The Mind Project

Team: 48
School: LAS CRUCES HIGH
Area of Science: Artificial intelligence
Team Members: Jeremy Wilson
Sponsoring Teacher: Marez
E-Mail: jwilson18node@yahoo.com

Table of Contents

1. Introduction.
2. Fig. 1, 2, 3.
3. Fig 4.
4. FFT.
9. Further development.
13. Source code.

Executive Summary

In the field of artificial intelligence the goal has always been if not preoccupied with, trying to simulate the way in which the human brain operates. In doing so computer scientists as well as ordinary people would benefit immensely from the knowledge and help gained by such A.I.s. What people have imagined in the movies and in books has not, in fact, been for the most part realistic because most are not aware that true artificial intelligence will be like a newborn baby that must learn as we do if not faster. Its brain can not be pure logic and we can not program it with all the knowledge it would need. If the dream of artificial intelligence is to come true there must be a new approach to creating it. In the brain there are things in biological terms called neural networks. In the 80s neural networks were what most computer scientists believed would propel the field of artificial intelligence forward but nothing much happened and in the 90s they had all but been forgotten. What was wrong with the neural networks was not the neural networks themselves but the one thing that they left out. Time! Yes it was time that computer scientists had failed to include in their models. Most computer scientists only cared about how efficient they learned static patterns and not a dynamic flow of data the way the brain does. Some neural networks tried to model time but to do this they recorded all the data and then tried to find the patterns within them. This was O.K. for data of sound recordings and voice recognition but when it came to vision it was pointless. The amount of data involved would be enormous even for a supercomputer. Even then what algorithms would they use and what about operating in real time. This is the problem that I have tried to solve. To create a neural network that incorporates time but this was not what I had in mind when I first thought of how to solve the problem of artificial intelligence. The concept of time as a basic principal for my hypothesis did not occur until late into the competition of the Supercomputing Challenge in 2006. Before then I had to come up with the principles behind it. The principal that I learned came from a book called On Intelligence by Jeff Hawkins. It was that neural networks can only learn static data but by using hierarchy it could be more efficient. Six months later when I reread it I saw that only half the book was about hierarchy and the other half was about time. That was when I combined the concept of time with hierarchy. A Time Hierarchy! The Time Hierarchy as its name implies can take a sequence of data in time and reduce its information as in a hierarchy to where one element at the top of the hierarchy would represent the entire pattern. In this way it would know the information without recording it.

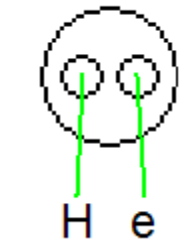
Introduction

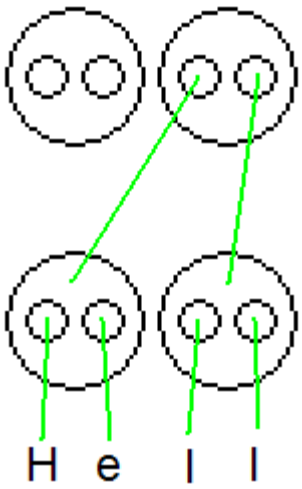
Proposal

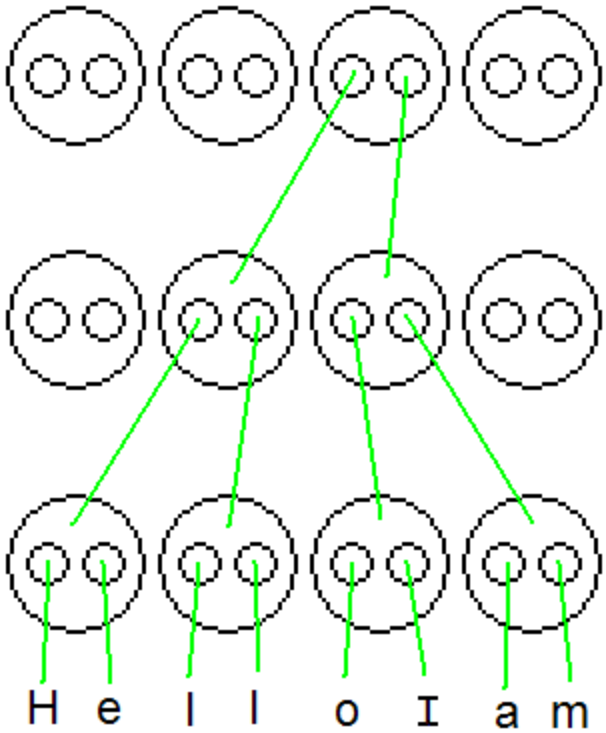
Problem:
Many Artificial intelligence programs today work on the principle that all the knowledge in them is preprogrammed. Information is not learned but imposed upon them. Even in neural networks the right answers must be known before training can take place and unsupervised neural networks do not always produce the right results that can be used in a meaningful way. Then there is the fact that real world problems such as vocal communication between man and machine can not be fully implemented by neural networks alone. They must have an expert system behind them doing all the logic.

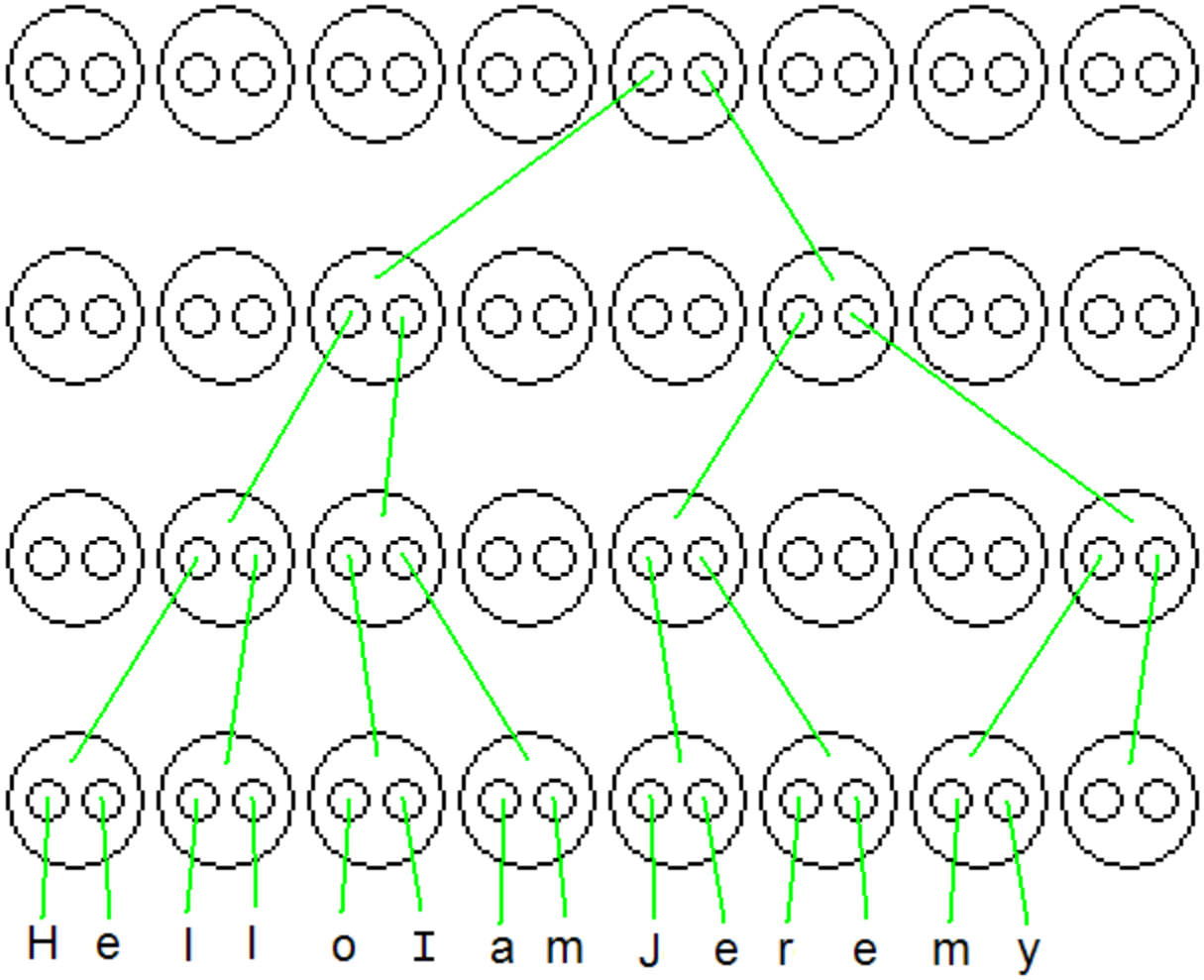
Purpose:
The purpose of this project is to create a type of neural network capable of performing all the tasks of traditional A.I. program without the burden of a cumbersome expert system.
This new Area of A.I. will be called Pure Neural Network A.I.

Procedure:
To create our neural network we will use the programming language Java. The data we will use to test our program will come from a WAVE sound file. Then we will transform it with an FFT. The frequency data will be input into the neural network. The output of the neural network will be turned back into sound with a reverse FFT.


H e


H e l l


H e l l o I a m


H e l l o I a m J e r e m y

My program implements two of these Time Hierarchy Neural Networks. One is used to learn patterns the other is used to reproduce the patterns by revising the direction of which the weights activate the nodes. Each neural network consists of 32 layers with 256 nodes per layer. The number of layers in the neural network is equal to the length of a sequence that can be learned.

Power of 2.
1/86 seconds.

2 power of 2 = 1/21.5 seconds

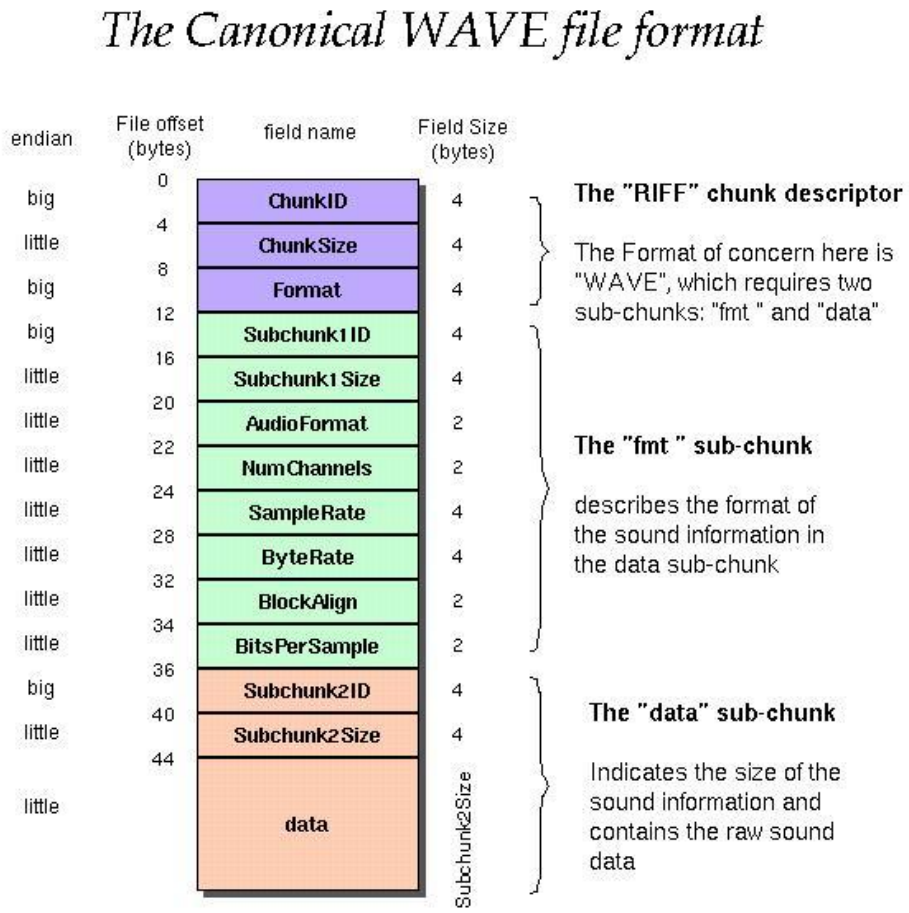
2 power of 4 = 1/5.375 seconds

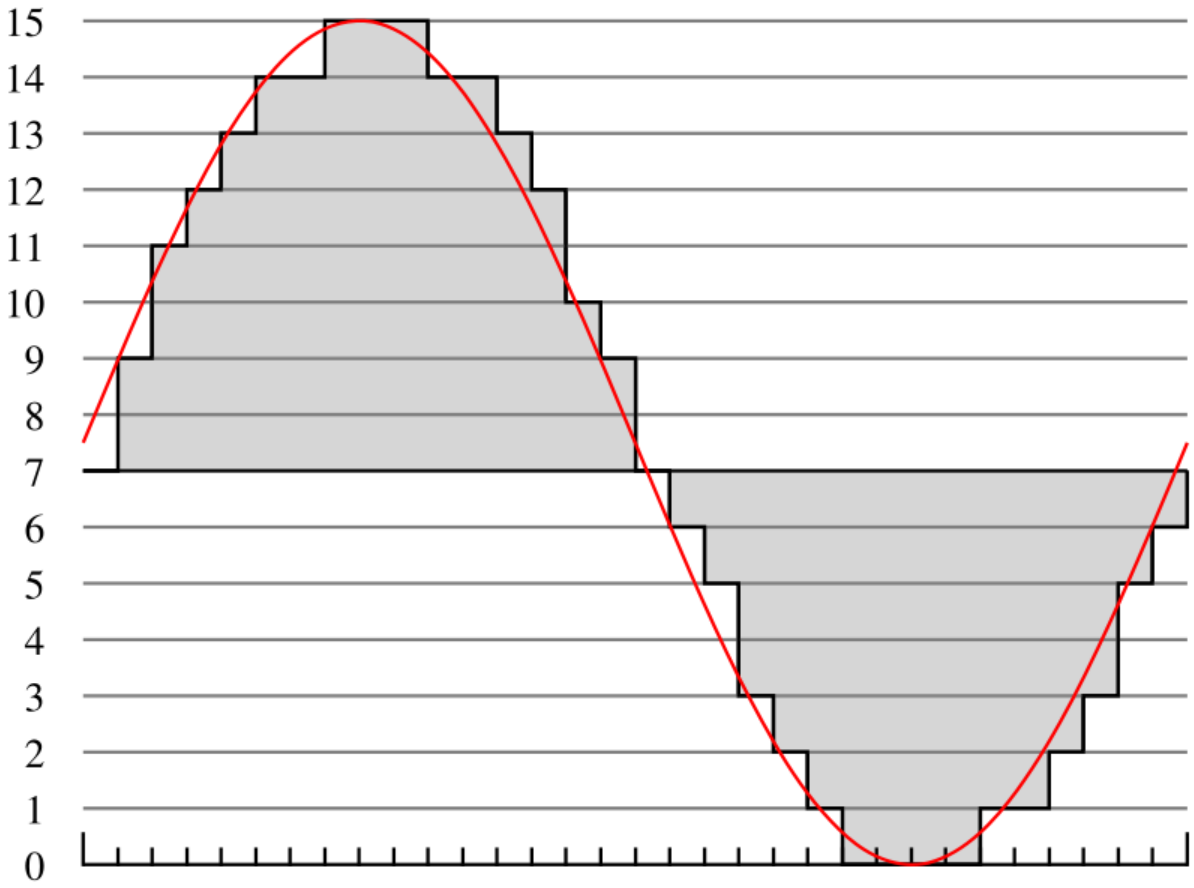
2 power of 8 = 2.97 seconds

2 power of 16 = 12.7 minutes

2 power of 32 = 1.583 years

The patterns that my program uses come from a WAVE file with 44000 samples a second.





512 of the 44000 samples are transformed though an FFT into amplitude values of 256 frequencies every 86th of a second.

44100 samples a second
1/86 second = 512 samples = 86 Hrz

44100/2=22050Hrz

44100/4=11025Hrz

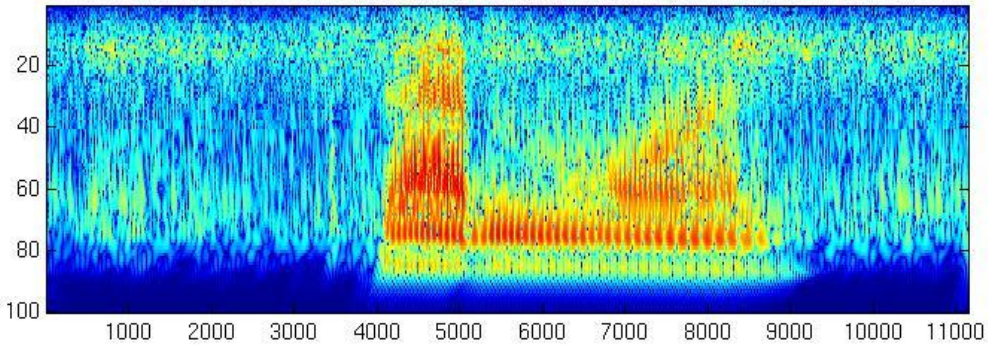
44100/6=7350Hrz

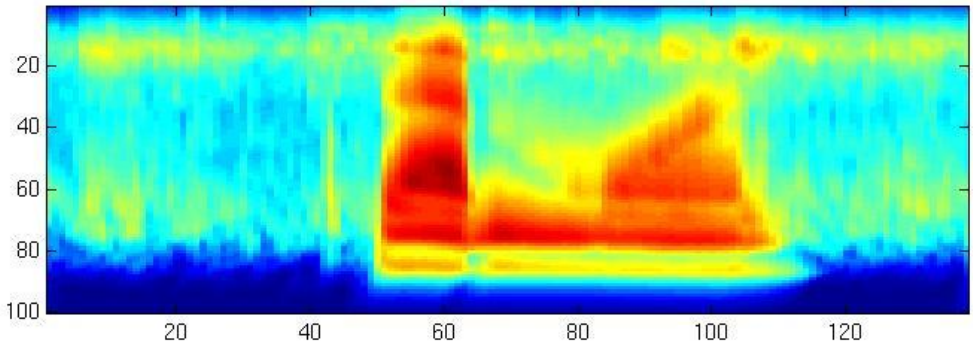
44100/8=5512.5Hrz

44100/10=4410Hrz

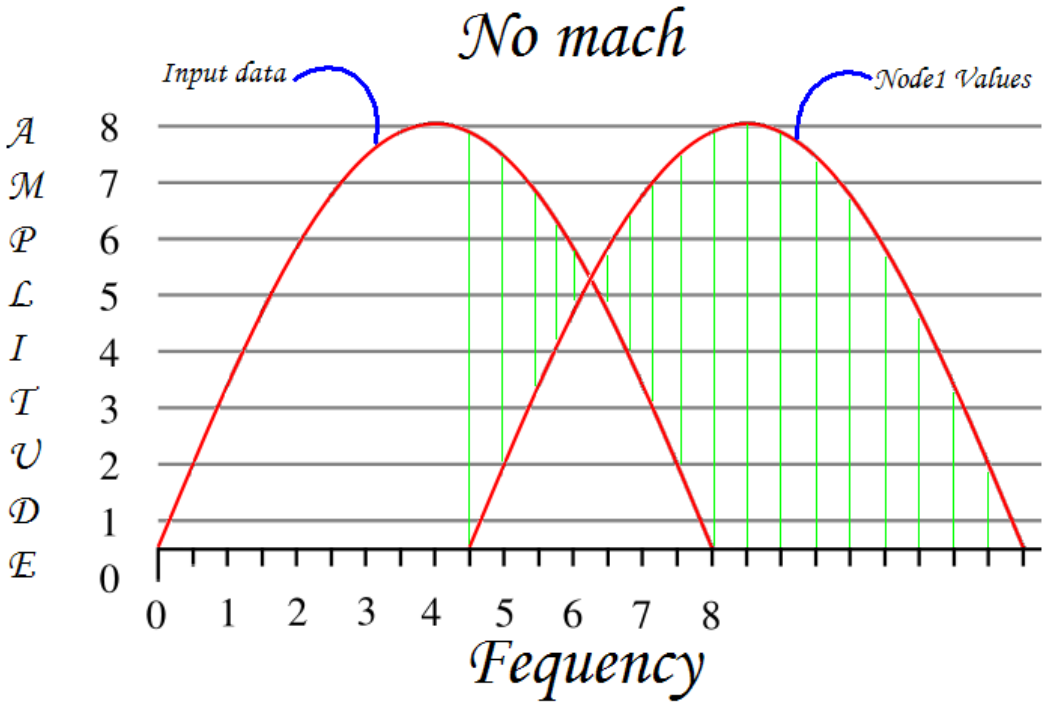
44100/12=3675Hrz

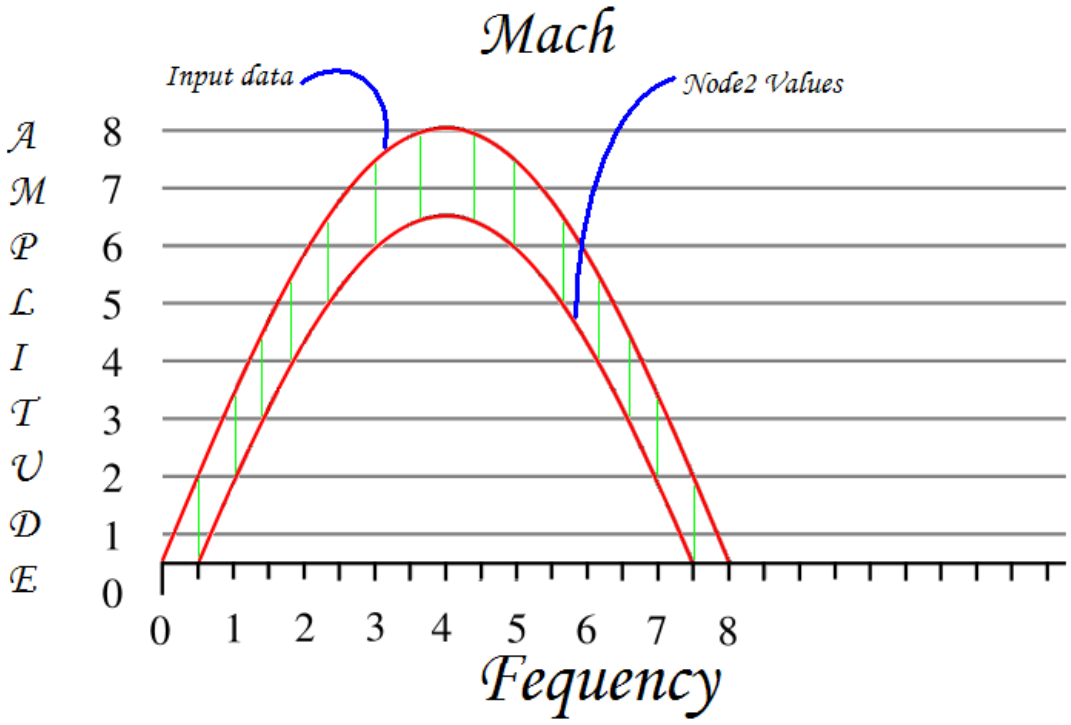
44100/512=86Hrz


20
40
60
80
100
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000

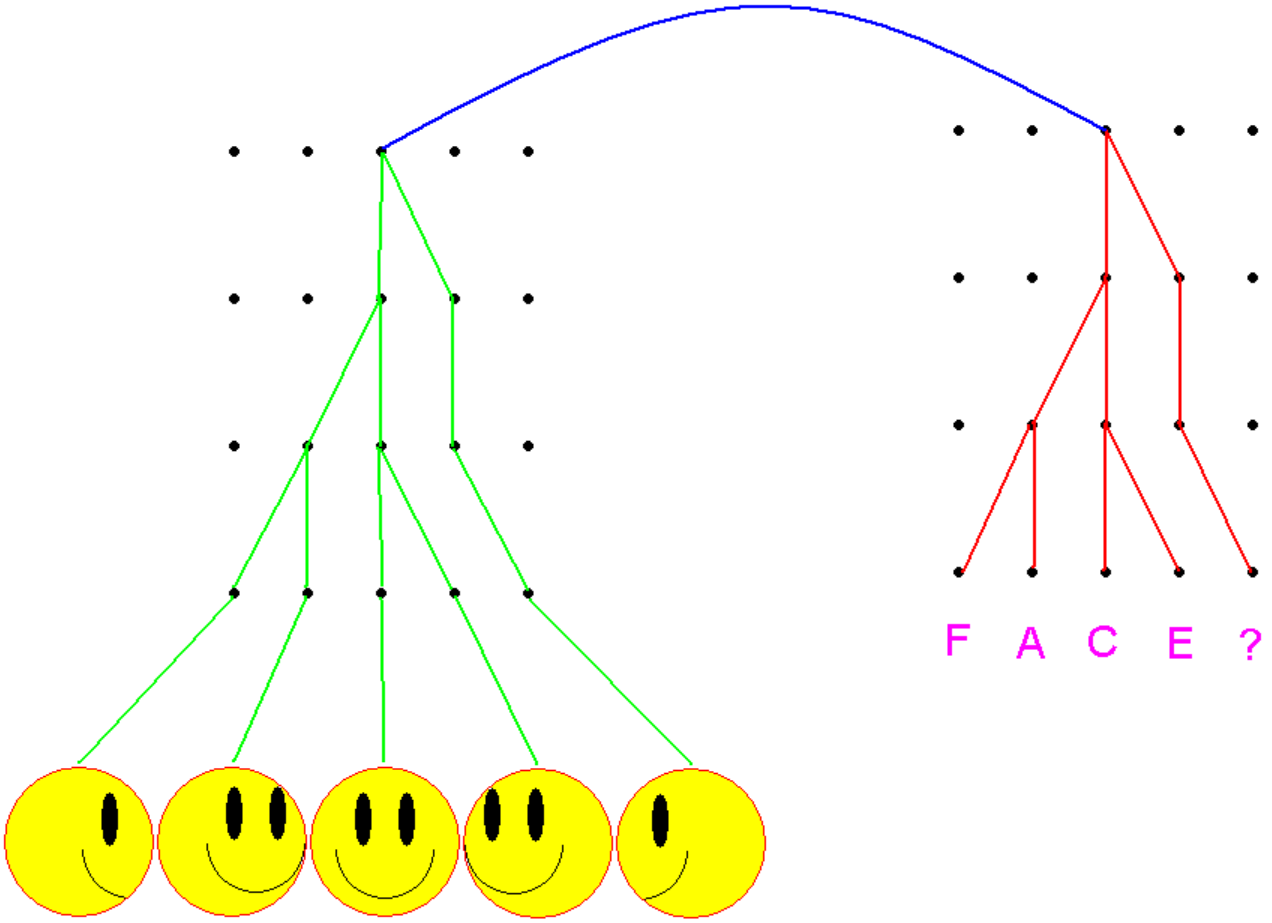

20
40
60
80
100
20
40
60
80
100
120

In the first layer of there neural network one node is selected to be on if its weights best mach the amplitude values of the FFT frequency data. Then the nodes weights are adjusted to better mach that FFT frequency data.





The reason that I have used sound instead of visual data is due to my limited abilities. I feel that that the true potential of this type of neural network can only be accomplished by combining a sound neural network with a vision neural network were as it can give an audio response to visual input.



Another application is robotics.
The retina of a robot would only be limited by memory.

size of retina equals # of nodes	*# 0f connection of a node equals # of pixles in retina*	*# of layers in neural network*
100*100*	100*100*	32*4

1 conection = 1 Byte

5*5*5*5*32*4
8KB

7*7*7*7*32*4
307.328KB

10*10*10*10*32*4
1.28MB

15*15*15*15*32*4
6.48MB

20*20*20*20*32*4
20.48MB

25*25*25*25*32*4
50MB

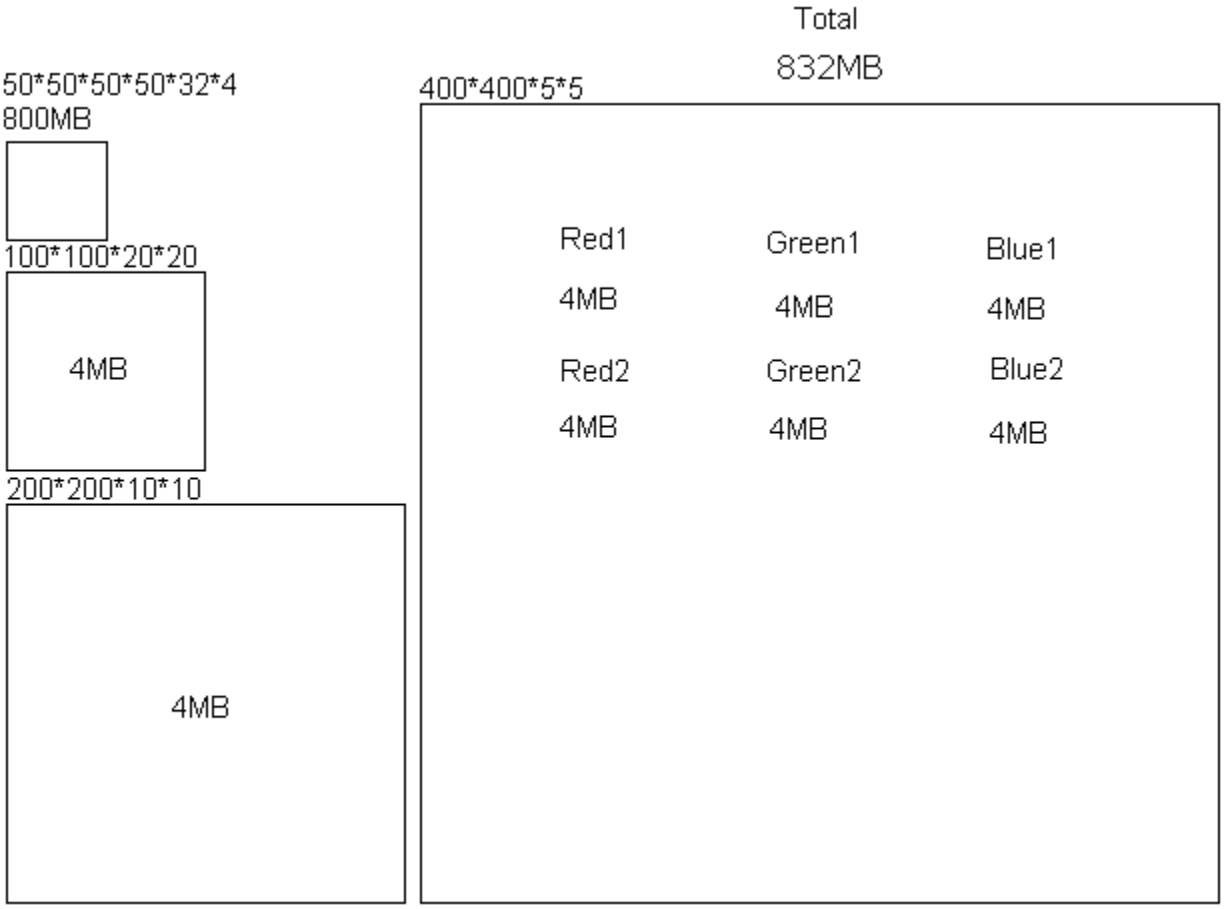
30*30*30*30*32*4
103.68MB

50*50*50*50*32*4
800MB

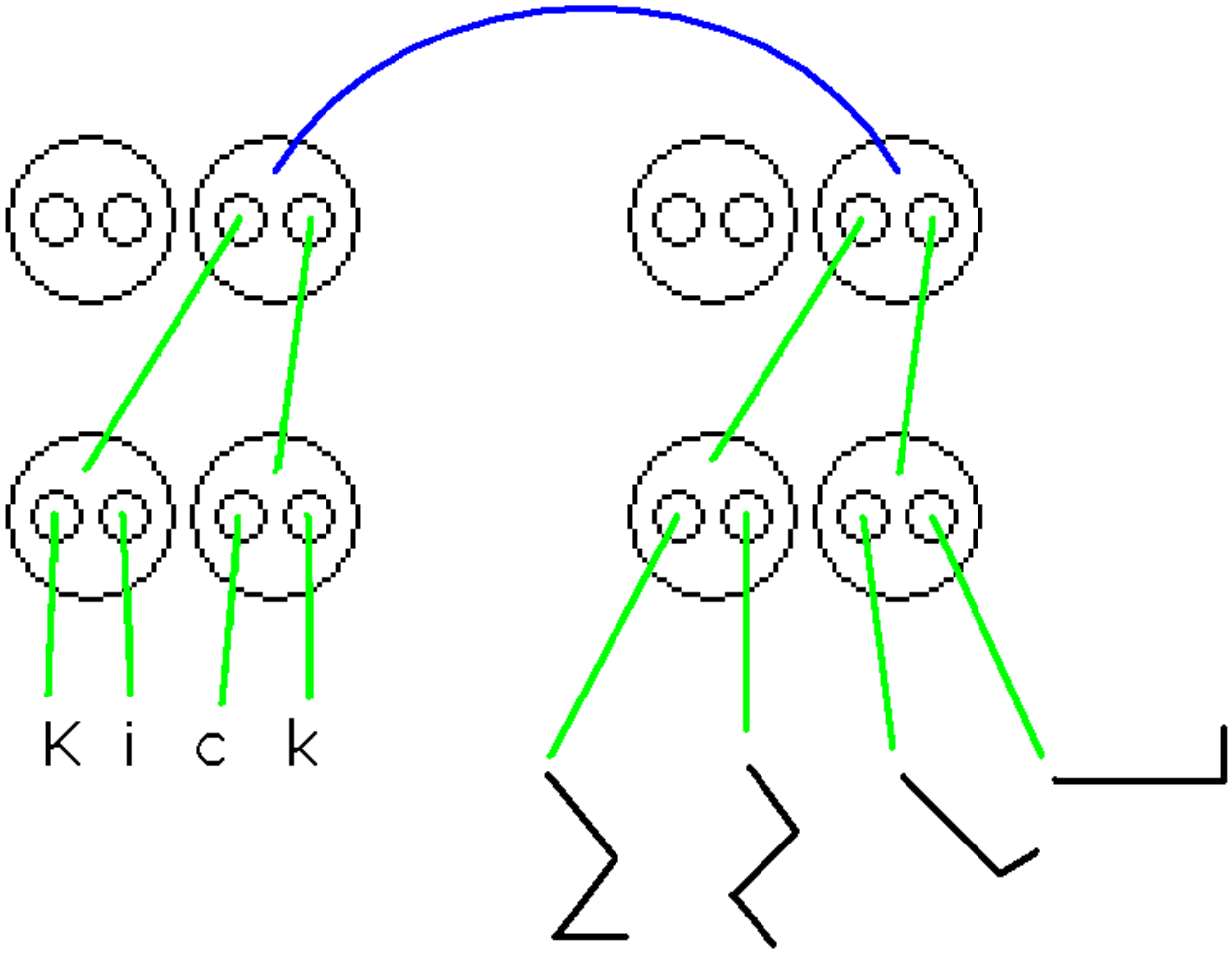
70*70*70*70*32*4
3.07328GB

100*100*100*100*32*4
12.8GB

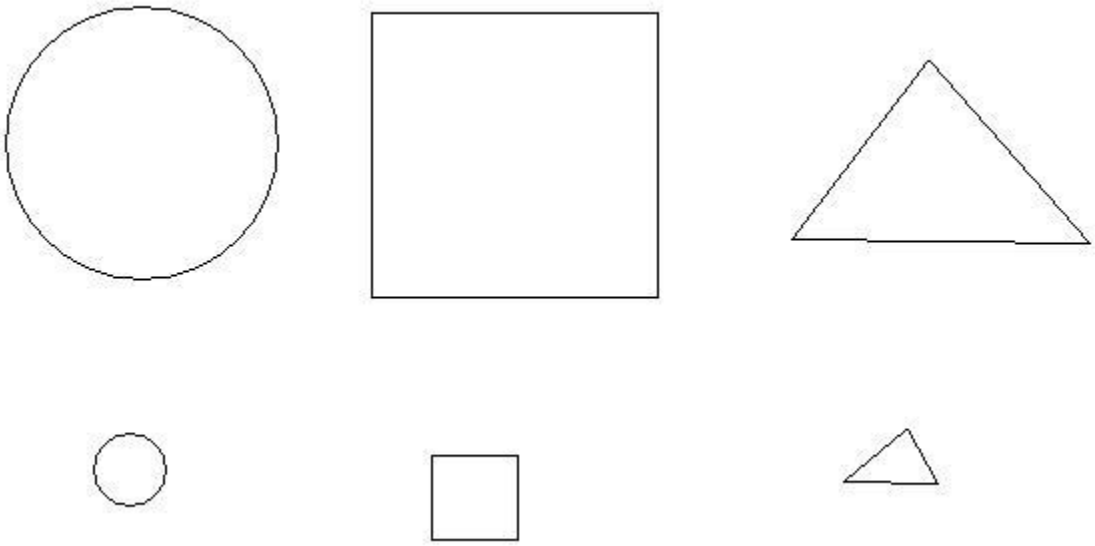
In the brain the average brain cell has 10,000 connections. For a robot with a big retina but limited memory there would be a sub neural network before the main neural network.



The Time Hierarchy Neural Network could also be used to control a robot.
A robot generates random movements.
A person tells the robot what it is doing.
The robot learns to associate the person's words with its own movements.



Genetic algorithms could also be used to find a desired set of weights values that would allow the neural network to respond to stimuli with sequences of patterns. When the neural network responds to the stimuli it will change its environment creating new stimuli to respond to creating a feedback loop of sequences instead of static patterns in normal neural networks.



Source code

```
import java.awt.*;
import javax.swing.*;
import java.applet.Applet;
import java.text.DecimalFormat;
import java.awt.event.*;

/*
 * AIProgram1.java
 * @author Jeremy Wilson
 * Created on March 1, 2007, 7:00 AM
 *
 */

public class AIProgram1 extends JFrame{
    int WAVEdata[],WAVEdataforFFT[];
    private JButton Start, Stop, Array;
    private TextArea display = new TextArea(10,40);
    long currentTime=0;

    Thread n1 = new Thread();
    Thread n2 = new Thread();
    Thread n3 = new Thread();

    //  n1.Time();
    //  n2.tick();
    //  n3.paint();

    public AIProgram1() {
        super( "A.I. Program 1.0" );

        Container container = getContentPane();
        container.setLayout( new FlowLayout() );
        Start = new JButton( "Start" );
        container.add( Start );
        Stop = new JButton( "Stop" );
        container.add( Stop );
        ButtonHandler handler = new ButtonHandler();
        Start.addActionListener( handler );
        Stop.addActionListener( handler );
```

```
        setSize(700,700);
        setVisible( true );
    }


    // public void tick() {
    //     long tick=0;

    //     for (long x=tick; x<tick+22; x++){
    //          currentTime = System.currentTimeMillis();

    //          if (currentTime == tick + 11)
    //               tick = currentTime;
    //      }
    //   }

    public void paint( Graphics g ) {
        super.paint(g);

        currentTime = System.currentTimeMillis();
        g.drawString("A speaker and a microphone is needed
to run this Application", 100, 100);
        g.drawString("Current Time " + currentTime, 100,
125);
        g.drawString("Sum of node wieghts", 100, 190);

        int number = 0;
        for (int x=100; x<612; x++){
            for (int y=200; y<264; y++){
                number = (int)(Math.random()*255);
                g.setColor(new
Color(number,number,number));
                g.fillRect(x,y,2,2);
            }
        }

        g.drawString("Node values One or Zero", 100, 290);
        int number2 = 0;
        for (int x=100; x<612; x++){
            for (int y=300; y<364; y++){
                number2 = (int)(Math.random()*2);
                if (number2 == 1)
                    g.setColor(new Color(255,255,255));
                else
                    g.setColor(new Color(0,0,0));
                g.fillRect(x,y,2,2);
            }
```

```
        }
    }
    //end of paint

    public static void main(String argv[]) {
        AIProgram1 application = new AIProgram1();
        application.setDefaultCloseOperation(
JFrame.EXIT_ON_CLOSE );
    }
    //end of main

    private class ButtonHandler implements ActionListener {
        public void actionPerformed( ActionEvent event ) {
            if (event.getActionCommand().equals("Stop"))
                System.exit(0);

            ReadWAVEdata RWD = new ReadWAVEdata();
            FFT fft = new FFT();
            //SoundNeuralNetwork SNN = new
SoundNeuralNetwork();

           // Time time = new Time();

           // n1.time(tick);

            int a=44;
            //if (currentTime%11==0);
            //a=a+512;
            // if (currentTime%11==0);


            RWD.ReadWAVEdata(size);

            int Wavedata[];
            Wavedata = new int[size];


            RWD.read(Wavedata);

            double realbitsamples[];
            double imaginarybitsamples[];
            fft.doFFT(realbitsamples,imaginarybitsamples);

            for(int x=a; x<a+512; x++) {
                realbitsamples [x] = Wavedata [x];
            }
```

```
            //if (currentTime%11==0);
            //fft.doFFT();
            //if (currentTime%11==0);
            //SNN.inputFrequencyData();
            //if (currentTime%11==0);
            //SNN.Do_soundneuralnetwork();

        }
        //end of actionPerformed
    }
//end of ButtonHandler
}
//end of AIProgram1
```

```
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

public class ReadWAVEdata {

    int WAVEdata[];
    int size=0;

    public ReadWAVEdata(){
        ReadWAVEdata application = new ReadWAVEdata();
        File name = new File("L:\\New A.I\\bug.wav");
        if ( name.exists())
            try{
                BufferedReader input = new BufferedReader(new
FileReader(name));
            }
            // process file processing problems
            catch( IOException ioException ) {
                JOptionPane.showMessageDialog( null, "FILE ERROR",
                        "FILE ERROR", JOptionPane.ERROR_MESSAGE );
            } else {
            JOptionPane.showMessageDialog( null,"file Does Not Exist",
                    "ERROR", JOptionPane.ERROR_MESSAGE );
            }
        WAVEdata = new int[input.length];
      //  size = input.length;
    }
//end of constuctor
    public void read(int[] WAVEdata) {
        for (int i=0; i<input.length; i++) {
            WAVEdata [i] = input [i];
        }
       // return WAVEdata[];
    }
}
```

```
// This code is repackaged after the code from Craig A.
Lindley, from Digital Audio with Java
// Site
ftp://ftp.prenhall.com/pub/ptr/professional_computer_scienc
e.w-022/digital_audio/
// Email
//package org.jscience.media.audio.dsp.monitors;

/**
 * This is a Java implementation of the fast Fourier
transform written by Jef
 * Poskanzer. The copyright appears above.
 */

/* libfft.c - fast Fourier transform library
 **
 ** Copyright (C) 1989 by Jef Poskanzer.
 **
 ** Permission to use, copy, modify, and distribute this
software and its
 ** documentation for any purpose and without fee is hereby
granted, provided
 ** that the above copyright notice appear in all copies
and that both that
 ** copyright notice and this permission notice appear in
supporting
 ** documentation.  This software is provided "as is"
without express or
 ** implied warranty.
 */
public class FFT {

    int bitspersample=8;
    int bitreverse[];
    double realbitsamples[], imaginarybitsamples[];
    private static final double TWOPI = 2.0 * Math.PI;

    public FFT() {
        bitreverse = new int[512];
        realbitsamples = new double [512];
        imaginarybitsamples = new double [512];

        for (int i = (1 << bitspersample) - 1; i >= 0; --i)
{
            int k = 0;

            for (int j = 0; j < bitspersample; ++j) {
```

```
                k *= 2;

                if ((i & (1 << j)) != 0) {
                    k++;
                }
            }
            bitreverse[i] = k;
        }
    }
//end of constuctor
    public void doFFT(double[] realbitsamples, double[]
imaginarybitsamples) {
        int n;
        int n2;
        int i;
        int k;
        int kn2;
        int l;
        int p;
        double ang;
        double s;
        double c;
        double tr;
        double ti;

        n2 = (n = (1 << bitspersample)) / 2;

        for (l = 0; l < bitspersample; ++l) {
            for (k = 0; k < n; k += n2) {
                for (i = 0; i < n2; ++i, ++k) {
                    p = bitreverse[k / n2];
                    ang = (TWOPI * p) / n;
                    c = Math.cos(ang);
                    s = Math.sin(ang);
                    kn2 = k + n2;

                    tr = (realbitsamples[kn2] * c) +
(imaginarybitsamples[kn2] * s);
                    ti = (imaginarybitsamples[kn2] * c) -
(realbitsamples[kn2] * s);

                    realbitsamples[kn2] = realbitsamples[k]
- tr;
                    imaginarybitsamples[kn2] =
imaginarybitsamples[k] - ti;
                    realbitsamples[k] += tr;
                    imaginarybitsamples[k] += ti;
```

```
                }
            }
            n2 /= 2;
        }
        for (k = 0; k < n; k++) {
            if ((i = bitreverse[k]) <= k) {
                continue;
            }

            tr = realbitsamples[k];
            ti = imaginarybitsamples[k];
            realbitsamples[k] = realbitsamples[i];
            imaginarybitsamples[k] =
imaginarybitsamples[i];
            realbitsamples[i] = tr;
            imaginarybitsamples[i] = ti;
        }
    }
//end of doFFT
}
```

```
import java.util.Arrays;
import javax.swing.JOptionPane;
import java.text.DecimalFormat;

public class SoundNeuralNetwork{

    double n[];
    int timestep[][];
    int inhibitnow;
    int inhibiter1[][];
    int suminhibiter1=0;
    int switchStrengths[][];
    int frequency[],sumfrequencies[];
    int nodeValue1[][],nodeValue2[][];
    int strengths1[][][],strengths2[][][];
    int sumStrengths1[][],sumStrengths2[][];
    int sumStrengths3[][],sumStrengths4[][];

    /** Creates a new instance of SoundNeuralNetwork */
    public SoundNeuralNetwork(){
        n = new double[32];
        timestep = new int[32][256];
        frequency = new int[256];
        sumfrequencies = new int[256];
        switchStrengths = new int[32][256];
        nodeValue1 = new int[32][256];
        nodeValue2 = new int[32][256];
        strengths1 = new int[32][256][256];
        strengths2 = new int[32][256][256];
        sumStrengths1 = new int[32][256];
        sumStrengths2 = new int[32][256];
        sumStrengths3 = new int[32][256];
        sumStrengths4 = new int[32][256];
        inhibiter1 = new int[32][256];

        for(int x=0; x<32; x++) {
            for(int y=0; y<256; y++) {
                for(int z=0; z<256; z++) {
                    n [x] = Math.pow(2, x);
                    inhibiter1 [x][y] = 0;
                    switchStrengths [x][y] = 0;
                    strengths1 [x][y][z] = (int)
(Math.random() *2);
                    strengths2 [x][y][z] = (int)
(Math.random() *2);
                    if (strengths1 [x][y][z] == 0)
                        strengths1 [x][y][z] = -1;
```

```
                    if (strengths2 [x][y][z] == 0)
                        strengths2 [x][y][z] = -1;
                }
            }
        }
    }
//end of constuctor
    public void inputFrequencyData(){
        int FFTfrequency[];
        FFTfrequency = new int[256];
        for(int x=0; x<256; x++) {
            frequency [x] = FFTfrequency [x];
            sumfrequencies [x] = 0;
        }
    }
//end of inputFrequencyData
    public void Do_soundneuralnetwork(){
//------sum of frequencies
        for(int y=0; y<256; y++) {
            for(int z=0; z<256; z++) {
                if (frequency [z] > strengths1 [0][y][z])
                    sumfrequencies [y] = frequency [z] -
strengths1 [0][y][z];
                if (frequency [z] < strengths1 [0][y][z])
                    sumfrequencies [y] = strengths1
[0][y][z] - frequency [z];
            }
        }
//------use frequency data
        int on, on2;
        on = sumfrequencies [0];
        on2 = 0;
        for(int y=0; y<256; y++) {
            nodeValue1 [0][y] = 0;
            sumStrengths2 [0][y] = 0;
        }
        for(int x=0; x<256; x++) {
            if (sumfrequencies [x] < on)
                on2 = x;
            if (sumfrequencies [x] < on)
                on = sumfrequencies [x];
        }
        on2 = (int) (Math.random() *256);
        nodeValue1 [0][on2] = 1;
        sumStrengths2 [0][on2] = 1;
        for(int x=0; x<256; x++) {
            for(int y=0; y<256; y++) {
```

```
                if ((nodeValue1 [0][x] == 1)&&(frequency
[y] > strengths1 [0][x][y]))
                    strengths1 [0][x][y] = strengths1
[0][x][y] + 1;
                if ((nodeValue1 [0][x] == 1)&&(frequency
[y] < strengths1 [0][x][y]))
                    strengths1 [0][x][y] = strengths1
[0][x][y] - 1;
                }
        }
//------activate nodevalues2
        for(int x=30; x<31; x++) {
            for(int y=0; y<256; y++) {
                nodeValue2 [x][y] = nodeValue1 [x][y];
            }
        }
//------sum of strengths
        for(int x=1; x<31; x++) {
            for(int y=0; y<256; y++) {
                sumStrengths1 [x][y] = 0;
                sumStrengths2 [x][y] = 0;
                sumStrengths3 [x][y] = 0;
                sumStrengths4 [x][y] = 0;
                for(int z=0; z<256; z++) {
                    if (nodeValue1 [x-1][z] == 1)
                        sumStrengths1 [x][y] =
sumStrengths1 [x][y] + strengths1 [x][y][z];
                    if (nodeValue1 [x-1][z] == 1)
                        sumStrengths2 [x][y] =
sumStrengths2 [x][y] + strengths2 [x][y][z];
                    if ((nodeValue2 [x+1][z] ==
1)&&(switchStrengths [x+1][z] == 0))
                        sumStrengths3 [x][y] =
sumStrengths3 [x][y] + strengths1 [x+1][y][z];
                    if ((nodeValue2 [x+1][z] ==
1)&&(switchStrengths [x+1][z] == 1))
                        sumStrengths4 [x][y] =
sumStrengths4 [x][y] + strengths2 [x+1][y][z];
                }
//--------------change strengths
                for(int z=0; z<256; z++) {
                    if ((sumStrengths1 [x][y] > 0)&&
                            (sumStrengths2 [x-1][z] > 0))
                        strengths1 [x][y][z] = strengths1
[x][y][z] + 1;

                    if ((sumStrengths1 [x][y] > 0)&&
```

```
                            (sumStrengths2 [x-1][z] <= 0))
                        strengths1 [x][y][z] = strengths1
[x][y][z] - 1;

                    if ((sumStrengths2 [x][y] > 0)&&
                            (sumStrengths2 [x-1][z] > 0))
                        strengths2 [x][y][z] = strengths2
[x][y][z] + 1;

                    if ((sumStrengths2 [x][y] > 0)&&
                            (sumStrengths2 [x-1][z] <= 0))
                        strengths2 [x][y][z] = strengths2
[x][y][z] - 1;
                }
//--------------find nodevalues
                if (sumStrengths1 [x][y] > 0)
                    nodeValue1 [x][y] = 1;
                if ((sumStrengths1 [x][y] <=
0)&&(sumStrengths2 [x][y] <= 0))
                    nodeValue1 [x][y] = 0;
                if ((sumStrengths3 [x][y] + sumStrengths4
[x][y]) > 0)
                    nodeValue2 [x][y] = 1;
                if ((sumStrengths3 [x][y] + sumStrengths4
[x][y]) <= 0)
                    nodeValue2 [x][y] = 0;
            }
        }
//------switch strengths
        for(int x=0; x<32; x++) {
            for(int y=0; y<256; y++) {
                if (nodeValue2 [x][y] == 1)
                    timestep [x][y] = timestep [x][y] + 1;
                if (nodeValue2 [x][y] == 0)
                    timestep [x][y] = 0;
            }
        }
        for(int x=0; x<32; x++) {
            for(int y=0; y<256; y++) {
                for(int z=0; z<256; z++) {
                    switchStrengths [x][y] = 0;
                    if ((timestep [x][y] % n [x] ==
0)&&(switchStrengths [x][y] == 0))
                        switchStrengths [x][y] = 1;
                    if ((timestep [x][y] % n [x] ==
0)&&(switchStrengths [x][y] == 0))
                        timestep [x][y] = 0;
```

```
                }
            }
        }
//------inhibit output
        int output[];
        output = new int[256];
        for(int x=0; x<10; x++) {
            for (int y=0; y<256; y++) {
                if (nodeValue2 [0][x] == 1)
                    output [y] = strengths1 [0][x][y];
            }
            for(int y=0; y<256; y++) {
                if (nodeValue1 [x][y] == 1)
                    suminhibiter1 = suminhibiter1 +
inhibiter1 [x][y];
            }
            for(int y=0; y<256; y++) {
                if ((inhibitnow == 1)&&(nodeValue1 [x][y]
== 1))
                    inhibiter1 [x][y] = inhibiter1 [x][y] -
1;
                if ((inhibitnow == 0)&&(nodeValue1 [x][y]
== 1))
                    inhibiter1 [x][y] = inhibiter1 [x][y] +
1;
            }
        }
        for(int x=0; x<256; x++) {
            if (suminhibiter1 < 0)
                output [x] = 0;
        }
    }
//end of Do_soundneuralnetwork
//
// -------------------------- Display test data -----------
---------------------
//
    public void display_test_data(){
        String output = "\n";
        DecimalFormat twoDigits = new  DecimalFormat("0");
        for(int x=0; x<10; x++) {
            for(int y=0; y<40; y++) {
                output += " " +
twoDigits.format(switchStrengths [x][y]);
                // nodeValue1 [x][y]
                // strengths1 [x][y][z]
                // sumStrengths1 [x][y]
```

```
                // switchStrengths [x][y]
                // timestep [x][y]
                // n [x]
                if (y == 39)
                    output +="\n";
            }
        }
        JOptionPane.showMessageDialog(null,output,
                "The product",
JOptionPane.INFORMATION_MESSAGE);
    }
}
```

```
public class Time {
    long currentTime=0;

    public Time() {
    }

    public void tick() {
        long tick=0;

        for (long x=tick; x<tick+22; x++){
            currentTime = System.currentTimeMillis();

            if (currentTime == tick + 11)
                tick = currentTime;
        }
    }

}
```