libraries and community resources that enable researchers to compress ML technology and developers to easily create and use powerful ML applications.

Keras is a small Python library for in-depth reading that can work beyond Theano or TensorFlow.

It is designed to make in-depth learning methods as fast and easy as possible in research and development.

It works on Python 2.7 or 3.5 and can work seamlessly on the GPUs and CPUs provided by the basic frameworks.

2. Opencv:

OpenCV was launched at Intel in 1999 by Gary Bradsky, and the first release was released in 2000. Vadim Pisarevsky joined Gary Bradsky to lead Intel's Russian OpenCV team. In 2005, OpenCV was used in Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued with the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the research. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is increasing day by day.

OpenCV supports a vast variety of languages such as C ++, Python, Java, etc., and is available on a variety of platforms including Windows, Linux, OS X, Android, and iOS. Fast GPU-based CUDA and OpenCL operating systems are also under active development.

OpenCV-Python is an OpenCV Python API, which combines the best features of the OpenCV C ++ API and Python language.

OpenCV-Python:

OpenCV-Python is a library of Python bonds designed to solve computer vision problems.

Python is a standard programming language designed by Guido van Rossum who became very popular very quickly, mainly because of its easy-to-read and code-readability. It enables the program editor to display ideas in a few lines of code without reducing readability.

Compared to languages like C / C ++, Python is slower. That is, Python can be easily extended with C / C ++, which allows us to write computer-generated code in C / C ++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C / C ++ code (because it is the real C ++ code running in the background) and second, it is easier to install in Python than C / C ++. OpenCV-Python is a Python package for the actual launch of OpenCV C ++.

OpenCV-Python uses Numpy, a highly customized library of numerical functionality with MATLAB-style syntax [7] All properties of the same OpenCV members are converted and removed from the same Numpy members. This also makes it easier to integrate with other Numpy libraries such as SciPy and Matplotlib.

NumPy:

NumPy is a Python library that offers a simple but powerful data structure: a n-dimensional array. This is the foundation on which almost all the power tools of Python data science tools are built.

Here are four key benefits that NumPy can bring to your code:

Extra speed: NumPy uses C-written algorithms that complete in nanoseconds rather than seconds.

A few pitfalls: NumPy helps you lower the loops and avoid getting stuck in iteration indicators.

Explicit code: With the exception of loops, your code will look exactly like the numbers you're trying to calculate.

Best quality: There are thousands of providers working to keep NumPy fast, friendly, and unobtrusive.

Because of these benefits, NumPy is a de facto level of multitasking in Python data science, and many popular libraries are built on it.

Neural Network:

The design of the neural network is based on the structure of the human brain. Just as we use our brain to identify patterns and classify different types of information, neural networks can be taught to perform similar functions in data.

Individual layers of neural networks can also be thought of as a type of filter that operates from the worst to the most hidden, which increases the chances of getting and producing the right result.

The human mind works in the same way. Whenever we receive new information, the brain tries to compare it with the known. The same concept is also used for deep neural networks.

Neural networks enable us to perform many tasks, such as merging, splitting, or reversing. With neural networks, we can collect or organize non-labeled data according to the similarities between the samples in this data. Or in the case of segregation, we can train the network in a labeled database to separate samples from this database into different categories.

IV. WORKING

DATASET: -

This dataset contains 9,312 images of different letters of the english alphabet.

And we use it to train our character recognition model.

Character formation patterns allow us to assign different characters on the basis of the characteristics shown in the training matrix above.

Like any other activity involving Data Science, Machine Learning, Computer Vision and Neural Network, this paper is divided into two phases:

Training

Testing and Posting

Steps to improve handwriting recognition

```
import matplotlib.pyplot as plt
```

```
import cv2
```

```
import numpy as np
```

```
from keras.models import Sequential
```

```
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
```

```
from keras.optimizers import SGD, Adam
```

```
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

```
from keras.utils import to_categorical
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
```

First, we do all the necessary importing mentioned above. We will see the use of all imports as we use them.

Read details:

We now read the data using `pd.read_csv ()` and print the first 10 images using `data.head (10)`.

Separate data from photos and labels:

```
X = data.drop ('0', axis = 1)
y = data ['0']
```

Separate information read in pictures and related labels. " 'Contains labels, so we omit column '0' from the read data file and use it in y to create labels.

Resetting data to csv file to display as image

In the section above, we separate data from training and testing data using `train_test_split ()`.

Also, we reconstructed the train data and image analysis to be displayed as an image, as in the original CSV file were present as 784 columns of pixel data. So we convert it into 28×28 pixels.

```
word_dict = {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G', 7: 'H', 8: 'I', 9: 'J', 10: 'K', 11: 'L', 12: 'M', 13: 'N', 14: 'O', 15: 'P', 16: 'Q', 17: 'R', 18: 'S', 19: 'T', 20: 'U', 21: 'V', 22: 'W', 23: 'X', 24: 'Y', 25: 'Z' }
```

All labels exist in the form of floating point values that convert them into whole values, so we create a `word_dict` dictionary to mark whole values and characters.

Editing the number of words in the database

Here we describe only the distribution of the alphabet.

First we convert the labels into whole values and then go into the calculation list according to the label. The checklist contains the number of pictures available in the database below each alphabet.

We now make an alphabetical list of all the letters using the `numbers ()` dictionary function.

We now use counters and alphabets to draw a horizontal bar structure.

Pushing data

Now we are mixing up some of the train set photos.

Shifting is done using a `mix function ()` so that we can display random images.

We then built 9 sites in 3×3 format and displayed restricted images of 9 alphabets.

Output for Character Recognition

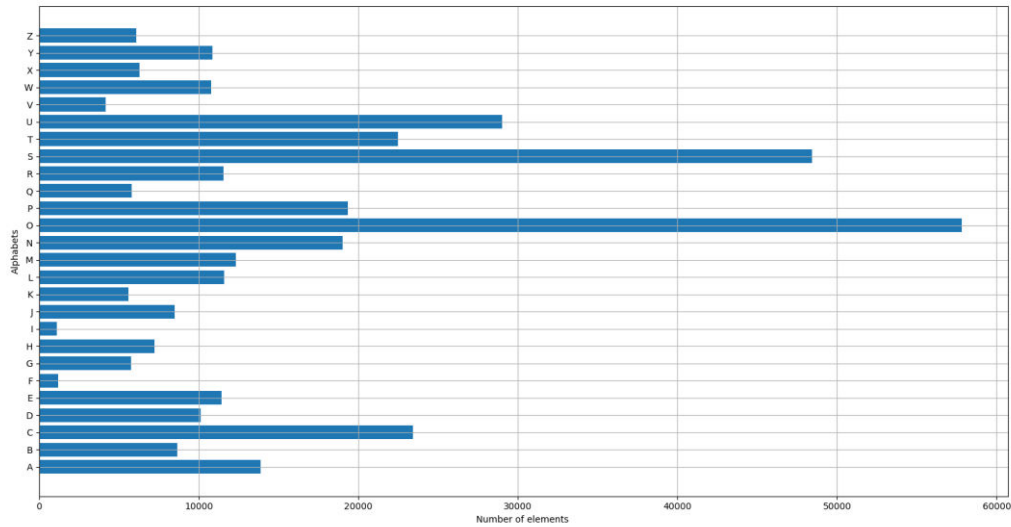


Figure 3: Plotting the number of alphabets in the dataset

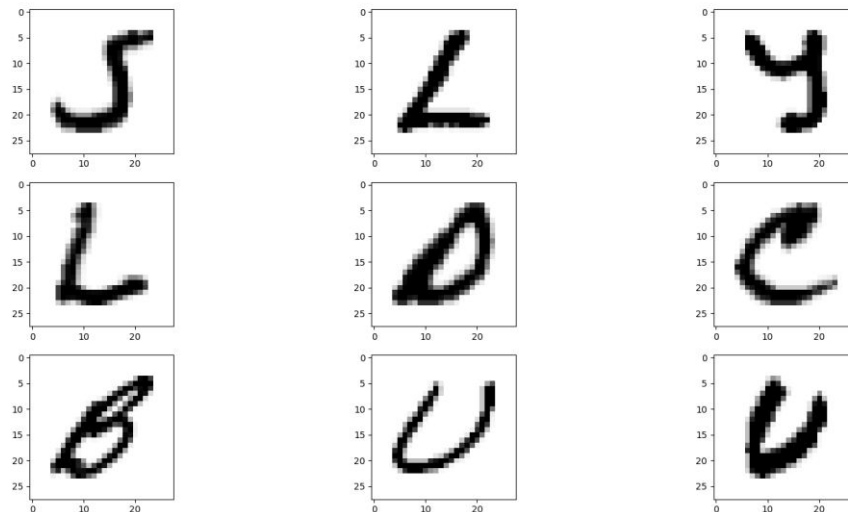


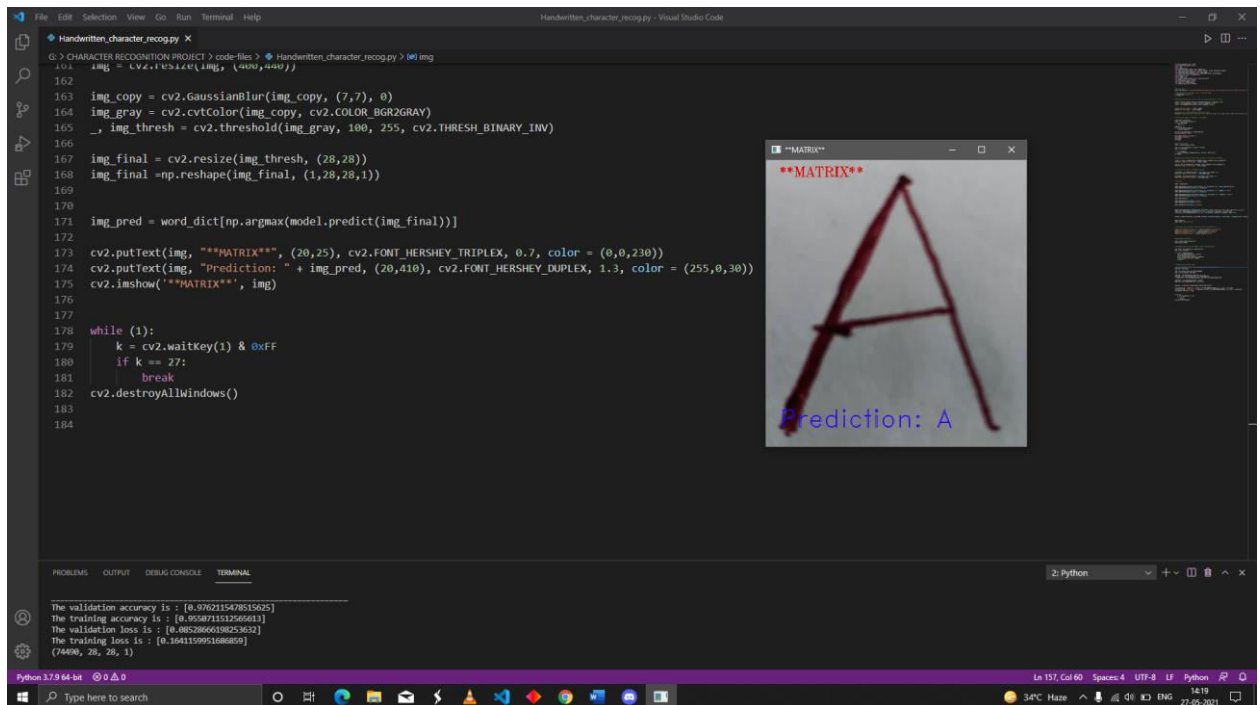
Figure 4: Displaying some images from the dataset

Epoch where all images are simultaneously processed by forwarding and reverting to the network, that is one Epoch

In this fig tree, we see that as the number of epochs increases the accuracy of the train also increases and the loss of training continues to decrease. And we can see that there are small signs of overcrowding, with the loss of validation less than the loss of training.

IV. RESULT

- Our research has an accuracy of 90%.
- It is detecting a single character at a time.



```

114 img = cv2.imread('A.jpg', cv2.IMREAD_GRAYSCALE)
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163 img_copy = cv2.GaussianBlur(img_copy, (7,7), 0)
164 img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)
165 _, img_thresh = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)
166
167 img_final = cv2.resize(img_thresh, (28,28))
168 img_final = np.reshape(img_final, (1,28,28,1))
169
170
171 img_pred = word_dict[np.argmax(model.predict(img_final))]
172
173 cv2.putText(img, "***MATRIX**", (20,25), cv2.FONT_HERSHEY_TRIPLEX, 0.7, color = (0,0,230))
174 cv2.putText(img, "Prediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (255,0,30))
175 cv2.imshow("***MATRIX**", img)
176
177
178 while (1):
179     k = cv2.waitKey(1) & 0xFF
180     if k == 27:
181         break
182 cv2.destroyAllWindows()
183
184
    
```

The validation accuracy is : [0.9702115478515025]
 The training accuracy is : [0.95871131266011]
 The validation loss is : [0.885296039826352]
 The training loss is : [0.1641159951686859]
 (7400, 28, 28, 1)

Figure 5 : On successful execution the code gives output

V. CONCLUSION

We have successfully developed handwriting recognition (Text Recognition) via Python, Tensorflow, and machine learning libraries.

Handwritten letters have been identified with more than 90% test accuracy.

This can also be extended to identify handwritten characters in other languages.

REFERENCES

- [1] P. Thompson, R. T. Batista-Navarro, G. Kontonatsios, J. Carter, E. Toon, J. McNaught, et al., "Text mining the history of medicine", PLoS ONE, vol. 11, no. 1, pp. 1-33, Jan. 2016.
- [2] K. D. Ashley and W. Bridewell, "Emerging AI & Law approaches to automating analysis and retrieval of electronically stored information in discovery proceedings", Artif. Intell. Law, vol. 18, no. 4, pp. 311-320, Dec. 2010.
- [3] C. C. Tappert, C. Y. Suen and T. Wakahara, "The state of the art in online handwriting recognition", IEEE Trans. Pattern Anal. Mach. Intell., vol. 12, pp. 787-808, Aug. 1990.
- [4] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi and F. Shafait, "High-performance OCR for printed English and Fraktur using LSTM networks", Proc. 12th Int. Conf. Document Anal. Recognit., pp. 683-687, Aug. 2013.
- [5] U. Bhattacharya and B. B. Chaudhuri, "Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals", IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 3, pp. 444-457, Mar. 2009.
- [6] A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network", Pattern Recognit., vol. 43, no. 7, pp. 2582-2589, Jul. 2010.



INNO  **SPACE**
SJIF Scientific Journal Impact Factor
Impact Factor: 7.542



ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 **9940 572 462**  **6381 907 438**  **ijircce@gmail.com**



www.ijircce.com

Scan to save the contact details