Design of a Neural Network Based Optical Character Recognition System for Musical Notes.

A.E. Akinwonmi, M.Tech., O.S. Adewale, Ph.D., B.K. Alese, Ph.D., and O.S. Adetunmbi, M.Tech.

Department of Computer Science, Federal University of Technology
P.M.B. 704 Akure, Nigeria.
E-mail: kaalfad@yahoo.com

ABSTRACT

Optical character recognition is the procedure by which the computer converts printed materials into ASCII files for editing, compact storage, fast retrieval, and for other purposes. In this study, a neural network approach was applied to perform high accuracy recognition on music score. There are three parts in the study. The first part designed an image preprocessor to obtain sets of images of 17 musical symbols consisting of notes and other glyphs for down-sampling using 8 different digitization procedures to serve as inputs. The second part is composed of the design and implementation of a neural network using a feed-forward with backward propagation. The network was trained using digitized down-sampled images of the symbol. The trained network was saved and tested with sample musical symbols for recognition. There were recognition errors. The recognized data was presented to the third part, which was designed as a Cakewalk Application Language (CAL) script generator. CAL scripts were obtained and presented to Cakewalk Audio Pro for Musical Instrument Digital Interfaces (MIDI) production.

(Keywords: optical, character, musical notes, neural networks, OCR, MIDI)

INTRODUCTION

The term optical character recognition (OCR) originally referred to the use of optical techniques involving mirrors and lenses while digital character recognition referred to the use of scanners and computer algorithms in image analysis. The two terms were originally believed to be separate areas. Though, very few applications still exist that use purely optical techniques, OCR as a term now covers digital character recognition applications as well (Wikipedia, 2005). Tanner, (2004) described OCR as a type of document image analysis where a scanned digital image of either machine printed or handwritten script is input into an OCR software engine and translating it into an editable machine readable digital format such a ASCII text (Avi-Itzhak et al., 1995 and Bedini et al., 1997).

OPTICAL CHARACTER RECOGNITION FOR MUSICAL NOTES

Music is an arrangement of or the art of combining or putting together sounds that please the ear as opposed to noise, which is a combination of sounds that displease the ear (i.e. occur with no discernable pattern). The musical tone (a sound with a distinctive quality) is the elementary token of music, which is defined by four properties including pitch, duration, intensity, and quality.

Musical signs and terms are means by which musicians or songwriters convey or store the idea of music. Music also stores its passive expression in written notes forms called musical notes. Within the context of this research, the term music is confined to those traditional Western musical styles, which originated about the eighteenth and early nineteenth centuries. In the referred style, musical notes are made up of five lines and spaces called the stave or staff. Musical sounds are represented on paper as notes.

The notes are conventionally named after the first seven letters of the English alphabet: A, B, C, D, E, F, G. Selected notes are then written on the lines and spaces of the musical staff. As such, musical notes represent some forms of characters or shapes. In addition, three symbols the flat (*b*), the sharp (♯), and the natural (♮) are placed next to notes to alter their pitch (Gerboth, 2005).

MUSICAL INSTRUMENT DIGITAL INTERFACE (MIDI) FORMAT

The computer system can process musical signs in two major forms. One form is as digital signal while the other form is as analogical or continuous form. Musical files are also represented in these forms. This research focused on the most common and portable digital representation of musical sound (i.e. MIDI). MIDI is an acronym for Musical Instrument Digital Interfaces format. This is the digital equivalence of the Musical notes so to say. MIDI is also the means by which computer communicates with most sound cards, keyboards, handheld devices such as mobile phones, PDAs and other electronic equipments. The compactness of MIDI music files makes them very useful for delivering music between devices and online (Bainbridge and Bell, 1997).

The Standard MIDI file format is a file interchange format defined by the MIDI Manufacturers Association (MMA). The format serves to allow for MIDI data exchange between different programs and devices. There are two common MIDI file format standards, MIDI Format 0 and MIDI Format 1. Format 0 MIDI files contain a single track, with all musical events stored in the single track while Format 1 MIDI files can store up to 256 tracks. There is a common language by means of which MIDI software packages communicate with each other. This language allows the software to edit MIDI files. Examples of the common MIDI software around are Cakewalk, Sonar, Cubase, Platinum, KOAN SSYEYO, and a host of others.

The simplest form is the digital composer utility which comes with GSM phones. Of the above list, Cakewalk and Sonar are the most popular in this country. The aim of the research is to design an effective and efficient optical character recognition system for musical note recognition. Specifically, the research is expected to design an optical character reader model that reads inputs from images of scanned simple musical notes, recognizes and plays out the scanned musical notes with steps and procedures for musical note document analysis, develop a model for error correction in optical musical note recognition, and develops an experimental template for further research into optical musical note recognition. In order to achieve the objectives of this research, a review of the related literature on optical character recognition techniques was carried out. This was be followed by a review of the published literature on MIDI. Thereafter, a neural network engine was designed with OCR capability to identify scanned musical notes using Visual Basic. Finally, the identified musical primitives were converted to CAL (Cakewalk Application Language) Script for MIDI generation.

CHOICE OF PROGRAMMING LANGUAGE

The inner workings of Neural Networks are complex, but they are known for stochastic applications (Dorffner, 1999). Consequently, it is believed that an implementation with a visual development kit would give a better understanding of its functions. Of course, other languages can give perhaps faster implementation of neural elements, for instance C and Pascal.

Notwithstanding, the points above Visual Basic Version 6.0 was used to develop the application. Visual Basic was chosen partly because of its simple constructs; Object-oriented features; and its ability to implement visual objects and components ability with minimal coding.

DESIGN AND IMPLEMENTATION OF THE OCR SYSTEM

This research applied the object-oriented features of Visual Basic to enhance modular techniques in the design of basic functional components. There are three basic functional components of the OCR system (Figure 1). These include the image preprocessor, the OCR engine and the CAL Script Generator. Other auxiliary components were coded into the system to enhance data collation, accessibility and testing. All of these components, though integrated, can run as reusable, separable and independent functional modules for demonstration/tutorial purpose. This approach has made the design and implementation of the system simple and easy while not compromising fidelity.

At one end, the image preprocessor accepted scanned images (black and white, line-art format) as input, passes its processed output to the OCR Neural Engine, resultant output is sent to the CAL Script generator, which generates a CAL Script.

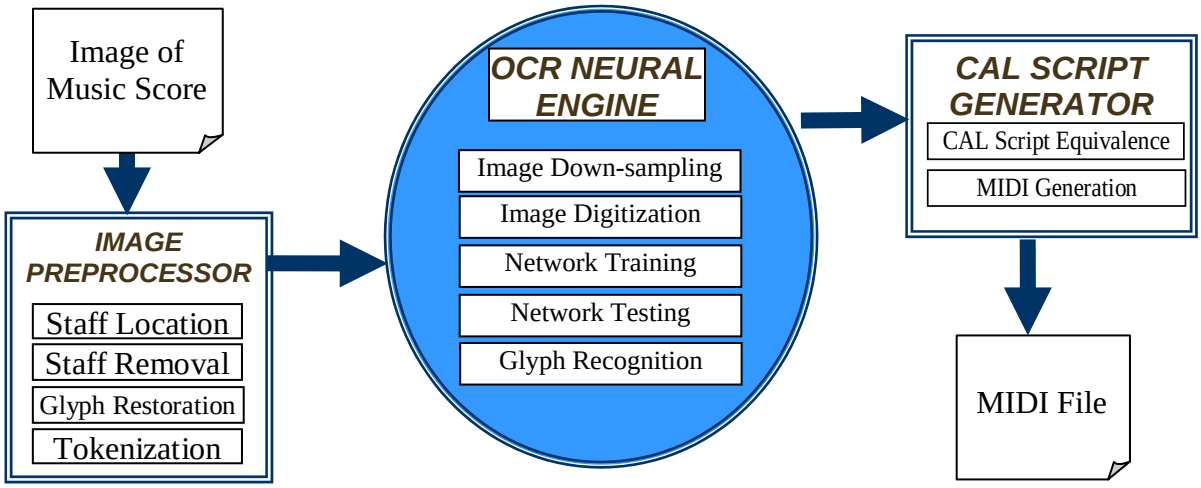


Figure 1: Conceptual Diagram of the System.

THE MULTIPLE DOCUMENT INTERFACE (MDI) FORM

The mdiNBOCRMN is the startup form for the application (Figure 2). It is also the MDI form for the Optical Character Recognition System for Musical Notes.

It has a main menu with basic options including "File", "Tools", and "Help". The File menu provides a means of exit from the package. The Tool menu has the options "Image Preprocessor", "Neural Engine", "CAL Script Generator", "Merge Data", and "Magnifier" with shortcuts "F2", "F4", "F5", "F6" and "F3" respectively. The options, in turn, activate functional modules within the package as their names indicate. It is possible to switch to any of the inherent functional modules of the software from this form, which serves as container (MDI) and parent to the modules.

Figure 2: Multiple Document Interface (MDI) Form (mdiNBOCRMN form).

IMAGE PREPROCESSOR

The image preprocessor provides the interface through which the scanned musical note images are treated for presentation to the neural engine. It is contained within the frmPreprocess form. It has a flexible, interactive, user friendly, and menu driven Graphic Users Interface (GUI) (Figure 3). It is modular in design. It is made up of menu options, which are set to negotiate with the parent Multiple Document Interface (MDI) form (mdiNBOCRMN form).

The menu functions are also supported with a tool-tip-text supported top-aligned toolbar and menu shortcuts keys. Furthermore, the preprocessor has functionality for performing basic file operations such as loading scanned images from storage, unloading images from memory, saving processed images to disk, and magnifying selected areas on the loaded image. The image formats include *.bmp, *.gif, *.bmp and *.jpg. Essentially, they must be in black and white presentation.

The preprocessor has a picture box control to contain images and give facilitates visual performance of necessary operations on loaded images. The picture box is also enhanced with both vertical and horizontal scroll bars to help reposition loaded images. Other functionalities of the preprocessor include a drawing tool consisting of a pen with adjustable pen size/width, an eraser, and a selection tool. These can be used to edit loaded image, introduce noise to the

image, and select a portion of it for data analysis or digitization. Moreover, the design has a preprocess wizard that helps to locate staff lines and remove them. However, in the process of stripping the staff lines, cross matching portions of musical glyphs are removed. This causes fragmentation of glyphs. Fragmentation of glyphs is corrected by a glyph reconstruction procedure within the preprocess wizard (Figure 4).

The preprocess wizard subsequently tokenizes contiguous regions of the image. There is also a status label, which indicates the inner processing of the wizard. In addition to routine for digitizing selected symbols and glyphs, there is also a command for saving digitized data. The image preprocessor provides a veritable fecundity for image editing, manipulation, visual sampling, and digitization. The output of the image preprocessor is redirected to the specialized OCR neural engine as either desired outputs for training the network or samples for recognition by trained network.

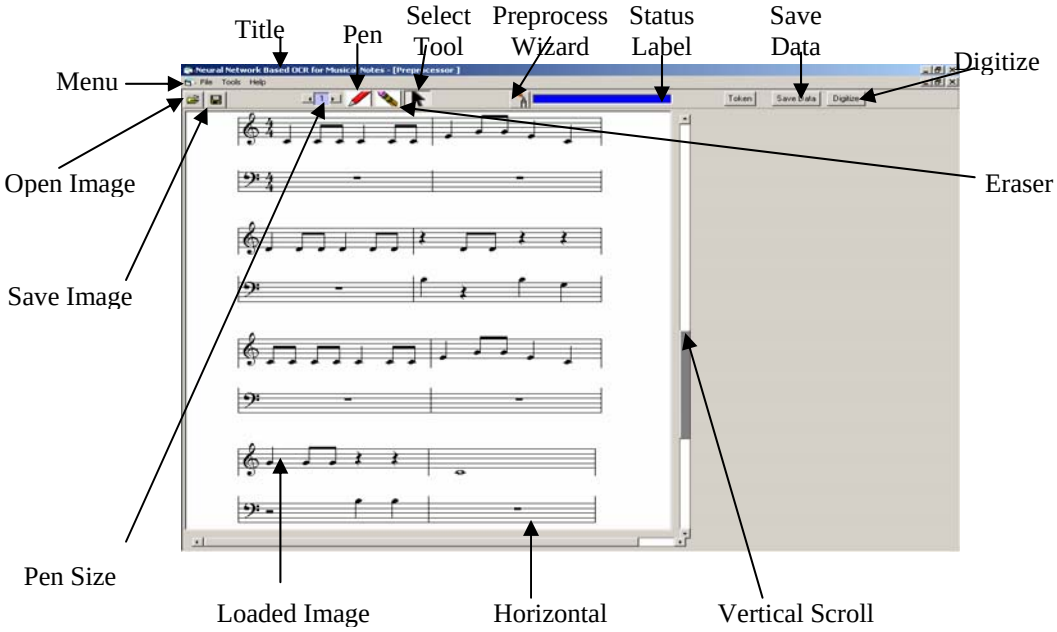


Figure 3: Preprocessor GUI showing Functions.

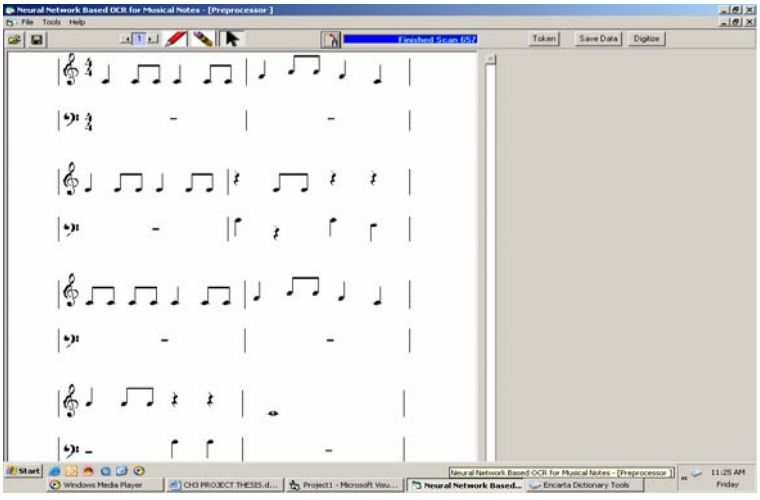


Figure 4: Preprocessor Showing musical image with staff lines removed.

THE OCR NEURAL ENGINE

The OCR Neural Engine is the heart of the NNBOCRMN. It operates from the frmNeuralEngine form, which is an MDI child form. The OCR Neural Engine provides the interface through which the digitized data of images from the preprocessor is presented to the OCR neural engine for either training or recognition (Figure 5). A network grid (flexgrid) is provided for visual display of this data, including input, desired output, actual output and scaled output. The number of rows in the grid represents the number of input data that can be fed into the network single input layer at any time. The flexgrid thus allows multiple presentations of input data for training though a restriction of 400 input data in an epoch was placed on the network for limited resources.

Some of the features of the flexgrid include 87 grid columns of which 3 are horizontally merged, thereby segmenting the network grid into four data zones. The 3 columns are dormant and in addition serve as labels during training and running. Eighty-one columns of the remainder hold input data for each of the maximum of 81 neurons permissible in the input layer of the network. The remaining 3 columns are designated 1 each for holding desired output data, actual output data and scaled output data.

Moreover, the OCR Neural Engine has a Downsampler module which helps to down-sample images from a picture box of 750 * 750 *twips* to 9 * 9 matrix. The down-sampling process also is visually implemented for comprehension. The down-sampled image is digitalized to an 81 bit-binary code. This code is fed as input to the 81 neural elements in the network. In order to enhance merging of data from different epoch values for each glyph into one file, a Rich Text Editor was designed and implemented with the application.

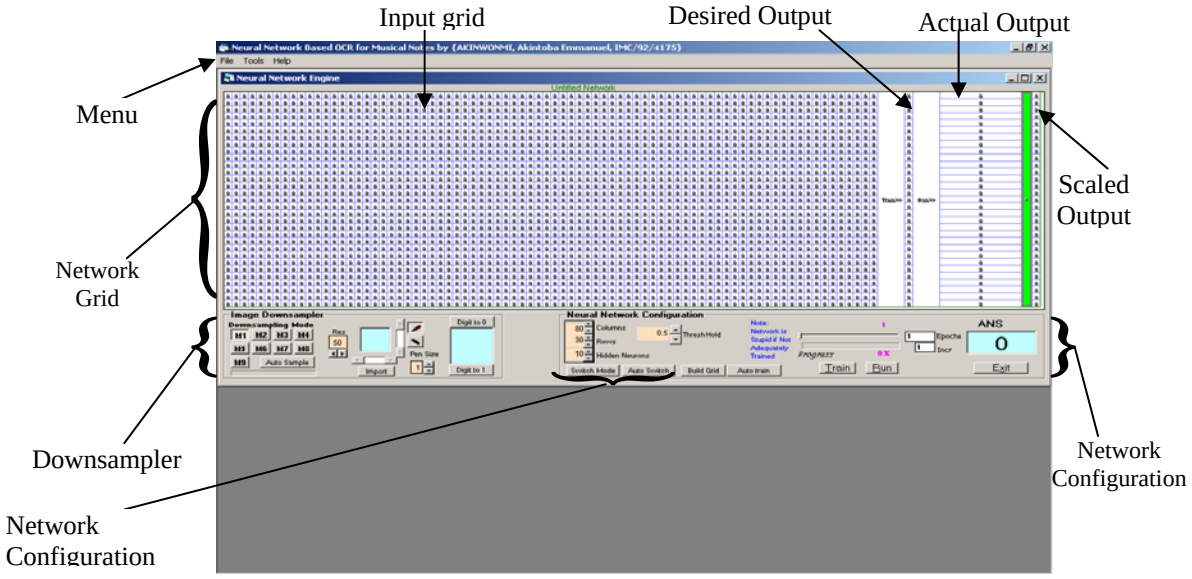


Figure 5: The OCR Neural Engine GUI.

DOWN-SAMPLING OF IMAGE

The downsampler is also comprised of a drawing tool consisting of a pen with adjustable pen size/width and an eraser to help manipulate images on the downsampler and in addition help introduce noise to the image. Down-sampling is the process of converting the number of tokens in the input data to conform to the number of inputs allowed by the network.

In the case of this research the image is down-sampled from a 750 * 750 matrix to a 9 * 9 matrix. Thus, there were 81 bits in the input data. Consequently, the input layer of the network contained 81 neurons to accept this input.

Of course, down-sampling does not preserve the originality of the input data. It in fact diminishes the quality of the original image. However, without down-sampling, in case of this research, a total of 562500 (750 * 750) neurons would be required to process the input image. This would require a lot of computing resource and slow down the processes. Effort was made to ensure that all input data underwent the same degree of down-sampling and scaling.

Throughout the research, a down-sampling ratio of approx 6944:1 was applied to the input data. In addition to down-sampling ratio, a resolving factor was used to improve the resolution of the down-sampled image.

DIGITIZATION OF DOWN-SAMPLED IMAGE

The following Figures (6, 7, and 8) show a pixel-to-matrix conversion of down-sampled images of a dotted semibreve, a quaver and a minim.

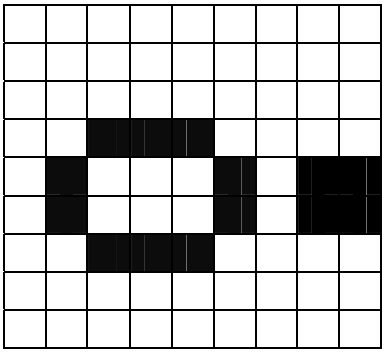


Figure 6: Down-Sampled Image of a Dotted Semibreve.

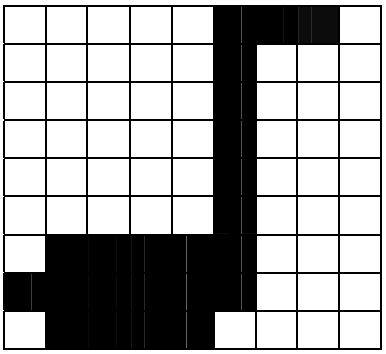


Figure 7: Down-Sampled Image of a Quaver.

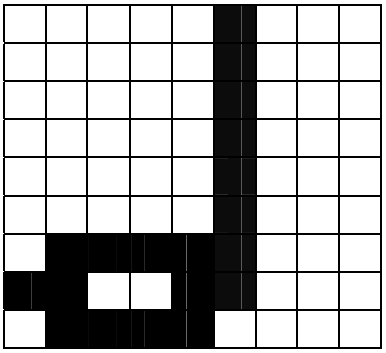


Figure 8: Down-Sampled Image of a Minim.

To convert these to input data, if a cell is black, its value is 1; and 0 if the cell is not black. This gave 81 Nodes for the input layer whereby the value for each node was one of the characters from the strings below (read right-to-left and top-down), resulting in:

000000000000000000000000000011100110
100010110100010000011100000000000000000
000 for the dotted semibreve,

011100000000100000000100000000100000000
100000000100000000111110000111111000011
110 for the Quaver and

000100000000100000000100000000100000000
100000000100000000111110000110011000011
110 for the Minim.

However, it is possible to have other values different from those above for the respective images considered. In that regard, this research also used a read-right-to-left and down-top, a read-left-to-right and down-top, a read-left-to-right and top-down and four other non-structured read approach to probe the network in the process of

finding optimum setup configuration. These were called digitization modes for convenience. The downsampler has 8 command buttons (M1 –M8) for setting each digitization mode.

SETTING UP OF THE NETWORK

Since musical conventions vary from civilization to civilization and from culture to culture (Gerboth, 2005) consequently it has been difficult to evolve a universal set of symbols for music. Even within conventions, there exist variations and non-standard styles. Various notations were looked at; more than 100 musical symbols were identified. Many of these either are non-general notations within the western style of notation or are combinations of two or more of the basic symbols. In order to avoid complicating the problem, this research was limited to the basic musical symbols which are notably more standard across dialect s, which include the staff, clefs (G & and F clefs ?), and time signatures (2/2, 3/4 and 4/4).

Others were the notes: minim h, crotchet q quaver e, semiquaver x and their rests (𝄻 , 𝄽 , 𝄾 and 𝄿).

Not all, glyphs studied also include the accidental signs: the flat (***b***), the sharp (**#**), and the natural (♮). The bar delineation sign (|) was also included among the symbols under study. The Neural Network was trained with digitized data of down-sampled images of the following 17 glyphs (Table 1).

Various images of each of the above glyphs were extracted from various scanned images of music that was loaded into the preprocessor. The extract was imported into the downsampler on the OCR Neural Engine and was down-sampled. Thereafter, the down-sampled images were converted into zeros and ones (81) digits in length. The digitized data were then loaded into the neural network grid.

TRAINING SET

For each symbol, a total of 48-images training set was digitized and inputted into the neural grid. Sixteen images for the desired glyph with the desired output set to 1 and different 2 sets of the undesired 16 symbols with the grid's desired output set to 0. This procedure was repeated for each of the symbols.

TRAINING EPOCHS

The Network was initially tested with the XOR problem, in which it learned at 1500 epochs. This gave a preliminary proof that the network was properly coded and that it could learn.

The network was thereafter trained for 10, 20, 50, 100, 200, 400, 600, 800, 1000,1200, 1400, 1600 , 1800, 2000, 3000, 4000, and 5000 epochs with the glyphs data to determine the epoch (an epoch is a complete training cycle for a training set of inputs) at which the network started to learn and to determine the range at which learning was optimum while avoiding over-learning and consequential memorization.

The results showed that learning started around 200 epochs. The network had reached optimum learning at 1600 epochs. Training for more than 1600 epochs led over-learning.

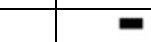
CHOOSING THE NUMBER OF HIDDEN NEURONS

Three approaches that were reported in Chesnut (2004) were applied:

Using less hidden Neurons than the input neurons. Precisely, using (No. of Input Neurons + No of Output Neurons) / 2. For this research, 81 input neurons and 1 output neuron per data item were used.

- No. of Hidden Neurons = No. of Input Neurons + No of Output Neurons) / 2
  = (81 + 1)/2
  = 41
  41 hidden neurons were used while the number was gradually increased.
- Using the same number of hidden neurons as there are input neurons. This research also used 81 hidden neurons for training.
- Using more hidden neurons than input neurons (The number of hidden neurons should be less than twice the input layer size). 100, 120, 140 and 160 hidden neurons were used.

Table 1: Images of the Selected Glyphs (Symbols).

SN	NAME	GLYPH	FUNCTION
1	G Clef	&	Shows treble and alto staff
2	F Clef	?	Shows tenor and bass
3	Time Signature(Two Two)	2/2	Specifies a time signature of 2 Minim per bar
4	Time Signature(Three Four)	3/4	Specifies a time signature of 3 Crotchets per bar
5	Time Signature(Four Four)	4/4	Specifies a time signature of 4 Crotchets per bar
6	Minim	**h**	A note with timing of half a semibreve (whole note)
7	Crotchet	**q**	A musical note that has the time value of one-quarter of a semibreve (whole note).
8	Quaver	**e**	A musical note that has the time value of one-eight a semibreve (whole note).
9	Semiquaver	**X**	A musical note that has the time value of one-sixteenth a semibreve (whole note).
10	minim rest		A musical rest that has the time value of a half note
11	Crotchet rest	**g**	A musical rest that has the time value of a Quarter of a semibreve (whole note).
12	Quaver rest		A musical rest that has the time value of one-eight of a semibreve (whole note).
13	Semiquaver rest		A musical rest that has the time value of one-sixteenth of a semibreve (whole note).
14	Flat Sign	***B***	A sign (*b*) placed next to a note to show that it is to be lowered by a half-step, or a note that is lowered a half-step
15	Sharp Sign	**#**	A sign (#) placed next to a note to show that it is to be raised by a half-step, or a note that is raised a half-step
16	Natural Sign	♮	A sign placed before a musical note in order to cancel a previous sharp or flat
17	Bar Delineation Sign	**|**	A bar is a fundamental unit of time into which a musical work is divided, according to the number of beats. The bar delineation sign bounds the beginning and end of a Bar on the staff

Of all trials, 140 hidden neurons gave the optimum learning (i.e. provision of optimal compromise between performance and time of convergence). Therefore, the network was set to 140 hidden neurons for training.

TRAINING THE NETWORK

After setting the parameters above, the following steps were taken:

- Each hidden neuron had a dendrite to each input Node. The value of the Hidden Neurons value was set to the total of the Weight of each dendrite multiply by the Activation of that dendrite.

- The total was passed through the Transfer function - a sigmoid (1 / (1 + Exp(Value * -1))). The purpose of this was to be able to select the values of the most contributing neurons. This function crunch a decimal to 1 or 0.

- Steps (a) and (b) were repeated to derive the Activation value of the Output Neurons from its dendrites to the Hidden Neurons giving (actual outputs) decimal values between 0 and 1.

- Training was next. It involved finding the error value for each Output Node.

  Output Neuron Error[x] =
  (Actual Output[x].Activation *
  (1 – Actual Output[x].Activation) *
  (Desired Outputs[x] - Actual Output[x].Activation)).

- The Hidden Neuron Error was calculated for each Neuron by totaling each dendrites Weight and multiplied by its Output Neurons Error. The resultant value was then multiplied by the current Neurons Activation.

- The next part involved using Output Error to adjust the Weights of the dendrites backward (Back-Propagation) from the Output Neurons to the Input Neurons dendrites. The new Weight of each Hidden-Output Dendrite value was set to:

  (HiddenActivation * LearningRate *
  OutputError + CurrentWeight) or:
  Product (HiddenActivation:
  LearningRate:OutputError)) + CurrentWeight

- Step (f) was repeated with new Weights for each Input-Hidden Dendrite. At this point, the Network was configured for training. Thereafter the network was trained for 1600 epochs. The trained network was saved with the name of the desired output for each symbol (for example: Minim.nM1 …….nM8, Sharp.nM1……nM8). Results depended on the digitization mode (M1, M2, M3, M4, M5, M6, M7, and M8) applied to the original image. For this research, M3 was used. It gave the best learning.

Essentially, one the objectives of training the network is to derive the input weight and dendrite strength for each neuron at which the network learns from training with each symbol. Once these are derived, the network is saved into a *.nM#) file

RUNNING THE NETWORK

The Original training data and other practical data (data not used during training) were used to run steps (a) – (c) above. The outputs were then recorded. The 17 symbols were recognized.

CAKEWALK APPLICATION LANGUAGE (CAL)

CAL stands for Cakewalk Application Language. It is an event-processing language which can be used to extend the functions of Cakewalk Pro Audio, SONAR™ and other similar packages. These are professional tools for authoring sound and music on the personal computer. They are usually equipped with custom editing commands. CAL blends elements of C and LISP, in particular. A typical CAL program is illustrates CAL's working.

EVENT PARAMETER VARIABLES

There are some predefined "delightful" variables allows access to events in Cakewalk's track buffers. They work only in the (for Each Event) expression, whereby they are automatically set to the value of the current event's parameters. This is equivalent to **Get**ting and **Set**ting event parameters.

They can be used to find out about the current event. They correspond to "Properties" in VB. In CAL different events have different parameters. While Note event has a key number, a velocity, and duration, A Patch event has only a patch change number. Accordingly, some of these

variables are defined only for particular kinds of events, whereas others like the variable for the time of the event apply to all kinds of events. The variables for all kinds of events include:

- Event.Chan: corresponds to the MIDI channel of the event (0..15).
- Event.Time: corresponds to starting time of the event
- Event.Kind: corresponds to kind of event it is. One of: NOTE, PATCH, CONTROL, CHANAFT, KEYAFT, WHEEL, SYSX

The following variables (Table 2) apply to particular kinds of events. They are used to get or set an event parameter for a particular kind of event.

Usage of the Event Kind and the Note.Key for example is demonstrated below:

The CAL script to make sure an event is a note event and if it is, add three to the key number (raise the pitch three half-steps) looks like the following:

(if (== Event.Kind NOTE); if it is a NOTE event then (= Note.Key (+ Note.Key 3)) ; change it!)

As long as condition evaluates to non-zero, action is implemented. However, if condition is zero the first time, action is never implemented.

BUFFER FUNCTIONS

These include “insert”, “delete” and “index”. They are used respectively to return the index of the event in the track buffer, delete the current event and insert a copy of the event at the specified index.

MUSICAL TIME FUNCTIONS

In professional audio authoring packages time is measured in three major formats: Beat; Tick and Raw Time. These are the conversion functions for dealing with times in Cakewalk. The following statements convert time between Measure:Beat:Tick and "raw” times (meas <time>), (beat <time>), (tick <time>) and (makeTime <measure> <beat> <tick>).

THE CAL SCRIPT GENERATOR

One of the modules of the application is the CAL Script Generator. It collects the recognized data output from the OCR Neural Engine and transforms it to the CAL script. Thereby terminating the sequence of processes needed to transform the musical note image to the script form. An [OCR Neural Engine]-[CAL Script Generator] interface was integrated into the program design to present the output of the OCR Neural Engine’s output to the CAL Script Generator wherein they are converted to Cal Script deliverable to Cakewalk Audio Authoring Package.

Table 2: Event Variable Parameters that Apply to Particular Kinds of Events (Cakewalk, 1999).

SN	Variable	Event.Kind	Meaning	Allowed values
1	Note.Key	NOTE	Key number (pitch)	0 .. 127
2	Note.Vel	NOTE	Velocity	0 .. 127
3	Note.Dur	NOTE	Duration	0 .. 65535
4	Patch.Num	PATCH	Patch number	0 .. 127
5	Patch.Bank	PATCH	Bank number	-1 .. 14383
6	Control.Num	CONTROL	Controller number	0 .. 127
7	Control.Val	CONTROL	Controller value	0 .. 127
8	KeyAft.Key	KEYAFT	Key number	0 .. 127
9	KeyAft.Val	KEYAFT	Value (pressure amount)	0 .. 127
10	ChanAft.Val	CHANAFT	Value (pressure amount)	0 .. 127
11	Wheel.Val	WHEEL	Wheel value	-8192 .. 8191

This package then generates the corresponding playable MIDI file from the CAL Script.

DATA – CAL SCRIPT CONVERSION

Conversion of the recognized data to CAL script is based on the fact that each Character could be identified along with its Time in HMSF and MBT formats, its kind, its Pitch (equivalent to its relative position on the Staff or its relative position on the keyboard) and it's Length if it is a note or a rest symbol. Each token of data passed by the OCR Neural Engine is of the structure (Figure 3):

Token {Kind, NTime, [Pitch],[Velocity], [Length]};

This represents the Kind, horizontal and vertical location of the glyphs, the Pitch (only applicable where Kind = Note), the Velocity (only applicable where Kind = Note), the Length (only applicable where Kind = Note or Kind = Rest)

For note C3 at the start of a bar:
Token {Kind, NTime, [Pitch],[Velocity], [Length]}is sent as Token ("Note", 00:00:00:00, "C3", 1.01.000). This interprets to a musical Note at the beginning (position "00:00:00:00") of a bass staff with a pitch of C3, having a loudness equivalent to the amount set on Cakewalk (0 - 127). The note has a length of 1 (a crochet). Each glyph is thus explicitly converted to this format playable by Cakewalk.

DISCUSSION

The factors affecting the accuracy of the system include quality of scanned image and neural network configuration. These are discussed here below.

Quality of Scanned Images

During scanning of archival score sheets, noise, distortion and skewing can occur. This will cause a decrease in accuracy of the system. Particularly, with old or defaced documents, the accuracy degenerates. Thus, it is very necessary to make sure that the neural engine is well trained using both clear images and noisy images. The combinations of both were used in this study. Degree of downsampling also affects accuracy. Since downsampling is simply a systematic reduction in the quality and size of the image, very vital portions could be lost during downsampling.

During staff removal, some portions of the original notes and glyphs were unavoidable removed, a restoration procedure was written to restore such portions using a technique that locates and examines vertical gaps. Where the vertical gaps were below a set threshold, such gaps were joined. The resulting images in this case could not have been the original images though they looked very close to the original images.

Neural Network Configuration

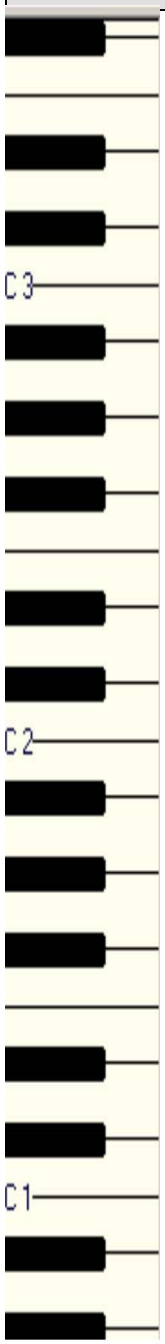
The downsampler uses a procedure to control dilatation and resolution in the images to be downsampled. Setting of the resolution was more on a trial and error basis since it is difficult to have the same picture quality for all practical data. In order to diminish the effect of this inconsistency in resolution with practical images, various resolutions were applied to the different training sets.

The neural engine was also implemented visually. The beauty of this is that though underlying workings of neural networks are hidden, it is possible to follow the processes visually to a reasonable extent. A first hand comparison could be made with various inputs and actual output readings. From the results obtained, the network could not learn when a low number of epochs were used for training. In contrast, with adequate training epochs, the network learned considerably. Restrain was exercised not to over train the network as this could lead to data memorization.

The Image Processor

The image preprocessor was designed to handle black and white images of 75dpi.This were loaded via a command dialog box. Thereafter, the preprocessor used horizontal and vertical projection techniques to locate staff lines, to locate notes and other glyphs, to locate and correct regions damaged during staff line removal. This procedure would be very accurate with lines or images that are not skewed. The fidelity of the preprocessor would obviously degenerate with increase in skew angle. It is believed that the, preprocessor would perform better with a very efficient de-skewing procedure.

Table 3: Token {Kind, NTime, [Pitch],[Velocity], [Length]} as interpreted by Cakewalk for some kinds of Notes.

Keys	Time in HMSF	MBT	Ch	Kind	Data / Pitch	Velocity	Note Length
	00:00:00:00	1.01.000	1	Note	F#3 or G*b*3	100	1:000
	00:00:00:00	1.01.000	1	Note	F3	100	1:000
	00:00:00:00	1.01.000	1	Note	E3	100	1:000
	00:00:00:00	1.01.000	1	Note	D#3 or E*b*3	100	1:000
	00:00:00:00	1.01.000	1	Note	D3	100	1:000
	00:00:00:00	1.01.000	1	Note	C#3 or D*b*3	100	1:000
	00:00:00:00	1.01.000	1	Note	C3	100	1:000
	00:00:00:00	1.01.000	1	Note	B2	100	1:000
	00:00:00:00	1.01.000	1	Note	A#2 or B*b*2	100	1:000
	00:00:00:00	1.01.000	1	Note	A2	100	1:000
	00:00:00:00	1.01.000	1	Note	G#2 or A*b*2	100	1:000
	00:00:00:00	1.01.000	1	Note	G2	100	1:000
	00:00:00:00	1.01.000	1	Note	F#2 or G*b*2	100	1:000
	00:00:00:00	1.01.000	1	Note	F2	100	1:000
	00:00:00:00	1.01.000	1	Note	E2	100	1:000
	00:00:00:00	1.01.000	1	Note	D#2 or E*b*2	100	1:000
	00:00:00:00	1.01.000	1	Note	D2	100	1:000
	00:00:00:00	1.01.000	1	Note	C#2 or D*b*2	100	1:000
	00:00:00:00	1.01.000	1	Note	C2	100	1:000
	00:00:00:00	1.01.000	1	Note	B1	100	1:000
	00:00:00:00	1.01.000	1	Note	A#1 or B*b*1	100	1:000
	00:00:00:00	1.01.000	1	Note	A1	100	1:000
	00:00:00:00	1.01.000	1	Note	G#1 or A*b*1	100	1:000
	00:00:00:00	1.01.000	1	Note	G1	100	1:000
	00:00:00:00	1.01.000	1	Note	F#1 or G*b*1	100	1:000
	00:00:00:00	1.01.000	1	Note	F1	100	1:000
	00:00:00:00	1.01.000	1	Note	E1	100	1:000
	00:00:00:00	1.01.000	1	Note	D#1 or E*b*1	100	1:000
	00:00:00:00	1.01.000	1	Note	D1	100	1:000
	00:00:00:00	1.01.000	1	Note	C#1 or D*b*1	100	1:000
	00:00:00:00	1.01.000	1	Note	C1	100	1:000
	00:00:00:00	1.01.000	1	Note		100	1:000
	00:00:00:00	1.01.000	1	Note		100	1:000
	00:00:00:00	1.01.000	1	Note		100	1:000
	00:00:00:00	1.01.000	1	Note		100	1:000

Moreover, preprocessing was entirely managed visually. i.e. black pixels were programmatically placed or located on picture box control using Pset() and Point() functions respectively. This method makes the image preprocessor suitable for teaching, demonstration and it enhanced easier management and better comprehension of the processes though it is rather slow and wasteful of memory. A direct file level processing would have been faster and would have facilitated a higher dpi scanning thus preserving the integrity of scanned image to a higher degree yet the basic workings of the processes would have been hid.

Model for Error Correction and Reduction

Due to the foregoing, a good number of recognition errors were identified during testing. These errors resulted from diminished image quality, distorted image, and failures of the preprocessor to effectively delineate or tokenize images. Consequent upon the above reasons, some symbols were wrongly recognized while others were not recognized at all.

In order to reduce or correct such errors the following steps should be taken:

- Reducing the instances of visual processing. So that less of the system resources could be consigned to image processing while more is committed to image fidelity and algorithm development;

- Avoidance of staff removal will likely preserve the original image quality. Sheridan and George (2004) shows that removal of the stave lines before symbol recognition is not the only first step in musical note recognition and may not be the best. They showed that instead of removing stave lines, more should be added. This is 'defacing' since it adds stave lines to the score at a 1/2 stave. However, the recognition procedure becomes more complex.

- The quality of image is affected by scanning settings. Pre-adjusting image hue, saturation and threshold on the scanner will likely help in determining the optimum settings for scanning.

- A neural network will likely give accurate output if adequate training using enough input data is performed. However, one constrain of the von Neumann machine is in the ability to implement parallel processing. Whereas parallel processing will enhance the performance of neural networks. Using multiple processors will likely enhance adequate training of networks.

CONCLUSIONS

Our studies showed that the accuracy of the neural network was highest with a particular digitization mode. Other neural network architectures that are suitable for solving classification problems for instances are the Learning vector quantization, the Counter – propagation, and the Probabilistic neural network.

REFERENCES

1. Avi-Itzhak, H.I., Diep, T.A., and Garland, H. 1995. "High Accuracy Optical Character Recognition Using Neural Networks with Centroid Dithering". *IEEE Transactions on Pattern Analysis and Machine Intelligence archive*. 17(2): 218 – 224. IEEE Computer Society: Washington, DC.

2. Bainbridge, D. and Bell, T.C. 1997. "Dealing with Superimposed Objects in Optical Music Recognition". *Sixth International Conference on Image Processing and its Applications*. Universities of Waikato and Canterbury: New Zealand.

3. Bedini, L., Bozzi, A. and Tonazzini, A. 1997. "Digital Techniques for Character Recognition in Old Documents". *ERCIM News*. No.28.

4. Bellini, P., Bruno, I., and Nesi, P. 2001. "Optical Music Sheet Segmentation". *Proceedings of the First International Conference on WEB Delivering of MUSIC*. 183–190.

5. Chesnut, C. 2004. "Chesnut Tablet PC OCR with Neural Network AI". http://www.generation5.org/content/2004/aiTabletOcr.asp.

6. Dorffner. 1999. "Internal Report for NEuroNet". http://www.kcl.ac.uk/neuronet.

7. Gerboth, W. 2005. "Musical Notation." Microsoft® Encarta® 2006 [CD]. Microsoft Corporation: Redmond, WA.

8. Sheridan, S. and Susan, G.E. 2004. "Defacing Music Scores for Improved Recognition".

Proceedings of the Second Australian Undergraduate Students' Computing Conference, 2004. School of Computer and Information Science, University of South Australia.

9. Tanner, S. 2004. "Deciding Whether Optical Character Recognition is Feasible". KDCS Digital Consultancy.

10. Wikipedia. 2005. "Neural Network". http://en.wikipedia.org/wiki/neural_network.

ABOUT THE AUTHORS

Mr. A. E. Akinwonmi attended the University of Lagos where he obtained a B.Sc (Hons) degree in Biochemistry in 1991. He later obtained a Post Graduate Diploma and M.Tech. degree in computer Science from The Federal University of Technology, Akure, Nigeria. He is currently a Senior Programmer.

Dr. O. S. Adewale *MIEEE* received Bachelor of Science (Combined Honors) in Computer Science with Mathematics from the then Ogun State University now Olabisi Onabanjo University, Ago-Iwoye, Nigeria; a Master of Technology in Computer Science; and a Ph.D. (Computer Science) from the Federal University of Technology, Akure, Nigeria. He is a senior lecturer in the Department of Computer Science, Federal University of Technology, Akure, Nigeria. He was an associate member of the International Centre of Theoretical Physics, Trieste, Italy between 2000 and 2005. He has published a number of articles at both local and international reputable journals. He currently sits on the First Bank of Nigeria Plc Nigeria Professorial Chair in Computer Science of the Federal University of Technology, Akure, Nigeria. His research areas among others include web-enabling applications, modeling and simulation, high-performance and high-availability computing, grid computing and tele-traffic engineering

DR. B. K. ALESE, *MNCS, MACM, MIEEE* is a Lecturer I in the Department of Computer Science, Federal University of Technology, Akure, Nigeria. He holds a Bachelor of Technology (B. Tech.) degree in Industrial Mathematics from The Federal University of Technology, Akure in 1997. He also holds a Master of Technology (M. Tech.) and Doctor of Philosophy (Ph.D.) degree in Computer Science from the same University in 2000 and 2004, respectively.

Dr. Alese Joined the services of the Federal University of Technology, Akure in 1998 as a Graduate Assistant. He was once the Assistant Director of the Post Graduate Diploma Program of University. He is a member of the University Senate and various committees in the University. His Research interests are Information Security, Quantum Communication, Computer Networks and Digital Signal Processing. He has more than 40 publications in both local and international journals and conference proceedings. He has attended many conferences and workshops. He is a member various Professional organizations such as Nigerian Computer Society (NCS), Association for Computing Machinery (ACM) and Institute of Electrical and Electronics Engineers (IEEE), New York. He is actively involved in Postgraduate Supervision as he has supervised and currently supervising many Postgraduate Students both at the Masters and Ph.D. levels.

Mr. O.S. Adetunmbi, *MIEEE* received his Bachelor of Technology and Master of Technology degrees in Computer Science from the Federal University of Technology in 1994 and 2000 respectively. Currently, he is a Lecturer at the Department of Computer Science, Federal University of Technology, Akure. He worked with the Department of Computer Science, University of Ado – Ekiti, Nigeria from 2001 to 2004. He was a Post Graduate Research Fellow of the Institute of Computing Technology, China Academy of Sciences, Beijing China. His Research interests are Information Security, Machine learning and natural Language Processing. He is a graduate student Member of IEEE.

SUGGESTED CITATION

Akinwonmi, A.E., O.S. Adewale, B.K. Alese, and O.S. Adetunmbi. 2008. "Design of a Neural network Based Optical Character Recognition System for Musical Notes". *Pacific Journal of Science and Technology*. 9(1):45-58.