**

**RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT INDONESIA MENGGUNAKAN TEKNOLOGI KINECT DAN METODE BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK**

TUGAS AKHIR - KI141502

Yohanes Aditya Sutanto

NRP 5112100135

Dosen Pembimbing

Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya 2016

*[Halaman ini sengaja dikosongkan]*

**

**RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT INDONESIA MENGGUNAKAN TEKNOLOGI KINECT DAN METODE BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK**

Yohanes Aditya Sutanto

NRP 5112100135

Dosen Pembimbing

Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya 2016

TUGAS AKHIR - KI141502

*[Halaman ini sengaja dikosongkan]*

**

**DESIGN AND IMPLEMENTATION OF INDONESIA SIGN LANGUAGE MODULE USING KINECT TECHNOLOGY AND BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK METHOD**

YOHANES ADITYA SUTANTO

NRP 5112100135

Advisor

Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

INFORMATICS DEPARTMENT

Faculty of Information Technology

Institut Teknologi Sepuluh Nopember

Surabaya 2016

Su

FINAL PROJECT - KI141502

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

**RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI KINECT DAN METODE *BACKPROPAGATION*-*GENETIC* *ALGORITHM* *NEURAL* *NETWORK***

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

pada

Bidang Studi Interaksi Grafis dan Seni

Program Studi S-1 Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Oleh:

**YOHANES ADITYA SUTANTO**

NRP. 5112100135

Disetujui oleh Pembimbing Tugas Akhir:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Wijayanti Nurul Khotimah, S.Kom., M.Sc. | | …………………… |
|  | NIP: 19860312 201212 2 004 | (pembimbing 1) | |
|  |  |  | |
| 2. | Dr.Eng. Nanik Suciati, S.Kom., M.Kom. | | …………………… |
|  | NIP: 19870213 201404 1 001 | (pembimbing 2) | |

**SURABAYA**

**JUNI, 2016**

*[Halaman ini sengaja dikosongkan]*

**RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI KINECT DAN METODE BACKPROPAGATION-GENETIC ALGORITHM NEURAL NETWORK**

Nama Mahasiswa : Yohanes Aditya Sutanto

NRP : 5112100135

Jurusan : Teknik Informatika FTIf-ITS

Dosen Pembimbing I : Wijayanti Nurul Khotimah, S.Kom, M.Sc.

**Dosen Pembimbing II :** **Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**

# Abstrak

*Bahasa isyarat adalah hal yang penting dalam komunikasi bagi orang yang menderita gangguan pendengaran. Kecepatan menguasai bahasa dan kemampuan mereka berinteraksi sangat dibutuhkan.**Mereka membutuhkan bahan pembelajaran yang tidak hanya berisi tentang komponen aural saja, namun juga secara visual karena lebih nyata.*

*Di sisi lain, teknologi berkembang pesat di segala aspek kehidupan. Berbagai macam terobosan teknologi baru telah diciptakan oleh manusia, salah satunya perangkat Kinect Sensor yang diciptakan untuk windows sekitar tahun 2012. Dengan menggunakan Kinect Sensor, manusia dapat melakukan interaksi dengan komputer tanpa sentuhan sama sekali.*

*Oleh karena itu, penulis memiliki ide untuk membuat aplikasi modul pengenalan Bahasa isyarat Indonesia berdasarkan Sistem Bahasa Isyarat Indonesia (SIBI) dengan menggunakan Kinect Sensor. Harapan aplikasi ini agar dapat membantu pengguna untuk belajar mengenai bahasa isyarat.*

Kata kunci: Kinect Sensor, bahasa isyarat, SIBI.

*[Halaman ini sengaja dikosongkan]*

DESIGN AND IMPLEMENTATION OF INDONESIA SIGN LANGUAGE MODULE USING KINECT TECHNOLOGY AND BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK METHOD

Name : Yohanes Aditya Sutanto

NRP : 5112100135

Major : Informatics Department, FTIf-ITS

Advisor I : Wijayanti Nurul Khotimah, S.Kom, M.Sc.

Advisor II : Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

# Abstract

Sign language is an important thing in the communication of deaf people. The speed of learning sign language and their skills to interact each other is very necessary. They need a media and the learning materials is not only about aural matter, but also visual.

On the other hand, technology growth very quickly. The new Many technology are invented, one of them is Leap Motion which invented in 2013. With Leap Motion Controller, human can interact with computer touchless.

Therefore, the author had the idea to create application that can be used to learn about Indonesian sign language based on Sistem Isyarat Bahasa Indonesia (SIBI). The expectation of this application is to help people know and learn about Indonesian sign language.

Keywords: : Kinect Sensor, Sign Language, SIBI

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

Puji dan syukur kepada Tuhan Yang Maha Esa atas rahmat dan kasih-Nya yang menyertai penulis selama proses pengerjaan tugas akhir ini sampai selesai.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang setinggi-tingginya kepada pihak-pihak yang telah membantu penulis dalam penyelesaian tugas akhir ini, terutama kepada: Kedua orang tua penulis, yang selalu mendukung penulis mulai dari awal kuliah sampai lulus.

1. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc. dan Ibu Dr.Eng. Nanik Suciati, S.Kom., M.Kom. yang telah bersedia untuk menjadi dosen pembimbing tugas akhir sehingga penulis dapat mengerjakan tugas akhir dengan arahan dan bimbingan yang baik dan jelas.
2. Teman-teman Mahasiswa Teknik Informatika 2012 yang telah berjuang bersama-sama selama menempuh pendidikan di Jurusan ini.
3. Kakak kelas Mahasiswa Teknik Informatika 2011 yang telah membantu dalam segala proses perkuliahan di TC ini.
4. Serta pihak-pihak lain yang turut membantu penulis baik secara langsung maupun tidak, yang namanya tidak penulis sebutkan disini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, mohon maaf apabila ada kesalahan dan kata-kata yang dapat menyinggung perasaan. Penulis berharap tugas akhir ini dapat menjadi media pembelajaran bahasa isyarat Indonesia.

Surabaya, Juni 2016

Penulis

# DAFTAR ISI

[BAB I PENDAHULUAN 1](#_Toc453251775)

[1.1 Latar Belakang 1](#_Toc453251776)

[1.2 Rumusan Permasalahan 2](#_Toc453251777)

[1.3 Batasan Permasalahan 2](#_Toc453251778)

[1.4 Tujuan 3](#_Toc453251779)

[1.5 Manfaat 3](#_Toc453251780)

[1.6 Metodologi 3](#_Toc453251781)

[1.7 Sistematika Penulisan 5](#_Toc453251782)

[BAB II DASAR TEORI 7](#_Toc453251783)

[2.1 Tunarungu 7](#_Toc453251786)

[2.2 Bahasa Isyarat 7](#_Toc453251787)

[2.3 Kinect 8](#_Toc453251788)

[2.4 Kinect SDK 10](#_Toc453251789)

[2.5 *Neural* *Network* 11](#_Toc453251790)

[2.6 *Backpropagation* 11](#_Toc453251791)

[2.7 *Backpropagation*-*Genetic* *Algorithm* *Neural* *Network* 13](#_Toc453251792)

[BAB III ANALISIS DAN PERANCANGAN SISTEM 15](#_Toc453251793)

[3.1 Analisis Perangkat Lunak 15](#_Toc453251796)

[3.1.1 Deskripsi Umum Perangkat Lunak 16](#_Toc453251797)

[3.1.2 Spesifikasi Kebutuhan Perangkat Lunak 16](#_Toc453251798)

[3.1.3 Identifikasi Pengguna 17](#_Toc453251799)

[3.2 Perancangan Perangkat Lunak 17](#_Toc453251800)

[3.2.1 Model Kasus Penggunaan 18](#_Toc453251801)

[3.2.2 Definisi Aktor 18](#_Toc453251802)

[3.2.3 Definisi Kasus Penggunaan 19](#_Toc453251803)

[3.2.4 Arsitektur Umum Sistem 23](#_Toc453251804)

[3.2.5 Rancangan Antarmuka Aplikasi 24](#_Toc453251805)

[3.2.6 Rancangan Proses Aplikasi 25](#_Toc453251806)

[BAB IV IMPLEMENTASI 29](#_Toc453251807)

[4.1 Lingkungan Pembangunan 29](#_Toc453251811)

[4.1.1 Lingkungan Pembangunan Perangkat Keras 29](#_Toc453251812)

[4.1.2 Lingkungan Pembangunan Perangkat Lunak 29](#_Toc453251813)

[4.2 Implementasi Antarmuka 29](#_Toc453251814)

[4.3 Implementasi Aplikasi 30](#_Toc453251815)

[4.3.1 Implementasi Pendeteksian Skeleton Pengguna 31](#_Toc453251816)

[4.3.2. Implementasi Proses Ekstraksi Fitur 32](#_Toc453251817)

[4.3.3 Implementasi Pembuatan Data Bahasa Isyarat Baru 37](#_Toc453251818)

[4.3.4 Implementasi Proses *Training* dan *Testing* 38](#_Toc453251819)

[4.3.4 Implementasi Proses Normalisasi Fitur 43](#_Toc453251820)

[4.3.6 Implementasi Proses Pembagian *Neural Network* 44](#_Toc453251821)

[BAB V PENGUJIAN DAN EVALUASI 47](#_Toc453251822)

[5.1. Lingkungan Pembangunan **Error! Bookmark not defined.**](#_Toc453251823)

[5.2. Skenario Pengujian 47](#_Toc453251824)

[5.2.1 Pengujian Skenario A1 dan Analisis 48](#_Toc453251825)

[5.2.2 Pengujian Skenario A2 dan Analisis 50](#_Toc453251826)

[5.2.3 Pengujian Skenario A3 dan Analisis 51](#_Toc453251827)

[5.2.4 Pengujian Skenario A4 dan Analisis 52](#_Toc453251828)

[5.2.5 Pengujian Skenario B1 dan Analisis **Error! Bookmark not defined.**](#_Toc453251829)

[5.2.6 Pengujian Skenario B2 dan Analisis **Error! Bookmark not defined.**](#_Toc453251830)

[5.2.7 Pengujian Skenario B3 dan Evaluasi **Error! Bookmark not defined.**](#_Toc453251831)

[5.2.8 Pengujian Skenario B4 dan Evaluasi **Error! Bookmark not defined.**](#_Toc453251832)

[5.3. Evaluasi 53](#_Toc453251833)

[5.3.1 Perbandingan Akurasi Model *Neural Network* Terhadap Algoritma yang Dipakai **Error! Bookmark not defined.**](#_Toc453251834)

[5.3.2 Perbandingan Akurasi Model Jika Ada Penambahan *Dataset* **Error! Bookmark not defined.**](#_Toc453251835)

[5.3.3 Perbandingan Akurasi Model Jika Menggunakan Kalibrasi **Error! Bookmark not defined.**](#_Toc453251836)

[5.3.4 Perbandingan Akurasi Model Terhadap Jumlah Iterasi **Error! Bookmark not defined.**](#_Toc453251837)

[5.3.5 Perbandingan *Runtime* Aplikasi Model *Neural Network* Terhadap Algoritma yang Dipakai **Error! Bookmark not defined.**](#_Toc453251838)

[BAB VI KESIMPULAN DAN SARAN 55](#_Toc453251839)

[6.1 Kesimpulan 55](#_Toc453251840)

[6.2 Saran 55](#_Toc453251841)

[Lampiran A Kode Sumber 58](#_Toc453251842)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR GAMBAR

[Gambar 2. 1 Leap Motion Controller **Error! Bookmark not defined.**](#_Toc423533208)

[Gambar 2. 2 Radius Deteksi Leap Motion Controller **Error! Bookmark not defined.**](#_Toc423533209)

[Gambar 2. 3 Contoh Jaringan Syaraf Feedforward 11](#_Toc423533210)

[Gambar 2. 4 Contoh Bahasa Isyarat Huruf **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533211)

[Gambar 3. 1 Diagram Kasus Penggunaan Aplikasi 18](#_Toc423533212)

[Gambar 3. 2 Arsitektur Sistem (*Training*) 23](#_Toc423533217)

[Gambar 3. 3 Arsitektur Sistem (*Testing*) 24](#_Toc423533218)

[Gambar 3. 4 Halaman Utama **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533219)

[Gambar 3. 5 Halaman *Training* **Error! Bookmark not defined.**](#_Toc423533220)

[Gambar 3. 6 Halaman Pengambilan Fitur Tangan Pengguna **Error! Bookmark not defined.**](#_Toc423533221)

[Gambar 3. 7 Halaman *Testing* **Error! Bookmark not defined.**](#_Toc423533222)

[Gambar 3. 8 Diagram Alir Mendeteksi Lokasi Objek **Error! Bookmark not defined.**](#_Toc423533223)

[Gambar 3. 9 Diagram Alir Ekstraksi Fitur Tangan Pengguna **Error! Bookmark not defined.**](#_Toc423533224)

[Gambar 3. 10Diagram Pohon Pembagian *Neural Network* **Error! Bookmark not defined.**](#_Toc423533225)

[Gambar 3. 11 *Pseudocode* Normalisasi Fitur **Error! Bookmark not defined.**](#_Toc423533227)

[Gambar 3. 12 *Pseudocode* Kalibrasi Tangan Pengguna **Error! Bookmark not defined.**](#_Toc423533228)

[Gambar 4. 1 Antarmuka Halaman Utama **Error! Bookmark not defined.**](#_Toc423533229)

[Gambar 4. 2 Antarmuka *Training* (Tabel Ekstraksi Fitur) **Error! Bookmark not defined.**](#_Toc423533230)

[Gambar 4. 3 Antarmuka *Training* (Pilihan *Training* Data) **Error! Bookmark not defined.**](#_Toc423533231)

[Gambar 4. 4 Antarmuka *Training* (Proses ekstraksi fitur) **Error! Bookmark not defined.**](#_Toc423533232)

[Gambar 4. 5 Antarmuka *Testing* **Error! Bookmark not defined.**](#_Toc423533233)

[Gambar 4. 6 Antarmuka *Testing* (Proses Ekstraksi Fitur) **Error! Bookmark not defined.**](#_Toc423533234)

[Gambar 4. 7 Antarmuka Kalibrasi **Error! Bookmark not defined.**](#_Toc423533235)

[Gambar 5. 1 Perbandingan Huruf A dan J **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533259)

[Gambar 5. 2 Perbandingan Huruf M dan N **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533260)

[Gambar 5. 3 Perbandingan Huruf Skenario Pengujian B1 **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533266)

[Gambar 5. 4 Perbandingan Huruf Skenario Pengujian B3 **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533272)

[Gambar 5. 5 Perbandingan Huruf I dan J **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533275)

[Gambar 5. 6 Perbandingan Huruf K dan V **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533276)

[Gambar 5. 7 Grafik Perbandingan Terhadap Algoritma **Error! Bookmark not defined.**](#_Toc423533277)

[Gambar 5. 8 Grafik Perbandingan Terhadap Proses Kalibrasi **Error! Bookmark not defined.**](#_Toc423533278)

[Gambar 5. 9 Grafik Perbandingan Terhadap Jumlah Iterasi **Error! Bookmark not defined.**](#_Toc423533279)

[Gambar 5. 10 Grafik Perbandingan *Runtime* Skenario A1 **Error! Bookmark not defined.**](#_Toc423533280)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR TABEL

[Tabel 3. 1 Definisi Kasus Penggunaan 18](#_Toc423531846)

[Tabel 3. 2 Definisi Kasus Penggunaan 19](#_Toc423531847)

[Tabel 3. 3 Spesifikasi Kasus Penggunaan Menambahkan Data Pelatihan 20](#_Toc423531848)

[Tabel 3. 4 Spesifikasi Kasus Penggunaan Latihan Gerakan Isyarat Huruf 21](#_Toc423531849)

[Tabel 3. 5 *Rule* Pembagian *Neural Network* 28](#_Toc423531859)

[Tabel 5. 1 Skenario Pengujian A1 49](#_Toc423531887)

[Tabel 5. 2 Tabel Perbandingan Akurasi Algoritma BP dan BPGA **Error! Bookmark not defined.**](#_Toc423531888)

[Tabel 5. 3 Tabel Perbandingan *Runtime* Algoritma (detik) **Error! Bookmark not defined.**](#_Toc423531889)

[Tabel 5. 4 Skenario Pengujian A2 50](#_Toc423531890)

[Tabel 5. 5 Terjemah Huruf Isyarat Skenario Pengujian A2 51](#_Toc423531891)

[Tabel 5. 6 Skenario Pengujian A3 51](#_Toc423531894)

[Tabel 5. 7 Terjemah Huruf Isyarat Pengujian Skenario A3 **Error! Bookmark not defined.**](#_Toc423531895)

[Tabel 5. 8 Perbandingan Akurasi Terhadap Jumlah Iterasi **Error! Bookmark not defined.**](#_Toc423531897)

[Tabel 5. 9 Skenario Pengujian B1 **Error! Bookmark not defined.**](#_Toc423531898)

[Tabel 5. 10 Terjemah Huruf Isyarat Pengujian Skenario B1 **Error! Bookmark not defined.**](#_Toc423531900)

[Tabel 5. 11 Tabel Terjemah Skenario Pengujian B2 **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%201).docx#_Toc423531901)

[Tabel 5. 12 Skenario Pengujian B2 **Error! Bookmark not defined.**](#_Toc423531902)

[Tabel 5. 13 Skenario Pengujian B3 **Error! Bookmark not defined.**](#_Toc423531903)

[Tabel 5. 14 Tabel Terjemahan Huruf Skenario B3 **Error! Bookmark not defined.**](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%201).docx#_Toc423531904)

[Tabel 5. 15 Terjemah Huruf Isyarat Pengujian Skenario B4 **Error! Bookmark not defined.**](#_Toc423531906)

[Tabel 5. 16 Skenario Pengujian B4 **Error! Bookmark not defined.**](#_Toc423531907)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR KODE SUMBER

[Kode Sumber 4. 1 Kode Sumber Pendeteksian Telapak Tangan 31](#_Toc423531949)

[Kode Sumber 4. 2. Kode Sumber Kelas *LeapMouse.cs* 32](#_Toc423531950)

[Kode Sumber 4. 3 Kode Sumber Ekstraksi Fitur Tangan 34](#_Toc423531951)

[Kode Sumber 4. 4 Fungsi Inisialisasi Bobot Pada Kelas *NeuralNetwork.cs* 38](#_Toc423531952)

[Kode Sumber 4. 5 Kelas Feedforward.cs **Error! Bookmark not defined.**](#_Toc423531953)

[Kode Sumber 4. 6 Fungsi GetErrorOutput Pada Kelas Backpropagation.cs **Error! Bookmark not defined.**](#_Toc423531954)

[Kode Sumber 4. 7 Fungsi ChromosomInit pada kelas GeneticAlgorithm.cs **Error! Bookmark not defined.**](#_Toc423531955)

[Kode Sumber 4. 8 Fungsi UpdateWeight di *Hidden* *Layer* Pada Kelas Backpropagation.cs **Error! Bookmark not defined.**](#_Toc423531956)

[Kode Sumber 4. 9 Fungsi GetErrorHidden Pada Kelas Backpropagation.cs **Error! Bookmark not defined.**](#_Toc423531957)

[Kode Sumber 4. 10 Fungsi UpdateWeightInput Pada Kelas Backpropagation.cs **Error! Bookmark not defined.**](#_Toc423531958)

[Kode Sumber 4. 11 Fungsi DoSelection pada kelas GeneticAlgorithm.cs **Error! Bookmark not defined.**](#_Toc423531959)

[Kode Sumber 4. 12 Fungsi DoCrossOver pada kelas GeneticAlgorithm.cs **Error! Bookmark not defined.**](#_Toc423531960)

[Kode Sumber 4. 13 Fungsi DoBackpropagation pada kelas GeneticAlgorithm.cs **Error! Bookmark not defined.**](#_Toc423531961)

[Kode Sumber 4. 14 Fungsi DoMutation pada kelas GeneticAlgorithm.cs **Error! Bookmark not defined.**](#_Toc423531962)

[Kode Sumber 4. 15 Fungsi Normalisasi Fitur 44](#_Toc423531963)

[Kode Sumber 4. 16 Potongan Kode Sumber Fungsi Mendapatkan Perbandingan Untuk Kalibrasi Tangan **Error! Bookmark not defined.**](#_Toc423531964)

[Kode Sumber 4. 17 Fungsi Kalibrasi Pada Fitur Pengguna **Error! Bookmark not defined.**](#_Toc423531965)

[Kode Sumber 4. 18Pembagian *Neural Network* **Error! Bookmark not defined.**](#_Toc423531966)

[Kode Sumber 7. 1 Kelas Backpropagation.cs 60](#_Toc423531994)

[Kode Sumber 7. 2 Kelas Chromosom.cs 61](#_Toc423531995)

[Kode Sumber 7. 3 Kelas ClassificationClass.cs 63](#_Toc423531996)

[Kode Sumber 7. 4 Kelas DataSet.cs 64](#_Toc423531997)

[Kode Sumber 7. 5 Kelas FeedForward.cs 67](#_Toc423531998)

[Kode Sumber 7. 6 Kelas GeneticAlgorithm 70](#_Toc423531999)

[Kode Sumber 7. 7 Kelas NeuralNetwork.cs 72](#_Toc423532000)

# BAB I PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan, batasan permasalahan, dan manfaat.

## 1.1 Latar Belakang

Bahasa isyarat adalah media bagi pada penderita Tunarungu untuk berkomunikasi dengan sekitarnya. Gerakan visual tubuh sangat membantu penderita agar yang ingin disampaikannya lebih mudah dimengerti oleh pasangan komunikasinya. Jika penderita berkomunikasi dengan gerakan bibir, tingkat keakuratan untuk mengartikan gerakan bibir tersebut lebih rendah dibandingkan dengan gerakan tubuh. Hal tersebut menjadi alasan untuk mengembangkan bahasa isyarat di Indonesia [1].

Dari survei yang dilakukan Multi Center Study di Asia Tenggara, Indonesia termasuk dalam 4 negara dengan prevalensi ketulian yang cukup tinggi yaitu 4,6 persen. Sedangkan 3 negara lainnya yakni Srilangka (8,8 persen), Myanmar (8,4 persen) dan India (6,3 persen)[2].

Di Indonesia ada dua dasar bahasa isyarat yang digunakan, salah satunya adalah Sistem Isyarat Bahasa Indonesia (SIBI). SIBI sudah menjadi bahasa isyarat Indonesia yang resmi. Di dalamnya terdapat posisi jari dan gerakan tangan untuk menggantikan kosa kata Bahasa Indonesia. Gerakan isyarat yang ada di dalam SIBI sudah diatus secara sistematis.

Di jaman sekarang, teknologi sudah berkembang sangat pesat. Sudah banyak hardware yang bisa membaca isyarat (gesture) yang ada dengan algoritma tertentu. Salah satunya adalah Kinect. Kinect adalah motion sensing input device yang dibuat oleh Microsoft untuk game consolenya Xbox 360 dan Xbox One serta untuk windows PC. Dengan bentuk seperti webcam, Kinect dapat membuat pengguna mengontrol dan berinteraksi dengan console atau komputernya tanpa perlu controller, tapi melalui natural user interface (menggunakan gesture dan perintah suara). Generasi pertama Kinect diperkenalkan bulan November tahun 2010 [3]

Sebelumnya sudah ada Tugas Akhir yang dibuat oleh Risal Andika Tridisaputra tentang pengenalan Bahasa isyarat Indonesia menggunakan teknologi leap motion, tetapi Tugas Akhir tersebut hanya sebatas membaca isyarat abjad. Oleh karena itu, Dengan kemampuan yang dimiliki Kinect, munculah ide untuk membuat software pengenalan bahasa isyarat menggunakan Kinect. Bahasa isyarat yang digunakan akan berpaku pada Sistem Isyarat Bahasa Indonesia (SIBI).

## 1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menentukan fitur dari Kinect yang bias digunakan untuk mendeteksi Bahasa isyarat yang diberikan?
2. Bagaimana menerapkan algoritma *Back Propagation Genetic Algorithm Neural Network* untuk mendeteksi Bahasa isyarat yang diberikan?
3. Bagaimana menentukan ketepatan gerakan isyarat yang dilakukan oleh penderita Tunarungu?.

## 1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Aplikasi yang akan dibuat adalah aplikasi yang berjalan pada sistem operasi Windows 7, Windows 8/8.1, dan Windows 10.
2. Aplikasi akan dikembangkan menggunakan *Kinect SDK* dengan menggunakan bahasa pemrograman C# dengan IDE Microsoft Visual Studio.
3. Versi *Kinect* sensor yang dipakai adalah *Kinect V1*
4. Algoritma *Neural Network* yang akan digunakan adalah *Back Propagation Genetic Algorithm.*
5. Topologi *Neural Network* adalah jenis *Multilayer Percepton* dengan 1 hidden layer.
6. Bahasa Isyarat yang digunakan berdasarkan pada Sistem Isyarat Bahasa Indonesia (SIBI).
7. Bahasa isyarat yang dikenali adalah kata.
8. Bahasa isyarat yang digunakan hanya yang bersifat statis (tidak terlalu banyak gerak).
9. Tidak semua isyarat akan dideteksi.

## 1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini antara lain:

1. Membuat aplikasi pengenalan bahasa isyarat menggunakan teknologi *Kinect*
2. Dapat mengimplementasikan algoritma *Back Propagation Genetic Algorithm Neural Network* sebagai *classifier* gerakan tangan dari pengguna yang akan dioleh dengan *Kinect*

## 1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah membuat aplikasi pengenalan bahasa isyarat yang nantinya akan digunakan sebagai media pembelajaran bahasa isyarat bagi penderita Tunarungu.

## 1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan menggunakan metodologi sebagai berikut:

1. Studi literatur

Tahap Studi literature merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literature, diskusi eksplorasi teknologi dan pustaka, serta pemahaman dasar teori yang digunakan pada topik Tugas Akhir. Literatur-literatur yang dimaksud disebutkan sebagai berikut:

1. Tunarungu
2. Bahasa Isyarat
3. Kinect
4. Kinect SDK
5. *Neural Network*
6. *Backpropagation*
7. *Backpropagation-Genetic Algorithm*
8. Perancangan perangkat lunak

Pada tahap ini diawali dengan melakukan analisis awal terharap permasalahan utama yang muncul pada topik Tugas Akhir. Kemudian dilakukan perancangan perangkat lunak yang meliputi penentuan data yang digunakan an proses-proses yang akan dilaksanakan. Langkah yang digunakan pada tahap ini adalah sebagai berikut:

1. Pendataan bahasa isyarat yang akan digunakan untuk aplikasi
2. Perancangan integrasi aplikasi dengan perangkat Kinect
3. Perancangan ektraksi fitur Kinect untuk mendapatkan data bahasa isyarat
4. Perancangan proses *training* dan *testing* data.
5. Implementasi dan pembuatan system

Pada tahap ini dilakukan implementasi integrase aplikasi dengan perangkat Kinect. Kemudian dilakukan implementasi ekstraksi fitur Kinect. Akhirnya dilakukan implementasi proses *training* dan *testing* data, Aplikasi ini dibangun menggunakan Microsoft Visual Studio 2015 dan Kinect SDK.

1. Uji coba dan evaluasi

Pada tahap ini dilakuka uji coba dengan menggunakan gerakan bahasa isyarat yang dilakukan penulis untuk mencoba aplikasi bisa berjalan atau tidak. Uji fungsionalitas untuk mengetahui apakah sudah memenuhi kebutuhan fungsional.

1. Penyusunan laporan Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan Tugas Akhir.

## 1.7 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan Tugas Akhir, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

**BAB II TINJAUAN PUSTAKA**

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan Tugas Akhir ini.

**BAB III ANALISIS DAN PERANCANGAN**

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

**BAB IV IMPLEMENTASI**

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi antarmuka aplikasi dan pembuatan kebutuhan fungsional aplikasi

**BAB V PENGUJIAN DAN EVALUASI**

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

**BAB VI PENUTUP**

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

# BAB II DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini. Pokok permasalahan yang akan di bahas mengenai teknologi yang mendukung dalam pembuatan tugas akhir seperti Kinect, Kinect SDK, *neural network*, algoritma *backpropagation*, algoritma *backpropagation-genetic*, dan pengetahuan umum mengenai bahasa isyarat huruf.



## 2.1 Tunarungu

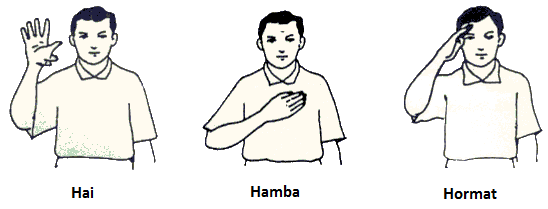
Tunarungu dapat diartikan sebagai keterbatasan yang dimiliki seseorang dalam mendengar sesuatu karena tidak berfungsinya. organ pendengaran yang dimilikinya. Ketunarunguan dapat dibedakan menjadi dua kategori yaitu tuli (deaf) dan kurang dapat mendengar (low hearing)[4]. Tuli adalah keadaan di mana organ pendengaran telah mengalami kerusakan yang sangat parah dan mengakibatkan tidak berfungsinya pendengaran. Sedangkan kurang dapat mendengar adalah keadaan di mana organ pendengaran mengalami kerusakan tetapi masih dapat berfungsi untuk mendengar.

## 2.2 Bahasa Isyarat

Bahasa isyarat adalah sarana komunikasi bagi penderita tunarungu. Bahasa Isyarat berkembang dan memiliki karakteristik sendiri di berbagai negara. Di Indonesia, bahasa isyarat yang digunakan berdasarkan pada Sistem Isyarat Bahasa Indonesia (SIBI). Ada 4 jenis bahasa isyarat dalam SIBI yaitu:

1. Isyarat Pokok: melambangkan sebuah kata atau konsep
2. Isyarat Tambahan: melambangkan awalan, akhiran, dan partikel (Imbuhan)
3. Isyarat Bentukan: Dibentuk dengan menggabungkan isyarat pokok dan isyarat tambahan.
4. Abjad Jari: dibentuk dengan jari – jari untuk mengeja huruf.

Pada Tugas Akhir ini, isyarat yang akan digunakan adalah isyarat pokok. Gambar 2.1 menunjukkan contoh Bahasa isyarat yang ada didalam SIBI.

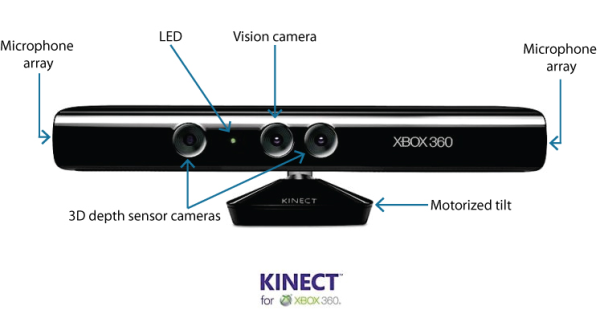


Gambar 2.1 Contoh Bahasa Isyarat yang ada didalam SIBI

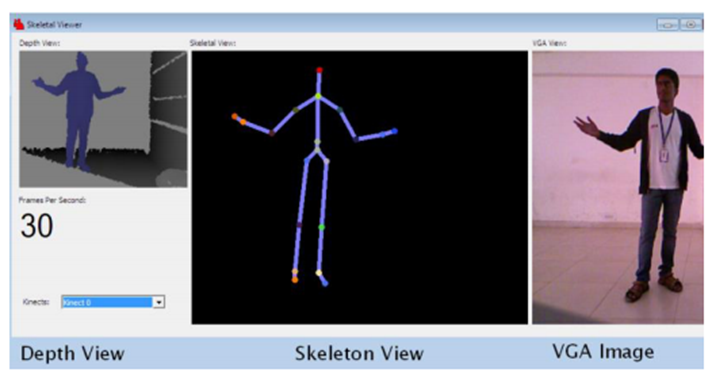
## 2.3 Kinect

Kinect adalah sebuah perangkat keras buatan Microsoft yang menggantikan input game controller pada Xbox dengan natural user interface berupa gesture dan perintah suara. RGB camera, 3d depth sensing system, multi-array microphone, dan motorized tilt merupakan komponen dasar dari Kinect yang dapat dilihat pada gambar 2.2.

Kinect dapat mendapatkan area sekitarnya dalam 3D dengan mengkombinasikan informasi dari depth sensor dan standard RGB camera. Hasil dari penggabungan tersebut adalah RGBD image dengan resolusi 640x480, dimana setiap pixel memiliki color information dan depth information. Pada bagian kiri Kinect memiliki laser infrared light source yang menghasilkan gelombang elektromagnetik dengan frekuensi 830 nm. Informasi akan dikodekan sebagai light pattern yang akan di “deformed” menjadi light reflects dari objek di depan Kinect. Berdasarkan proses deformation yang didapat oleh sensor di sisi kanan RGB camera sebuah depth map dibuat seperti pada Gambar 2.3[5].



Gambar 2.2 Microsoft Kinect



Gambar 2.3 *Depth, Skeleton,* dan *VGA view*

## 2.4 Kinect SDK

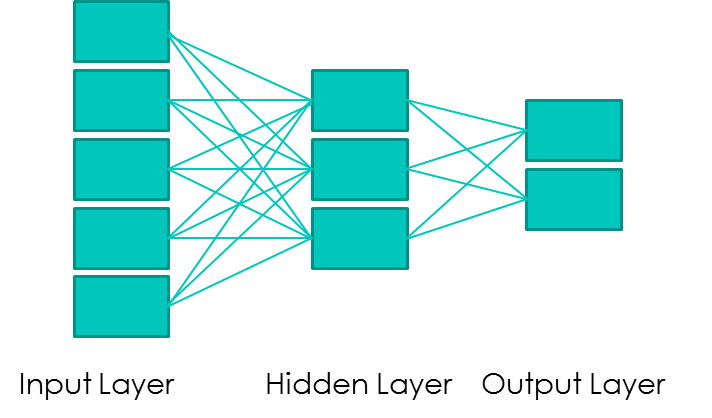
Kinect SDK adalah library yang dibuat oleh Microsoft untuk pengembangan aplikasi perangkat lunak yang menggunakan Kinect sebagai alat input utama. Kinect SDK dapat diimplementasikan dengan bahasa pemrograman C#, C++, dan JavaScript. Library ini memiliki beberapa fitur diantaranya skeleton tracking, Thumb Tracking, end of hand tracking, open/close hand gesture dan lainnya [3]. Pada Kinect SDK yang akan digunakan di Tugas Akhir ini, memiliki 20 skeletoh joint yang dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Skeleton Joints* yang diketahui Kinect

## 2.5 *Neural* *Network*

*Neural network* atau jaringan syaraf neural biasanya terdiri dari 3 bagian yang disebut lapisan atau *layer* [5]. Lapisan *input* terhubung dengan lapisan *hidden* yang juga terhubung dengan lapisan *output*. *Node* yang ada pada lapisan *input* merepresentasikan fitur yang ada pada *network*. *Node* yang ada pada lapisan *hidden* didapatkan dengan aktifitas dari lapisan input dan bobot dari hubungan antara lapisan *input* dan *unit* di lapisan *hidden*. Aktifitas yang terjadi pada lapisan *hidden* tergantung dari aktiftias yang terjadi pada unit di lapisan *hidden* dan bobot antara lapisan *hidden* dan lapisan *output*. Contoh dari bentuk jaringan syaraf dengan 1 lapisan *hidden* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Contoh Jaringan Syaraf Feedforward

## 2.6 *Backpropagation*

*Backpropagation* (BP) adalah algoritma *training* yang menggunakan *forward network* atau biasanya disebut *Multi Layer Perceptron* (MLP). Jaringan syaraf yang menggunakan algoritma BP memprediksi suatu set tertentu dengan mempelajari pada setiap contoh yang dimasukkan ke dalam jaringan syaraf[6]. Langkah algoritma BP yaitu:

1. Mencari *error* di *node* ke-j pada lapisan *output* menggunakan rumus pada Persamaan 2.1. *Target* adalah kelas yang diinginkan.

|  |  |
| --- | --- |
|  | (2.1) |
|  |  |

1. Update bobot pada lapisan *hidden* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *hidden* dan *output*).

|  |  |
| --- | --- |
|  | (2.2) |
|  |  |

1. Mencari *error* di *node* ke-j pada lapisan *hidden* menggunakan rumus pada Persamaan 2.3. *Node* A terhubung ke *node* B dan *node* C. Error dari *node* B dan *node* C dibutuhkan untuk menghasilkan *error* pada *node* A.

|  |  |
| --- | --- |
|  | (2.3) |
|  |  |

1. Update bobot pada lapisan *input* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *input* dan *hidden*).

## 2.7 *Backpropagation*-*Genetic* *Algorithm* *Neural* *Network*

Sejak Rumelhalt mengenalkan *Backpropagation* (BP) pada tahun 1986, *neural* *network* BP telah banyak digunakan untuk data *training*, dan menjadi bagian penting pada kemajuan *neural* *network*. Fungsi performa yang digunakan pada BP adalah MSE (*Mean Square Error*).

Rumus yang digunakan untuk mengganti berat dari jaringan pada algoritma BP standar terdapat pada Persamaan 2.4.

|  |  |
| --- | --- |
|  | (2.4) |

Di mana *k* adalah jumlah iterasi, *Wlij* adalah bobot yang menghubungkan *node* pada layer *i* dan *j*, adalah bilangan positif yang disebut *learning* *rate* yang digunakan untuk mengatur langkah-langkah pembelajaran dan biasanya bernilai bilangan positif sangat kecil.

Algoritma BP adalah metode yang efektif, namun mempunyai beberapa kekurangan yaitu: (1) algoritma BP menggunakan *gradient* *descent* untuk meminimalisir MSE dimana membuat menjadi *local* *minimum*; (2) Performa dari algoritma ini sangat sensitif terhadap pengaturan dari *learning* *rate*. Jika *learning* *rate* terlalu tinggi, algoritma ini mungkin menjadi tidak stabil. Jika *learning* *rate* terlalu kecil, algoritma ini membutuhkan waktu untuk menyatu; (3) inisialisasi berat dan bias pada jaringan sangat penting dalam proses konvergensi dari jaringan BP, di mana dihasilkan secara acak, dan terkadang membuat jaringan tidak dapan mencapai tujuan *training*.

Algoritma *Backpropagation* – *Genetic* *Algorithm* *Neural* *Network* (BPGANN) merupakan algoritma gabungan dari *Back Propagation* dan *Genetic Algorithm* (GA) dengan menggunakan metode GA untuk mencari bobot inisial dari suatu *neural network* dan mempercepat konvergensi, karena algoritma BP membuat konvergensi menjadi pelan. Langkah-langkah algoritma genetik seperti berikut:

1. Menggunakan metode GA untuk mencari kromosom paling unggul sebagai acuan bobot awal pada *neural network*.
2. Menggunakan algoritma BP untuk melatih *neural network* dengan bobot awal yang sudah ada.
3. Jika performa *neural network* semakin berbeda maka lanjut ke langkah 5, jika tidak maka lakukan langkah 4.
4. *Update* bobot dari *neural network* sebagai populasi awal dari GA dan menggunakannya untuk mencari kromosom paling unggul dari bobot yang dilatih oleh algoritma BP, lalu kembali ke langkah nomor 2.
5. Tentukan apakah kondisi pada saat iterasi memuaskan atau tidak, jika sudah memuaskan maka proses pelatihan berhenti, jika tidak maka kembali ke langkah nomor 2 [7].

# BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat.



## 3.1 Analisis Perangkat Lunak

Bahasa isyarat merupakan sarana komunikasi untuk penderita tunarungu. Walaupun agak sulit untuk mengarikan isyarat yang diberikan, hal tersebut telah membantu penderita tunarungu untuk berkomunikasi dengan sekitar. Namun masih banyak yang belum mengerti apa arti isyarat bagi yang diberi ato memberi isyarat.

Aplikasi ini dibangun dengan tujuan membantu pengguna untuk mempelajari Bahasa isyarat yang ada. Isyarat yang dipakai sesuai dengan SIBI yang nantinya akan didapat dari training data yang didapat. Dengan menggunakan Microsoft Visual Studio dan Kinect, penulis mengekstraksi *skeleton joints* yang ditangkap oleh Kinect kemudian akan dikalkulasi menjadi fitur – fitur yang akan diolah. Kemudian sebuah *classifier* digunakan untuk mengolah fitur yang telah dikalkulasi dan akan menjadi sebuah *Neural Network* yang akan digunakan untuk menentukan arti Bahasa Isyarat yang diberikan.

Untuk tahap testing data tidak jauh berbeda dengan tahap training data. Setelah mengolah fitur yang didapat dari kalkulasi *skeleton joints*, fitur akan dimasukkan sebagai input *Neural Network* dan akan menghasilkan output arti Bahasa Isyarat yang diberikan

### 3.1.1 Deskripsi Umum Perangkat Lunak

Tugas akhir yang akan dikembangkan adalah sebuah modul pengenalan Bahasa iysarat dengan menggunakan Kinect. Aplikasi ini menggunakan Kinect SDK dan dijalankan dengan perangkat keras Kinect V1.

Pengguna utama adalah semua orang yang ingin belajar Bahasa isyarat. Pengguna dapat mempelajari isyarat yang sudah ada dalam aplikasi ataupun memberikan isyarat baru sesuai dengan SIBI. Jika isyarat yang diberikan tidak ada dalam aplikasi maka tidak ada output yang diberikan.

### 3.1.2 Spesifikasi Kebutuhan Perangkat Lunak

Kebutuhan sistem yang akan dibuat ini melibatkan dua hal, yakni kebutuhan fungsional maupun kebutuhan non-fungsional. Dimana masing-masing berhubungan dengan keberhasilan dalam pembuatan aplikasi tugas akhir ini.

#### 3.1.2.1 Kebutuhan Fungsional Perangkat Lunak

Pada sistem ini, terdapat beberapa kebutuhan fungsional yang mendukung untuk jalannya aplikasi. Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:.

1. Mendeteksi *Skeleton* Pengguna

Aplikasi dapat mendeteksi pengguna yang sedang berdiri di depan Kinect .

1. Mengekstraksi Fitur *Skeleton*

Aplikasi dapat mendeteksi lokasi objek berupa *Skeleton* yang akan diekstraksi menjadi fitur-fitur untuk proses klasifikasi pada saat melakukan *training* dan *testing*.

1. Menerjemahkan Bahasa Isyarat

Aplikasi dapat menerjemahkan bahasa isyarat yang sesuai dengan fitur-fitur dari skeleton pengguna pada saat melakukan *testing*.

#### 3.1.2.2 Kebutuhan Non-Fungsional

Pada sistem ini, terdapat beberapa kebutuhan non-fungsional yang mendukung dan menambah performa untuk jalannya aplikasi.

Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:

1. Penyesuaian Intensitas Cahaya

Intensitas cahaya merupakan salah satu hal penting yang perlu diperhatikan ketika kita akan menggunakan sebuah sensor. Kinect juga sangat terpengaruh dengan intensitas cahaya. Jika terlalu gelap atau terang, Kinect tidak dapat mendeteksi pengguna. Jika kurang terang, Data skeleton yang diambil Kinect tidak stabil. Maka dari itu, untuk penggunaan aplikasi ini sebaiknya berada di ruangan yang memiliki Intensitas cahaya yang cukup dan usahakan tidak langsung terkena matahari.

1. Posisi Peletakan Kinect

Posisi peletakan Kinect disesuaikan dengan pengguna sehingga menghasilkan fitur – fitur yang sempurna. Jarak optimal pengguna dengan Kinect adalah 0.6 meter – 1.8 meter.

### 3.1.3 Identifikasi Pengguna

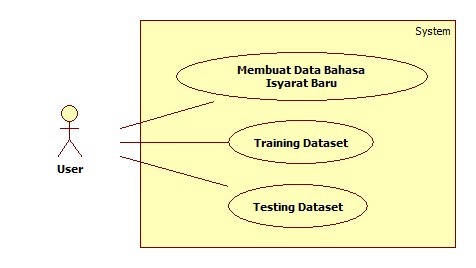
Dalam aplikasi tugas akhir ini, pengguna yang akan terlibat hanya terdapat satu orang saja, yakni orang yang akan melakukan pengenalan bahasa isyarat.

## 3.2 Perancangan Perangkat Lunak

Subbab ini membahas bagaimana rancangan dari aplikasi tugas akhir ini. Hal yang dibahas meliputi model kasus penggunaan, definisi aktor, definisi kasus penggunaan, arsitektur umum sistem, rancangan antarmuka aplikasi, dan rancangan proses aplikasi.

### 3.2.1 Model Kasus Penggunaan

Dari hasil analisa deskripsi umum perangkat lunak dan spesifikasi kebutuhan perangkat lunak yang telah dijelaskan, maka model kasus penggunaan untuk aplikasi pengenalan bahasa isyarat pada Gambar 3.1.



Gambar 3.1 Diagram Kasus Penggunaan Aplikasi

### 3.2.2 Definisi Aktor

Aktor yang terdapat dalam sistem aplikasi ini terlihat pada Tabel 3.1.

Tabel 3.1 Definisi Kasus Penggunaan

|  |  |  |
| --- | --- | --- |
| No | Nama | Deskripsi |
| 1 | Pengguna | Merupakan aktor yang bertugas untuk menambahkan data pelatihan dan melakukan latihan gerakan isyarat statis, seluruh fungsionalitas yang ada di dalam sistem dapat digunakan oleh pengguna. |

### 3.2.3 Definisi Kasus Penggunaan

Pada Gambar 3.1 telah dijelaskan bahwa aktor yang dalam hal ini disebut pengguna mempunyai tiga kasus penggunaan, yakni menambahkan data bahasa isyarat baru, melakukan pelatihan data set dan percobaan *dataset*. Detail mengenai kasus penggunaan tersebut dapat dilihat pada Tabel 3.2.

Tabel 3.2 Definisi Kasus Penggunaan

|  |  |  |  |
| --- | --- | --- | --- |
| No | Kode Kasus Penggunaan | Nama Kasus Penggunaan | Keterangan |
| 1 | UC-01 | Membuat Data Bahasa Isyarat Baru | Pengguna membuat data Bahasa isyarat baru. Dapat berupa Bahasa isyarat baru atau yang sudah ada. |
| 2 | UC-02 | *Training Dataset* | Pengguna mengaktifkan mode pelatihan *dataset* |
| 3 | UC-03 | *Testing Dataset* | Pengguna melakukan percobaan *dataset* dengan melakukan gerakan bahasa isyarat yang ada. |

#### 3.2.3.1 Kasus Penggunaan Membuat Data Bahasa Isyarat Baru

Spesifikasi kasus penggunaan membuat data bahasa isyarat baru dapat dilihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi Kasus Penggunaan Membuat Data Bahasa Isyarat Baru

|  |  |
| --- | --- |
| Nama Kasus Penggunaan | Membuat Data Bahasa Isyarat Baru |
| Nomor | UC-01 |
| Deskripsi | Kasus penggunaan aktor untuk membuat data Bahasa isyarat baru |
| Aktor | Pengguna |
| Kondisi Awal | Pengguna sudah menjalankan aplikasi dan perangkat Kinect telah tersambung |
| Alur Normal | 1. User memasukkan nama Bahasa isyarat yang akan dibuat didalam textbox yang sudah disediakan.  2. User menekan tombol kontrol “Create File”  3. Sistem menerima inputan dan menunggu 5 detik untuk perangkat Kinect menemukan skeleton user.  4. User masuk kedalam daerah yang dapat ditangkap Kinect dan mencari lokasi yg tepat sehingga Kinect mendapatkan skeleton user, kemudian melakukan gerakan Bahasa isyarat yang akan dibuat.  A1. Kinect tidak menemukan skeleton user.  5. Kinect menemukan skeleton user, sistem mengekstrak data skeleton selama 10 detik untuk dikalkulasi.  A2. Kinect kehilangan skeleton saat proses kalkulasi  6. Sistem menyimpan hasil kalkulasi dalam sebuah file. |
| Alur Alternatif | A1. Kinect tidak menemukan skeleton user.  1. Sistem memberikan notifikasi skeleton user tidak ditemukan.  A2. Kinect kehilangan skeleton saat proses kalkulasi  1. Sistem menghentikan proses kalkulasi  2. Sistem memberikan notifikasi skeleton hilang |
| Kondisi Akhir | Aplikasi akan membuat data set baru |

#### 3.2.3.2 Kasus Penggunaan *Training Dataset*

Spesifikasi kasus penggunaan *training dataset* dapat dilihat pada Tabel 3.4.

Tabel 3.4 Spesifikasi Kasus Penggunaan *Training Dataset*

|  |  |
| --- | --- |
| Nama Kasus Penggunaan | *Training Dataset* |
| Nomor | UC-02 |
| Deskripsi | Kasus penggunaan aktor untuk melakukan pelatihan data set |
| Aktor | Pengguna |
| Kondisi Awal | pengguna sudah menjalankan aplikasi dan sudah ada *dataset* yang dibuat aplikasi pada tahap sebelumnya |
| Alur Normal | 1. User menekan tombol kontrol “Train Dataset” yang ada pada aplikasi  2. Sistem memulai tahap training *dataset* yang sudah ada  3. Sistem membuat file *neural network* dalam format .xml |
| Alur Alternatif | - |
| Kondisi Akhir | Aplikasi membuat file *neural network* dalam format .xml |

#### Kasus *Penggunaan Testing Dataset*

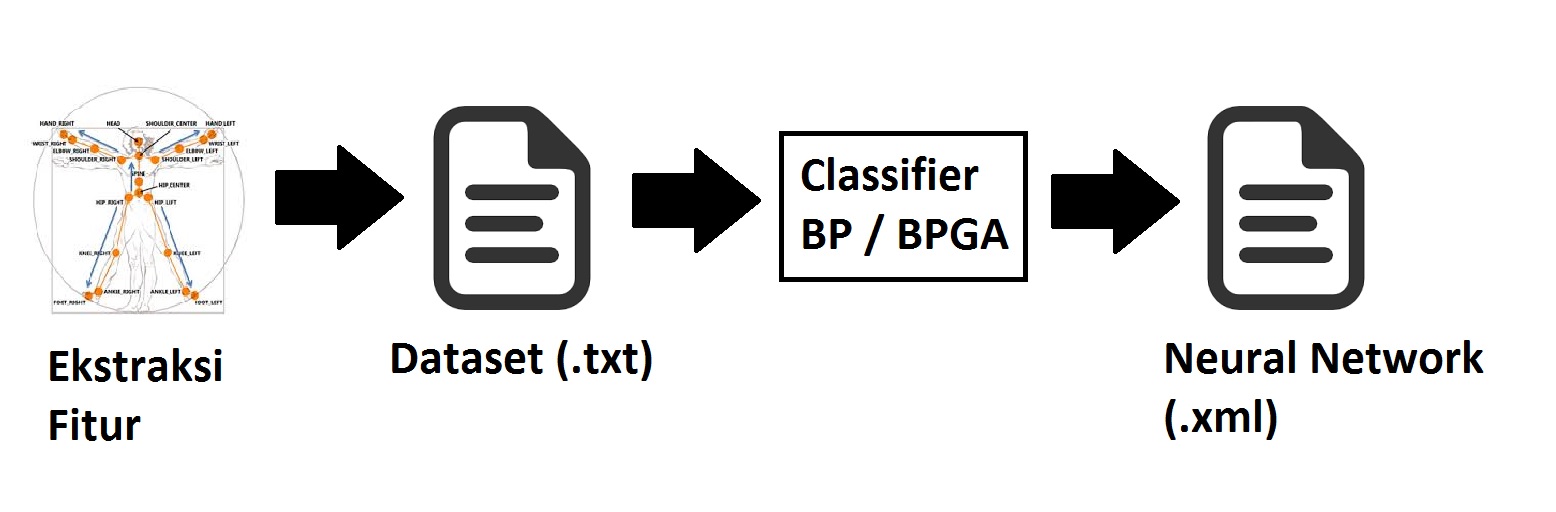
Spesifikasi kasus penggunaan *Testing Dataset* dapat dilihat pada Tabel 3.5.

Tabel 3.5 Spesifikasi Kasus Penggunaan *Testing Dataset*

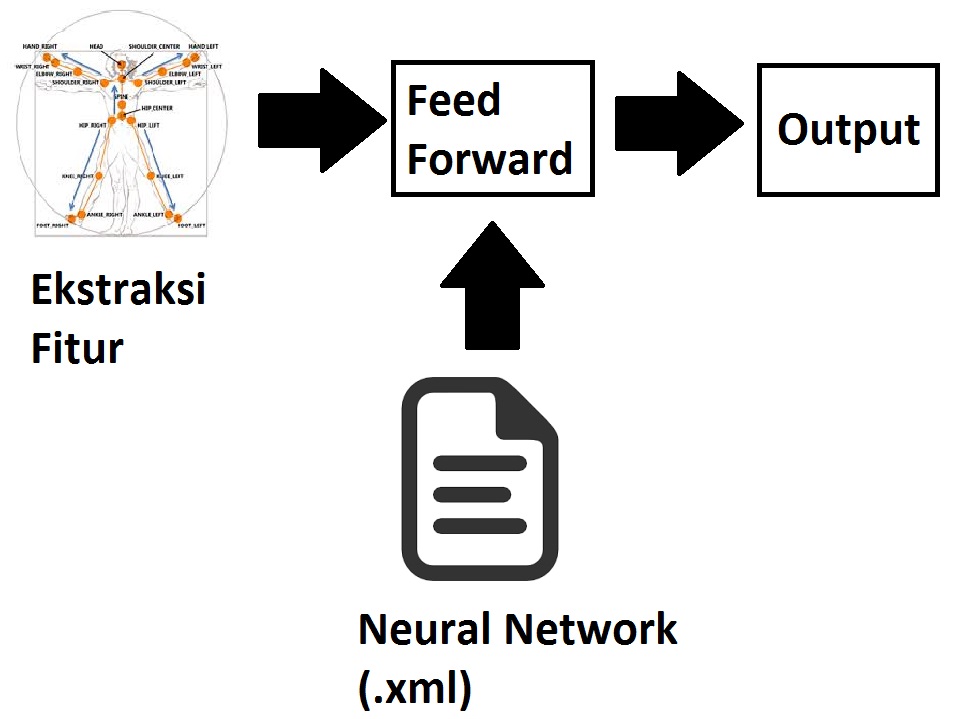
|  |  |
| --- | --- |
| Nama Kasus Penggunaan | *Testing Dataset* |
| Nomor | UC-03 |
| Deskripsi | Kasus penggunaan aktor untuk melakukan percobaan *dataset* |
| Aktor | Pengguna |
| Kondisi Awal | pengguna sudah menjalankan aplikasi dan sudah ada *dataset* yang dibuat aplikasi pada tahap sebelumnya |
| Alur Normal | 1. User menekan tombol kontrol “Start Testing” yang ada pada aplikasi  2. Sistem memulai tahap *testing dataset* dengan *neural network* yang sebelumnya dibentuk  3. User melakukan Bahasa isyarat didepan Kinect.  4. Kinect mengambil data skeleton yang didapat  5. Sistem mengekstrak data skeleton yang kemudian akan dijadikan input dapat *neural network*  6. Sistem mendapatkan output dari *neural network* dan kemudian menampilkan. |
| Alur Alternatif | - |
| Kondisi Akhir | Aplikasi memberikan output bahasa isyarat |

### 3.2.4 Arsitektur Umum Sistem

Arsitektur sistem pada aplikasi yang akan dibuat terlihat pada Gambar 3.2 dan Gambar 3.3.



Gambar 3.2 Arsitektur Sistem (*Training*)

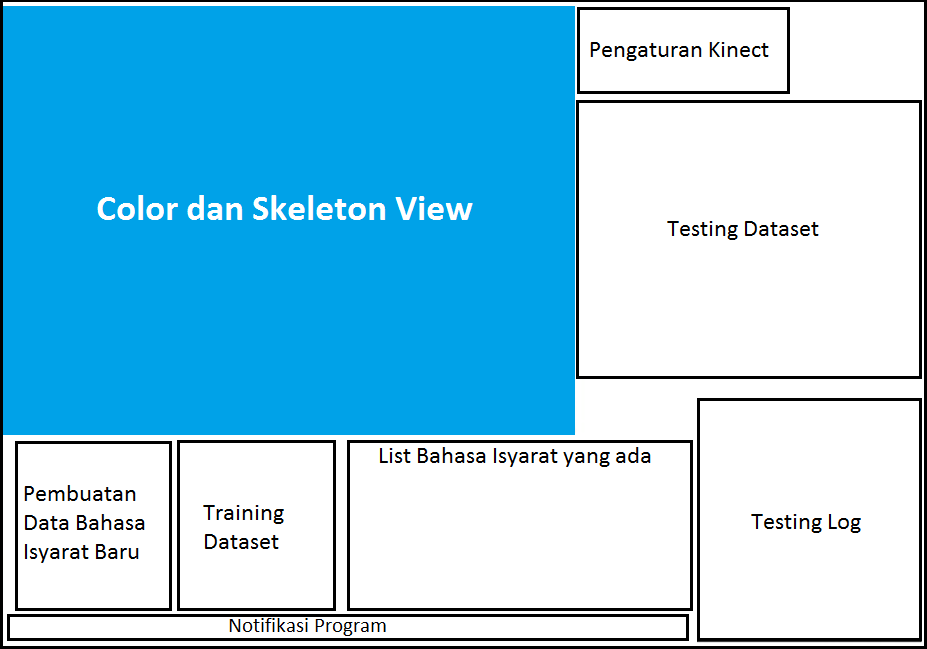


Gambar 3.3 Arsitektur Sistem (*Testing*)

### 3.2.5 Rancangan Antarmuka Aplikasi

Rancangan antarmuka aplikasi diperlukan untuk memberikan gambaran umum kepada pengguna bagaimana sistem yang ada dalam aplikasi ini berinteraksi dengan pengguna. Selain itu rancangan ini juga memberikan gambaran bagi pengguna apakah tampilan yang sudah disediakan oleh aplikasi mudah untuk dipahami dan digunakan, sehingga akan muncul kesan *user experience* yang baik dan mudah.

Rancangan antarmuka apikasi ini hanya memiliki 1 windows dengan beberapa elemen didalamnya seperti *color view* dan *skeleton view* dari Kinect, dan kontrol – kontrol lainnya yang sekiranya bisa dipahami pengguna.



Gambar 3.4 Rancangan Antarmuka Aplikasi

### 3.2.6 Rancangan Proses Aplikasi

Pada rancangan proses aplikasi akan dijelaskan mengenai proses yang terjadi dalam sistem untuk memenuhi fungsionalitas yang ada pada aplikasi. Proses ini penting agar aplikasi dapat berjalan secara baik dan benar.

3.2.6.1 Rancangan Proses Ekstraksi Fitur

Proses ekstraksi fitur skeleton sangat dibutuhkan bagi pengguna untuk melakukan *training* maupun *testing.* Saat pembuatan *dataset* baru, aplikasi akan menunggu 5 detik untuk mendeteksi skeleton pengguna, kemudian akan ada 2 detik tambahan untuk melakukan gerakan isyarat. Kemudian aplikasi akan mengambil total 100 data dalam waktu 10 detik yang akan dihitung dan dicari rata – ratanya kemudian disimpan dalam file .txt. Rancangan proses pengambilan fitur skeleton dapat dilihat pada Gambar 3.9.

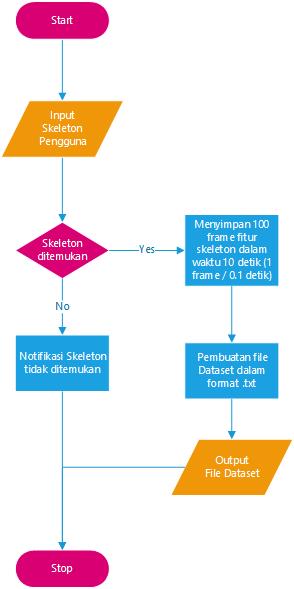
Untuk melakukan ekstraksi skeleton pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 9 fitur skeleton. 9 fitur tersebut merupakan *skeleton joints* yang diketahui oleh Kinect. 9 *skeleton joints* tersebut adalah sebagai berikut:

1. Bahu kanan (Vektor SR)
2. Siku tangan kanan (Vektor ER)
3. Pergelangan tangan kanan (Vektor WR)
4. Telapak tangan kanan (Vektor HR)
5. Bahu kiri (Vektor SL)
6. Siku tangan kiri (Vektor EL)
7. Pergelangan tangan kiri (Vektor WL)
8. Telapak tangan kiri (Vektor HL)
9. Bahu tengah (Vektor SC)

Kemudian ke 10 fitur tersebut diolah untuk dijadikan atribut *dataset* yang berjumlah 28 buah. 28 Atribut yang dimaksud dapat dilihat pada Tabel 3.6.

#### 3.2.6.3 Rancangan Proses Pembagian *Neural Network*

Proses pembagian *neural* *network* pada aplikasi ini dilakukan dengan membagi bahasa isyarat menjadi 2 golongan sesuai dengan percobaan penulis menggunakan *dataset* yang sebelumnya sudah dibuat. Rule Pembagian *neural network*  dapat dilihat pada Tabel 3.7.



Gambar 3.9. Diagram Alir Proses Ekstraksi Fitur

Tabel 3.6 Atribut *Dataset*

|  |  |  |
| --- | --- | --- |
| Vector3 (x,y,z) | Angle (float) | Distance (float) |
| ER - SR | SC - SR - ER | HR - HL |
| WR - ER | SR - ER - WR |
| HR - WR | ER - WR - HR |
| EL - SL | SC - SL - EL |
| WL - EL | SL - EL - WL |
| HL - WL | EL - WL - HL |
| HL - HR |  |

Tabel 3.7 *Rule* Pembagian *Neural Network*

|  |  |  |
| --- | --- | --- |
| *Rule* | *If* | *Then* |
| 1 | ER – WR (y) < 0.53f | Gol1.xml |
| 2 | *else* | Gol2.xml |

#### 3.2.6.4 Rancangan Proses Normalisasi Fitur

Normalisasi fitur dilakukan dengan menggunakan Persamaan 3.1. Dilakukan proses *feature* *scalling* untuk membuat fitur dalam *range* tertentu sehingga fitur lebih proporsional. Nilai maksimum dan minimum setiap fitur didapatkan dari hasil pengamatan yang dilakukan oleh penulis. *Pseudocode* untuk normalisasi fitur seperti pada Persamaan 3.1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
|  | | (3.1) | |

Keterangan:

X = nilai fitur tangan sebelum normalisasi

X’ = nilai fitur tangan setelah proses normalisasi

Xmin = nilai fitur tangan minimum

Xmax = nilai fitur tangan maksimum.

# BAB IV IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada modul. Namun, pada hasil akhir mungkin saja terjadi perubahan kecil. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C# dengan tambahan menggunakan Kinect SDK.



## 4.1 Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

### 4.1.1 Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah PC dengan spesifikasi sebagai berikut.

* Prosesor Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
* Memori (RAM) 8,00 GB
* Kinect Sensor

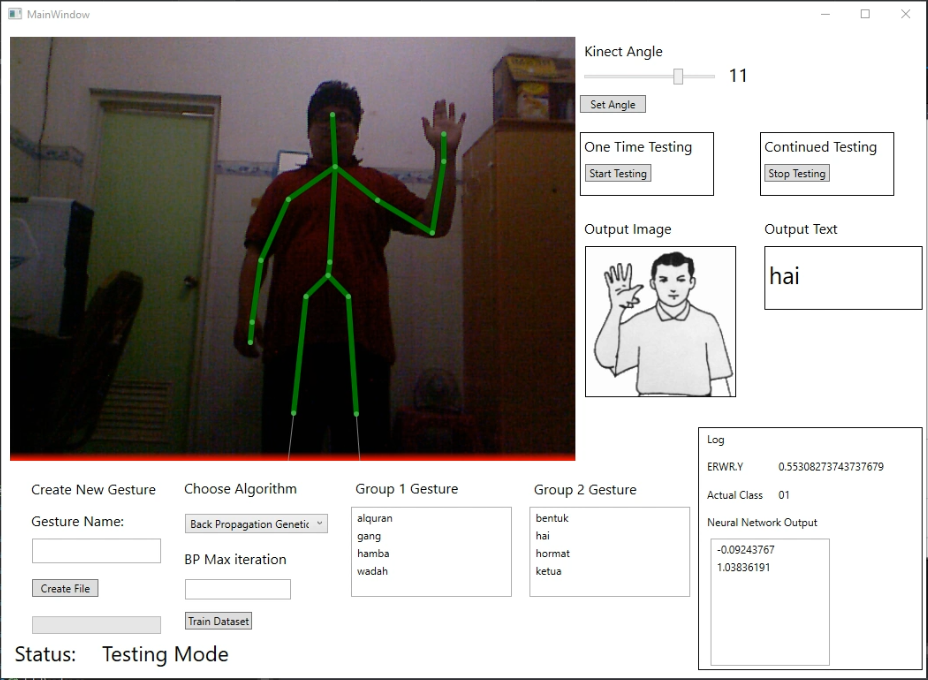
### 4.1.2 Lingkungan Pembangunan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk membuat aplikasi ini sebaga berikut.

* Microsoft Visual Studio 2015
* Windows 10 64 bit sebagai sistem operasi
* Kinect SDK

## 4.2 Implementasi Antarmuka

Seperti yang telah dijelaskan pada subbab rancangan antarmuka aplikasi, aplikasi modul pengenalan bahasa isyarat yang akan dibuat hanya akan memiliki 1 window utama yang sudah mencakup semua fungsionalitas aplikasi yang dibutuhkan. Tampilan antarmuka aplikasi dapat dilihat pada Gambar 4.1.



Gambar 4.1 Antarmuka Aplikasi

## 4.3 Implementasi Aplikasi

Pada subbab ini akan dibahas mengenai implementasi aplikasi dari kasus penggunaan ke dalam baris kode. Dijelaskan juga dengan fungsi yang dibutuhkan untuk menunjang aplikasi ini agar dapat berjalan sebagaimana mestinya. Implementasi ini dilakukan menggunakan Microsoft Visual Studio 2015 dengan bahasa pemrograman C#.

### 4.3.1 Implementasi Pendeteksian Skeleton Pengguna

Untuk menjalankan aplikasi ini tentunya membutuhkan perangkat keras Kinect, sehingga dibutuhkan suatu proses untuk mendeteksi *skeleton* pengguna. Sebelum mendeteksi *skeleton,* Kinect harus diintegrasikan dengan program terlebih dahulu.

Kode sumber proses integrasi Kinect dapat dilakukan seperti pada Kode Sumber 4. 1. Sementara kode sumber untuk mendeteksi *skeleton* pengguna dapat dilakukan seperti pada Kode Sumber 4.2

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  20  21  22  23  24  25  26  27  28  29 | if (KinectSensor.KinectSensors.Count > 0)  {  StatusDetail.Content = "Kinect Found";  mainSensor = KinectSensor.KinectSensors[0];    if (mainSensor.IsRunning)  {  StatusDetail.Content = "Reset Kinect";  StopKinect(mainSensor);  }  if (mainSensor.Status == KinectStatus.Connected)  {  StatusDetail.Content = "Setup Kinect";  Microsoft.Samples.Kinect.WpfViewers.KinectSensorManager kinectManager = new Microsoft.Samples.Kinect.WpfViewers.KinectSensorManager();  kinectManager.KinectSensor = mainSensor;  kinectManager.ColorStreamEnabled = true;  kinectManager.SkeletonStreamEnabled = true;  kinectManager.DepthStreamEnabled = true;  kinectManager.ElevationAngle = KinectDefaultAngle;  ColorView.KinectSensorManager =  SkeletonView.KinectSensorManager = kinectManager;  StatusDetail.Content = "Idle";  mainSensor.AllFramesReady += MainSensor\_AllFramesReady;  InitClassificationClass();  }  else StatusDetail.Content = "Kinect Not Found";  }  } |

Kode Sumber 4. 1 Kode Sumber Integrasi Kinect

|  |  |
| --- | --- |
| 1  2  3  4  5  6  8  8  9  10  11  12  13  14  15 | private Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)  {  using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())  {  if (skeletonFrameData == null)  return null;  skeletonFrameData.CopySkeletonDataTo(allSkeletons);  //get the first tracked skeleton  Skeleton first = (from s in allSkeletons  where s.TrackingState == SkeletonTrackingState.Tracked  select s).FirstOrDefault();  return first;  }  } |

Kode Sumber 4. 2. Kode Sumber Deteksi *Skeleton* Pengguna

### 4.3.2 Implementasi Proses Ekstraksi Fitur

Untuk melakukan ekstraksi tangan pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 9 fitur *skeleton* yang kemudian dijadikan 7 atribut *vector3* dan 14 atribut *double*. Pada

|  |  |
| --- | --- |
| 1  2  3  4  5  6  8  8  9  10  11  12  13  14  15 | for (int i = 0; i < 100; i++)  {  if (first == null)  {  result = false;  return;  }  //Right Body SR.SetVector(first.Joints[JointType.ShoulderRight].Position); ER.SetVector(first.Joints[JointType.ElbowRight].Position); WR.SetVector(first.Joints[JointType.WristRight].Position); HR.SetVector(first.Joints[JointType.HandRight].Position);  //Left Body SL.SetVector(first.Joints[JointType.ShoulderLeft].Position); EL.SetVector(first.Joints[JointType.ElbowLeft].Position); WL.SetVector(first.Joints[JointType.WristLeft].Position); HL.SetVector(first.Joints[JointType.HandLeft].Position);  //Center Body SC.SetVector(first.Joints[JointType.ShoulderCenter].Position);    SRER += ER - SR;  ERWR += WR - ER;  WRHR += HR - WR;  SLEL += EL - SL;  ELWL += WL - EL;  WLHL += HL - WL;  HRHL += HL - HR;  Vector3 v1, v2;  double res;  //SC-SR-ER  v1 = SC - SR;  v1 = v1.Normalize();  v2 = ER - SR;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SCSRER += (double)Math.Acos(res);  //SR-ER-WR  v1 = SR - ER;  v1 = v1.Normalize();  v2 = WR - ER;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SRERWR += (double)Math.Acos(res);  //ER-WR-HR  v1 = ER - WR;  v1 = v1.Normalize();  v2 = HR - WR;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  ERWRHR += (double)Math.Acos(res);  //SC-SL-EL  v1 = SC - SL;  v1 = v1.Normalize();  v2 = EL - SL;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SCSLEL += (double)Math.Acos(res);  //SL-EL-WL  v1 = SL - EL;  v1 = v1.Normalize();  v2 = WL - EL;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SLELWL += (double)Math.Acos(res);  //EL-WL-HL  v1 = EL - WL;  v1 = v1.Normalize();  v2 = HL - WL;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  ELWLHL += (double)Math.Acos(res);  //Distance HR - HL  DisHRHL += Vector3.Distance(HR, HL);  Thread.Sleep(10);  }  SRER /= 100;  ERWR /= 100;  WRHR /= 100;  SLEL /= 100;  ELWL /= 100;  WLHL /= 100;  HRHL /= 100;  SCSRER /= 100;  SRERWR /= 100;  ERWRHR /= 100;  SCSLEL /= 100;  SLELWL /= 100;  ELWLHL /= 100;  DisHRHL /= 100;  AllFeatureNormalize(); |

Kode Sumber 4.3 dijelaskan mengenai proses ekstraksi fitur pada setiap *frame* dari Kinect. Setiap atribut akan diambil rata-ratanya dari 100 *frame* skeleton yang ditangkap oleh Kinect.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  8  8  9  10  11  12  13  14  15 | for (int i = 0; i < 100; i++)  {  if (first == null)  {  result = false;  return;  }  //Right Body SR.SetVector(first.Joints[JointType.ShoulderRight].Position); ER.SetVector(first.Joints[JointType.ElbowRight].Position); WR.SetVector(first.Joints[JointType.WristRight].Position); HR.SetVector(first.Joints[JointType.HandRight].Position);  //Left Body SL.SetVector(first.Joints[JointType.ShoulderLeft].Position); EL.SetVector(first.Joints[JointType.ElbowLeft].Position); WL.SetVector(first.Joints[JointType.WristLeft].Position); HL.SetVector(first.Joints[JointType.HandLeft].Position);  //Center Body SC.SetVector(first.Joints[JointType.ShoulderCenter].Position);    SRER += ER - SR;  ERWR += WR - ER;  WRHR += HR - WR;  SLEL += EL - SL;  ELWL += WL - EL;  WLHL += HL - WL;  HRHL += HL - HR;  Vector3 v1, v2;  double res;  //SC-SR-ER  v1 = SC - SR;  v1 = v1.Normalize();  v2 = ER - SR;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SCSRER += (double)Math.Acos(res);  //SR-ER-WR  v1 = SR - ER;  v1 = v1.Normalize();  v2 = WR - ER;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SRERWR += (double)Math.Acos(res);  //ER-WR-HR  v1 = ER - WR;  v1 = v1.Normalize();  v2 = HR - WR;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  ERWRHR += (double)Math.Acos(res);  //SC-SL-EL  v1 = SC - SL;  v1 = v1.Normalize();  v2 = EL - SL;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SCSLEL += (double)Math.Acos(res);  //SL-EL-WL  v1 = SL - EL;  v1 = v1.Normalize();  v2 = WL - EL;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  SLELWL += (double)Math.Acos(res);  //EL-WL-HL  v1 = EL - WL;  v1 = v1.Normalize();  v2 = HL - WL;  v2 = v2.Normalize();  res = Vector3.DotProduct(v1, v2);  ELWLHL += (double)Math.Acos(res);  //Distance HR - HL  DisHRHL += Vector3.Distance(HR, HL);  Thread.Sleep(10);  }  SRER /= 100;  ERWR /= 100;  WRHR /= 100;  SLEL /= 100;  ELWL /= 100;  WLHL /= 100;  HRHL /= 100;  SCSRER /= 100;  SRERWR /= 100;  ERWRHR /= 100;  SCSLEL /= 100;  SLELWL /= 100;  ELWLHL /= 100;  DisHRHL /= 100;  AllFeatureNormalize(); |

Kode Sumber 4.3 Kode Sumber Ekstraksi Fitur *Skeleton*

### 4.3.3 Implementasi Pembuatan Data Bahasa Isyarat Baru

Dalam implementasi pembuatan data bahasa isyarat baru, hasil ektraksi fitur akan dibagi menjadi 2 golongan sesuai dengan syarat pembagian *neural network* pada Tabel 3.7. Implementasi dapat dilihat seperti pada Kode Sumber 4.4

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | StringBuilder sb = new StringBuilder();  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", SRER.X.ToString("0.00000"), System.Environment.NewLine, SRER.Y.ToString("0.00000"), System.Environment.NewLine, SRER.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", ERWR.X.ToString("0.00000"), System.Environment.NewLine, ERWR.Y.ToString("0.00000"), System.Environment.NewLine, ERWR.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", WRHR.X.ToString("0.00000"), System.Environment.NewLine, WRHR.Y.ToString("0.00000"), System.Environment.NewLine, WRHR.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", SLEL.X.ToString("0.00000"), System.Environment.NewLine, SLEL.Y.ToString("0.00000"), System.Environment.NewLine, SLEL.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", ELWL.X.ToString("0.00000"), System.Environment.NewLine, ELWL.Y.ToString("0.00000"), System.Environment.NewLine, ELWL.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", WLHL.X.ToString("0.00000"), System.Environment.NewLine, WLHL.Y.ToString("0.00000"), System.Environment.NewLine, WLHL.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", HRHL.X.ToString("0.00000"), System.Environment.NewLine, HRHL.Y.ToString("0.00000"), System.Environment.NewLine, HRHL.Z.ToString("0.00000")));  sb.AppendLine(String.Format("{0}", SCSRER.ToString("0.00000")));  sb.AppendLine(String.Format("{0}", SRERWR.ToString("0.00000")));  sb.AppendLine(String.Format("{0}", ERWRHR.ToString("0.00000")));  sb.AppendLine(String.Format("{0}", SCSLEL.ToString("0.00000")));  sb.AppendLine(String.Format("{0}", SLELWL.ToString("0.00000")));  sb.AppendLine(String.Format("{0}", ELWLHL.ToString("0.00000")));  sb.Append(String.Format("{0}", DisHRHL.ToString("0.00000")));  int gol = 0;  if (ERWR.Y < 0.53)  gol = 1;  else  gol = 2;  result = true;    int count = 0;  string fullPath;  if (gol == 1)  {  filename = string.Format("gol1.{0}", filename);  fullPath = @path + "gol1-normalize\\" + filename;  }  else  {  filename = string.Format("gol2.{0}", filename);  fullPath = @path + "gol2-normalize\\" + filename;  }  while (File.Exists(fullPath))  {  string[] temp = filename.Split('.');  count++;  if(gol == 1)  filename = String.Format("gol1.{0}.{1}.txt", temp[1], count.ToString());  else  filename = String.Format("gol2.{0}.{1}.txt", temp[1], count.ToString());  if (gol == 1)  fullPath = @path + "gol1-normalize\\" + filename;  else  fullPath = @path + "gol2-normalize\\" + filename;  }  File.AppendAllText(fullPath, sb.ToString()); |

Kode Sumber 4.4 Implementasi Pembuatan Data Bahasa Isyarat Baru

### 4.3.4 Implementasi Proses *Training* dan *Testing*

Pada proses *Training* dan *Testing*, penulis menggunakan *classifier* dengan metode *Back Propagation-Genetic Algorithm* yang pada Tugas Akhir sebelumnya sudah diimplementasikan oleh Risal Andika Tidisaputra. Dengan melakukan beberapa modifikasi kode, aplikasi ini dapat menggunakannya. Untuk proses *training*, Implementasi dapat dilihat pada Kode Sumber 4.5. Output dari proses *training* adalah sebuah *neural network* dalam format .xml yang nantinya akan digunakan untuk proses *testing*. Implementasi proses *Testing* dapat dilihat pada Kode Sumber 4.6.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102 | float avgError1 = 0;  float avgError2 = 0;  NeuralNetwork nn1 = new NeuralNetwork();  NeuralNetwork nn2 = new NeuralNetwork();  DirectoryInfo info1 = new DirectoryInfo(string.Format("{0}gol1-normalize\\", path));  DirectoryInfo info2 = new DirectoryInfo(string.Format("{0}gol2-normalize\\", path));  DataSetList dsl1 = new DataSetList();  DataSetList dsl2 = new DataSetList();  FileReader fr1 = new FileReader();  FileReader fr2 = new FileReader();  dsl1 = fr1.ReadFile(info1.FullName, cc1);  dsl2 = fr2.ReadFile(info2.FullName, cc2);  StatusDetail.Content = "Training Dataset";  if (algorithm.Equals("BP"))  {  nn1.InitNetwork(dsl1[0].AttributeCount, dsl1[0].AttributeCount / 2, cc1.TargetCount);  nn2.InitNetwork(dsl2[0].AttributeCount, dsl2[0].AttributeCount / 2, cc2.TargetCount);  nn1.Seed = nn2.Seed = 0;  nn1.InitWeight();  nn2.InitWeight();  BackPropagation bp1 = new BackPropagation();  BackPropagation bp2 = new BackPropagation();  bp1.Init(nn1, dsl1, cc1);  bp2.Init(nn2, dsl2, cc2);  avgError1 = bp1.Run(Int32.Parse(iteration\_text.Text));  avgError2 = bp2.Run(Int32.Parse(iteration\_text.Text));  }  else if(algorithm.Equals("BPGA"))  {  GeneticAlgorithm ga1 = new GeneticAlgorithm();  GeneticAlgorithm ga2 = new GeneticAlgorithm();  ga1.Init(dsl1, cc1);  ga2.Init(dsl2, cc2);  //Gunakan ga.Run() jika sebelumnya tidak memiliki network yang bagus  //Chromosom fitChrom1 = ga1.Run();  //Chromosom fitChrom2 = ga2.Run();  //Gunakan chromosom yang sudah ada jika sudah memiliki network yang bagus  loadNet = new NeuralNetwork();  NNtoXMLReader nnr = new NNtoXMLReader(loadNet);  algoTest = nnr.read("gol1ga.xml");  seleksi1 = nnr.chromosom;  loadNet = new NeuralNetwork();  nnr = new NNtoXMLReader(loadNet);  algoTest = nnr.read("gol2ga.xml");  seleksi2 = nnr.chromosom;  for(int i = 0; i < dsl1.Count; i++)  {  int popCount = 0;  for(int j = 0; j < seleksi1.Count(); j++)  {  if(seleksi1[j] == 0)  {  dsl1[i].RemoveBit(j - popCount);  popCount++;  }  }  }  for (int i = 0; i < dsl2.Count; i++)  {  int popCount = 0;  for (int j = 0; j < seleksi2.Count(); j++)  {  if (seleksi2[j] == 0)  {  dsl2[i].RemoveBit(j - popCount);  popCount++;  }  }  }  nn1.InitNetwork(dsl1[0].AttributeCount, dsl1[0].AttributeCount / 2, cc1.TargetCount);  nn2.InitNetwork(dsl2[0].AttributeCount, dsl2[0].AttributeCount / 2, cc2.TargetCount);  nn1.Seed = 0;  nn2.Seed = 0;  nn1.InitWeight();  nn2.InitWeight();  BackPropagation bp1 = new BackPropagation();  BackPropagation bp2 = new BackPropagation();  bp1.Init(nn1, dsl1, cc1);  bp2.Init(nn2, dsl2, cc2);  avgError1 = bp1.Run(Int32.Parse(iteration\_text.Text));  avgError2 = bp2.Run(Int32.Parse(iteration\_text.Text));  }  NNtoXMLWriter nnw1 = new NNtoXMLWriter(nn1, avgError1);  NNtoXMLWriter nnw2 = new NNtoXMLWriter(nn2, avgError2);  if (algorithm.Equals("BP"))  {  nnw1.Write("gol1.xml", algorithm, null);  nnw2.Write("gol2.xml", algorithm, null);  }  else if(algorithm.Equals("BPGA"))  {  nnw1.Write("gol1ga.xml", algorithm, seleksi1);  nnw2.Write("gol2ga.xml", algorithm, seleksi2);  } |

Kode Sumber 4.5 Implementasi Proses *Traning*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108 | int gol;  DataSetList dsl = new DataSetList();  List<float> fitur = new List<float>();  string xmlName = "";  fitur.Add((float)SRER.X);  fitur.Add((float)SRER.Y);  fitur.Add((float)SRER.Z);  fitur.Add((float)ERWR.X);  fitur.Add((float)ERWR.Y);  fitur.Add((float)ERWR.Z);  fitur.Add((float)WRHR.X);  fitur.Add((float)WRHR.Y);  fitur.Add((float)WRHR.Z);  fitur.Add((float)SLEL.X);  fitur.Add((float)SLEL.Y);  fitur.Add((float)SLEL.Z);  fitur.Add((float)ELWL.X);  fitur.Add((float)ELWL.Y);  fitur.Add((float)ELWL.Z);  fitur.Add((float)WLHL.X);  fitur.Add((float)WLHL.Y);  fitur.Add((float)WLHL.Z);  fitur.Add((float)HRHL.X);  fitur.Add((float)HRHL.Y);  fitur.Add((float)HRHL.Z);  fitur.Add((float)SCSRER);  fitur.Add((float)SRERWR);  fitur.Add((float)ERWRHR);  fitur.Add((float)SCSLEL);  fitur.Add((float)SLELWL);  fitur.Add((float)ELWLHL);  fitur.Add((float)DisHRHL);  if (ERWR.Y < 0.53)  {  if (algorithm.Equals("BP"))  xmlName = "gol1.xml";  else if (algorithm.Equals("BPGA"))  xmlName = "gol1ga.xml";  gol = 1;  }  else  {  if (algorithm.Equals("BP"))  xmlName = "gol2.xml";  else if (algorithm.Equals("BPGA"))  xmlName = "gol2ga.xml";  gol = 2;  }  DataSet ds = new DataSet(fitur.Count);  for (int i = 0; i < fitur.Count; i++)  {  ds[i] = fitur[i];  }  dsl.Add(ds);  FeedForward ff = new FeedForward();  loadNet = new NeuralNetwork();  NNtoXMLReader nnr = new NNtoXMLReader(loadNet);  Translate t = new Translate();  algoTest = nnr.read(xmlName);  for (int i = 0; i < dsl.Count; i++)  {  if (algoTest.Equals("BPGA"))  {  int popCount = 0;  for (int j = 0; j < nnr.chromosom.Length; j++)  {  if (nnr.chromosom[j] == 0)  {  dsl[i].RemoveBit(j - popCount);  popCount++;  }  }  }  }  loadNet = new NeuralNetwork();  nnr = new NNtoXMLReader(loadNet);  nnr.read(xmlName);  ff = new FeedForward();  ff.Init(loadNet, dsl);  for (int i = 0; i < dsl.Count; i++)  ff.Run(i);  int[] actualClass = new int[loadNet.OutputLayer.Count];  actualClass = ff.GetActualClass();  t = new Translate(actualClass);  string actualClassLog = "";  for(int i = 0; i < actualClass.Length; i++)  actualClassLog += actualClass[i];  if (gol == 1)  outputText.Content = t.Result(cc1);  else if (gol == 2)  outputText.Content = t.Result(cc2);  string imageFullPath = @imagePath + outputText.Content + ".bmp";  if (File.Exists(imageFullPath))  outputImage.Source = (ImageSource)new ImageSourceConverter().ConvertFrom(imageFullPath);  Log\_ERWRY.Content = ERWR.Y.ToString();  outputList.Items.Clear();  for (int i = 0; i < loadNet.OutputLayer.Count(); i++)  {  outputList.Items.Add(loadNet.OutputLayer[i].Input);  } "; |

Kode Sumber 4.6 Implementasi Proses Testing

### 4.3.4 Implementasi Proses Normalisasi Fitur

Proses normalisasi fitur mengimplementasikan *pseudocode* pada Persamaan 3.1**Error! Reference source not found.**, di mana setiap fitur pengguna dibagi dengan nilai maksimum. Nilai maksimum didapatkan penulis pada saat melakukan pengamatan pada setiap fitur. Normalisasi ini dilakukan supaya *training neural network* berjalan dengan cepat. Setiap fitur bernilai antara 0 sampai 1. Implementasi dapat dilihat pada Kode Sumber 4.7. Implementasi fungsi FeatureNorm() dapat dilihat pada Kode Sumber 4.8.

### 4.3.6 Implementasi Proses Pembagian *Neural Network*

Proses pembagian *neural* *network* pada aplikasi ini dilakukan dengan membagi bahasa isyarat yang ada menjadi 2 golongan. Untuk golongan pertama ditentukan dengan melihat nilai Y pada *vector* ER – WR. Jika Y kurang dari 0.53 maka bahasa isyarat termasuk golongan 1. Selain itu bahasa isyarat akan masuk golongan 2. Implementasi pembagian *Neural Network* dapat dilihat pada Kode Sumber 4.9.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | private void AllFeatureNormalize()  {  SRER = Features.FeatureNorm(SRER);  ERWR = Features.FeatureNorm(ERWR);  WRHR = Features.FeatureNorm(WRHR);  SLEL = Features.FeatureNorm(SLEL);  ELWL = Features.FeatureNorm(ELWL);  WLHL = Features.FeatureNorm(WLHL);  HRHL = Features.FeatureNorm(HRHL);  SCSRER /= Math.PI;  SRERWR /= Math.PI;  ERWRHR /= Math.PI;  SCSLEL /= Math.PI;  SLELWL /= Math.PI;  ELWLHL /= Math.PI;  DisHRHL /= 3.4641f;  } |

Kode Sumber 4.7 Fungsi Normalisasi Fitur

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public class Features  {  public static double Normalize(double x)  {  return (x + 2) / 4;  }  public static Vector3 FeatureNorm(Vector3 u)  {  return new Vector3(Normalize(u.X), Normalize(u.Y), Normalize(u.Z));  }  } |

Kode Sumber 4.8 Implementasi fungsi FeatureNorm() pada class Features

|  |  |
| --- | --- |
| 1  2  3  4  5 | int gol = 0;  if (ERWR.Y < 0.53)  gol = 1;  else  gol = 2; |

Kode Sumber 4.9 Implementasi Pembagian *Neural Network*

# BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsional secara keseluruhan. Pengujian dilakukan dengan beberapa skenario. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

## Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah PC yang memiliki spesifikasi sebagai berikut.

* Prosesor Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
* Memori (RAM) 8,00 GB
* Kinect

## Skenario Pengujian

Pengujian dilakukan terhadap 8 bahasa isyarat yang sudah disediakan. Ke 8 bahasa isyarat tersebut adalah sebagai berikut:

1. Alquran
2. Bentuk
3. Gang
4. Hai
5. Hamba
6. Hormat
7. Ketua
8. Wadah

Dari ke 8 bahasa isyarat tersebut, dibagi menjadi 2 golongan. Golongan pertama berisikan Alquran, Gang, Hamba, dan Wadah. Sedangkan golongan kedua berisikan Bentuk, Hai, Hormat, dan Ketua. Pembagian golongan tersebut bersadarkan ketentuan pada Tabel 3.7.

Skenario pengujian yang dilakukan dibagi menjadi 2 skenario A dan skenario B. Pada skenario A, *neural* *network* yang digunakan pada saat *testing* adalah hasil *training* dengan data *skeleton* penulis. Sedangkan skenario B, *neural* *network* yang digunakan pada saat *testing* adalah hasil *training* dengan data *skeleton* penulis dan satu pengguna.

1. Pengujian skenario A1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh penulis dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis.
2. Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh pengguna dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis.
3. Pengujian skenario B1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh penulis dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis dan 80 *dataset* yang diambil dari data *skeleton* pengguna.
4. Pengujian skenario B2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh pengguna dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis dan 80 *dataset* yang diambil dari data *skeleton* pengguna.

### Pengujian Skenario A1 dan Analisis

Pada pengujian skenario A1 uji coba dilakukan oleh penulis. Uji coba ini menggunakan *dataset* sebanyak 160 buah, semua *dataset* tersebut dilatih dari *skeleton* penulis. Skenario dapat dilihat pada Tabel 5.1.

Tabel 5.1 Skenario Pengujian A1

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi A1 |
| Kode | SP-A1 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Jumlah *Dataset* | 160 *dataset* |
| Penguji | Penulis |
| Prosedur Pengujian | Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 95% |

Hasil yang didapat cukup memuaskan. Kesalahan output terjadi pada bahasa isyarat Hamba dan Hormat. Menurut pengamatan penulis, kesalahan tersebut terjadi karena posisi *skeleton* yang kurang tepat dan ketidakstabilan Kinect dalam mengambil data *skeleton*. Hasil dari skenario pengujian A1 dapat dilihat pada Tabel 5.2.

Tabel 5.2 Terjemahan Isyarat Kata Skenario Pengujian A1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Target Kelas | | | | | | | | |
| Hasil Uji Coba |  | Alquran | Bentuk | Gang | Hai | Hamba | Hormat | Ketua | Wadah |
| Alquran | 5 |  |  |  |  |  |  |  |
| Bentuk |  | 5 |  |  |  |  |  |  |
| Gang |  |  | 5 |  |  |  |  |  |
| Hai |  |  |  | 5 |  |  |  |  |
| Hamba |  |  |  |  | 4 |  |  |  |
| Hormat |  |  |  |  |  | 4 |  |  |
| Ketua |  |  |  |  |  | 1 | 5 |  |
| Wadah |  |  |  |  | 1 |  |  | 5 |

### Pengujian Skenario A2 dan Analisis

Pada pengujian skenario A2 uji coba dilakukan oleh pengguna. Uji coba ini menggunakan *dataset* sebanyak 160 buah, semua *dataset* tersebut dilatih dari *skeleton* penulis. Skenario dapat dilihat pada Tabel 5.3.

Tabel 5.3 Skenario Pengujian A2

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi A2 |
| Kode | SP-A2 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Jumlah *Dataset* | 160 *dataset* |
| Penguji | Pengguna |
| Prosedur Pengujian | Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 82.5% |

Hasil yang didapatkan turun dari hasil pengujian sebelumnya sebesar sebelumnya. Hal ini terjadi karena *skeleton* setiap orang tidak sama ketika ditekuk dan sebab lainnya. Kesalahan output terjadi pada isyarat Alquran. Ketika diterjemahkan mengeluarkan output isyarat Ketua. Isyarat Alquran dan Ketua berada pada golongan yang berbeda. Ketika melakukan perhitungan fitur, menghasilkan nilai yang tidak sesuai sehingga *dataset* yang didapat diterjemahkan dengan *neural* *network* yang tidak sesuai. Hasil dari skenario pengujian A2 dapat dilihat pada Tabel 5.4.

Tabel 5.4 Terjemahan Isyarat Kata Skenario Pengujian A2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Target Kelas | | | | | | | | |
| Hasil Uji Coba |  | Alquran | Bentuk | Gang | Hai | Hamba | Hormat | Ketua | Wadah |
| Alquran | 4 |  |  |  |  |  |  |  |
| Bentuk |  | 2 |  | 2 |  |  |  |  |
| Gang |  | 3 | 5 |  |  |  |  |  |
| Hai |  |  |  | 3 |  |  |  |  |
| Hamba |  |  |  |  | 4 |  |  |  |
| Hormat |  |  |  |  |  | 5 |  |  |
| Ketua | 1 |  |  |  |  |  | 5 |  |
| Wadah |  |  |  |  | 1 |  |  | 5 |

### Pengujian Skenario B1 dan Analisis

Pada pengujian skenario B1 uji coba dilakukan oleh penulis. Uji coba ini menggunakan *dataset* sebanyak 240 buah, 160 *dataset* dilatih dari *skeleton* penulis dan 80 *dataset* dilatih dari *skeleton* pengguna. Skenario dapat dilihat pada Tabel 5.5.

Tabel 5.5 Skenario Pengujian B1

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi B1 |
| Kode | SP-B1 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Jumlah *Dataset* | 240 *Dataset* |
| Penguji | Penulis |
| Prosedur Pengujian | Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 97.5% |

Hasil yang didapat meningkat 2.5% dari skenario pengujian A1. Kesalahan output terjadi pada bahasa isyarat Hai. Kesalahan yang terjadi karena posisi *skeleton* yang kurang tepat sehingga terjadi perbedaan saat perhitungan. Hasil dari skenario pengujian B1 dapat dilihat pada Tabel 5.6.

Tabel 5.6 Terjemahan Isyarat Kata Skenario Pengujian B1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Target Kelas | | | | | | | | |
| Hasil Uji Coba |  | Alquran | Bentuk | Gang | Hai | Hamba | Hormat | Ketua | Wadah |
| Alquran | 5 |  |  |  |  |  |  |  |
| Bentuk |  | 5 |  |  |  |  |  |  |
| Gang |  |  | 5 |  |  |  |  |  |
| Hai |  |  |  | 4 |  |  |  |  |
| Hamba |  |  |  |  | 5 |  |  |  |
| Hormat |  |  |  |  |  | 5 |  |  |
| Ketua |  |  |  | 1 |  |  | 5 |  |
| Wadah |  |  |  |  |  |  |  | 5 |

### Pengujian Skenario B2 dan Analisis

Pada pengujian skenario B2 uji coba dilakukan oleh pengguna. Uji coba ini menggunakan *dataset* sebanyak 240 buah, 160 *dataset* dilatih dari *skeleton* penulis dan 80 *dataset* dilatih dari *skeleton* pengguna. Skenario dapat dilihat pada Tabel 5.5.

Tabel 5.7 Skenario Pengujian B2

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi B2 |
| Kode | SP-B2 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Jumlah *Dataset* | 240 *Dataset* |
| Penguji | Pengguna |
| Prosedur Pengujian | Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 95% |

Hasil yang didapat setelah penambahan *dataset* pengguna meningkat cukup signifikan. Hal ini terjadi karena saat proses pelatihan memakai *dataset* yang lebih bervariasi. Kesalahan output yang terjadi pada skenario pengujian B2 disebabkan karena kurang tepatnya posisi *skeleton* saat pengambilan data. Hasil skenario pengujian B2 dapat dilihat pada Tabel 5.8.

Tabel 5.8 Terjemahan Isyarat Kata Skenario Pengujian B2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Target Kelas | | | | | | | | |
| Hasil Uji Coba |  | Alquran | Bentuk | Gang | Hai | Hamba | Hormat | Ketua | Wadah |
| Alquran | 5 |  |  |  |  |  |  |  |
| Bentuk |  | 5 |  |  |  |  |  |  |
| Gang |  |  | 5 |  |  |  |  |  |
| Hai |  |  |  | 4 |  |  |  |  |
| Hamba |  |  |  |  | 4 |  |  |  |
| Hormat |  |  |  | 1 |  | 5 |  |  |
| Ketua |  |  |  |  |  |  | 5 |  |
| Wadah |  |  |  |  | 1 |  |  | 5 |

## Evaluasi

Subbab ini membahas mengenai evaluasi terhadap pengujian-pengujian yang telah dilakukan. Dalam hal ini evaluasi menunjukkan data akurasi dari hasil pengujian akurasi *neural network* yang telah dilakukan sebelumnya.

### Evaluasi Pengujian Akurasi *Neural Network*

Evaluasi pengujian akurasi *neural network* dilakukan dengan menampilkan data hasil pengujian yang telah dipaparkan pada subbab 5.2. dalam hal ini, pengujian disusun dalam bentuk grafik yang dapat dilihat pada Gambar 5.1. dari data yang terdapat pada grafik tersebut, diketahui bahwa aplikasi yang dibuat telah memiliki akurasi yang cukup baik yaitu lebih dari 80%.

Gambar 5.1 Hasil Pengujian Tingkat Akurasi *Neural Network*

# BAB VI KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

## Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Akurasi yang didapat dengan menggunakan metode *Back Propagation – Genetic Algorithm* untuk membentuk sebuah *classifier* memiliki hasil yang sangat baik.
2. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80%.
3. Kinect merupakan perangkat keras yang sangat sensitif terhadap gerakan, perbedaan yang sangat kecil saja dapat membuat hasil berbeda. Selain itu Kinect juga sensitif terharap Intensitas cahaya, Kestabilan *skeleton* dapat dijaga dengan menggunakan Kinect di ruangan yang memiliki cahaya yang cukup.

## 6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Kinect yang digunakan dalam Tugas Akhir ini adalah Kinect V1 yang memiliki fitur-fitur yang sangat terbatas. Saat ini sudah ada Kinect V2 yang memiliki fitur tambahan seperti mendeteksi gerakan jadi. Untuk membuat penelitian yang sama lebih baik menggunakan Kinect V2 sehingga fitur yang digunakan akan lebih baik dan akurat.
2. Memperbanyak model *neural network* dengan cara melakukan *training* dari berbagai pengguna.

# 

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | A. W. Yanuardi, P. Samudra and J. A. Purnama P., "Indonesian Sign Language Computer Application for The Deaf," *IEEE,* 2010. |
| [2] | N. Ulfah, "5.000 Bayi Indonesia Lahir Tuli Setiap Tahun," Detik Health, 9 Januari 2010. [Online]. Available: http://health.detik.com/read/2010/01/09/155558/1274969/763/. [Accessed 26 Desember 2014]. |
| [3] | L. Motion, "Leap Motion," Leap Motion Inc., 2012. [Online]. Available: https://developer.leapmotion.com/documentation/csharp/index.html. [Accessed 22 Desember 2014]. |
| [4] | S. Aliyu, M. Mohandes and M. Deriche, "Arabic Sign Language Recognition using the Leap Motion Controller," *IEEE,* 2014. |
| [5] | O. Mencer. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise\_96/journal/vol4/cs11/report.html. [Accessed 6 Juni 2015]. |
| [6] | Aberdeen's Robert Gordon University, "www.rgu.ac.uk," [Online]. Available: https://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf. [Accessed 2 Maret 2015]. |
| [7] | S. Wang, S. Yin and M. Jiang, "Hybrid Neural Network Based On GA-BP for Personal Credit Scoring," *Fourth International Conference on Natural Computation,* 2008. |
| [8] | E. Rakun, M. Andriarni, I. W. Wiprayoga, K. Danniswara and A. Tjandra, "Combining Depth Image and Skeleton Data from Kinect for Recognizing Words in the Sign System for Indonesian Language (SIBI)," *IEEE,* 2013. |

# Lampiran A Kode Sumber

|  |
| --- |
| class BackPropagation  {  private float learningRate = 0.03f;  private FeedForward feedForward;  private NeuralNetwork lastStableNetwork;  public NeuralNetwork Network{  set{  feedForward.Network = value;  }  get{  return feedForward.Network;  }  }  public ListDataSet DataSetList  {  get{  return feedForward.DataSetList;  }  }  private ClassificationClass classificationClass;  public void Initialise(NeuralNetwork nn, ListDataSet dsl, ClassificationClass cc){  feedForward = new FeedForward();  feedForward.Initialise(nn, dsl);  classificationClass = cc;  }  public float Run(int maxIter)  {  float lastAvgError = float.MaxValue;  float avgError =0;  for (int it = 0; it < maxIter; it++){  avgError = 0;  for (int k = 0; k < DataSetList.Count; k++){  Network.InitaliseInput();  feedForward.Run(k);  float[] errorOutput = GetErrorOutput(k);  float totalError = 0;  for (int i = 0; i < errorOutput.Length; i++){  totalError += Math.Abs(errorOutput[i]);  }  avgError += totalError;  }  avgError /= DataSetList.Count;  }  return avgError;  }  private float[] GetErrorOutput(int index){  float[] outputError = new float[classificationClass.TargetCount];  for (int i = 0; i < classificationClass.TargetCount; i++){  outputError[i] = classificationClass.GetTarget(DataSetList[index].ClassName)[i] - Network.OutputLayer[i].Input;  }  UpdateWeightHidden(outputError, index);  return outputError;  }  private void UpdateWeightInput(float[] hiddenError, int index){  for (int i = 0; i < Network.InputLayer.Count; i++){  for (int j = 0; j < Network.HiddenLayer.Count; j++){  float newWeight = Network.InputLayer[i].GetWeight(j) + (learningRate \* hiddenError[j] \* Network.InputLayer[i].Input);  Network.InputLayer[i].SetWeight(j, newWeight);  }  }  }  private void UpdateWeightHidden(float[] outputError, int index){  for (int i = 0; i < Network.HiddenLayer.Count; i++){  for (int j = 0; j < Network.OutputLayer.Count; j++){  float newWeight = Network.HiddenLayer[i].GetWeight(j) + (this.learningRate \* outputError[j] \* Network.HiddenLayer[i].Input);  Network.HiddenLayer[i].SetWeight(j, newWeight);  }  }  GetErrorHidden(outputError, index);  }  private void GetErrorHidden(float[] outputError, int index){  float[] hiddenError = new float[Network.HiddenLayer.Count];  for (int i = 0; i < Network.HiddenLayer.Count; i++){  float linear = 0;  for (int j = 0; j < Network.OutputLayer.Count; j++){  linear += outputError[j] \* Network.HiddenLayer[i].GetWeight(j);  }  hiddenError[i] = Network.HiddenLayer[i].Input \* (1 - Network.HiddenLayer[i].Input) \* linear;  }  UpdateWeightInput(hiddenError, index);  } |

Kode Sumber 7. 1 Kelas Backpropagation.cs

|  |
| --- |
| class Chromosom  {  private int[] bit;  public int[] Bit{  set{  bit = value;  }  get{  return bit;  }  }  public int this[int index]{  set{  bit[index] = value;  }  get{  return bit[index];  }  }  public int Length{  get{  return bit.Length;  }  }  private float fitnessValue = 0;  public float FitnessValue{  set{  fitnessValue = value;  }  get{  return fitnessValue;  }  }  public Chromosom(int[] newBit){  bit = newBit;  }  public void Print(){  for (int i = 0; i < bit.Length; i++){  Console.Write(bit[i] + " ");  }  }  } |

Kode Sumber 7. 2 Kelas Chromosom.cs

|  |
| --- |
| class ClassificationClass  {  private List<string> classList = new List<string>();  private int[] actualClass;  public int TargetCount  {  get  {  int factor = 1;  while (Math.Pow(2, factor) < classList.Count)  factor++;  return factor;  }  }  public void Add(string className)  {  int index = GetIndex(className);  if (index == -1)  {  classList.Add(className);  }  }  public void Clear()  {  classList.Clear();  }  public int GetIndex(string className)  {  return classList.IndexOf(className);  }  public int[] GetTarget(int index)  {  int[] target = new int[TargetCount];  int bitCount = target.Length - 1;  while (index > 0)  {  target[bitCount--] = (index % 2);  index = index / 2;  }  return target;  }  public int[] GetTarget(string className)  {  return GetTarget(GetIndex(className));  }  } |

Kode Sumber 7. 3 Kelas ClassificationClass.cs

|  |
| --- |
| class DataSet  {  public string ClassName { set; get; }  private List<float> \_attribute;  public float this[int index]  {  set  {  \_attribute[index] = value;  }  get  {  return \_attribute[index];  }  }  public int AttributeCount  {  get  {  return \_attribute.Count;  }  }  public DataSet()  {  \_attribute = new List<float>();  }  public DataSet(int attributeCount)  {  \_attribute = new List<float>();  for (int i = 0; i < attributeCount; i++)  \_attribute.Add(0);  }  public DataSet(DataSet ds)  {  \_attribute = new List<float>();  for (int i = 0; i < ds.AttributeCount; i++)  {  \_attribute.Add(ds[i]);  }  this.ClassName = ds.ClassName;  }  public void RemoveBit(int index)  {  \_attribute.RemoveAt(index);  }  } |

Kode Sumber 7. 4 Kelas DataSet.cs

|  |
| --- |
| class FeedForward  {  private NeuralNetwork neuralNetwork;  public NeuralNetwork Network  {  set{  neuralNetwork = value;  }  get{  return neuralNetwork;  }  }  private ListDataSet dataSetList;  public ListDataSet DataSetList  {  get{  return dataSetList;  }  }    public void Initialise(NeuralNetwork nn, ListDataSet dsl){  neuralNetwork = nn;  dataSetList = dsl;  }  public void Run(int index){  neuralNetwork.InitaliseInput();  DoInputLayer(index);  }  public void Initialise(NeuralNetwork nn, ListDataSet dsl){  neuralNetwork = nn;  dataSetList = dsl;  }  public void Run(int index){  neuralNetwork.InitaliseInput();  DoInputLayer(index);  }  private void DoInputLayer(int index){  for (int i = 0; i < dataSetList[index].AttributeCount; i++){  neuralNetwork.InputLayer[i].Input = dataSetList[index][i];  }  DoHiddenLayer();  }  private void DoHiddenLayer()  {  for (int i = 0; i < neuralNetwork.InputLayer.Count; i++){  for (int j = 0; j < neuralNetwork.HiddenLayer.Count; j++){  neuralNetwork.HiddenLayer[j].Input += neuralNetwork.InputLayer[i].GetOutput(j);  }  }  for (int j = 0; j < neuralNetwork.HiddenLayer.Count; j++){  neuralNetwork.HiddenLayer[j].Input = Sigmoid(neuralNetwork.HiddenLayer[j].Input);  }  DoOutputLayer();  }  private void DoOutputLayer(){  for (int i = 0; i < neuralNetwork.HiddenLayer.Count; i++){  for (int j = 0; j < neuralNetwork.OutputLayer.Count; j++)  {  neuralNetwork.OutputLayer[j].Input += neuralNetwork.HiddenLayer[i].Input \* neuralNetwork.HiddenLayer[i].GetWeight(j);  }  }  }  private float Sigmoid(float val){  return 1 / (1.0f + (float)Math.Exp(-val));  } |

Kode Sumber 7. 5 Kelas FeedForward.cs

|  |
| --- |
| class GeneticAlgorithm  {  const int individualCount = 3;  private Random random = new Random();  private ClassificationClass classificationClass;  private BackPropagation backPropagation;  private NeuralNetwork network;  private ListDataSet listDataSet;  private List<Chromosom> chromosoms;  public void Initialize(ListDataSet lds, ClassificationClass cc){  listDataSet = lds;  classificationClass = cc;  }  public Chromosom Run(){  ChromosomInit();  int iteration = 3;  for (int i = 0; i < iteration; i++){  DoBackPropagation();  DoSelection();  DoCrossOver();  int chanceMutation = GetRandom();  if (chanceMutation < GetRandom()){  DoMutation();  }  }    int index = 0;  for (int i = 1; i < chromosoms.Count; i++){  if (chromosoms[i].FitnessValue > chromosoms[index].FitnessValue){  index = i;  }  }  Chromosom fittestChromosom = chromosoms[index];  return fittestChromosom;  }    private void ChromosomInit()  {  chromosoms = new List<Chromosom>(individualCount);  for (int i = 0; i < individualCount; i++){  chromosoms.Add(new Chromosom(GetRandomBinary()));  }  }  private void DoSelection(){  chromosoms.Sort((val1, val2) => val1.FitnessValue.CompareTo(val2.FitnessValue));  }    private void DoCrossOver(){  for (int i = 0; i < chromosoms.Count; i += 2){  int nextIndex = i + 1;  if (nextIndex >= chromosoms.Count)  nextIndex = i - 1;  int[] bit1 = chromosoms[i].Bit;  int[] bit2 = chromosoms[nextIndex].Bit;  for (int j = bit1.Length / 2; j < bit1.Length; j++){  int t = bit1[j];  bit1[j] = bit2[j];  bit2[j] = t;  }  chromosoms[i].Bit = bit1;  chromosoms[nextIndex].Bit = bit2;  }  }    private void DoMutation()  {  for (int i = 0; i < chromosoms.Count; i++){  int chanceChromoMut = GetRandom();  if (chanceChromoMut <= GetRandom()){  int bitIndex = GetRandom(chromosoms[i].Length);  if (chromosoms[i][bitIndex] == 0) chromosoms[i][bitIndex] = 1;  else chromosoms[i][bitIndex] = 0;  }  }  }    private void DoBackPropagation(){  for (int i = 0; i < chromosoms.Count; i++){  ListDataSet lds = new ListDataSet(listDataSet);  for (int j = 0; j < lds.Count; j++){  int popCount = 0;  for (int k = 0; k < chromosoms[i].Length; k++){  if (chromosoms[i][k] == 0){  lds[j].RemoveBit(k - popCount);  popCount++;  }  }  }  NeuralNetwork nn = new NeuralNetwork();  nn.InitialiseNetwork(lds[0].AttributeCount, lds[0].AttributeCount / 2, classificationClass.TargetCount);  nn.InitialiseWeight();    BackPropagation bp = new BackPropagation();  bp.Initialise(nn, lds, classificationClass);  bp.Run(5000);    FeedForward ff = new FeedForward();  ff.Initialise(nn, lds);    int totalCorrect = 0;  for (int j = 0; j < lds.Count; j++){  ff.Run(j);  bool correct = true;  int[] targetClass = classificationClass.GetTarget(lds[j].ClassName);  for (int k = 0; k < ff.GetActualClass().Length; k++){  if (targetClass[k] != ff.GetActualClass()[k])  correct = false;  }  if (correct)totalCorrect++;  }  chromosoms[i].FitnessValue = totalCorrect / (float)lds.Count;  }  }  } |

Kode Sumber 7. 6 Kelas GeneticAlgorithm

|  |
| --- |
| class NeuralNetwork  {  private Random random = new Random();  public int Seed{  set{  random = new Random(value);  }  }  public float treshold = 0.5f;  public List<Node> InputLayer { set; get; }  public List<Node> HiddenLayer { set; get; }  public List<Node> OutputLayer { set; get; }  public NeuralNetwork(){  }  public NeuralNetwork(NeuralNetwork newNN){  newNN.InitaliseInput();  InitialiseNetwork(newNN.InputLayer.Count, newNN.HiddenLayer.Count, newNN.OutputLayer.Count);  for (int i = 0; i < newNN.InputLayer.Count; i++){  InputLayer[i] = newNN.InputLayer[i];  }  for (int i = 0; i < newNN.HiddenLayer.Count; i++){  HiddenLayer[i] = newNN.HiddenLayer[i];  }  }  public void InitialiseNetwork(int inputLayerCount, int hiddenLayerCount, int outputLayerCount){  InputLayer = new List<Node>();  HiddenLayer = new List<Node>();  OutputLayer = new List<Node>();  for (int i = 0; i < inputLayerCount; i++){  InputLayer.Add(new Node(hiddenLayerCount));  }  for (int i = 0; i < hiddenLayerCount; i++){  HiddenLayer.Add(new Node(outputLayerCount));  }  for (int i = 0; i < outputLayerCount; i++){  OutputLayer.Add(new Node(1));  OutputLayer[i].SetWeight(0, 1);  }  }  public void InitaliseInput()  {  for (int i = 0; i < InputLayer.Count; i++){  InputLayer[i].Input = 0;  }  for (int i = 0; i < HiddenLayer.Count; i++){  HiddenLayer[i].Input = 0;  }  for (int i = 0; i < OutputLayer.Count; i++){  OutputLayer[i].Input = 0;  }  }  public void InitialiseWeight(){  for (int i = 0; i < InputLayer.Count; i++){  for (int j = 0; j < HiddenLayer.Count; j++){  InputLayer[i].SetWeight(j, GetRandom() / 100.0f);  }  }  for (int i = 0; i < HiddenLayer.Count; i++){  for (int j = 0; j < OutputLayer.Count; j++){  HiddenLayer[i].SetWeight(j, GetRandom() / 100.0f);  }  }  }  int GetRandom(){  return random.Next(0, 30);  }  } |

Kode Sumber 7. 7 Kelas NeuralNetwork.cs

# BIODATA PENULIS

Penulis, Risal Andika Tridisaputra lahir di Jember, Jawa Timur, pada tanggal 9 Juni 1993. Penulis adalah anak ke-3 dari 3 bersaudara dan dibesarkan di Kota Surabaya. Penulis menempuh pendidikan formal di SDN AL-Falah Surabaya (1999-2005), SMPN 32 Surabaya (2005-2008), dan SMAN 2 Surabaya (2008-2011). Pada tahun 2011, penulis menempuh pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember, Surabaya, Jawa Timur, Indonesia.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Interaksi Grafis dan Seni dan memiliki kompetensi pada beberapa subjek seperti Desain Web, Pemrograman Android, Pemrograman Windows Phone, dan Big Data. Selama berada di dunia akademi kampus, penulis aktif sebagai asisten dosen untuk mata kuliah Interaksi Manusia dan Komputer. Selain itu penulis juga aktif dalam bidang nonakademik. Organisasi mahasiswa yang pernah diikuti penulis adalah menjadi Staf Departemen Dalam Negeri pada Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) periode 2012-2013, Ketua Sponsorship dalan acara Schematics 2013 yang diselenggarakan oleh Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) periode 2013-2014, dan menjadi peserta di berbagai kegiatan jurusan. Penulis juga pernah menjadi finalis pada beberapa ajang perlombaan seperti, VOCOMFEST tahun 2014, ENUMERATION tahun 2014, dan DISCOVERY tahun 2015. Penulis dapat dihubungi melalui alamat email risal.andika@gmail.com