Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 4, Issue, 04, pp.155-161, April, 2012

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

ISSN: 0975-833X

RESEARCH ARTICLE

ONLINE /OFFLINE HANDWRITTEN GREGG SHORTHAND RECOGNITION USING CANN

Rajasekaran. R[1] and Dr. K. Ramar[*2]

[1]F X Polytechnic College, Tharuvai, Tirunelveli
[*2]Einstein College of Engineering, Tirunelveli

ARTICLE INFO

Article History:
Received 09th January, 2012
Received in revised form
04th February, 2012
Accepted 07th March, 2012
Published online 30th April, 2012

Key words:

Hand written,
Gregg Shorthand,
Competitive Artificial Neural Network,
Shorthand Script Recognition

ABSTRACT

The need to process documents on paper by computer has led to an area of research that may be referred to as Document Image Understanding (DIU).The goal of DIU system is to convert a raster image representation of a document. For Example, A hand written Gregg shorthand character or word is converted into an appropriate Symbolic Character in a computer (It may be a scanned character or online written character).Thus it involves many disciplines of computer science including image processing, pattern recognition, natural language processing, artificial intelligence, neural networks and database system. The ultimate goal of this paper is to recognize hand written Gregg shorthand character and Gregg shorthand word.

INTRODUCTION

We consider that it is necessary to convert the handwritten Gregg shorthand character or word in to its equivalent English word or character. In this paper, A Gregg character or Gregg word can be recognised and converted into its equivalent English character or word. In this paper a single shorthand character or word drawn into the text area can be digitalized and it can be learned or recognized. If learning mode is selected the computer learns the character, if it is a recognition mode then the computer Compares the drawn character with already stored patterns using CANN(Competitive Artificial Neural Networks) and displays the result in probabilistic method as well as characteristic method. There are 24 main characters and 19000 frequently used words in Gregg shorthand. In this paper various types of Gregg shorthand characters and words are tested and the results were presented. Our future proposal is to recognise and convert all those 19000 frequently used Gregg shorthand words. Recognition rate for online and offline Gregg shorthand was compared. Efficiency of offline is better over online. The rest of the paper is organized as follows: Section (2) focuses on Artificial Neural networks and Section (3) emphasizes on algorithm CANN for Gregg shorthand recognition and Section (4) for the Experimental results and conclusion.

II. ARTIFICIAL NEURAL NETWORK (ANN)

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, doesn't appear to regenerate. Because this type of cell is the only part of the body that isn't slowly replaced, it is assumed that these cells are what provide us with our abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as neurons. Each of these neurons can connect with up to 200,000 other neurons, although 1,000 to 10,000 are typical.

The power of the human mind comes from the sheer numbers of these basic components and the multiple connections between them. It also comes from genetic programming and learning. The individual neurons are complicated. They have a myriad of parts, sub-systems, and control mechanisms. They convey information via a host of electrochemical pathways. There are over one hundred different classes of neurons, depending on the classification method used. Together these neurons and their connections form a process which is not binary, not stable, and not synchronous. In short, it is nothing like the currently available electronic computers, or even artificial neural networks. These artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism. They do it in a primitive way. But for the software engineer who is trying to solve problems, neural computing was never about replicating human brains. It is about machines and a new way to solve problems.

On the contrary, neural network researchers are seeking an understanding of nature's capabilities for which people can engineer solutions to problems that have not been solved by traditional computing. To do this, the basic unit of neural networks, the artificial neurons, simulates the four basic functions of natural neurons. Figure 1 shows a fundamental representation of an artificial neuron.

**Corresponding author:* rajasekar2000in@rediffmail.com;

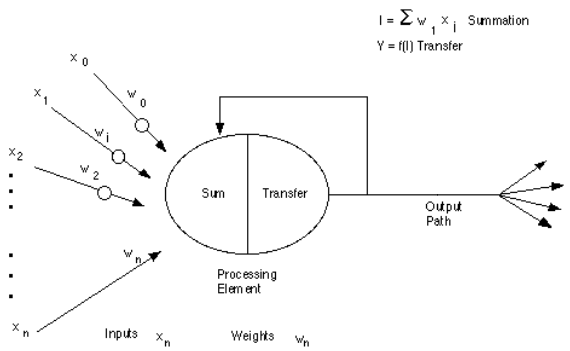

Figure 1 A Basic Artificial Neuron.

In Figure 1, various inputs to the network are represented by the mathematical symbol, x(n). Each of these inputs are multiplied by a connection weight. These weights are represented by w(n). In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output. This process lends itself to physical implementation on a large scale in a small package. This electronic implementation is still possible with other network structures which utilize different summing functions as well as different transfer functions. Some applications require "black and white," or binary, answers. These applications include the recognition of text, the identification of speech, and the image deciphering of scenes. These applications are required to turn real-world inputs into discrete values. These potential values are limited to some known set, like the ASCII characters or the most common 50,000 English words. Because of this limitation of output options, these applications don't always utilize networks composed of neurons that simply sum up, and thereby smooth, inputs. These networks may utilize the binary properties of ORing and ANDing of inputs. These functions, and many others, can be built into the summation and transfer functions of a network. Other networks work on problems where the resolutions are not just one of several known values. These networks need to be capable of an infinite number of responses. Applications of this type include the "intelligence" behind robotic movements. This "intelligence" processes inputs and then creates outputs which actually cause some device to move. That movement can span an infinite number of very precise motions. These networks do indeed want to smooth their inputs which, due to limitations of sensors, come in non-continuous bursts, say thirty times a second. To do that, they might accept these inputs, sum that data, and then produce an output by, for example, applying a hyperbolic tangent as a transfer functions. In this manner, output values from the network are continuous and satisfy more real world interfaces.

Other applications might simply sum and compare to a threshold, thereby producing one of two possible outputs, a zero or a one. Other functions scale the outputs to match the application, such as the values minus one and one. Some functions even integrate the input data over time, creating time-dependent networks. Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins. There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help. The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full blown sense of being truly self learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab.

Supervised Training

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined. The current commercial network development packages provide tools to monitor how well an artificial neural network is converging on the ability to predict the right answer. These tools allow the training process to go on for days, stopping only when the system reaches some statistically desired point, or accuracy. However, some networks never learn. This could be because the input data does not contain the specific information from which the desired output is derived. Networks also don't converge if there is not enough data to enable complete learning. Ideally, there should be enough data so that part of the data can be held back as a test. Many layered networks with multiple nodes are capable of memorizing data. To monitor the network to determine if the system is simply memorizing its data in some non significant way, supervised training needs to hold back a set of data to be used to test the system after it has undergone its training. (Note: memorization is avoided by not having too many processing elements.) If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs. Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back-propagation. These various learning techniques are explored in greater depth later in this report.

Yet, training is not just a technique. It involves a "feel," and conscious analysis, to insure that the network is not over trained. Initially, an artificial neural network configures itself with the general statistical trends of the data. Later, it continues to "learn" about other aspects of the data which may be spurious from a general viewpoint. When finally the system has been correctly trained, and no further learning is needed, the weights can, if desired, be "frozen." In some

systems this finalized network is then turned into hardware so that it can be fast. Other systems don't lock themselves in but continue to learn while in production use.

Unsupervised or Adaptive Training

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adoption. At the present time, unsupervised learning is not well understood. This adoption to the environment is the promise which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field. Yet, at the present time, the vast bulk of neural network work is in systems with supervised learning. Supervised learning is achieving results.

Supervised vs. Unsupervised learning

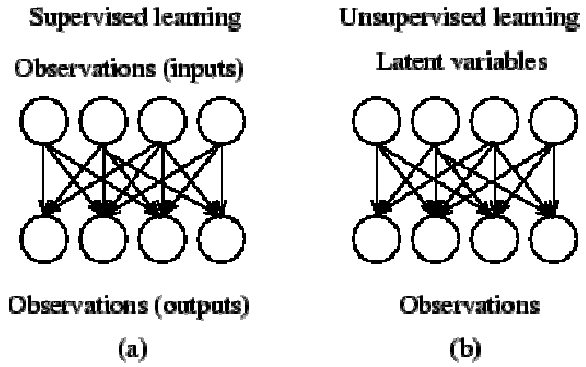

Figure 2: The causal structure of (a) supervised and (b) unsupervised learning.

In supervised learning, one set of observations, called inputs, is assumed to be the cause of another set of observations, called outputs, while in unsupervised learning all observations are assumed to be caused by a set of latent variables.From a theoretical point of view, supervised and unsupervised learning differ only in the causal structure of the model. In supervised learning, the model defines the effect one set of observations, called inputs, has on another set of observations, called outputs. In other words, the inputs are assumed to be at the beginning and outputs at the end of the causal chain. The models can include mediating variables between the inputs and outputs. In unsupervised learning, all the observations are assumed to be caused by latent variables, that is, the observations are assumed to be at the end of the causal chain. In practice, models for supervised learning often leave the probability for inputs undefined. This model is not needed as long as the inputs are available, but if some of the input values are missing, it is not possible to infer anything about the outputs. If the inputs are also modeled, then missing inputs cause no problem since they can be considered latent variables as in unsupervised learning. Figure 2 illustrates the difference in the causal structure of supervised and unsupervised learning. It is also possible to have a mixture of the two, where both input observations and latent variables are assumed to have caused the output observations.

This paper illustrates both supervised and unsupervised learning mechanisms of Gregg shorthand recognition

Existing System Vs CANN

The recognition of characters from scanned images of documents has been a problem that has received much attention in the fields of image processing, pattern recognition and artificial intelligence. Classical methods in pattern recognition do not as such suffice for the recognition of visual characters due to the following reasons:

1. The 'same' characters differ in sizes, shapes and styles from person to person and even from time to time with the same person.
2. Like any image, visual characters are subject to spoilage due to noise.
3. There are no hard-and-fast rules that define the appearance of a visual character. Hence rules need to be heuristically deduced from samples.

As such, the human system of vision is excellent in the sense of the following qualities:

1. The human brain is adaptive to minor changes and errors in visual patterns. Thus we are able to read the handwritings of many people despite different styles of writing.
2. The human vision system learns from experience: Hence we are able to grasp newer styles and scripts with amazingly high speed.
3. The human vision system is immune to most variations of size, aspect ratio, color, location and orientation of visual characters.

In contrast to limitations of classical computing, Artificial Neural Networks (ANNs), that were first developed in the mid 1900's serve for the emulation of human thinking in computation to a meager, yet appreciable extent. Of the several fields wherein they have been applied, *humanoid computing* in general and *pattern recognition* in particular have been of increasing activity. The recognition of visual (optical) characters is a problem of relatively amenable complexity when compared with greater challenges such as recognition of human faces. ANNs have enjoyed considerable success in this area due to their humanoid qualities such as adapting to changes and learning from prior experience. So we have decided to take this ANN for our project. The strategy used for recognition can be broadly classified into three categories, namely: structural, statistical and hybrid. Structural techniques use some qualitative measurements as features. They employ different methods such as rule-based, graph-theoretic and heuristic methods for classification. Statistical techniques use some quantitative measurements as features and an appropriate statistical method for recognition. In hybrid approach, these two techniques are combined at appropriate stages for representation of characters and utilizing them for recognition. Depending on the problem,

anyone of these techniques can be utilized while accommodating the variations in handwriting.

III. CANN ALGORITHM

This Paper uses a large (but simple) two-layer neural network to learn and recognize patterns. The hand-drawn Gregg image is digitized onto a grid of input neurons. Each possible answer is represented by a single output neuron. Every input neuron is linked directly to every output neuron (there are no hidden layers). As in most neural networks, the data (or programming) is encoded in the links between neurons. If a link between an input neuron and an output neuron is positive, that means that if the input is on then the total score for that output neuron is increased by a small amount. If the link is negative, then it follows that if that input is on, the corresponding output has its score reduced by an amount. The output neuron with the highest score (and thus the best match) is considered the winner. This is known as a competitive artificial neural network (CANN).

This paper's learning procedure is extremely simple. It is simple arithmetic.

1. All links between active input neurons and the selected output neuron have their weights increased by one.
2. All links between inactive inputs neurons and the selected output neuron have their weights decreased by one.

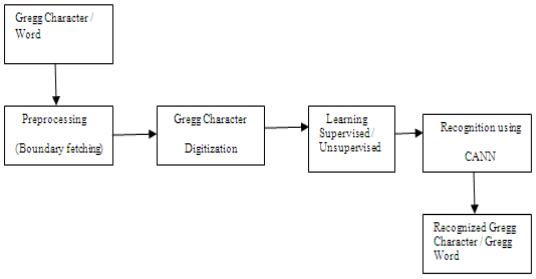

Figure 3 Gregg Shorthand recognition systems

Pre-processing

This process is the initial process. The handwritten parts are found, (from text area or scanned images or other picture file) separated, and transformed into matrices.

Gregg Character Digitization

The process of digitization is important for the neural network used in the system. In this process, the input image is sampled into a binary window which forms the input to the recognition system. In the above figure, the alphabet *A* has been digitized into *6X8=48* digital cells, each having a single color, either black or white. It becomes important for us to encode this information in a form meaningful to a computer. For this, we assign a value *+1* to each black pixel and *0* to each white pixel and create the binary image matrix *I* which is shown in the figure 4 So much of conversion is enough for neural networking which is described next. Digitization of an image into a binary matrix of specified dimensions makes the input image invariant of its actual dimensions. Hence an image of whatever size gets transformed into a binary matrix of fixed pre-determined dimensions. This establishes uniformity in the dimensions of the input and stored patterns as they move through the recognition system.

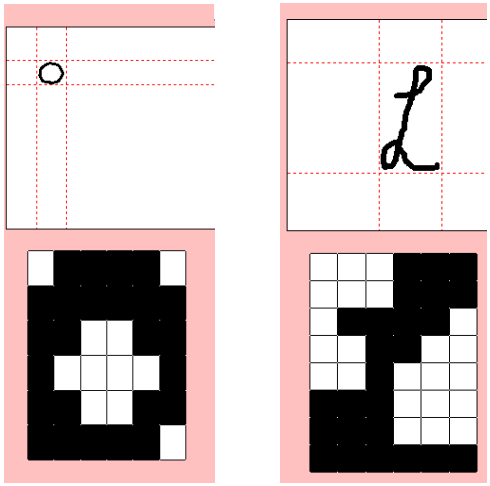

Figure 4. (a) (b)
(a) Gregg Character 'a' and its digitization
(b) Gregg word 'aback' and its digitization

Learning Mechanism

In the employed system, a highly simplified architecture of artificial neural networks is used. For purpose of easy understanding, the learning mechanism of the neural network is described first and its architecture is described next, in section [4.]. In the used method, various characters are *taught* to the network in a supervised manner. A character is presented to the system and is assigned a particular *label*. Several variant patterns of the same character are taught to the network under the same label. Hence the network learns various possible variations of a single pattern and becomes adaptive in nature. During the *training process*, the input to the neural network is the input matrix *G* defined as follows:

$If\ I(i,j) = 1\ Then\ G(i,j) = 1$
Else:
$If\ I(i,j) = 0\ Then\ G(i,j) = -1$

The input matrix *G* is now fed as input to the neural network. It is typical for any neural network to learn in a supervised or unsupervised manner by adjusting its *weights*. In the current method of learning, each candidate character taught to the network possesses a corresponding weight matrix. For the *kth* character to be taught to the network, the weight matrix is denoted by *Wk*. As learning of the character progresses, it is this weight matrix that is updated. At the commencement of teaching (supervised training), this matrix is initialized to zero. Whenever a character is to be taught to the network, an input pattern representing that character is submitted to the network. The network is then *instructed* to identify this pattern as, say, the *kth* character in a *knowledge base* of characters. That means that the pattern is assigned a label *k*. In accordance with this, the weight matrix *Wk* is updated in the following manner:

for all i=1 to x
{
for all j=1 to y
{

Wk(i,j)=Wk(i,j)+G(i,j)
}
}
Here *x* and *y* are the dimensions of the matrix *Wk* (and G).

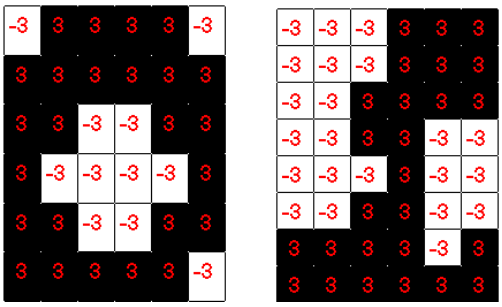

Figure 5 Weight Matrix of Digitized Character/word

Figure. 5 gives the weight matrix, say, '*Wa'* corresponding to the Gregg alphabet 'a'. and the corresponding weight matrix 'Waback' for the Gregg word 'aback' The matrix is has been updated thrice to learn the alphabet *a.* It should be noted that this matrix is specific to the alphabet *a* alone. Other characters shall each have a corresponding weight matrix.

Gregg Character Recognition

The overall architecture of the recognition system is shown below.

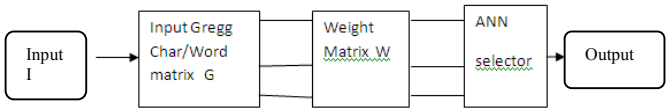

Figure 6 Architecture of Gregg Recognition system

In this system, the candidate pattern *I* is the input. The block '*G'* provides the input matrix *G* to the weight blocks *Wk* for each *k*. There are totally *n* weight blocks for the totally *n* characters to be taught (or already taught) to the system.

Digitized Score (ψ)

This statistic is a product of corresponding elements of the weight matrix *Wk* of the *kth* learnt pattern and an input pattern *I* as its *candidate*. It is formulated using the as follows:

$$\psi(k) = \sum_{i=1}^{x}\sum_{j=1}^{y} W_k(i,j) * I(i,j)$$

It should be noted that unlike in the training process where *M* was the processed input matrix, in the recognition process, the binary image matrix *I* is directly fed to the system for recognition.

Weight Score (μ)

This statistic simply gives the sum total of all the positive elements of the weight matrix of a learnt pattern. It may be formulated as follows (with m (*k*) initialized to *0* each time).

```
for i=1 to x
{
  for j=1 to y
  {
    if W_k(i,j) > 0 then
    {
      μ(k) = μ(k) + W_k(i,j)
    }
  }
}
```

Recognition Quotient (Q)

This statistic gives a measure of how well the recognition system identifies an input pattern as a *matching* candidate for one of its many learnt patterns. It is simply given by:

$$Q(k) = \frac{\psi(k)}{\mu(k)}$$

The greater the value of Q, the more confidence does the system bestow on the input pattern as being similar to a pattern already known to it. The classification of input patterns now follows the following trivial procedure:-

1. For an input candidate pattern *I*, calculate the recognition quotient (*Q*(*k*)) for each learnt pattern *k*.
2. Determine the value of *k* for which *Q*(*k*) has the maximum value.
3. Too low maximum value of *Q*(*k*) (say less than 0.5) indicates poor recognition. In such a case:
   - Conclude that the candidate pattern does not exist within the knowledge base OR
   - Teach the candidate pattern to the network till a satisfactory value of *Q*(*k*) is obtained.
4. Conditionally, identify the input candidate pattern as being akin to the *kth* learnt pattern OR proceed with the training for better performance.

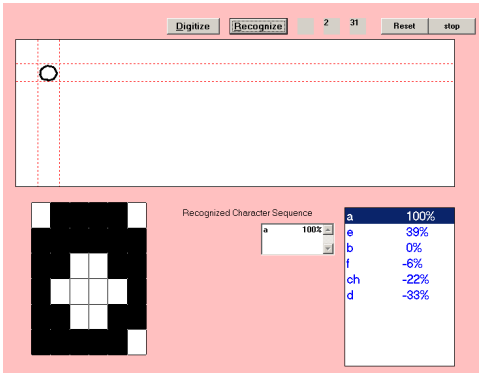

Figure 7. Recognition of Gregg character 'a"

In the above figure the selector gives an output *k* by making the best selection as in Step 4 of the mentioned algorithm. The adaptive performance of the network can easily be tested by an example: Gregg character 'a', and by giving repeated learning to the neural network we can get up to 100% recognition ratio.

Performance Analysis

The neural system has some direct advantages that become apparent at this stage:

1. The method is highly adaptive; recognition is tolerant to minor errors and changes in patterns.
2. The knowledge base of the system can be modified by teaching it newer characters or teaching different variants of earlier characters.
3. The system is highly general and is invariant to size and aspect ratio.
4. The system can be made user specific: User-profiles of characters can be maintained, and the system can be made to recognize them as per the orientation of the user.

The dimensions of the input matrix need to be adjusted for performance. Greater the dimensions, higher the resolution and better the recognition. This however Increase the time-complexity of the system which can be a sensitive issue with slower computers. Typically, *32X32* matrices have been empirically found sufficient for the recognition of Gregg handwritten characters. For Intricate scripts, greater resolution of the matrices is required. As already illustrated in the previous example, efficient supervised teaching is essential for the proper performance. Neural expert systems are therefore typically used where human-centered training is preferred against rigid and inflexible system-rules.

EXPERIMENTAL RESULTS & CONCLUSIONS

24 Gregg alphabet characters and 3 sample frequently used words were taken for this experiment.

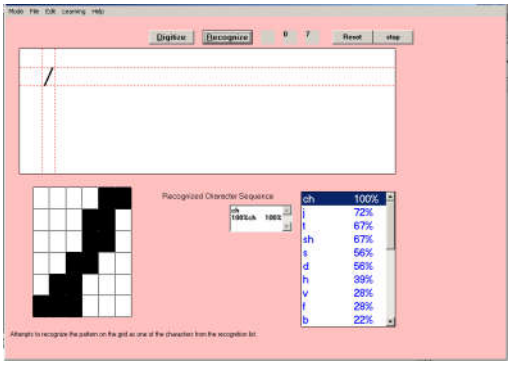

Figure 8. Recognition of Gregg Character "ch"

Table 1.Comparision between recognition of offline & online Gregg alphabet characters

GREGG ALPHABET SYMBOL	ENGLISH ALPHABET	OFFLINE RECOGNITION TIME IN ms	RECOGNITION QUOTIENT IN %	ONLINE RECOGNITION TIME IN ms	RECOGNITION QUOTIENT IN %
	b	5	100	5	83
	ch	5	100	5	72
	d	7	100	8	72
	e	6	100	7	61
	f	6	100	6	83
	g	6	100	6	94
	h	7	100	7	67
	i	6	100	7	61
	j	6	100	6	78
	k	4	100	6	100
	l	7	100	5	78
	m	6	100	5	68
	n	8	100	6	56
	ng	6	100	6	67
	o	7	100	7	67
	p	6	100	6	78
	r	7	100	7	79
	s	8	100	6	66
	sh	6	100	6	83
	t	7	100	7	72
	th	6	100	7	72

	u	6	100	8	68
	v	7	100	7	67

Findings:-

1. Recognition time of Gregg characters or Gregg word for both online or offline is more or less same.
2. Algorithm gives 100% accuracy for offline characters or words
3. Algorithm gives an average result of 73% recognition quotient Q accuracy for online characters or words.
4. If the user drawn character size varies Recognition quotient Q for online characters also varies.
5. Online recognition accuracy may come to 100% by giving proper learning either supervised or unsupervised to the network.

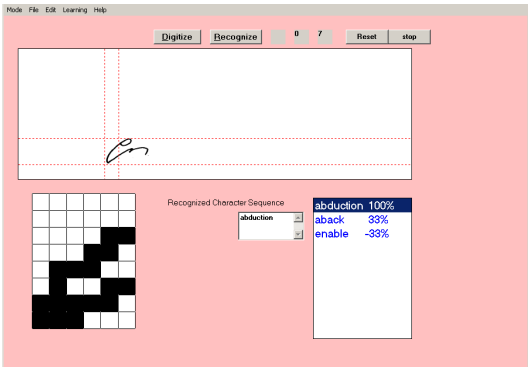

Figure 9. Recognition of Gregg word "abduction"

Table 2. Comparision between recognition of offline & online Gregg frequently used words

GREGG ALPHABET SYMBOL	ENGLISH WORD	OFFLINE RECOGNITION	RECOGNITION QUOTIENT IN %	ONLINE RECOGNITION	RECOGNITION QUOTIENT IN %
	abduction	7	100	6	79
	enable	6	100	6	78
	aback	8	100	9	64

A simplistic approach for recognition of Gregg visual characters using artificial neural networks has been described. The advantages of neural computing over classical methods have been outlined. Despite the computational complexity involved, artificial neural networks offer several advantages in pattern recognition & classification in the sense of emulating adaptive human intelligence to a small extent. Our Future work is to recognise rest of 19000 frequently used words of Gregg shorthand

REFERENCES

[1] Anil K. Jain, Jianchang Mao, K. M. Mohiuddin, *Artificial Neural Networks: A Tutorial, Computer,* v.29 n.3, p.31-44, March 1996

[2] Simon Haykin, Neural Networks: A *comprehensive foundation, 2nd* Edition, Prentice Hall, 1998

[3] Alexander J. Faaborg, Using Neural *Networks to Create an Adaptive Character Recognition System,* March 2002, available at: http://web. media.mit.edu/~faabor /research/cornell/hci_neuralnet work_finalPaper.pdf

[4] E. W. Brown, Character Recognition *by Feature Point Extraction,* unpublished paper authored at Northeastern University, 1992, available at: http://www.ccs.neu.edu/home/feneric/charrecnn.html

[5] Dori, Dov, and Alfred Bruckstein, ed. *Shape, Structure and Pattern Recognition*. New Jersey: World Scientific Publishing Co., 1995.

[6] Gorsky, N.D. "Off-line Recognition of Bad Quality Handwritten Words Using Prototypes." *Fundamentals in Handwriting Recognition*. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[7] Impedovo, Sebastiano. "Frontiers in Handwriting Recognition." *Fundamentals in Handwriting Recognition*. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[8] Licolinet, Eric, and Olivier Baret. "Cursive Word Recognition: Methods and Strategies." *Fundamentals in Handwriting Recognition"*. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[9] Simon, J.C. "On the Robustness of Recognition of Degraded Line Images." *Fundamentals in Handwriting Recognition*". Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[10] Wang, Patrick Shen-Pei. "Learning, Representation, Understanding and Recognition of Words - An Intelligent Approach." *Fundamentals in Handwriting Recognition*. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[11] Yaeger, Larry S., Brandyn J. Webb, and Richard F. Lyon. "*Combining Neural Networks and Context-Driven Search for Online, Printed Handwriting Recognition in the Newton.*" *A.I. Magazine*. 19(1): 73-89, 1998 Spring.

[12] Young, Tzay Y., and King-Sun Fu, ed. *Handbook of Pattern Recognition and Image Processing*. New York: Academic Press, Inc., 1996.

[13] *Visual Character Recognition using Artificial Neural Networks* by *Shashank Araokar*

[14] *A knowledge-based approach for recognition of handwritten Pitman shorthand language strokes* by P NAGABHUSHAN and BASAVARAJ S ANAMI.