On the Application of the Nearest Neighbour Classifier Algorithm to Character Pattern Recognition

Bedole Omolu
% Department of Computer Science & Mathematics,
Lagos State University, Lagos,
Nigeria.
Email: bedolfomolu@yahoo.co.uk
&
Bamidele Oluwade
Department of Computer Science, Faculty of Communication and Information Sciences,
University of Ilorin, Ilorin, Kwara State,
Nigeria.
Email: deleoluwade@yahoo.com

ABSTRACT
The Nearest Neighbour Classifier algorithm and Back-propagation algorithm are two well-known learning algorithms in neural networks and artificial intelligence. In this paper, an experimental application of the nearest-neighbour classifier algorithm to character pattern recognition is carried out. This algorithm is chosen because of its relative ease of usage and its adaptability to image processing applications. The system is trained to recognize patterns of letters of the English alphabet (A–Z). Lots of example patterns representing each letter are extracted and memorized by the system. When the system is presented with an input pattern representing a letter, it classifies the pattern into the appropriate category based on the memorized patterns via Hamming distance metric. The system also computes in percentage, the degree of accuracy involved in the classification. The system achieved a high level of accuracy (over 90%, up to 100%) in recognizing the input patterns. The system makes use of GUI (Graphical User Interface), and as such, Microsoft Visual Basic 6.0 was employed as the implementation language for the solution. The system memory was implemented in a database using Microsoft Access.

Keywords: Character Pattern Recognition, English alphabet, Nearest Neighbour Classifier algorithm, Hamming Distance, Recognition Accuracy.

African Journal of Computing & ICT Reference Format:

Bedole Omolu & Bamidele Oluwade (2017), On the Application of the Nearest Neighbour Classifier Algorithm to Character Pattern Recognition,
Afr. J. Comp. & ICT, Vol.10, Nos. 1& 2, pp. 1 - 17.

I. INTRODUCTION

The Nearest Neighbour Classifier algorithm and Back-propagation algorithm are two well-known learning algorithms in neural networks and artificial intelligence. An (artificial) neural network is simply, a massive (large number of) interconnected network of simple processing elements (called neurons), that has a natural propensity for storing experiential knowledge and making it available for use. Neural network, also referred to as neuro-computer, connectionist network, parallel distributed processor etc, is an information processing paradigm that is inspired by the way biological systems, such as the brain, process information. The key element of this paradigm is the novel structure of the neural network, which is composed of neurons working in unison to solve specific problems. An artificial neural network learns by example, and is usually configured for a specific application such as pattern recognition, sales forecasting, industrial process control, etc. [**1, 2**]. Tasks for which neural networks are useful include: pattern classification, clustering/grouping, function approximation, signature analysis, prediction/forecasting, optimization, control, monitoring, and content-addressable memory (or associative memory).

There are several learning rules, including Error-correction learning, Hebbian learning, Competitive learning, and Boltzmann Learning [**2**]. Also, there are several learning algorithms utilizing the learning rules. The Back-Propagation learning algorithm is one of the most general-purpose, and commonly used neural network system learning algorithms. The back-propagation algorithm achieves its generality because of the gradient-descent technique used to train the network. However, this algorithm is seemingly more sophisticated in nature, compared to the Nearest Neighbor Algorithm which appears relatively easier to use [**3**]. Hence the Nearest Neighbor Classifier Algorithm is experimented with in this paper.

The task of pattern classification is to assign an input pattern (like a speech waveform, or handwritten symbol) represented by a feature vector to one of many pre-specified classes. Well-known applications include character recognition, speech recognition, blood cell classification, etc. In clustering (also known as unsupervised pattern classification), there are no training data with known class labels. A clustering algorithm explores the similarity between the patterns and places similar patterns in a cluster (group). Well-known clustering applications include data mining, data compression, and exploratory data analysis.

In this paper, an experimental application of the nearest-neighbour classifier algorithm to character pattern recognition is carried out. Pattern recognition refers to the assignment of patterns to their respective classes. It involves the categorization of input data into identification classes via extraction of significant features or attributes of the data from a background of irrelevant detail. A pattern recognition system receives data in form of raw measurement which collectively form a stimuli vector. An example is the problem of recognizing the patterns of letters of the English alphabet. A pattern recognizer must be able to learn. Its basic task is to learn general concepts on the basis of specific examples.

The Nearest Neighbour Classifier algorithm relies on distance metrics [**4, 5**]. There are a plethora of metrics for determining the similarity between input patterns (vectors). Commonly used measures include: Euclidean distance, Dot Product (or Inner Product), City Block distance, Chessboard distance, and Hamming distance [**2, 6**]. The Hamming distance measure is one of the most popular and simplest techniques for computing pattern similarity in this application, and is hereby used. The primary purpose of the present paper is to provide an experimental validation of the Nearest Neighbour Classifier algorithm via the use of Hamming distance. Although the Hamming distance metric is relatively an old metric, it still achieves useful results for basic recognition tasks.

Given two binary patterns (vectors) $\mathbf{x}_i$ and $\mathbf{x}_j$, the Hamming distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed as follows [**7**]:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left| \sum x_i - x_j \right| \quad (1)$$

When the system is presented with an input pattern during the classification phase, the system takes the feature vector extracted from the pattern and computes its Hamming distance for each of the patterns stored in the database, the smallest distance giving the best match. The Hamming distance metric measures the number of bit positions in which two binary patterns differ. The degree of accuracy involved in classification is the ratio of the number of bit positions in which two patterns are similar to the total number of bit positions, expressed as

percentage. Thus if the total number of bit positions is N, and the Hamming distance is K, then

$$\text{Percentage Accuracy} = \frac{(N - K)}{N} \times 100\%$$

The training phase of the Nearest-Neighbour Classifier algorithm consists only of storing the feature vectors and class labels of the training samples. In the actual classification phase, the same feature as before is computed for the test sample whose class is unknown. Given the feature vector gotten from the unknown pattern, its distance to each of the patterns in the database is computed, the smallest distance giving the best match. The system is trained to recognize patterns of letters of the English alphabet (A–O). Lots of example patterns representing each letter are extracted and memorized by the system. When the system is presented with an input pattern representing a letter, it classifies the pattern into the appropriate category based on the memorized patterns via the use of Hamming distance metric. The system also computes in percentage, the degree of accuracy involved in the classification. The trained patterns are shown in Figure 1.

To design a system for pattern recognition of letters A-Z, or subsets of these letters, each letter is coded as a pattern of 0's and 1's on a grid, which might be, say 15x10, 128x128, or 8x8. In this paper, a grid of 8x8 is used, because an 8-bit code similar to the 8-bit EBCDIC (Extended Binary Coded Decimal Interchange Code) is envisaged. Although the use of a higher dimensional grid increases the resolution of the system, it however requires large training set and large amount of memory for storing the training patterns. A smaller dimensional grid, like the 8x8 grid, offers poor resolution of the input patterns but has the following advantages:

(i) Smaller training set is required to learn to distinguish objects.

(ii) The amount of memory required for storing the training patterns is reduced.

The basic procedure in the system design involves the following:

1. Collection of lots of example input patterns for each letter.

2. An appropriate architecture is selected for the system, with an input layer consisting of source nodes equal in number to the pixels of an input image (64 in this paper), and an output layer consisting of neurons equal in number to the number of class categories (26 in this paper).

3. Splitting the examples set into two subsets: a training set, and a test set. The training set is used to train the network. This phase of the network design is called learning. In training the network, the desired response is defined by the "identity" of the particular letter whose image is presented to the network as the input signal.

4. The recognition performance of the trained network is tested using the test set.

Areas of applications of pattern recognition system include seismic data analysis, optical character recognition, signature verification, fingerprint identification, person identification, recognition of chemical structures etc [**3, 8, 9**].

II. PATTERN RECOGNITION PROCESS AND NEAREST NEIGHBOUR CLASSIFIER ALGORITHM

The pattern recognition process involves the following sequential phases, as depicted in Figure 2, namely:

(i) Filtering: This refers to removal of unwanted information from the input. This is usually aided using a sensor such as a digital camera, microphone or thermometer.
(ii) Feature Extraction: This is a process of studying and deriving useful information from the filtered input pattern. It is carried out using a feature extractor.
(iii) Classification: Here, the input pattern is assigned to a particular category. Classifiers typically rely on distance metrics and probability theory. Classification is accomplished using a classifier.

Basically, there are four (4) approaches to pattern recognition. These are statistical approach, structural approach, syntactic approach and neural network approach [**3**]. The statistical approach is based on conditional probability. That is, for pattern classification, measurements of the input patterns (components of the feature vector) is taken and an estimate of the likelihood or probability of the pattern belonging to a particular

class. In structural approach, recognition is by graph matching i.e. classes are represented by graphs or similar data structures like trees, arrays, matrices etc.

In the syntactic approach, classes are represented by grammars which have been constructed from pattern primitives such as symbols and terminals. Recognition is done by parsing the input to see if it can be generated by the grammars. Finally, in the neural network approach, the features extracted from the pattern is taken and used as input to a neural network. This approach has the advantage that neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.

This adaptability and learning characteristics enable neural network models to deal with imprecise data and ill-defined situations. Thus, a suitably trained neural network has the ability to generalize inputs even if they do not appear in the training data. Also, neural networks have good fault-tolerance. That is, since there are many neurons (processing elements) linked together, failures of a few neurons would not greatly affect the overall performance of the system. The procedure used in this paper is along the line of the neural network approach.

The back-propagation algorithm used in [**10**] is known to be the most widely applied paradigm for training neural network systems. However, the back-propagation algorithm is not commonly used for image processing applications [**11**]. This is because the neural network system tends to be overly sensitive to the position and scale of the object in the image, as shown in Figure 3.

This paper therefore presents a neural network-based system using the Nearest Neighbour Classifier algorithm [**12, 13**]. This algorithm is relatively simpler**,** and can be used to build a position-independent and scale-independent network. The training phase of the Nearest-Neighbour Classifier algorithm consists only of storing the feature vectors and class labels of the training samples. In the actual classification phase, the same feature as before is extracted from the test sample whose class is unknown.

The algorithm used for the character recognition system based on the Nearest-Neighbour Classifier algorithm is as follows:

Step 0: Start.
Step 1: Collect lots of sample patterns of each letter to be recognized by the system.
Step 2: Use a subset of the sample set to train the system.
Step 3: The system extracts and memorizes the training patterns and their associated classes.
Step 4: Determine the appropriate data representation scheme to be used in modeling the patterns.
Step 5: Storage of each memorized pattern in the database as a vector.
Step 6: The other subset of the sample patterns are used to test the recognition performance of the system
Step 7: Use the Hamming Distance metric to generalize distorted input patterns by computing the similarity between the input pattern (vector) and each of the stored patterns. The input pattern is assigned to the class of the pattern that is most similar to the input pattern. That is, if the system is presented with a pattern (vector) not seen during training, it computes the Hamming distance of the pattern for each of the stored pattern and assigns the input pattern to the class of the stored pattern with the least Hamming distance.
Step 8: Stop.

III. METHODOLOGY

The following is the pseudocode used in implementing the system.

```
//Training phase: store input patterns and their associated
class category
repeat
   get inputPattern and associatedPatternClass
   store inputPattern and associatedPatternClass in the
database
until there are no more training example patterns in the
training set

//Recognition phase: get an input pattern and
//assign it to one of the memorised (stored) class
categories
//and compute the percentage accuracy in the
classification
get inputPattern
for i = 1 to numberOfDatabaseRow
   if inputPattern = storedPattern(i) then
      {
         inputPatternClass = storedPatternClass(i)
         percentageAccuracy = 100
      }
   else
      {
```

```
minDistance = 64
for j = 1 to numberOfDatabaseRow
   {
      compute dist(j) = hammingDist(inputPattern,
storedPattern(j))
      if dist(j) < minDistance then
         {
            minDistance = dist(j)
            inputPatternClass = storedPatternClass(j)
         }
      percentageAccuracy = 100*(64-dist(j))/64
   }
}
```

The design of the solution is discussed under the following subheadings: input design, output design, memory design, and process design.

INPUT DESIGN

The input interface for the system consists of an 8x8 grid where the training pattern is drawn, and a text box where the associated class is entered. Each pixel (square) of the grid represents an input node (neuron), so there are 64 input neurons altogether. Each node can be in either state 1 when the pixel is black, or state 0 when the pixel is white.

OUTPUT DESIGN

The output interface also consists of an 8x8 grid where the pattern to be classified is entered. Each pixel can be in either state 1 when the pixel is black, or state 0 when the pixel is white. On the right side of the output interface is the system response frame showing the class category of the recognized input pattern and the percentage accuracy in the classification.

MEMORY DESIGN

The system memory is implemented in a database using Microsoft Access. When an input pattern (vector) is presented to the system during the training phase, the pattern is stored in the database as a string of 0's and 1's. Each pattern of 0's and 1's is associated with a class category. A class is analogous to a set and can be associated with multiple patterns representing members of the set.

PROCESS DESIGN

This is the description of the system functionality. In the training phase of the system, several input patterns (and their associated class category) are presented to the system and are subsequently stored in the database. In the recognition phase, the system is presented with an input pattern and will assign the pattern to one of the stored class categories. To assign an input pattern to a class, the pattern is compared with each stored pattern. If the input pattern matches one of the stored patterns, it is assigned to the class of that stored pattern, and the percentage accuracy is 100; if the input pattern does not match any of the stored patterns, its similarity to each stored pattern is computed and the input pattern is assigned to the class of the most similar stored pattern. The Hamming distance measure is the best technique for computing pattern similarity in this application [**11**]. The percentage accuracy is computed as the ratio of the number of similar positions to the total number of positions (i.e. 64), expressed as a percentage.

The system is trained (in a supervised manner) with a subset of patterns of the English alphabet shown in Figure 1. Results obtained (Figure 4) from the application of the system to the patterns indicate that the system performs well. The Hamming distance metric is also shown to be effective in the classification of distorted patterns in the test set. As can be seen in Figure 3, an input pattern can be thought of as a two-dimensional grid of pixels. In its simplest form, the colour of a pixel is a binary indicator of the illumination status of the pixel. A black colour could be interpreted as logic 1, while a white colour is interpreted as logic 0. For images more complex than simple character patterns, a gray-shade (or colour) pixel representation is often needed, but that requires additional information about each pixel. For this application, binary information will be sufficient to construct patterns of all the characters which the system will ever process. Since there are eight rows, an 8-bit system similar to the 8-bit EBCDIC is used. The representation of the input pattern as 8-bit words is depicted in Table 1. A technique for representing bits in compressed form, and its application to various n-bit words can be found in [**14, 15, 16**].

So going row-wise, the image in Figure 3(a) is represented as 00000000 00000000 11100000 10100000 11100000 10100000 10100000 00000000. By concatenating the eight row vectors of pixels in Figure 3(a), which comprise the character matrix, into a single 64-bit vector, a pattern that is suitable for input to an associative-memory neural system is formed. Moreover, by considering the input as a vector, the network has been provided with the inherent ability to compare patterns of new characters with patterns of characters it has already learnt to recognize.

IV. IMPLEMENTATION AND RESULTS

The system makes use of GUI (Graphical User Interface), and as such, Microsoft Visual Basic 6.0 is adopted for implementing the solution.

Visual Basic offers the following advantages:

1. It allows creation of GUI (Graphical User Interface) applications.

2. It allows quick and easy creation of Window-based applications.

Thus, using Visual Basic facilitates the system implementation since it has the tools for the GUI components. Visual Basic provides us with the flexibility of:

1. Creating the GUI by drawing objects such as the input grid, and buttons.

2. Setting the properties of these objects to refine their appearances.

3. Making the GUI components respond to user events by attaching code to them.

The system memory is simulated in Microsoft Access database (See Appendix). Although one could use a simple file to store the data, this would require you to design separate routines to insert data into the file, delete data from the file, etc. By using a database to control all these actions, the designer can concentrate on what the application will do rather than how to access the data. This also reduces the amount of code to be written. Visual Basic makes it easier to access databases by providing several data controls and objects that assist designers in the process. Using a database also offers the advantage of simultaneous universal access to the system.

SYSTEM DESCRIPTION
The system is highly interactive and menu-driven. On starting the system, the main menu comes up (See Appendix). The main menu has three buttons: Training Mode, Use Mode, and Database View.
1. Training Mode: On clicking this button, the user is presented with the "Training Interface" as shown.

2. Use Mode: On clicking this button, the user is presented with the "Classification Interface" shown.

3. Database View: On clicking this button, the user is presented with the system memory view interface shown**.**

TRAINING INTERFACE
The training interface (see Appendix) is where the user trains the network with different patterns. It has an 8x8 grid, a text box and three buttons. Each pattern to be learnt by the network is drawn on the grid using the mouse, and the associated class is typed in the text box. On clicking the "Learn Pattern" button, the system stores in the database, the pattern and the associated class. Clicking the "Refresh" button clears the grid of the drawn pattern. Clicking the "Close" button closes the training interface and takes the user back to the main menu.

CLASSIFICATION INTERFACE
The classification interface (see Appendix) is where the system is presented with a pattern for classification. This interface has an 8x8 grid, three buttons, and the "System Response" frame that contains the "Pattern Class" and "Accuracy" labels. The pattern to be classified is drawn on the left grid using the mouse. On clicking the "Classify Pattern" button, the system produces a response by displaying in the system response frame, the class associated with the pattern, and the accuracy involved. Clicking the "Refresh" button clears the grid of the drawn pattern. Clicking the "Close" button closes the classification interface and takes the user back to the main menu.

SYSTEM MEMORY VIEW INTERFACE
This interface (see Appendix**)** enables the system user to view the content of the system memory. The interface has three buttons: "Delete Current Row", "Delete All Rows", and "Close".

Clicking the "Delete Current Row" button deletes a selected row from the database. Clicking the "Delete All Rows" button clears the database of all its contents. Clicking the "Close" button closes the interface and takes the user back to the main menu. The first 10 rows of the entire 65 columns of the system memory is as shown in the Appendix.

TRAINING DATA
The system was trained to recognise the character patterns for the uppercase letters of (subset of) the English alphabet (A–Z). If presented with any of the patterns, the

system classifies it into the appropriate category with 100% accuracy. This is irrespective of the scale and the axes. However, the system can classify noisy patterns to one of the stored pattern classes, as well as computing the degree of accuracy (i.e similarity to the nearest neighbour).

V. DISCUSSION AND CONCLUSION

In this paper, an application of the Nearest Neighbour Classifier algorithm to character pattern recognition system has been presented. It has also been shown how the Hamming distance metric is used to generalize distorted patterns. The test case is the characters of the English alphabet. The generalization ability of the system is tested by presenting the system with patterns not seen during the training phase. The Nearest Neighbour Classifier algorithm offers the advantage of "one-shot" learning, against the iterative manner in which the Back-Propagation algorithm is employed. Although the Hamming distance metric is relatively an old metric, it still achieves useful results for basic recognition tasks.

As can be seen from Figure 4, two As were successfully identified with accuracy 100% and 98.4% respectively. Both characters C and F were also identified 100%. The characters B, D and E recorded high recognition rates of 96.9%, 96.9% and 98.4% respectively. The programming language used for the Graphical User Interface-based system was Microsoft Visual Basic while the system memory was implemented in a database using Microsoft Access.

The input image to the system is drawn on a grid of 8x8 pixels. Though, using a higher dimensional grid increases the resolution of the system, it requires large training set and large amount of memory for storing the training patterns. In a real-life application, the system could be constrained in such a manner that if the percentage accuracy is above a certain minimum value, then the response is accepted; otherwise, it is rejected. Further work may involve application of other metrics to the given characters and a comparative analysis of the accuracies obtained with those recorded using the Hamming distance metric.

ACKNOWLEDGEMENT

This paper is a revised part of an M.Sc. Dissertation earlier written by the first author under the supervision of the second author [9]. The authors will like to thank Professor Leslie Smith for his critical comments on an earlier version of the paper. He is of the Centre for Cognitive & Computational Neuroscience, Department of Computing Science and Mathematics, University of Stirling, Scotland.

REFERENCES

[1]. Phillips, Winfred (2000), The Extraordinary Future, Consortium on Cognitive Science Instruction (The Mind Project), http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=247 <last accessed in 2017>

[2]. Haykin, Simon (1994), Neural Networks-a Comprehensive Foundation, Macmillan College Publishing Company Inc.

[3]. Tveter, Donald (1998), The Pattern Recognition Basis of Artificial Intelligence, Wiley-IEEE Computer Society.

[4]. Alkasassbeh, Mouhammd; Altarawneh, Ghada A. and Hassanat, Ahmad B. (2015), On Enhancing the Performance of Nearest Neighbour Classifiers using Hassanat Distance Metric, Canadian Journal of Pure and Applied Sciences (CJPAS), Vol. 9, Issue 1, pp. 3291-3298.

[5]. Prasath, V. B. Surya; Alfeilat, Haneen Arafat Abu; Lasassmeh, Omar and Hassanat, Ahmad B. A. (2017), Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier – A Review', https://arxiv.org/pdf/1708.04321.pdf

[6]. Fisher, R; Perkins, S; Walker, A. and Wolfart, E. (2003), Glossary-Distance Metrics, http://homepages.inf.ed.ac.uk/rbf/HIPR2/metric.htm <last accessed in 2016>

[7]. Berlekamp, E. R. (1968), Algebraic Coding Theory, McGraw-Hill Book Company, New York.

[8]. Olszewski, Robert T. (2001), Generalized Feature Eextraction for Structural Pattern Recognition in Time Series Data, Ph.D. Thesis, Computer Science Department,

Carnegie Mellon University; http://reports-archive.adm.cs.cmu.edu/anon/2001/CMU-CS-01-108.pdf < last accessed in 2017>

[9]. Omolu, Bedole (2007), Application of Neural Encoding to a Two-Dimensional Character Pattern Recognition System, M.Sc. Dissertation, Department of Computer Science, University of Ibadan, Nigeria.

[10]. Ojo, A. K. and Sofoluwe, A. B. (2003), A Backprop Variant and some Practical Applications, Journal of Computer Science & Its Applications (Journal of the Nigeria Computer Society), Vol. 9, No. 1, pp. 82-98.

[11]. Skapura, David M. (1996), Building Neural Networks, ACM Press.

[12]. Kataria, A. and Sigh, M. D. (2013), A Review of Data Classification using K-Nearest Neighbour Algorithm, International Journal of Emerging Technology and Advanced Engineering, 3(6), pp. 354-360.

[13]. Moise, Izabela; Pournaras, Evangelos; and Helbing, Dirk (2015), K-Nearest Neighbour Classifier, https://www.ethz.ch/content/dam/ethz/special-interest/gess/computational-social-science-dam/documents/education/Spring2015/datascience/k-Nearest-Neighbour-Classifier.pdf < last accessed in 2017>

[14]. Oluwade, Bamidele ('Dele) (2000), A Novel Groupoid Code, Journal of Science Research (Journal of the Faculty of Science, University of Ibadan, Nigeria), Vol. 6, No. 1, pp. 1-3.

[15]. Oluwade, Bamidele ('Dele) (2009), A Binary Text Compression Algorithm based on Partitioning, Advances in Computer Science and Engineering, Vol. 3, No. 2, pp. 165-174.

[16]. Oluwade, Bamidele ('Dele) (2010), Application of a Data Compression Technique to the American Standard Code for Information Interchange (ASCII), International Journal of Information Science and Computer Mathematics, Vol. 1, No. 1, pp. 1-7.

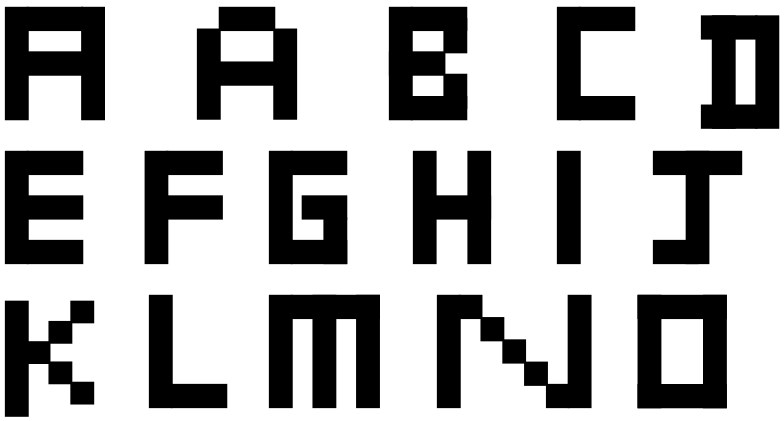


Figure 1: Trained Patterns

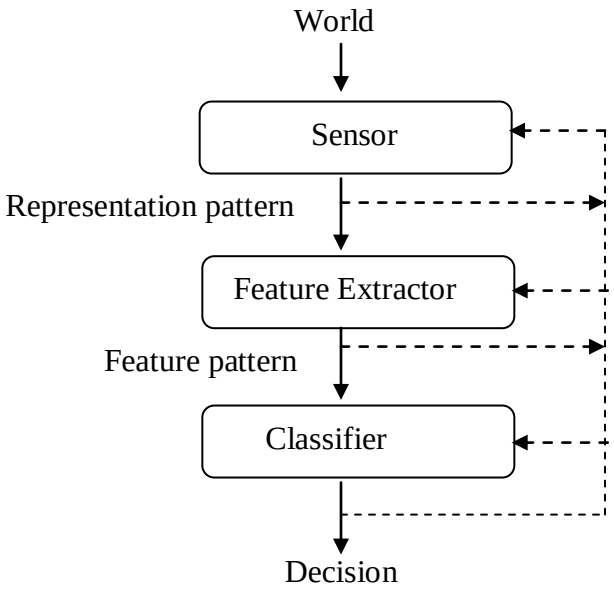
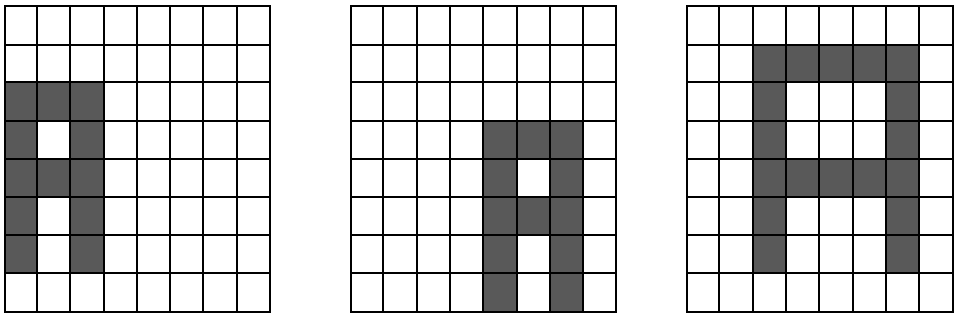


Figure 2: Pattern recognition system model [**9**]



(a) (b) (c)

Figure 3: Input Pattern: Three different patterns of character A. The pattern in Figure 3(b) is of the same size as that of Figure 3(a) but with a shift in position.

Table 1: Representation of Input Pattern as 8-bit Words

Figure No.	8-bit Representation
1(a)	00000000, 00000000, 11100000, 10100000, 11100000, 10100000, 10100000, 00000000
1(b)	00000000, 00000000, 00000000, 00001110, 00001010, 00001110, 00001010, 00001010
1(c)	00000000, 00111110, 00100010, 00100010, 00111110, 00100010, 00100010, 00000000

Table 2(a): Row-wise 8-bit Representation for Figure 3(a)

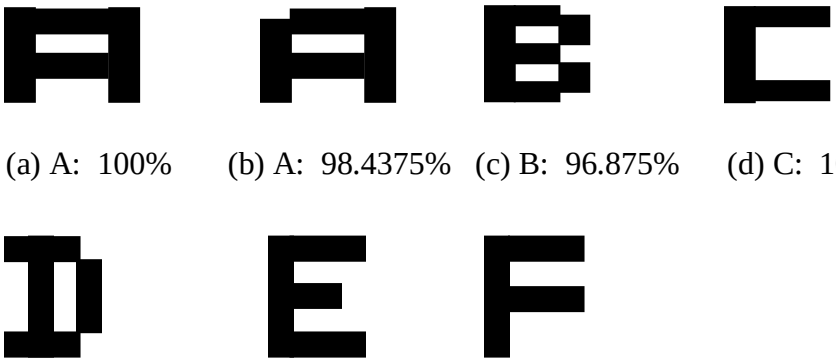
ROW NUMBER	8-BIT REPRESENTATION
1	00000000
2	00000000
3	11100000
4	10100000
5	11100000
6	10100000
7	10100000
8	00000000

Table 2(b): Row-wise 8-bit Representation for Figure 3(b)

ROW NUMBER	8-BIT REPRESENTATION
1	00000000
2	00000000
3	00000000
4	00001110
5	00001010
6	00001110
7	00001010
8	00001010

Table 2(c): Row-wise 8-bit Representation for Figure 3(c)

ROW NUMBER	8-BIT REPRESENTATION
1	00000000
2	00111110
3	00100010
4	00100010
5	00111110
6	00100010
7	00100010
8	00000000



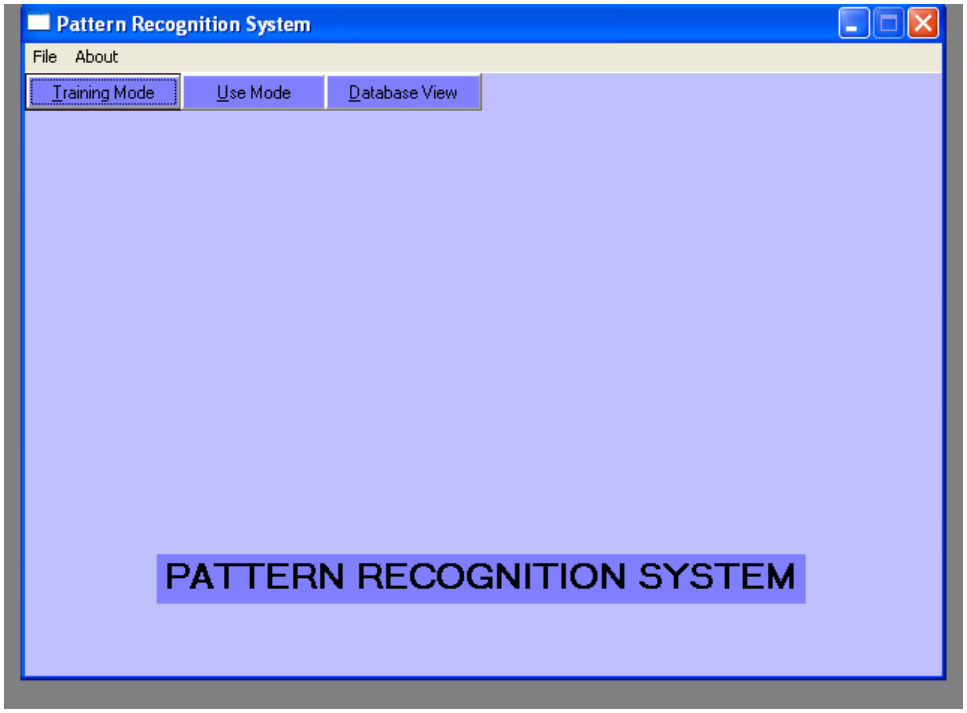
(a) A: 100% (b) A: 98.4375% (c) B: 96.875% (d) C: 100%

(e) D: 96.875% (f) E: 98.4375% (g) F: 100%

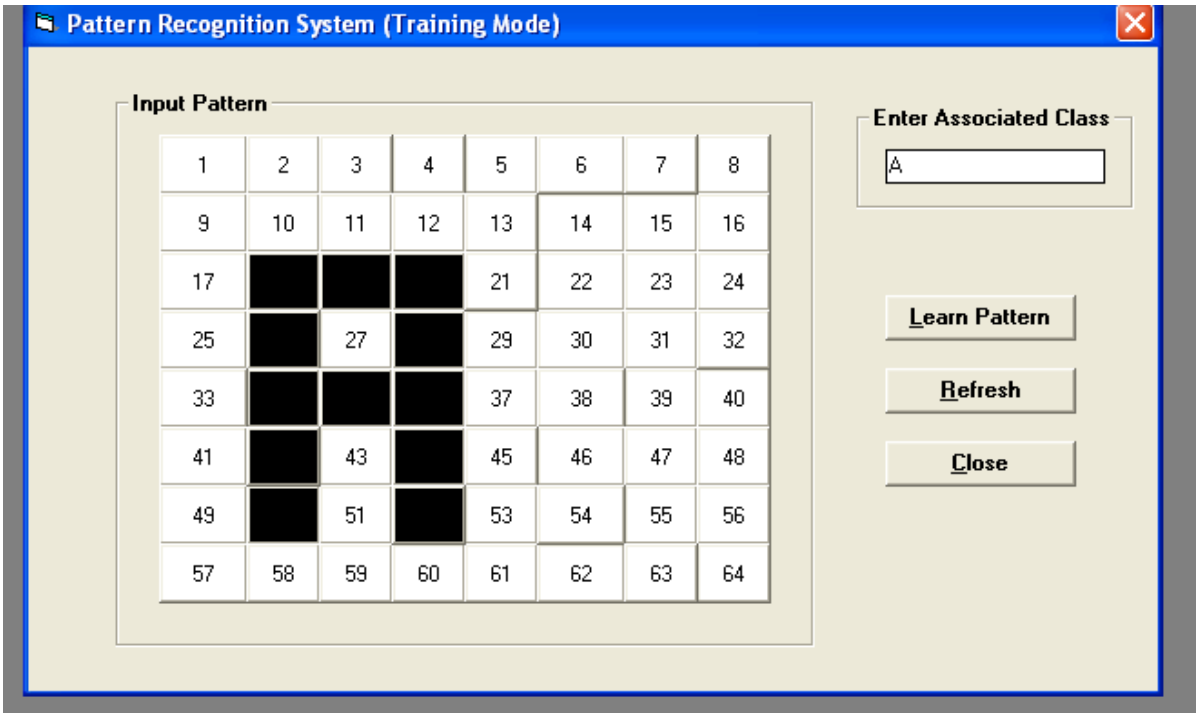
Figure 4: Results

APPENDIX

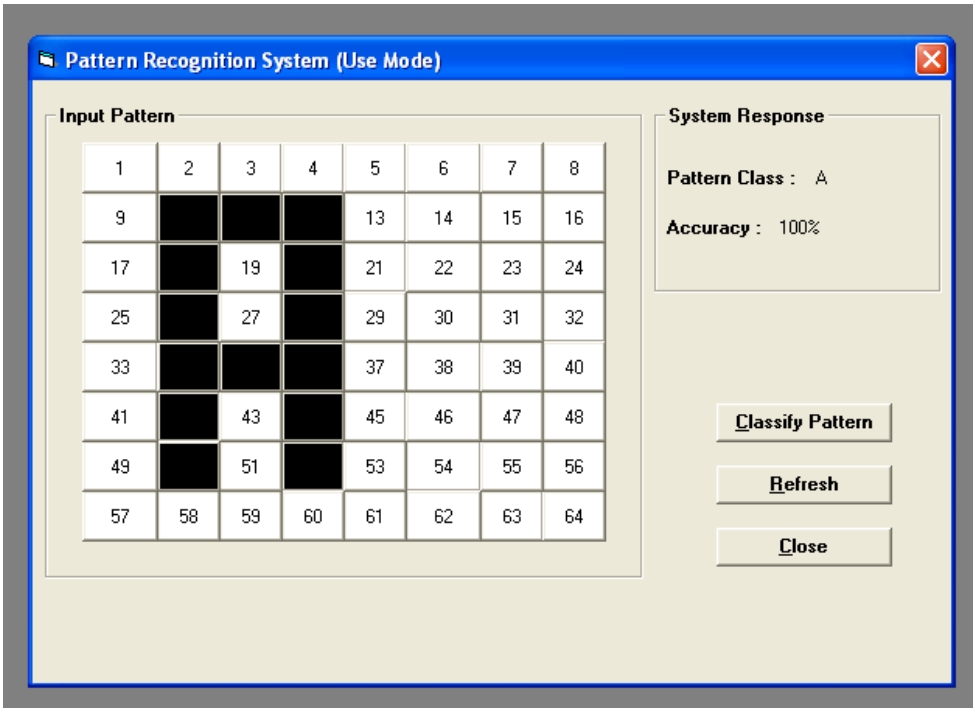
MAIN MENU

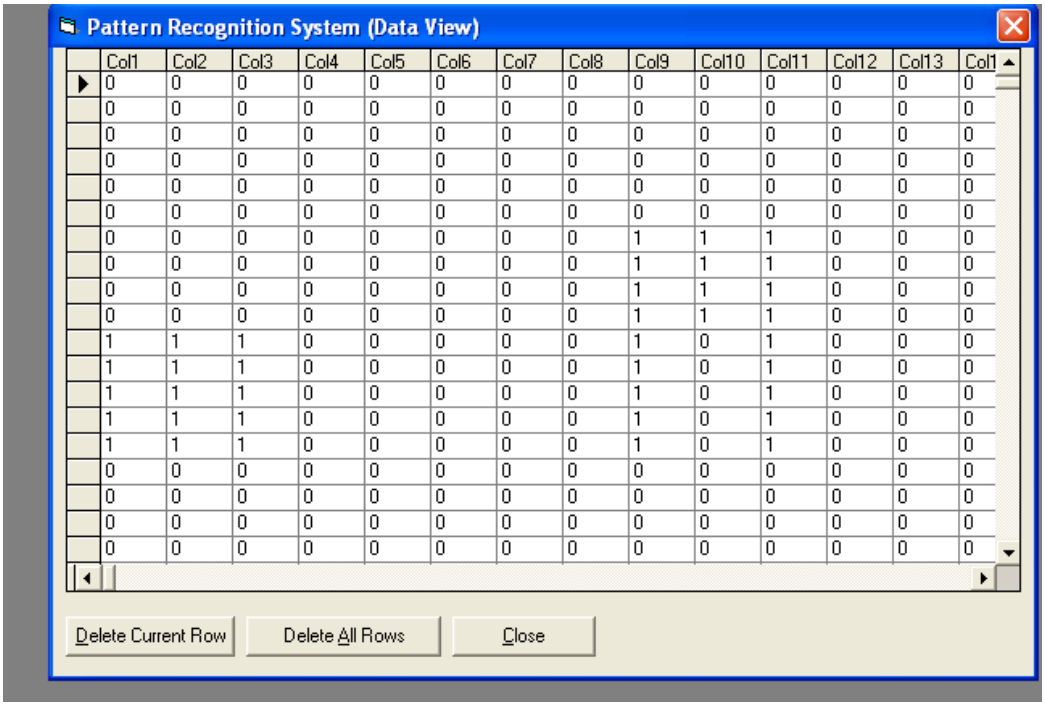


TRAINING INTERFACE

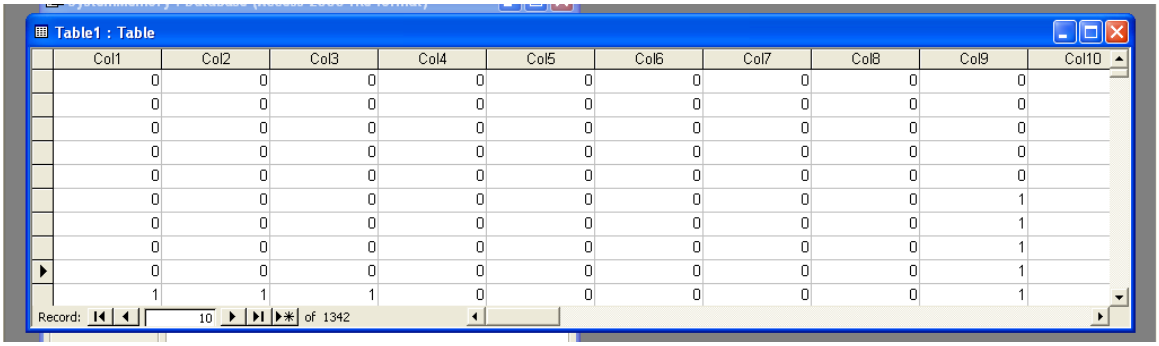


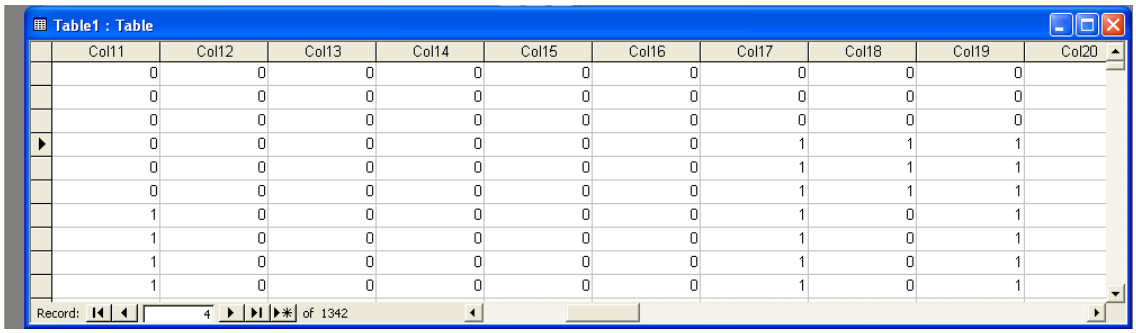
CLASSIFICATION INTERFACE

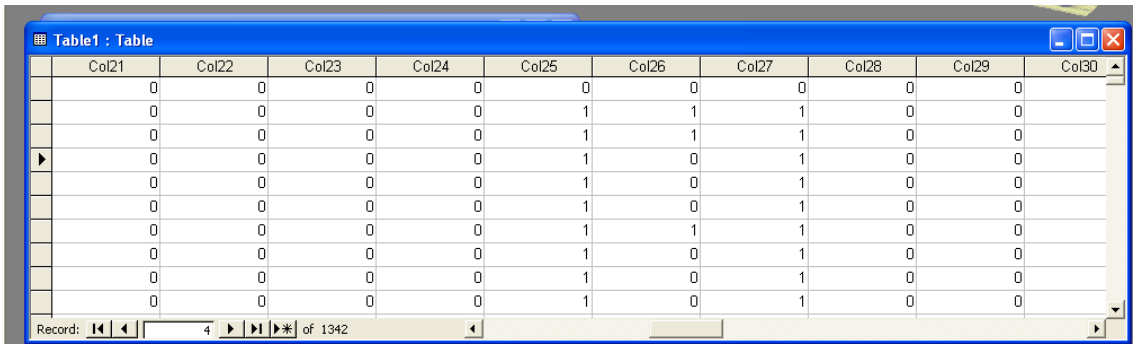




SYSTEM MEMORY VIEW INTERFACE.







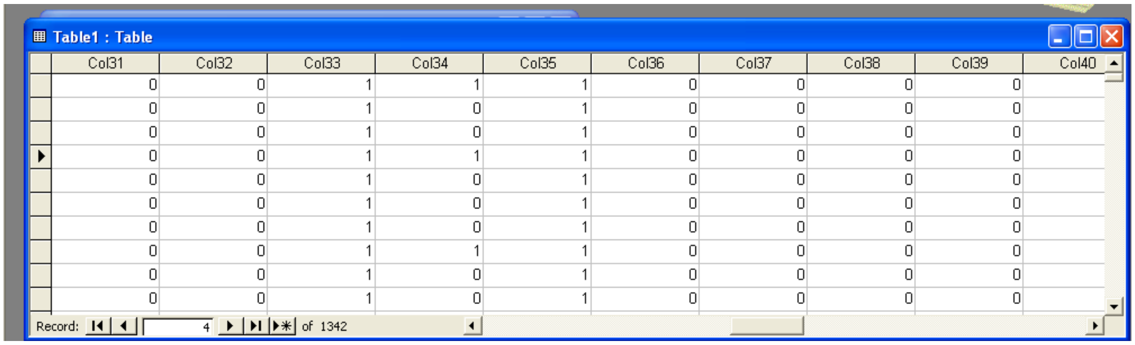


Table1 : Table

Col31	Col32	Col33	Col34	Col35	Col36	Col37	Col38	Col39	Col40
0	0	1	1	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	1	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	1	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	0	1	0	0	0	0	

Record: 4 of 1342

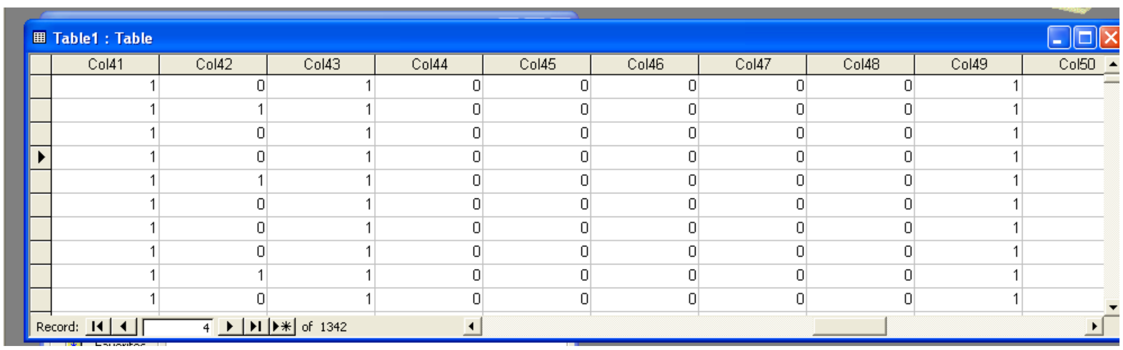


Table1 : Table

Col41	Col42	Col43	Col44	Col45	Col46	Col47	Col48	Col49	Col50
1	0	1	0	0	0	0	0	1	
1	1	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	1	
1	1	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	1	
1	1	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	1	

Record: 4 of 1342

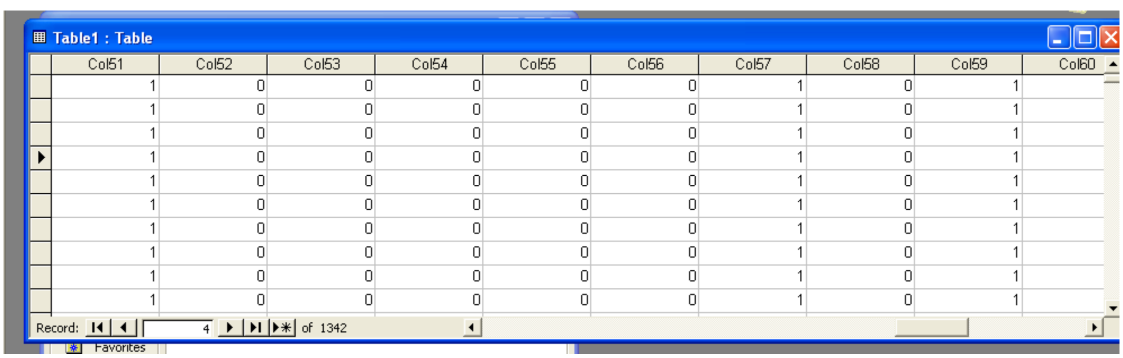


Table1 : Table

Col51	Col52	Col53	Col54	Col55	Col56	Col57	Col58	Col59	Col60
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	1	0	1	

Record: 4 of 1342

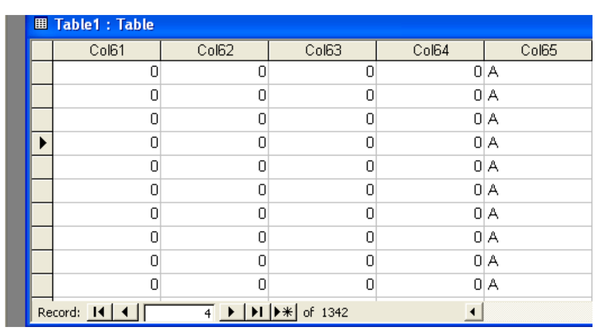


Table1 : Table

Col61	Col62	Col63	Col64	Col65
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A
0	0	0	0	A

Record: 4 of 1342