**

**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]*

# C:\Users\Risal Andika T\Downloads\RISAL\LEMPENG_CAP.pngLEMBAR PENGESAHAN

**RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI LEAP MOTION 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:

**RISAL ANDIKA TRIDISAPUTRA**

NRP. 511110033

Disetujui oleh Pembimbing Tugas Akhir:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Wijayanti Nurul Khotimah, S.Kom., M.Sc. | | …………………… |
|  | NIP: 19860312 201212 2 004 | (pembimbing 1) | |
|  |  |  | |
| 2. | Ridho Rahman Hariadi, S.Kom., M.Sc. | | …………………… |
|  | NIP: 19870213 201404 1 001 | (pembimbing 2) | |

**SURABAYA**

**JUNI, 2015**

*[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 2015

Penulis

# DAFTAR ISI

[1.1 Latar Belakang 1](#_Toc452968718)

[1.2 Rumusan Permasalahan 2](#_Toc452968719)

[1.3 Batasan Permasalahan 2](#_Toc452968720)

[1.4 Tujuan 3](#_Toc452968721)

[1.5 Manfaat 3](#_Toc452968722)

[2.1 Tunarungu 5](#_Toc452968723)

[2.2 Bahasa Isyarat 5](#_Toc452968724)

[2.3 Kinect 6](#_Toc452968725)

[2.4 Kinect SDK 8](#_Toc452968726)

[2.5 *Neural* *Network* 9](#_Toc452968727)

[2.6 *Backpropagation* 9](#_Toc452968728)

[2.7 *Backpropagation*-*Genetic* *Algorithm* *Neural* *Network* 10](#_Toc452968729)

[3.1 Analisis Perangkat Lunak 14](#_Toc452968730)

[3.1.1 Deskripsi Umum Perangkat Lunak 15](#_Toc452968731)

[3.1.2 Spesifikasi Kebutuhan Perangkat Lunak 15](#_Toc452968732)

[3.1.3 Identifikasi Pengguna 16](#_Toc452968733)

[3.2 Perancangan Perangkat Lunak 16](#_Toc452968734)

[3.2.1 Model Kasus Penggunaan 16](#_Toc452968735)

[3.2.2 Definisi Aktor 17](#_Toc452968736)

[3.2.3 Definisi Kasus Penggunaan 17](#_Toc452968737)

[3.2.4 Arsitektur Umum Sistem 22](#_Toc452968738)

[3.2.5 Rancangan Antarmuka Aplikasi 23](#_Toc452968739)

[3.2.6 Rancangan Proses Aplikasi 24](#_Toc452968740)

[4.1 Lingkungan Pembangunan 29](#_Toc452968741)

[4.1.1 Lingkungan Pembangunan Perangkat Keras 29](#_Toc452968742)

[4.1.2 Lingkungan Pembangunan Perangkat Lunak 29](#_Toc452968743)

[4.2 Implementasi Antarmuka 30](#_Toc452968744)

[4.2.1 Implementasi Antarmuka Halaman Utama 30](#_Toc452968745)

[4.2.2 Implementasi Antarmuka Halaman *Training* 30](#_Toc452968746)

[4.2.3 Implementasi Antarmuka Halaman *Testing* 32](#_Toc452968747)

[4.3 Implementasi Aplikasi 34](#_Toc452968748)

[4.3.1 Implementasi Pendeteksian Lokasi Objek 34](#_Toc452968749)

[4.3.2. Implementasi Proses Ekstraksi Fitur 35](#_Toc452968750)

[4.3.3 Implementasi Algoritma BPGA 37](#_Toc452968751)

[4.3.4 Implementasi Proses Normalisasi Fitur 45](#_Toc452968752)

[4.3.5 Implementasi Proses Kalibrasi Tangan 45](#_Toc452968753)

[4.3.6 Implementasi Proses Pembagian *Neural Network* 48](#_Toc452968754)

[5.1. Lingkungan Pembangunan 51](#_Toc452968755)

[5.2. Skenario Pengujian 51](#_Toc452968756)

[5.2.1 Pengujian Skenario A1 dan Analisis 52](#_Toc452968757)

[5.2.2 Pengujian Skenario A2 dan Analisis 54](#_Toc452968758)

[5.2.3 Pengujian Skenario A3 dan Analisis 56](#_Toc452968759)

[5.2.4 Pengujian Skenario A4 dan Analisis 56](#_Toc452968760)

[5.2.5 Pengujian Skenario B1 dan Analisis 58](#_Toc452968761)

[5.2.6 Pengujian Skenario B2 dan Analisis 60](#_Toc452968762)

[5.2.7 Pengujian Skenario B3 dan Evaluasi 61](#_Toc452968763)

[5.2.8 Pengujian Skenario B4 dan Evaluasi 63](#_Toc452968764)

[5.3. Evaluasi 66](#_Toc452968765)

[5.3.1 Perbandingan Akurasi Model *Neural Network* Terhadap Algoritma yang Dipakai 66](#_Toc452968766)

[5.3.2 Perbandingan Akurasi Model Jika Ada Penambahan *Dataset* 66](#_Toc452968767)

[5.3.3 Perbandingan Akurasi Model Jika Menggunakan Kalibrasi 67](#_Toc452968768)

[5.3.4 Perbandingan Akurasi Model Terhadap Jumlah Iterasi 68](#_Toc452968769)

[5.3.5 Perbandingan *Runtime* Aplikasi Model *Neural Network* Terhadap Algoritma yang Dipakai 68](#_Toc452968770)

[6.1 Kesimpulan 71](#_Toc452968771)

[6.2 Saran 71](#_Toc452968772)

*[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 9](#_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 17](#_Toc423533212)

[Gambar 3. 2 Arsitektur Sistem (*Training*) 22](#_Toc423533217)

[Gambar 3. 3 Arsitektur Sistem (*Testing*) 23](#_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 30](#_Toc423533229)

[Gambar 4. 2 Antarmuka *Training* (Tabel Ekstraksi Fitur) 31](#_Toc423533230)

[Gambar 4. 3 Antarmuka *Training* (Pilihan *Training* Data) 31](#_Toc423533231)

[Gambar 4. 4 Antarmuka *Training* (Proses ekstraksi fitur) 32](#_Toc423533232)

[Gambar 4. 5 Antarmuka *Testing* 32](#_Toc423533233)

[Gambar 4. 6 Antarmuka *Testing* (Proses Ekstraksi Fitur) 33](#_Toc423533234)

[Gambar 4. 7 Antarmuka Kalibrasi 33](#_Toc423533235)

[Gambar 5. 1 Perbandingan Huruf A dan J 55](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533259)

[Gambar 5. 2 Perbandingan Huruf M dan N 56](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533260)

[Gambar 5. 3 Perbandingan Huruf Skenario Pengujian B1 59](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533266)

[Gambar 5. 4 Perbandingan Huruf Skenario Pengujian B3 63](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533272)

[Gambar 5. 5 Perbandingan Huruf I dan J 65](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533275)

[Gambar 5. 6 Perbandingan Huruf K dan V 65](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%202).docx#_Toc423533276)

[Gambar 5. 7 Grafik Perbandingan Terhadap Algoritma 67](#_Toc423533277)

[Gambar 5. 8 Grafik Perbandingan Terhadap Proses Kalibrasi 67](#_Toc423533278)

[Gambar 5. 9 Grafik Perbandingan Terhadap Jumlah Iterasi 68](#_Toc423533279)

[Gambar 5. 10 Grafik Perbandingan *Runtime* Skenario A1 69](#_Toc423533280)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR TABEL

[Tabel 3. 1 Definisi Kasus Penggunaan 17](#_Toc423531846)

[Tabel 3. 2 Definisi Kasus Penggunaan 18](#_Toc423531847)

[Tabel 3. 3 Spesifikasi Kasus Penggunaan Menambahkan Data Pelatihan 18](#_Toc423531848)

[Tabel 3. 4 Spesifikasi Kasus Penggunaan Latihan Gerakan Isyarat Huruf 20](#_Toc423531849)

[Tabel 3. 5 *Rule* Pembagian *Neural Network* 27](#_Toc423531859)

[Tabel 5. 1 Skenario Pengujian A1 53](#_Toc423531887)

[Tabel 5. 2 Tabel Perbandingan Akurasi Algoritma BP dan BPGA 53](#_Toc423531888)

[Tabel 5. 3 Tabel Perbandingan *Runtime* Algoritma (detik) 54](#_Toc423531889)

[Tabel 5. 4 Skenario Pengujian A2 54](#_Toc423531890)

[Tabel 5. 5 Terjemah Huruf Isyarat Skenario Pengujian A2 55](#_Toc423531891)

[Tabel 5. 6 Skenario Pengujian A3 56](#_Toc423531894)

[Tabel 5. 7 Terjemah Huruf Isyarat Pengujian Skenario A3 57](#_Toc423531895)

[Tabel 5. 8 Perbandingan Akurasi Terhadap Jumlah Iterasi 57](#_Toc423531897)

[Tabel 5. 9 Skenario Pengujian B1 58](#_Toc423531898)

[Tabel 5. 10 Terjemah Huruf Isyarat Pengujian Skenario B1 59](#_Toc423531900)

[Tabel 5. 11 Tabel Terjemah Skenario Pengujian B2 60](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%201).docx#_Toc423531901)

[Tabel 5. 12 Skenario Pengujian B2 61](#_Toc423531902)

[Tabel 5. 13 Skenario Pengujian B3 62](#_Toc423531903)

[Tabel 5. 14 Tabel Terjemahan Huruf Skenario B3 62](file:///D:\TA\BUKU\BUKU\Buku%20iSyarat%20(Revisi%201).docx#_Toc423531904)

[Tabel 5. 15 Terjemah Huruf Isyarat Pengujian Skenario B4 64](#_Toc423531906)

[Tabel 5. 16 Skenario Pengujian B4 64](#_Toc423531907)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR KODE SUMBER

[Kode Sumber 4. 1 Kode Sumber Pendeteksian Telapak Tangan 34](#_Toc423531949)

[Kode Sumber 4. 2. Kode Sumber Kelas *LeapMouse.cs* 35](#_Toc423531950)

[Kode Sumber 4. 3 Kode Sumber Ekstraksi Fitur Tangan 37](#_Toc423531951)

[Kode Sumber 4. 4 Fungsi Inisialisasi Bobot Pada Kelas *NeuralNetwork.cs* 37](#_Toc423531952)

[Kode Sumber 4. 5 Kelas Feedforward.cs 39](#_Toc423531953)

[Kode Sumber 4. 6 Fungsi GetErrorOutput Pada Kelas Backpropagation.cs 40](#_Toc423531954)

[Kode Sumber 4. 7 Fungsi ChromosomInit pada kelas GeneticAlgorithm.cs 41](#_Toc423531955)

[Kode Sumber 4. 8 Fungsi UpdateWeight di *Hidden* *Layer* Pada Kelas Backpropagation.cs 41](#_Toc423531956)

[Kode Sumber 4. 9 Fungsi GetErrorHidden Pada Kelas Backpropagation.cs 42](#_Toc423531957)

[Kode Sumber 4. 10 Fungsi UpdateWeightInput Pada Kelas Backpropagation.cs 42](#_Toc423531958)

[Kode Sumber 4. 11 Fungsi DoSelection pada kelas GeneticAlgorithm.cs 42](#_Toc423531959)

[Kode Sumber 4. 12 Fungsi DoCrossOver pada kelas GeneticAlgorithm.cs 43](#_Toc423531960)

[Kode Sumber 4. 13 Fungsi DoBackpropagation pada kelas GeneticAlgorithm.cs 45](#_Toc423531961)

[Kode Sumber 4. 14 Fungsi DoMutation pada kelas GeneticAlgorithm.cs 45](#_Toc423531962)

[Kode Sumber 4. 15 Fungsi Normalisasi Fitur 46](#_Toc423531963)

[Kode Sumber 4. 16 Potongan Kode Sumber Fungsi Mendapatkan Perbandingan Untuk Kalibrasi Tangan 47](#_Toc423531964)

[Kode Sumber 4. 17 Fungsi Kalibrasi Pada Fitur Pengguna 48](#_Toc423531965)

[Kode Sumber 4. 18Pembagian *Neural Network* 49](#_Toc423531966)

[Kode Sumber 7. 1 Kelas Backpropagation.cs 76](#_Toc423531994)

[Kode Sumber 7. 2 Kelas Chromosom.cs 77](#_Toc423531995)

[Kode Sumber 7. 3 Kelas ClassificationClass.cs 79](#_Toc423531996)

[Kode Sumber 7. 4 Kelas DataSet.cs 80](#_Toc423531997)

[Kode Sumber 7. 5 Kelas FeedForward.cs 83](#_Toc423531998)

[Kode Sumber 7. 6 Kelas GeneticAlgorithm 86](#_Toc423531999)

[Kode Sumber 7. 7 Kelas NeuralNetwork.cs 88](#_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.

*[Halaman ini sengaja dikosongkan]*

# 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.

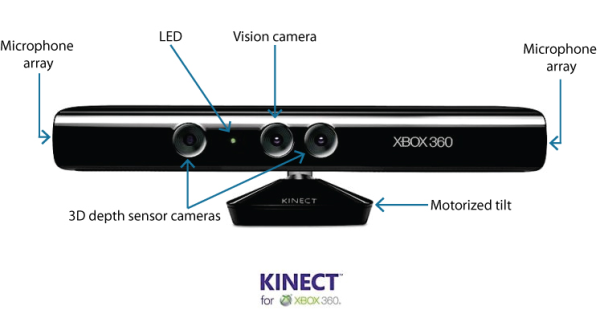


Gambar 2.1 Contoh Bahasa Isyarat yang ada didalam SIBI

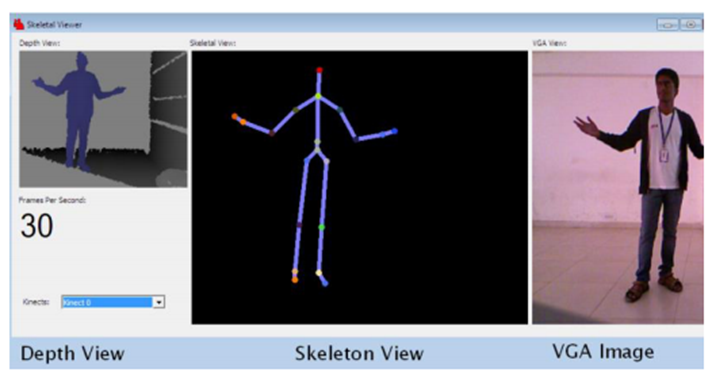
## 2.3 Kinect

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].



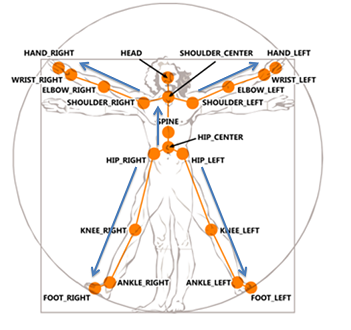
Gambar 2.2 Microsoft Kinect



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

## 2.4 Kinect SDK

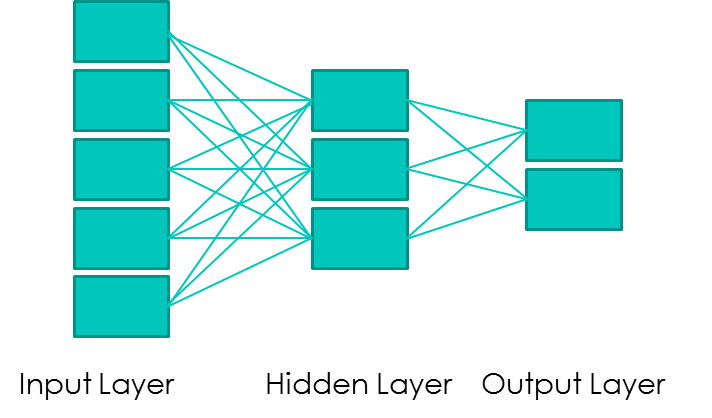
Kinect SDK adalah library yang dibuat oleh Microsoft untuk pengembangan aplikasi perangkat lunak yang menggunakan Kinect sebagai alat input utama. Kinect SDK dapat diimplementasikan dengan bahasa pemrograman C#, C++, dan JavaScript. Library ini memiliki beberapa fitur diantaranya skeleton tracking, Thumb 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].

0

# 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. 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 satu kebutuhan non-fungsional yang mendukung dan menambah performa untuk jalannya aplikasi.

Fungsu itu adalah posisi peletakan Kinect. Aplikasi ini menggunakan perangkat keras Kinect dengan posisi menghadap pengguna. Posisi peletakan Kinect disesuaikan dengan pengguna sehingga menghasilkan fitur – fitur yang sempurna

### 3.1.3 Identifikasi Pengguna

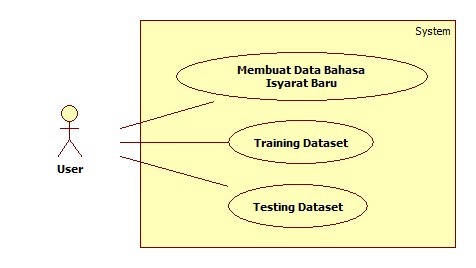
Dalam aplikasi tugas akhir ini, pengguna yang akan terlibat hanya terdapat satu orang saja, yakni orang yang akan melakukan pengenalan bahasa isyarat.

## 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 iSyarat 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 huruf, 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 data set |
| 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

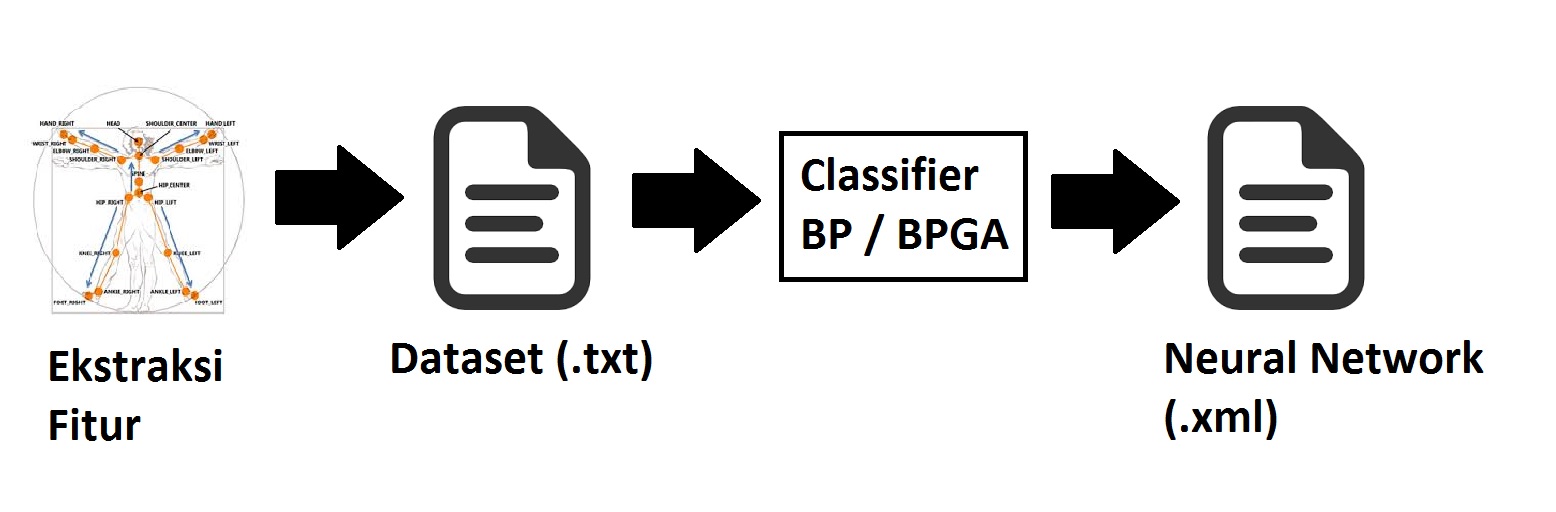
Spesifikasi kasus penggunaan Testing Dataset dapat dilihat pada Tabel 3.5.

Tabel 3.5 Spesifikasi Kasus Penggunaan Testing Dataset

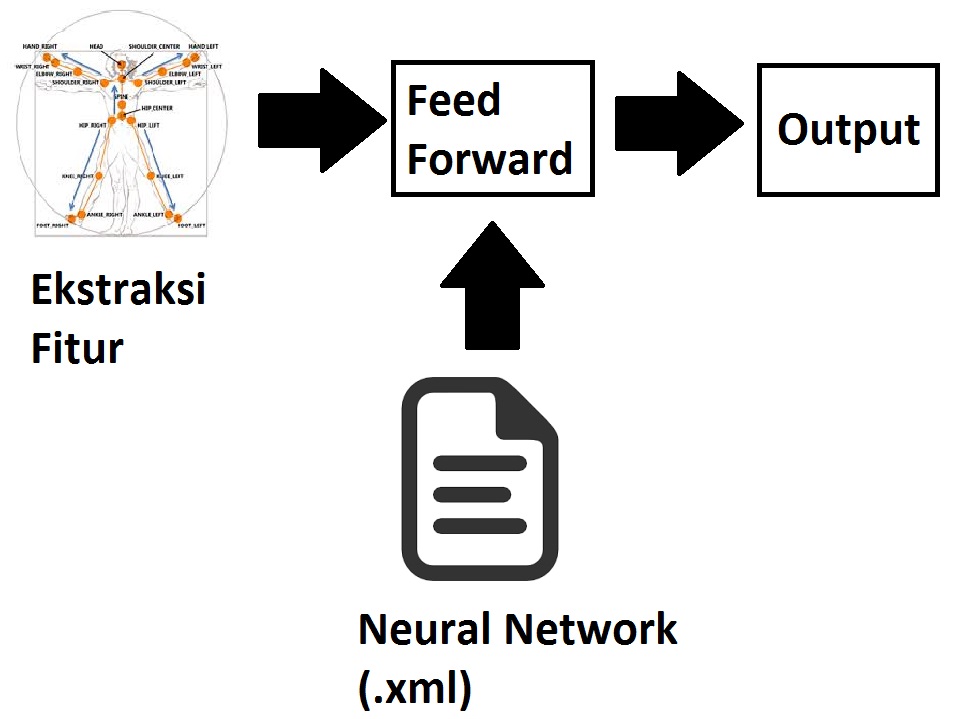
|  |  |
| --- | --- |
| Nama Kasus Penggunaan | Testing Dataset |
| Nomor | UC-03 |
| Deskripsi | Kasus penggunaan aktor untuk melakukan percobaan dataset |
| Aktor | Pengguna |
| Kondisi Awal | pengguna sudah menjalankan aplikasi dan sudah ada dataset yang dibuat aplikasi pada tahap sebelumnya |
| Alur 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*)

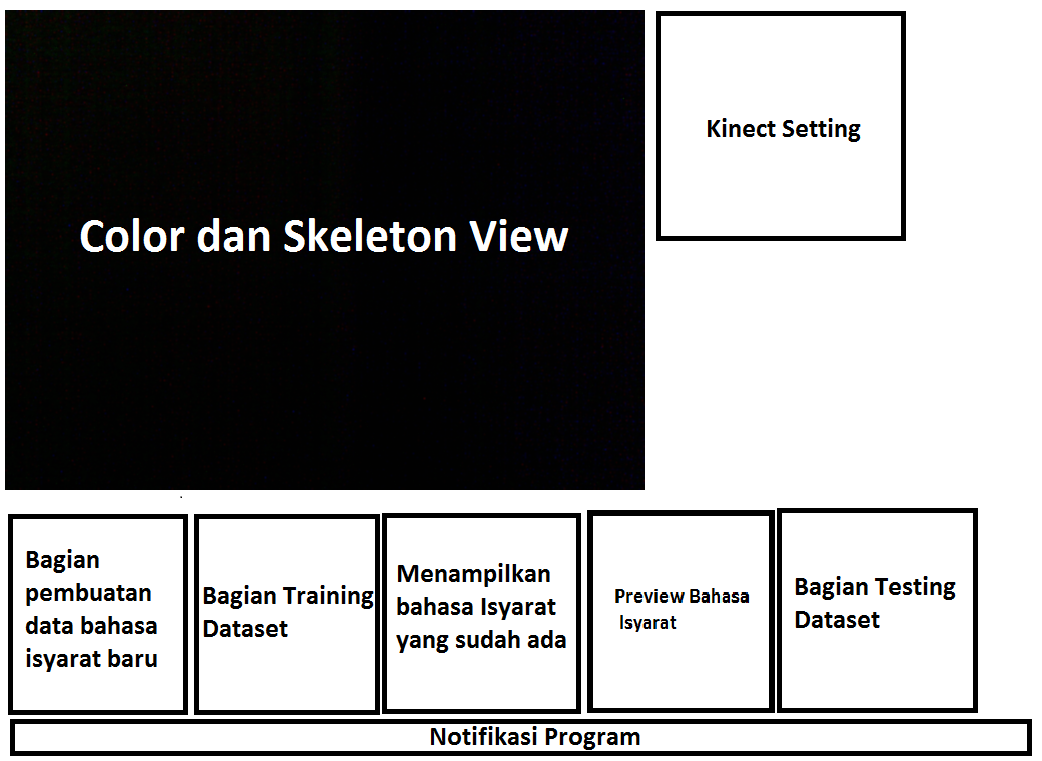


Gambar 3. 3 Arsitektur Sistem (*Testing*)

### 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.

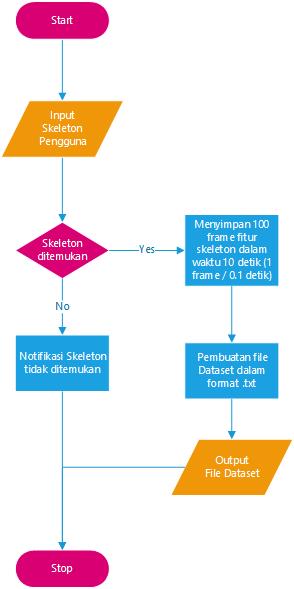
Untuk melakukan ekstraksi skeleton pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 9 fitur skeleton. 9 fitur tersebut merupakan *skeleton joints* yang diketahui oleh Kinect. 9 *skeleton joints* tersebut adalah sebagai berikut:

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) <= 0.1254f | 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 **Error! Reference source not found.**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
|  | | (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

Pada subbab ini akan dibahas mengenai hasil implementasi yang dilakukan berdasarkan rancangan antarmuka. Nantinya akan digunakan Leap Motion untuk menggerakkan kursor dalam perpindahan menu dan proses pengenalan bahasa isyarat. Dalam implementasi Antarmuka, digunakan Microsoft Visual Studio 2015 dengan bahasa pemrograman C#.

### 4.2.1 Implementasi Antarmuka Halaman Utama

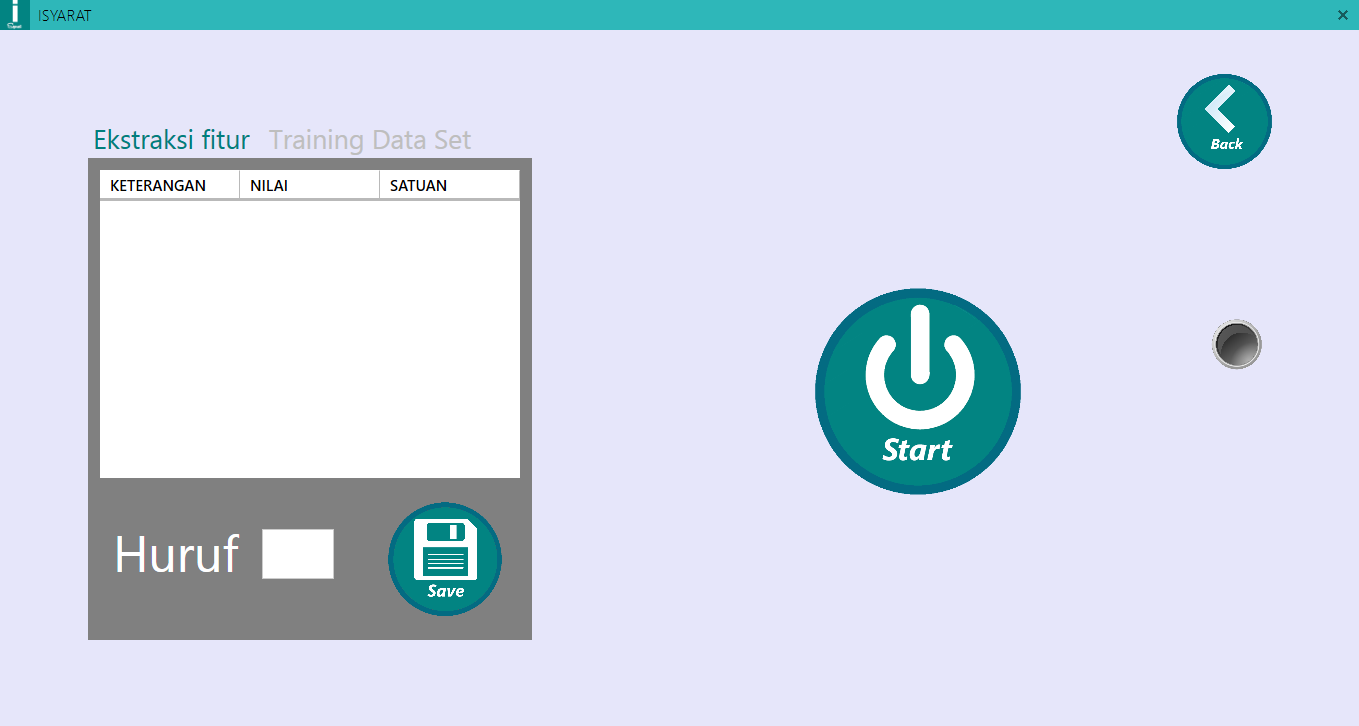
Pada halaman utama aplikasi iSyarat ini, terdapat 2 tombol untuk menuju ke menu *training* dan menu *testing*. Cara untuk memilih tombol hanya dengan menggerakkan kursor (menggunakan Leap Motion) ke tombol yang ingin dipilih. Tampilan antarmuka halaman utama seperti pada Gambar 4. 1.



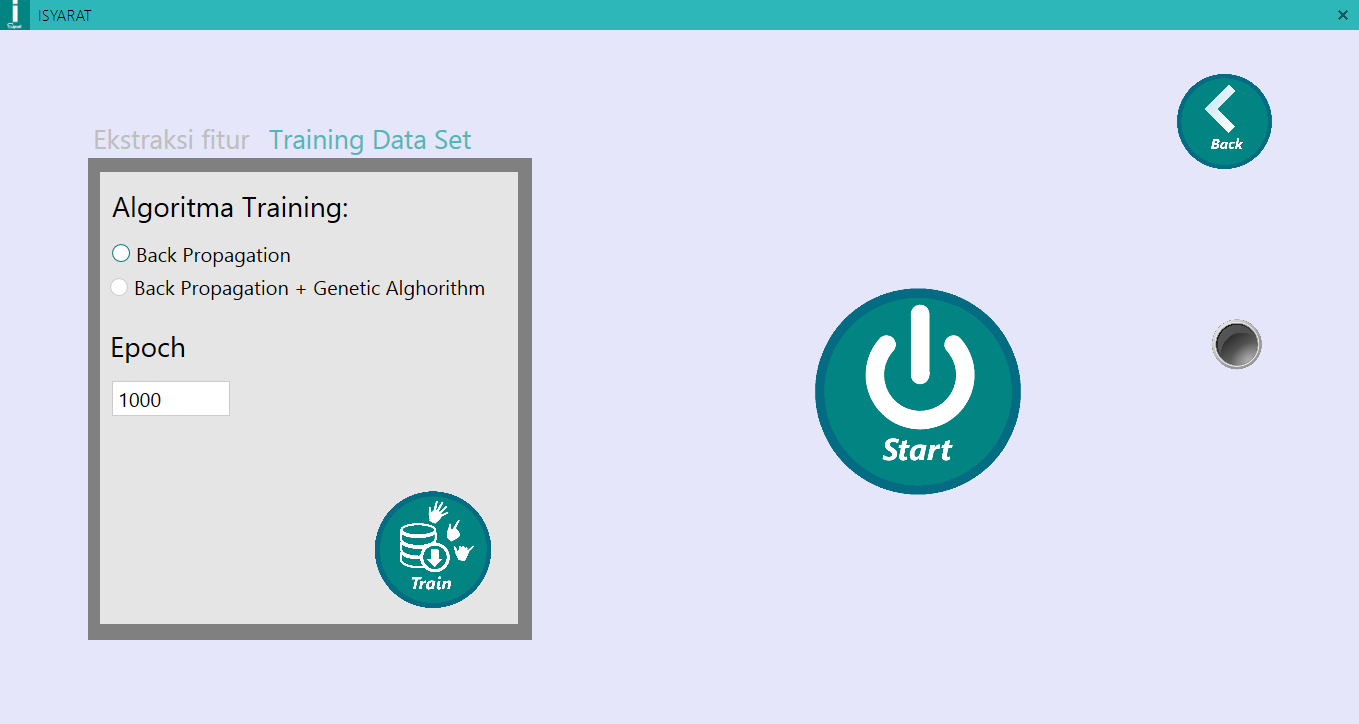
Gambar 4. 1 Antarmuka Halaman Utama

### 4.2.2 Implementasi Antarmuka Halaman *Training*

Pada halaman *training* ini terdapat tabel ekstraksi fitur, tombol menyimpan *dataset*, tombol untuk mulai merekam fitur tangan, dan tombol untuk kembali ke halaman sebelumnya yaitu halaman utama seperti pada Gambar 4. 2. Ketika tab “*training* *dataset*” dipilih, maka akan muncul pilihan untuk melakukan *training* data seperti pilihan algoritma, dan jumlah iterasi yang dilakukan dalam menjalankan algoritma tersebut seperti pada Gambar 4. 3. Jika memilih tombol *start*, maka akan tampil antarmuka seperti pada Gambar 4. 4.



Gambar 4. 2 Antarmuka *Training* (Tabel Ekstraksi Fitur)



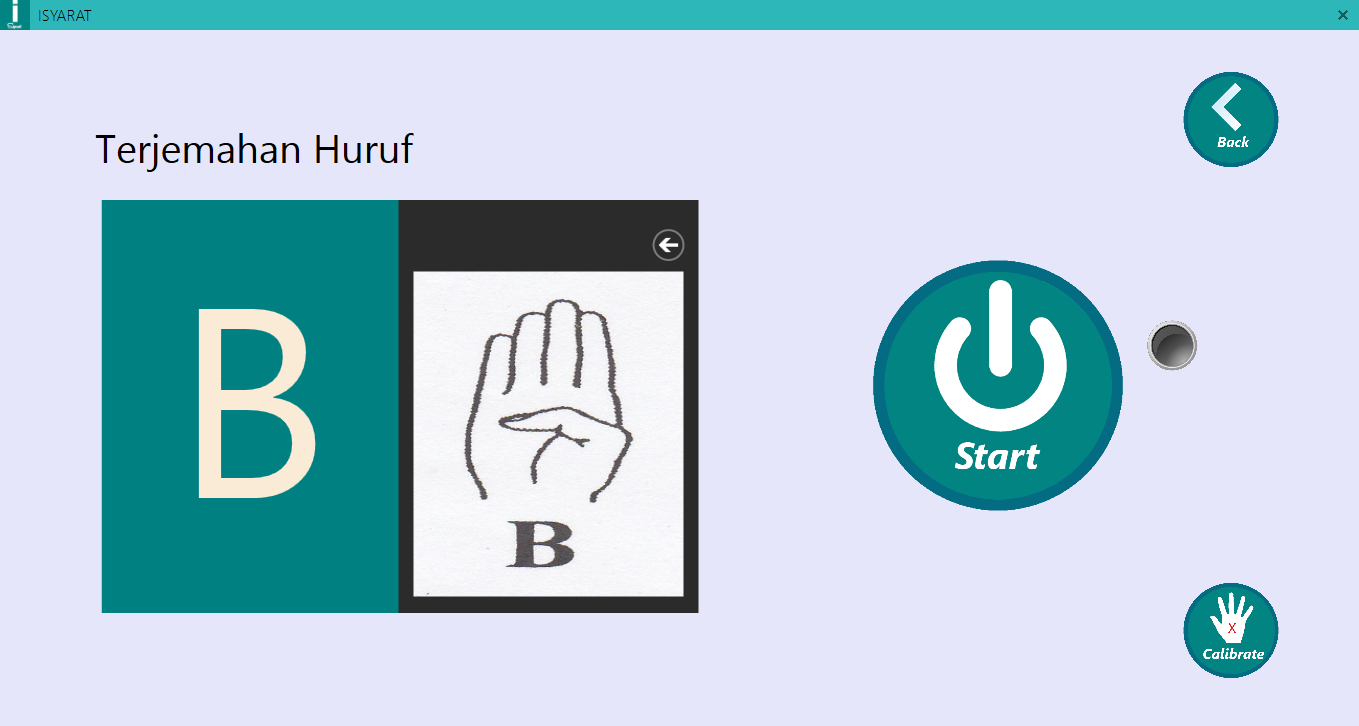
Gambar 4. 3 Antarmuka *Training* (Pilihan *Training* Data)



Gambar 4. 4 Antarmuka *Training* (Proses ekstraksi fitur)

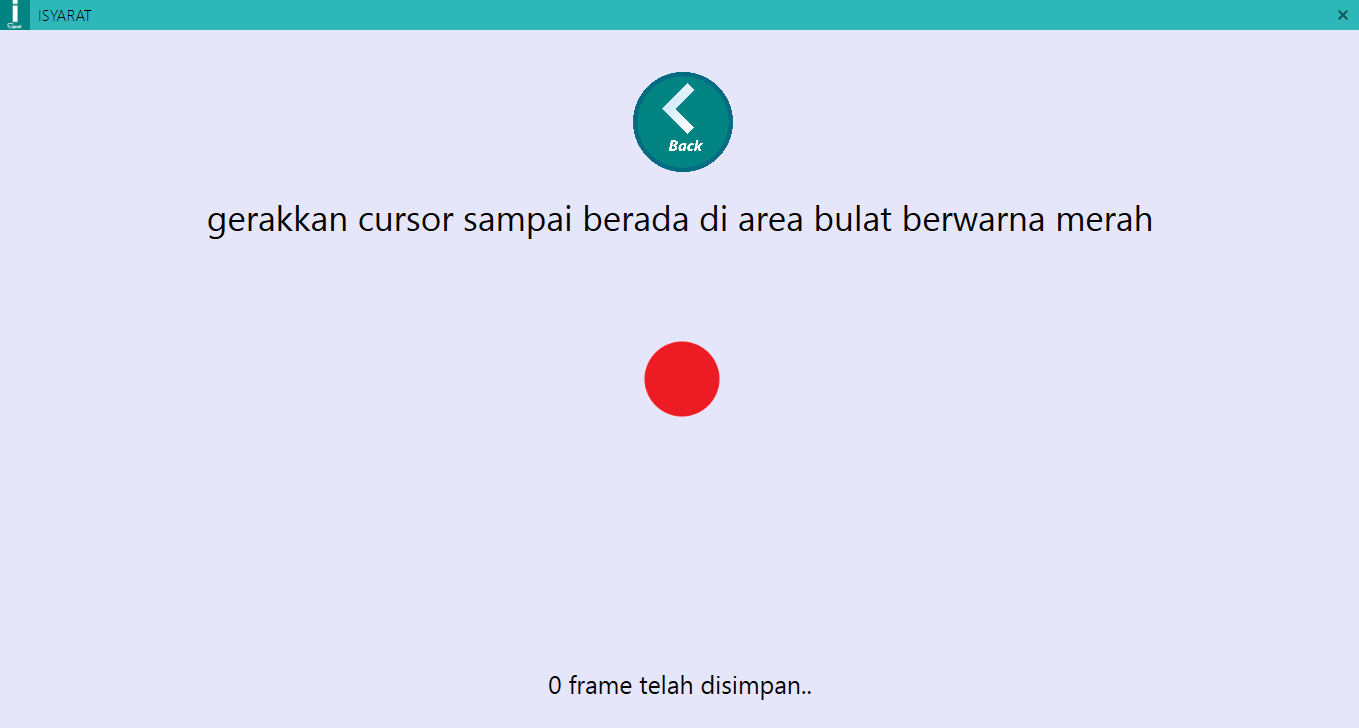
### 4.2.3 Implementasi Antarmuka Halaman *Testing*

Pada halaman *testing*, terdapat terjemahan dari hasil *testing* beserta gambarnya. Selain itu, juga terdapat tombol untuk memulai *testing*, tombol kembali ke menu sebelumnya, dan juga tombol untuk melakukan kalibrasi. Tampilan antarmuka halaman *testing* dapat dilihat pada Gambar 4. 5.



Gambar 4. 5 Antarmuka *Testing*

Lalu, jika memilih tombol *start*, maka akan tampil halaman pengambilan fitur jari seperti pada Gambar 4. 6. Sedangkan, jika memilih tombol kalibrasi, maka akan tampil halaman untuk kalibrasi seperti pada Gambar 4. 7.



Gambar 4. 6 Antarmuka *Testing* (Proses Ekstraksi Fitur)



Gambar 4. 7 Antarmuka Kalibrasi

## 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 2012 dengan bahasa pemrograman C#.

### 4.3.1 Implementasi Pendeteksian Lokasi Objek

Untuk menjalankan aplikasi ini tentunya membutuhkan perangkat Leap Motion Controller, sehingga dibutuhkan suatu proses untuk mendeteksi lokasi objek tangan pengguna dan merepresentasikanya kedalam aplikasi ini sebagai kursor.

Kode sumber proses pendeteksian posisi telapak tangan kanan pengguna, dapat dilakukan seperti pada Kode Sumber 4. 1. Sementara kode sumber untuk menggerakkan kursor dapat dilakukan seperti pada Kode Sumber 4. 2

|  |
| --- |
| void newFrameHandler(Leap.Frame frame, float deltaTime)  {  if (frame.Hands.Count != 0){  leapMouse.Run(frame.Hands.Rightmost.PalmPosition);  }  } |

Kode Sumber 4. 1 Kode Sumber Pendeteksian Telapak Tangan

|  |
| --- |
| public LeapMouse(Image img, MainWindow w){  this.img = img;  this.win = w;  }  public void Run(Vector v){  double velocity = 3.2;  img.Margin = new Thickness(v.x \* velocity + win.Width / 2, v.z \* velocity + win.Height / 2, 0, 0);  } |

Kode Sumber 4. 2. Kode Sumber Kelas *LeapMouse.cs*

### 4.3.2. Implementasi Proses Ekstraksi Fitur

Untuk melakukan ekstraksi tangan pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 34 fitur tangan. Pada Kode Sumber 4. 3 dijelaskan mengenai proses ekstraksi fitur pada setiap *frame* dari Leap Motion. Setiap fitur akan diambil rata-ratanya dari 10 *frame* tangan yang ditangkap oleh Leap Motion Controller.

|  |
| --- |
| handSphereRadius += hand.SphereRadius;  handPitch += Math.Abs(hand.Direction.Pitch \* (float)(180.0 / Math.PI));  handYaw += Math.Abs(hand.Direction.Yaw \* (float)(180.0 / Math.PI));  handRoll += Math.Abs(hand.Direction.Roll \* (float)(180.0 / Math.PI));  jari1PalmX += (hand.Fingers[0].TipPosition.x - hand.PalmPosition.x);  jari2PalmX += (hand.Fingers[1].TipPosition.x - hand.PalmPosition.x);  jari3PalmX += (hand.Fingers[2].TipPosition.x - hand.PalmPosition.x);  jari4PalmX += (hand.Fingers[3].TipPosition.x - hand.PalmPosition.x);  jari5PalmX += (hand.Fingers[4].TipPosition.x - hand.PalmPosition.x);  jari1PalmY += (hand.Fingers[0].TipPosition.y - hand.PalmPosition.y);  jari2PalmY += (hand.Fingers[1].TipPosition.y - hand.PalmPosition.y);  jari3PalmY += (hand.Fingers[2].TipPosition.y - hand.PalmPosition.y);  jari4PalmY += (hand.Fingers[3].TipPosition.y - hand.PalmPosition.y);  jari5PalmY += (hand.Fingers[4].TipPosition.y - hand.PalmPosition.y);  jari1PalmZ += (hand.Fingers[0].TipPosition.z - hand.PalmPosition.z);  jari2PalmZ += (hand.Fingers[1].TipPosition.z - hand.PalmPosition.z);  jari3PalmZ += (hand.Fingers[2].TipPosition.z - hand.PalmPosition.z);  jari4PalmZ += (hand.Fingers[3].TipPosition.z - hand.PalmPosition.z);  jari5PalmZ += (hand.Fingers[4].TipPosition.z - hand.PalmPosition.z);  jari1ke2X += hand.Fingers[1].TipPosition.x - hand.Fingers[0].TipPosition.x;  jari1ke3X += hand.Fingers[2].TipPosition.x - hand.Fingers[0].TipPosition.x;  jari1ke4X += hand.Fingers[3].TipPosition.x - hand.Fingers[0].TipPosition.x;  jari1ke5X += hand.Fingers[4].TipPosition.x - hand.Fingers[0].TipPosition.x;  jari1ke2Y += hand.Fingers[1].TipPosition.y - hand.Fingers[0].TipPosition.y;  jari1ke3Y += hand.Fingers[2].TipPosition.y - hand.Fingers[0].TipPosition.y;  jari1ke4Y += hand.Fingers[3].TipPosition.y - hand.Fingers[0].TipPosition.y;  jari1ke5Y += hand.Fingers[4].TipPosition.y - hand.Fingers[0].TipPosition.y;  jari1ke2Z += hand.Fingers[1].TipPosition.z - hand.Fingers[0].TipPosition.z;  jari1ke3Z += hand.Fingers[2].TipPosition.z - hand.Fingers[0].TipPosition.z;  jari1ke4Z += hand.Fingers[3].TipPosition.z - hand.Fingers[0].TipPosition.z;  jari1ke5Z += hand.Fingers[4].TipPosition.z - hand.Fingers[0].TipPosition.z;  jari2ke3X += hand.Fingers[2].TipPosition.x - hand.Fingers[1].TipPosition.x;  jari2ke3Y += hand.Fingers[2].TipPosition.y - hand.Fingers[1].TipPosition.y;  jari2ke3Z += hand.Fingers[2].TipPosition.z - hand.Fingers[1].TipPosition.z; |

Kode Sumber 4. 3 Kode Sumber Ekstraksi Fitur Tangan

### 4.3.3 Implementasi Algoritma BPGA

Dalam implementasi algoritma *backpropagation*, setiap bobot pada *layer* *input* maupun bobot pada *layer* *hidden* akan dideklarasikan terlebih dahulu secara acak. Penulis memberikan *range* angka acak seperti pada Kode Sumber 4. 4.

|  |
| --- |
| 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 4. 4 Fungsi Inisialisasi Bobot Pada Kelas *NeuralNetwork.cs*

Setelah itu, dilakukan proses *feedforward* seperti pada Kode Sumber 4. 5. Tahap algoritma *feedforward* ini adalah:

|  |
| --- |
| 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));  } |

Pada *layer* *input*, *input* *layer* ke-i adalah hasil ekstraksi fitur ke-i.

Pada *layer* *hidden*, *input* dari *hidden* *layer* ke-j dijumlahkan dengan *output* dari *input* *layer* ke-i (*linear*), lalu lakukan fungsi pengaktifan *sigmoid*.

Pada *layer* *output*, *input* dari *output* *layer* ke-j dijumlahkan dengan perkalian antara *input* *hidden* *layer* ke-i dengan bobot yang acak (*dot product*) [7].

Kode Sumber 4. 5 Kelas Feedforward.cs

Lalu, akan dilakukan proses *backpropagation*. Tahap algoritma *backpropagation* ini adalah:

Menghitung *error* di *output layer* menggunakan rumus pada Persamaan 2.1. Implementasi dapat dilihat pada Kode Sumber 4. 6.

*Update* bobot di *hidden* *layer* menggunakan rumus pada Persamaan 2.2. Implementasi dapat dilihat pada Kode Sumber 4. 8.

Menghitung *error* di *hidden* *layer* menggunakan rumus pada Persamaan 2.3. Implementasi dapat dilihat pada Kode Sumber 4. 9.

*Update* bobot di *input* *layer* menggunakan rumus pada Persamaan 2.2 Implementasi dapat dilihat pada Kode Sumber 4. 10 [7].

Sedangkan algoritma genetik mempunyai tahap-tahap seperti berikut:

1. Deklarasi kromosom seperti pada Kode Sumber 4. 7.

2. Sebanyak iterasi, lakukan *backpropagation* seperti Kode Sumber 4. 13.

3. Lalu, lakukan seleksi kromosom seperti pada Kode Sumber 4. 11.

4. Lalu, lakukan penyilangan kromosom seperti pada Kode Sumber 4. 12.

5. Lalu, lakukan mutasi kromosom seperti pada Kode Sumber 4. 14 [5].

|  |
| --- |
| 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;  } |

Kode Sumber 4. 6 Fungsi GetErrorOutput Pada Kelas Backpropagation.cs

|  |
| --- |
| private void ChromosomInit()  {  chromosoms = new List<Chromosom>(individualCount);  for (int i = 0; i < individualCount; i++){  chromosoms.Add(new Chromosom(GetRandomBinary()));  }  } |

Kode Sumber 4. 7 Fungsi ChromosomInit pada kelas GeneticAlgorithm.cs

|  |
| --- |
| 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);  } |

Kode Sumber 4. 8 Fungsi UpdateWeight di *Hidden* *Layer* Pada Kelas Backpropagation.cs

|  |
| --- |
| 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 4. 9 Fungsi GetErrorHidden Pada Kelas Backpropagation.cs

|  |
| --- |
| 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);  }  }  } |

Kode Sumber 4. 10 Fungsi UpdateWeightInput Pada Kelas Backpropagation.cs

|  |
| --- |
| private void DoSelection(){  chromosoms.Sort((val1, val2) => val1.FitnessValue.CompareTo(val2.FitnessValue));  } |

Kode Sumber 4. 11 Fungsi DoSelection pada kelas GeneticAlgorithm.cs

|  |
| --- |
| 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;  }  } |

Kode Sumber 4. 12 Fungsi DoCrossOver pada kelas GeneticAlgorithm.cs

|  |
| --- |
| 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 4. 13 Fungsi DoBackpropagation pada kelas GeneticAlgorithm.cs

|  |
| --- |
| 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;  }  }  } |

Kode Sumber 4. 14 Fungsi DoMutation pada kelas GeneticAlgorithm.cs

### 4.3.4 Implementasi Proses Normalisasi Fitur

Proses normalisasi fitur mengimplementasikan *pseudocode* pada **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. 15.

### 4.3.5 Implementasi Proses Kalibrasi Tangan

Proses kalibrasi tangan mengimplementasikan *pseudocode* pada **Error! Reference source not found.**. Kalibrasi tangan dilakukan karena ukuran tangan setiap pengguna berbeda-beda. Implementasi mendapatkan perbandingan tangan penulis dan pengguna dapat dilihat pada Kode Sumber 4. 16. Sedangkan implementasi proses kalibrasi tangan pengguna dapat dilihat pada Kode Sumber 4. 17.

|  |
| --- |
| public void Normalized(List<float> kalibrasiList)  {  for (int i = 0; i < dataSetList.Count; i++)  {  for (int j = 0; j < dataSetList[i].AttributeCount; j++)  {  if (j == 0 || j == 1 || j == 2 || j == 3){  dataSetList[i][j] /= 180.0f;  }  else if (j >= 4 && j <= 18){  dataSetList[i][j] /= 100.0f;  }  else if(j>=19 && j<=22){  dataSetList[i][j] /= 200.0f;  }  else if (j >= 23 && j <= 26){  dataSetList[i][j] /= 100.0f;  }  else if (j >= 27 && j <= 30){  dataSetList[i][j] /= 200.0f;  }  else if(j==31){  dataSetList[i][j] /= 100.0f;  }  else if (j == 32){  dataSetList[i][j] /= 100.0f;  }  else if (j == 33){  dataSetList[i][j] /= 100.0f;  }  }  } |

Kode Sumber 4. 15 Fungsi Normalisasi Fitur

|  |
| --- |
| if (isCalibratingOnGoing){  pointable = frame.Pointables.Frontmost;  hands = frame.Hands;  hand = hands[0];  if (countCalibrasi<10){  kalibrasiLebarTangan += Math.Abs(hand.Fingers[0].TipPosition.x - hand.PalmPosition.x) + Math.Abs(hand.Fingers[4].TipPosition.x - hand.PalmPosition.x);  kalibrasiPanjangTangan += Math.Abs(hand.Fingers[2].TipPosition.z - hand.PalmPosition.z);  }  countCalibrasi++;  if (countCalibrasi == 10){  kalibrasiPanjangTangan /= 10;  kalibrasiLebarTangan /= 10;  }  } |

Kode Sumber 4. 16 Potongan Kode Sumber Fungsi Mendapatkan Perbandingan Untuk Kalibrasi Tangan

|  |
| --- |
| public void Calibrated(List<float> multiplier){  for (int i = 0; i < dataSetList.Count; i++){  for (int j = 0; j < dataSetList[i].AttributeCount; j++){  if (j == 0 || j == 1 || j == 2 || j == 3){  }  else if (j == 4 || j == 5 || j == 7 || j == 8 || j==9){  dataSetList[i][j] \*= multiplier[35];  }  else if (j == 9 || j == 10 || j == 11 || j == 12 || j == 13){  }  else if (j == 15 || j == 16 || j == 17 || j==18 || j==19){  dataSetList[i][j] \*= multiplier[34];  }  else if (j >= 19 && j <= 22){  dataSetList[i][j] \*= multiplier[35];  }  else if (j >= 23 && j <= 26){  }  else if (j >= 27 && j <= 30){  dataSetList[i][j] \*= multiplier[34];  }  else if (j == 31){  dataSetList[i][j] \*= multiplier[35];  }  else if (j == 32){  }  else if (j == 33){  dataSetList[i][j] \*= multiplier[34];  }  }  } |

Kode Sumber 4. 17 Fungsi Kalibrasi Pada Fitur Pengguna

### 4.3.6 Implementasi Proses Pembagian *Neural Network*

Proses pembagian *neural* *network* pada aplikasi ini dilakukan dengan membagi 26 huruf menjadi 3 bagian. Bagian pertama untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x sama atau lebih dari letak jari telunjuk pada sumbu x, yaitu huruf E, I, J, M, N, S, dan T. Bagian kedua untuk bahasa isyarat huruf yang letak ibu jari pada sumbu x lebih kecil dari letak jari telunjuk pada sumbu x, yaitu huruf A, C, D, G, H, O, P, Q, X, dan Y. Bagian ketiga yaitu untuk bahasa isyarat huruf yang jari telunjuk atau jari tengahnya tegak, yaitu huruf B, F, K, L, R, U, V, W, dan Z. Kode sumber untuk proses penggolongan *neural netw*ork dapat dilihat pada Kode Sumber 4. 18.

|  |
| --- |
| if ((jari2PalmZ <= -75.0f || jari3PalmZ <= -75.0f) && jari3PalmX >= -25.0f){  filename = "gol3\_v6\_15k.xml";  sum = 2;  }  else if ((jari1PalmX <= (kalibrasiList[4]\*-1)+5.0f || jari3PalmX <= -25.0f)){  filename = "gol2\_v6\_15k.xml";  sum = 1;  }  else{  filename = "gol1\_v6\_15k.xml";  sum = 0;  } |

Kode Sumber 4. 18Pembagian *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 laptop Lenovo G410S yang memiliki spesifikasi sebagai berikut.

* Prosesor Intel(R) Core i5 CPU @ 2,50GHz
* Memori (RAM) 4,00 GB
* Leap Motion

## Skenario Pengujian

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* dari tangan pengguna yang sama. Sedangkan skenario B, *neural* *network* yang digunakan pada saat *testing* berbeda dengan pada saat *training*.

1. Pengujian skenario A1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset.
2. Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
3. Pengujian skenario A3 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
4. Pengujian skenario A4 merupakan pengujian akurasi terhadap jumlah iterasi yang dilakukan pada algoritma GABP. Pengujian ini dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
5. Pengujian skenario B1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 260 dataset. Pengujian ini tanpa menggunakan kalibrasi.
6. Pengujian skenario B2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi.
7. Pengujian skenario B3 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 520 dataset. Pengujian ini menggunakan kalibrasi.
8. Pengujian skenario B4 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh tangan orang lain dengan menggunakan 520 dataset. Pengujian ini menggunakan kalibrasi.

### Pengujian Skenario A1 dan Analisis

Pada pengujian skenario A1 ini penulis akan melakukan uji coba terhadap akurasi jika testing dilakukan dengan algoritma genetik terlebih dahulu. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan 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* |
| Kalibrasi | Tanpa kalibrasi |
| Jumlah *Dataset* | 260 *dataset* |
| Penguji | Penulis |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 86,9%. |

Dapat dilihat jika menggunakan algoritma genetik terlebih dahulu untuk menyeleksi fitur tangan yang dipakai, akurasi bertambah sekitar 10%. Pada proses uji coba ini, fitur yang digunakan algoritma genetik pada *neural network* tangan bagian 1 (huruf E, I, J, M, N, S, dan T) adalah fitur 2, 5, 6, 9, 10, 11, 12, 16, 18, 19, 20, 25, 28, 31, 32, dan 34. Sedangkan pada bagian 2 (huruf A, C, D, G, H, O, P, Q, X, dan Y) adalah fitur 1,3,4, 6, 8, 12, 14, 15, 16, 18, 21, 22, 23, 28, 29, 30, dan 31. Sedangkan pada bagian 3 (huruf B, F, K, L, R, U, V, W, dan Z) adalah fitur 2, 3, 4, 5, 6, 7, 9, 12, 13, 14, 15, 16, 19, 21, 25, 28, 31, dan 34. Tabel perbandingan akurasi dapat dilihat pada Tabel 5. 2.

Tabel 5. 2 Tabel Perbandingan Akurasi Algoritma BP dan BPGA

|  |  |  |
| --- | --- | --- |
| **Hasil Uji Coba** | **Algoritma** | |
| **Backpropagation** | **BPGA** |
| **Benar** | 100 | 113 |
| **Salah** | 30 | 17 |
| **Total Pengujian** | 130 | 130 |
| **Akurasi** | 76,90% | 86,90% |

Menurut analisis yang dilakukan oleh penulis, jika *neural* *network* yang digunakan menggunakan algoritma genetik terlebih dahulu, dengan jumlah iterasi yang sama, proses pembuatan *neural* *network* menggunakan algoritma genetik lebih cepat. Tabel perbandingan kecepatan pembuatan *neural* *network* dapat dilihat pada Tabel 5. 3.

Tabel 5. 3 Tabel Perbandingan *Runtime* Algoritma (detik)

|  |  |  |
| --- | --- | --- |
| ***Neural Network*** | **Algoritma** | |
| **Backpropagation** | **BPGA** |
| **Golongan 1** | 406.56 | 290.4 |
| **Golongan 2** | 539.8708 | 385.622 |
| **Golongan 3** | 589.3062 | 420.933 |

### Pengujian Skenario A2 dan Analisis

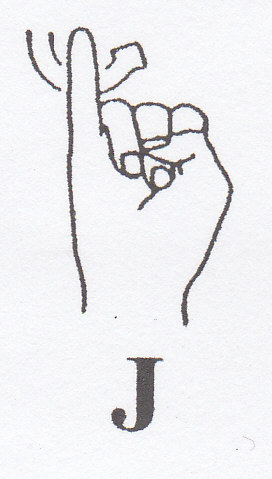
Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma BP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Skenario uji coba dapat dilihat pada Tabel 5. 4. Sedangkan hasil uji coba dapat dilihat pada Tabel 5. 5. Pada skenario pengujian A2, huruf J terdeteksi sebagai huruf A sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking (jari ke-5) sebagai ibu jari (jari ke-1) seperti pada Gambar 5. 1. Selain itu, huruf M terdeteksi sebagai huruf N sebanyak 3 kali. Hal ini terjadi dikarenakan bentuk bahasa isyarat huruf M dan N yang serupa seperti pada Gambar 5. 2. Selain itu, dengan menggunakan kalibrasi, akurasi bertambah sekitar 4%.

Tabel 5. 4 Skenario Pengujian A2

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi A2 |
| Kode | SP-A2 |
| Algoritma | *Backpropagation* |
| Kalibrasi | Kalibrasi |
| Jumlah *Dataset* | 260 *dataset* |
| Penguji | Penulis |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 80,80%. |

Tabel 5. 5 Terjemah Huruf Isyarat Skenario Pengujian A2





Gambar 5. 1 Perbandingan Huruf A dan J



Gambar 5. 2 Perbandingan Huruf M dan N

### Pengujian Skenario A3 dan Analisis

Pengujian skenario A3 merupakan pengujian akurasi menggunakan algoritma GABP terhadap bahasa isyarat huruf yang dilakukan oleh penulis dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Pada skenario pengujian A3, akurasi terjemah bahasa isyarat naik 13% dari yang semula 80,80% menjadi 93,80%. Skenario uji coba dapat dilihat pada Tabel 5. 6. Sedangkan hasil uji coba dapat dilihat pada Tabel 5. 7.

Tabel 5. 6 Skenario Pengujian A3

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi *Testing* A3 |
| Kode | SP-A3 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Kalibrasi | Kalibrasi |
| Jumlah *Dataset* | 260 *dataset* |
| Penguji | Penulis |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 93,80%. |

### Pengujian Skenario A4 dan Analisis

Pengujian skenario A4 merupakan pengujian akurasi menggunakan algoritma GABP terhadap iterasi yang dilakukan dengan menggunakan 260 dataset. Pengujian ini menggunakan kalibrasi. Tabel perbandingan akurasi dapat dilihat pada Tabel 5. 8. Pada pengujian skenario A4 ini, didapatkan hasil iterasi yang paling optimal adalah ketika iterasi *backpropagation* yang dilakukan saat sudah mendapatkan kromosom yang baik dengan algoritma genetika adalah pada jumlah iterasi 15000.

Tabel 5. 7 Terjemah Huruf Isyarat Pengujian Skenario A3



Tabel 5. 8 Perbandingan Akurasi Terhadap Jumlah Iterasi

|  |  |
| --- | --- |
| Jumlah Iterasi | Akurasi |
| 10000 | 83,84 % |
| 15000 | 93,80 % |
| 20000 | 87,60 % |

### Pengujian Skenario B1 dan Analisis

Pada pengujian skenario B1 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing* yang dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 91.5 mm dan lebar telapak tangan 51.6 mm) menggunakan algoritma *backpropagation-genetic algorithm*. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Skenario dapat dilihat pada Tabel 5. 9. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 10.

Tabel 5. 9 Skenario Pengujian B1

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi B1 |
| Kode | SP-B1 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Kalibrasi | Tanpa kalibrasi |
| Jumlah *Dataset* | 260 *dataset* |
| Penguji | Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 66,9%. |

Dari hasil pengujian ini, akurasi hanya mencapai 66,9%. Menurut analisis yang dilakukan oleh penulis, penyebab dari kecilnya akurasi tersebut dikarenakan tangan pengguna yang lebar dan panjang telapak tangannya lebih kecil dari tangan penulis dan tanpa melakukan kalibrasi terlebih dahulu. Huruf E terdeteksi sebagai huruf N sebanyak 3 kali juga terdeteksi sebagai huruf T sebanyak 2 kali. Huruf B terdeteksi sebagai huruf W sebanyak 3 kali. Selain bentuk tangan yang tidak sama, huruf yang dilakukan uji coba juga serupa seperti pada Gambar 5. 3.



Gambar 5. 3 Perbandingan Huruf Skenario Pengujian B1

Tabel 5. 10 Terjemah Huruf Isyarat Pengujian Skenario B1



### Pengujian Skenario B2 dan Analisis

Pada pengujian skenario B2 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing* yang dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 91.5 mm dan lebar telapak tangan 51.6 mm) menggunakan algoritma *backpropagation-genetic algorithm*. Uji coba ini menggunakan *dataset* sebanyak 260 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan kalibrasi terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 12. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 11.

Tabel 5. 11 Tabel Terjemah Skenario Pengujian B2



Tabel 5. 12 Skenario Pengujian B2

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi B2 |
| Kode | SP-B2 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Kalibrasi | Kalibrasi |
| Jumlah *Dataset* | 260 *dataset* |
| Penguji | Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 70,80%. |

Pada pengujian ini, dapat dilihat bahwa pada *neural network* golongan 1 (huruf E, I, J, M, N, S) sering terjadi salah terjemah. Menurut analisa penulis, hal ini dikarenakan fitur ke 21, 22, dan 23 (jarak jari ke-1 dengan jari ke 3, 4, dan 5 pada sumbu X) diseleksi. Namun, akurasi bertambah sekitar 3,9% setelah dilakukan kalibrasi.

### Pengujian Skenario B3 dan Evaluasi

Pada pengujian skenario B3 ini, penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing*. *Testing* ini dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 110,56 mm dan lebar telapak tangan 74.3 mm). Uji coba ini menggunakan *dataset* sebanyak 520 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan mengalibrasi tangan pengguna terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 14. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 13

Tabel 5. 14 Skenario Pengujian B3

Tabel 5. 13 Tabel Terjemahan Huruf Skenario B3

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi B3 |
| Kode | SP-B3 |
| Algoritma | *Backpropagation* |
| Kalibrasi | Kalibrasi |
| Jumlah *Dataset* | 520 *dataset* |
| Penguji | Pengguna dengan panjang jari tengah 91.5 mm dan lebar telapak tangan 51.6 mm |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 68,5 %. |

Gambar 5. 4 Perbandingan Huruf Skenario Pengujian B3



Pada skenario pengujian B3, huruf I terdeteksi sebagai huruf J sebanyak 5 kali, huruf J terdeteksi sebagai huruf I sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking pengguna sebagai ibu jari pada saat melakukan isyarat huruf I dan J seperti pada Gambar 5. 5. Selain itu, huruf D terdeteksi sebagai huruf O sebanyak 4 kali, huruf I terdeteksi sebagai huruf N sebanyak 2 kali dan huruf T sebanyak 2 kali, huruf K terdeteksi sebagai huruf V sebanyak 4 kali, Huruf S terdeteksi sebagai huruf N sebanyak 2 kali dan huruf T sebanyak 1 kali, dan huruf X terdeteksi sebagai huruf O sebanyak 5 kali. Menurut analisis yang dilakukan oleh penulis, hal ini terjadi karena huruf yang dilakukan *testing* serupa seperti pada Gambar 5. 4.

### Pengujian Skenario B4 dan Evaluasi

Pada pengujian skenario B4 ini penulis akan melakukan uji coba terhadap akurasi yang didapatkan pada saat melakukan *testing*. *Testing* ini dilakukan oleh orang lain (panjang jari tengah dari telapak tangan adalah 110,56 mm dan lebar telapak tangan 74.3 mm). Uji coba ini menggunakan *dataset* sebanyak 520 buah, semua *dataset* tersebut dilatih oleh tangan penulis. Uji coba ini dilakukan dengan mengalibrasi tangan pengguna terlebih dahulu. Skenario dapat dilihat pada Tabel 5. 16. Sedangkan hasil uji coba akurasi setiap huruf dapat dilihat seperti pada Tabel 5. 15.

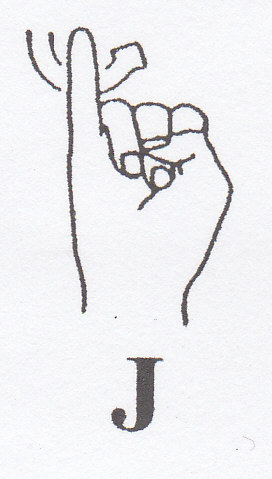
Tabel 5. 15 Terjemah Huruf Isyarat Pengujian Skenario B4



Tabel 5. 16 Skenario Pengujian B4

|  |  |
| --- | --- |
| Nama Skenario Pengujian | Pengujian Akurasi B4 |
| Kode | SP-B4 |
| Algoritma | *Backpropagation-Genetic Algorithm* |
| Kalibrasi | Kalibrasi |
| Jumlah *Dataset* | 520 *dataset* |
| Penguji | Orang lain dengan panjang jari tengah 110,56 mm dan lebar telapak tangan 74.3 mm |
| Prosedur Pengujian | Pengguna melakukan uji coba 26 huruf isyarat sebanyak 5 kali |
| Hasil yang Diperoleh | Akurasi 76,15%. |

Pada skenario pengujian B4, huruf I terdeteksi sebagai huruf J sebanyak 5 kali, huruf J terdeteksi sebagai huruf I sebanyak 5 kali. Hal ini terjadi karena Leap Motion mendeteksi jari kelingking pengguna sebagai ibu jari pada saat melakukan isyarat huruf I dan J seperti pada Gambar 5. 5. Selain itu, huruf K terdeteksi sebagai huruf V sebanyak 4 kali. Hal ini terjadi dikarenakan bentuk bahasa isyarat huruf K dan huruf V yang serupa seperti pada Gambar 5. 6. Namun, ketika melakukan uji coba bahasa isyarat V, hasil terjemah tidak terdeteksi sebagai huruf K.



Gambar 5. 5 Perbandingan Huruf I dan J



Gambar 5. 6 Perbandingan Huruf K dan V

## Evaluasi

Pada subbab ini, akan dijelaskan mengenai perbandingan dalam bentuk grafik terhadap uji coba yang telah dilakukan sebelumnya. Beberapa parameter perbandingan diantara lain:

1. Akurasi model *neural network* Terhadap algoritma yang dipakai
2. Akurasi model jika ada penambahan *dataset.*
3. Akurasi model terhadap proses kalibrasi.
4. Akurasi model terhadap iterasi yang dilakukan.
5. *Runtime* model *neural network* Terhadap algoritma yang dipakai.

### 5.3.1 Perbandingan Akurasi Model *Neural Network* Terhadap Algoritma yang Dipakai

Pada pengujian skenario A1, selisih akurasi yang didapatkan *neural network* menggunakan algoritma genetik dan tanpa menggunakan algoritma genetik adalah sekitar 10%. Grafik perbandingan akurasi dari skenario A1 dapat dilihat seperti pada Gambar 5. 7. Selain itu, jika pengujian dilakukan kalibrasi tangan terlebih dahulu seperti pada skenario A2 dan A3, selisih akurasi sekitar 13%. Grafik perbandingan akurasi dari skenario A2 dan A3 dapat dilihat seperti pada Gambar 5. 7.

### 5.3.2 Perbandingan Akurasi Model Jika Ada Penambahan *Dataset*

Pada pengujian skenario B3 dan B4, selisih akurasi yang didapatkan *neural network* menggunakan algoritma genetik dan tanpa menggunakan algoritma genetik adalah sekitar 7,65 %. Penambahan *dataset* dilakukan dengan tujuan untuk memperkaya model pada *neural* *network*. Grafik perbandingan akurasi dapat dilihat pada Gambar 5. 7.

Gambar 5. 7 Grafik Perbandingan Terhadap Algoritma

### 5.3.3 Perbandingan Akurasi Model Jika Menggunakan Kalibrasi

Pada pengujian skenario B1 dan B2, selisih akurasi yang didapatkan *neural network* tanpa menggunakan kalibrasi dan menggunakan kalibrasi terlebih dahulu adalah sekitar 3,9 %. Grafik perbandingan akurasi dapat dilihat pada Gambar 5. 8.

Gambar 5. 8 Grafik Perbandingan Terhadap Proses Kalibrasi

### 5.3.4 Perbandingan Akurasi Model Terhadap Jumlah Iterasi

Pada pengujian skenario A4, didapatkan hasil iterasi yang paling optimal adalah ketika iterasi *backpropagation* yang dilakukan saat sudah mendapatkan kromosom yang baik dengan algoritma genetika adalah pada jumlah iterasi 15000 yaitu mencapai 93,80%. Perbandingan akurasi dapat dilihat pada Gambar 5. 9.

Gambar 5. 9 Grafik Perbandingan Akurasi Terhadap Jumlah Iterasi

### 5.3.5 Perbandingan *Runtime* Aplikasi Model *Neural Network* Terhadap Algoritma yang Dipakai

Pada pengujian skenario A1, *runtime* pada saat melakukan training pada *neural network* menggunakan algoritma genetika lebih cepat daripada tanpa menggunakan algoritma genetika terlebih dahulu. Grafik perbandingan *runtime* dapat dilihat pada Gambar 5. 10.

Gambar 5. 10 Grafik Perbandingan *Runtime* Skenario A1

*[Halaman ini sengaja dikosongkan]*

# 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. Dari sisi akurasi, penerapan algoritma GABP pada aplikasi ini lebih akurat daripada algoritma BP. Algoritma GABP dapat meningkatkan akurasi *neural network* sekitar 7-13%.
2. Dari sisi *runtime*, penerapan algoritma GABP pada aplikasi ini lebih cepat daripada algoritma BP.
3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi di atas 80% jika pada saat *testing* menggunakan data *training* tangan yang sama. Namun, tangan yang ada pada data *training* berbeda pada saat *testing*, akurasi aplikasi ini menjadi sekitar 65-75%.
4. Leap Motion merupakan alat yang sangat sensitif terhadap gerakan, sehingga diperlukan rata-rata dalam pengambilan fitur di aplikasi ini.

## 6.2 Saran

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

1. Perlu adanya penelitian lebih lanjut tentang proses kalibrasi dari jari tangan setiap pengguna. Sehingga dapat meningkatkan nilai akurasi aplikasi ini. Hal ini harus dilakukan karena pada saat *testing* terdapat perbedaan bentuk, panjang, lebar masing-masing fitur setiap jari pengguna.
2. Memperbanyak model *neural network* dengan cara melakukan *training* dari berbagai jari 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