



TUGAS AKHIR - IF184802

**IMPLEMENTASI JARINGAN SARAF TIRUAN UNTUK
PENGENALAN KARAKTER PADA STUDI KASUS PER-
MASALAHAN SPOJ HARD IMAGE RECOGNITION**

CYNTHIA DEWI TEJAKUSUMA
NRP 05111540000074

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

[Halaman ini sengaja dikosongkan]



TUGAS AKHIR - IF184802

**IMPLEMENTASI JARINGAN SARAF TIRUAN UNTUK
PENGENALAN KARAKTER PADA STUDI KASUS PER-
MASALAHAN SPOJ HARD IMAGE RECOGNITION**

CYNTHIA DEWI TEJAKUSUMA
NRP 05111540000074

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

[Halaman ini sengaja dikosongkan]



UNDERGRADUATE THESES - IF184802

**IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORK
FOR CHARACTER RECOGNITION IN STUDY CASE
OF SPOJ PROBLEMS HARD IMAGE RECOGNITION**

CYNTHIA DEWI TEJAKUSUMA

NRP 05111540000074

Supervisor 1

Rully Soelaiman, S.Kom., M.Kom.

Supervisor 2

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

INFORMATICS DEPARTMENT

Faculty of Information Technology and Communication

Institut Teknologi Sepuluh Nopember

Surabaya, 2019

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

IMPLEMENTASI JARINGAN SARAF TIRUAN UNTUK PENGENALAN KARAKTER PADA STUDI KASUS PERMASALAHAN SPOJ HARD IMAGE RECOGNITION

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada

Bidang Studi Algoritma Pemrograman
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

Cynthia Dewi Tejakusuma

NRP. 05111540000074

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom.

NIP. 19700213194021001

(Pembimbing 1)

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

NIP. 198603122012122004

(Pembimbing 2)

SURABAYA
JANUARI 2019

[*Halaman ini sengaja dikosongkan*]

ABSTRAK

IMPLEMENTASI JARINGAN SARAF TIRUAN UNTUK PENGENALAN KARAKTER PADA STUDI KASUS PERMASALAHAN SPOJ HARD IMAGE RECOGNITION

Nama : Cynthia Dewi Tejakusuma
NRP : 05111540000074
Departemen : Departemen Informatika,
Fakultas Teknologi Informasi dan Komunikasi, ITS
Pembimbing I : Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II : Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Abstrak

Komputer semakin dirancang untuk memiliki kemampuan yang sama dengan manusia. Salah satu contoh kemampuan yang manusia kembangkan pada komputer adalah pengenalan karakter. Manusia dapat dengan mudah mengenali karakter pada citra yang mengalami transformasi, stretching, ataupun derau acak misalnya pada pembacaan nomor polisi kendaraan atau captcha. Manusia harus memberi algoritma pada komputer untuk proses pengenalan karakter itu sendiri agar dapat memiliki kemampuan yang diinginkan. Dengan kemampuan tersebut, komputer dapat membantu kegiatan manusia sehari-hari. Pada tugas akhir ini, penulis akan merancang penyelesaian permasalahan pengenalan karakter pada studi kasus permasalahan SPOJ Hard Image Recognition (HIR) dengan mengimplementasikan jaringan saraf tiruan menggunakan bahasa Python dan C++. Data yang digunakan adalah citra yang dapat diambil pada situs daring penilaian SPOJ Hard Image Recognition (HIR) itu sendiri. Setelah melalui beberapa trial and error, penulis mendapatkan arsitektur jaringan saraf tiruan yang optimal, ya-

itu yang dapat mengenali karakter pada data SPOJ terbanyak atau mendapatkan nilai tertinggi pada SPOJ. Aplikasi yang dibuat berhasil mendapatkan nilai tertinggi pada SPOJ Hard Image Recognition (HIR), yaitu 32.

Kata Kunci: *jaringan saraf tiruan; klasifikasi digit; pengenalan karakter*

ABSTRACT

IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORK FOR CHARACTER RECOGNITION IN STUDY CASE OF SPOJ PROBLEMS HARD IMAGE RECOGNITION

Name : Cynthia Dewi Tejakusuma
Student ID : 05111540000074
Department : Informatics Department,
Faculty of Information Technology and Communication, ITS
Supervisor I : Rully Soelaiman, S.Kom., M.Kom.
Supervisor II : Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Abstract

Computers are increasingly designed to have the same capabilities as humans. One example of the capability that human develop on computers is character recognition. Humans can easily recognize characters in images that experience transformation, stretching, or random noise such as reading the vehicle's police number or captcha. Humans must give an algorithm to a computer for character recognition process itself in order to have the desired capability. With the character recognition ability, computers can help human daily activities. In this thesis, the author will design a solution to the problem of character recognition in a study case of SPOJ Hard Image Recognition by implementing artificial neural network using Python and C++. After going through a number of trial and error, the author got the optimal artificial neural network architecture, which can recognize the characters in the most SPOJ data or get the highest score on SPOJ. The proposed application successfully

achieved a score of 32 on SPOJ Hard Image Recognition.

Keywords: artificial neural network; character recognition; digit classification

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir dan laporan akhir dalam bentuk buku ini. Penelitian tugas akhir ini dilakukan untuk mengeksplorasi topik yang menarik perhatian penulis serta memenuhi salah satu syarat dalam mendapatkan gelar Sarjana Komputer di Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember. Penulis memiliki harapan bahwa apa yang penulis kerjakan dapat membawa manfaat bagi perkembangan ilmu pengetahuan di bidang komputer dan bagi penulis sendiri selaku peneliti.

Penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membimbing dan memberi dukungan baik secara langsung maupun tidak langsung selama proses penggerjaan tugas akhir ini maupun selama menempuh masa studi. Pihak tersebut antara lain:

1. Bapak Sungkono Tedjakusuma dan Ibu Juniatyi Susilaningtyas, selaku orang tua penulis, serta keluarga besar yang selalu memberikan doa dan semangat sehingga penulis dapat menyelesaikan tugas akhir dalam rentang waktu yang diharapkan.
2. Bapak Rully Soelaiman S.Kom., M.Kom., selaku pembimbing penulis yang memberikan dukungan baik berupa didikan dan ajaran, maupun semangat dan nasihat selama menempuh masa studi dan penggerjaan tugas akhir. Berkat bimbingan dan ketersediaannya untuk berdiskusi, penulis dapat menyelesaikan tugas akhir dalam rentang waktu yang diharapkan.
3. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc., selaku pembimbing penulis yang memberikan dukungan berupa ilmu, arahan, dan semangat selama penggerjaan tugas akhir.

4. Seluruh dosen dan karyawan Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember yang telah memberi ilmu dan waktunya untuk mempersiapkan penulis agar siap untuk masuk ke dalam dunia kerja.
5. Teman-teman, kakak-kakak, dan adik-adik mahasiswa Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember yang senantiasa membantu, menemani, memberi semangat dan kenangan selama kurang lebih 3,5 tahun masa studi.
6. Teman-teman *administrator* Laboratorium Pemrograman 2 Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember yang selalu menemani dan membantu penulis ketika ada kesusahan selama masa studi.
7. Achmad Ibnu Malik Al Chasni, Hendry Wiranto, dan Pradipta Baskara yang sudah menemani penulis selama menempuh masa studi dan membantu penulis dalam pengerjaan tugas akhir.

Penulis mohon maaf jika masih ada kekurangan pada tugas akhir ini. Penulis juga berharap tugas akhir ini dapat memberikan kontribusi dan manfaat bagi pembaca.

Surabaya, Januari 2019

Cynthia Dewi Tejakusuma

DAFTAR ISI

ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xv
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxiii
DAFTAR KODE SUMBER	xxv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	3
1.5 Manfaat	3
1.6 Metodologi	3
1.7 Sistematika Penulisan	5
BAB II DASAR TEORI	7
2.1 Deskripsi Permasalahan	7
2.1.1 Parameter Masukan	8
2.1.2 Batasan Permasalahan	8
2.1.3 Keluaran Permasalahan	8
2.2 Deskripsi Umum	9
2.2.1 Online Judge	9
2.2.2 Pengenalan Pola	9
2.2.3 Citra Biner	10
2.2.4 Praproses Citra	10
2.2.5 Segmentasi	11

2.2.6	Depth-First Search (DFS)	12
2.2.7	Penapisan Ukuran	12
2.2.8	Normalisasi Ukuran	12
2.2.9	Jaringan Saraf Tiruan (JST)	13
2.2.10	Fungsi Aktivasi	15
2.2.11	Fungsi Optimasi	17
2.2.12	Struct	18
BAB III	DESAIN	21
3.1	Deskripsi Umum Sistem	21
3.2	Desain Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan	26
3.2.1	Desain Praproses	27
3.2.2	Desain Persiapan Masukan Jaringan Saraf Tiruan	28
3.2.3	Desain Pemanggilan Fungsi Pembuatan Jaringan Saraf Tiruan	30
3.2.4	Desain Pelatihan Jaringan Saraf Tiruan	32
3.3	Desain Pengujian Jaringan Saraf Tiruan	36
3.3.1	Desain Fungsi Main	36
3.3.2	Desain Fungsi DFS	39
3.3.3	Desain Fungsi Cek Digit	39
3.3.4	Desain Fungsi Load Model	40
3.3.5	Desain Fungsi Fungsi Aktivasi	42
3.3.6	Desain Fungsi Prediksi	43
3.4	Desain Program Tambahan	44
3.4.1	Desain Program Konversi Berkas TXT ke PNG	44
3.4.2	Desain Program Scaling Citra	45

3.4.3	Desain Program Transformasi Komponen JST ke dalam C++	46
BAB IV IMPLEMENTASI	49
4.1	Lingkungan implementasi	49
4.1.1	Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan	49
4.1.2	Program Pengujian Jaringan Saraf Tiruan . .	49
4.2	Implementasi Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan	50
4.2.1	Penggunaan Library	50
4.2.2	Implementasi Praproses dan Persiapan Masa- sukan Jaringan Saraf Tiruan	51
4.2.3	Implementasi Pemanggilan Fungsi Pembu- atan Jaringan Saraf Tiruan	53
4.2.4	Implementasi Pelatihan Jaringan Saraf Tiruan	53
4.3	Implementasi Program Pengujian Jaringan Saraf Ti- ruan	54
4.3.1	Penggunaan Library, Variabel Global, dan Struct	54
4.3.2	Implementasi Fungsi Main	56
4.3.3	Implementasi Fungsi DFS	58
4.3.4	Implementasi Fungsi Cek Digit	59
4.3.5	Implementasi Fungsi Load Model	60
4.3.6	Implementasi Fungsi Fungsi Aktivasi	61
4.3.7	Implementasi Fungsi Prediksi	62
4.4	Implementasi Program Tambahan	63
4.4.1	Implementasi Program Konversi Berkas TXT ke PNG	63
4.4.2	Implementasi Program Scaling Gambar . .	64
4.4.3	Implementasi Program Inisiasi Array C++ .	65

BAB V UJI COBA DAN EVALUASI	69
5.1 Lingkungan Uji Coba	69
5.1.1 Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan	69
5.1.2 Program Pengujian Jaringan Saraf Tiruan .	69
5.2 Uji Coba Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan	69
5.2.1 Uji Coba Kebenaran Tahap Praproses . . .	70
5.2.2 Uji Coba Kebenaran Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan	71
5.3 Uji Coba Program Pengujian Jaringan Saraf Tiruan	75
5.3.1 Uji Coba Kebenaran Lokal	75
5.3.2 Uji Coba Kebenaran Situs Penilaian Daring SPOJ	76
BAB VI KESIMPULAN DAN SARAN	87
6.1 Kesimpulan	87
6.2 Saran	89
DAFTAR PUSTAKA	91
LAMPIRAN A: Gambar Pelatihan Jaringan Saraf Tiruan	93
LAMPIRAN B: Hasil Prediksi Uji Coba Lokal Program C++	95
BIODATA PENULIS	109

DAFTAR GAMBAR

Gambar 2.1	Contoh Citra Masukan PNG	7
Gambar 2.2	Contoh Citra Masukan TXT	8
Gambar 2.3	Contoh Hasil Segmentasi pada Program C++	11
Gambar 2.4	Struktur JST	13
Gambar 2.5	Ilustrasi Perhitungan Output pada JST	14
Gambar 2.6	Grafik Fungsi Aktivasi <i>Identity</i> [11]	15
Gambar 2.7	Grafik Fungsi Aktivasi <i>Sigmoid</i> [11]	16
Gambar 2.8	Grafik Fungsi Aktivasi <i>Tanh</i> [11]	17
Gambar 2.9	Grafik Fungsi Aktivasi <i>ReLU</i> [11]	17
Gambar 3.1	Diagram Umum Penyelesaian	22
Gambar 3.2	Proses Persiapan Dataset	23
Gambar 3.3	Proses Pelatihan Jaringan Saraf Tiruan	24
Gambar 3.4	Proses Pengujian Jaringan Saraf Tiruan	26
Gambar 3.5	Praproses pada Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST	27
Gambar 3.6	<i>Pseudocode</i> Tahap Praproses pada Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST	28
Gambar 3.7	Persiapan Masukan JST	29
Gambar 3.8	<i>Pseudocode</i> Tahap Persiapan Masukan JST	30
Gambar 3.9	Pemanggilan Fungsi Pembuatan dan Pelatihan JST	31
Gambar 3.10	<i>Pseudocode</i> Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan JST	32
Gambar 3.11	Pelatihan JST	33
Gambar 3.12	<i>Pseudocode</i> Tahap Pelatihan JST (Bg.1)	35

Gambar 3.13	<i>Pseudocode</i> Tahap Pelatihan JST (Bg.2)	36
Gambar 3.14	<i>Pseudocode</i> Fungsi Main Program Pengujian JST (Bg.1)	37
Gambar 3.15	<i>Pseudocode</i> Fungsi Main Program Pengujian JST (Bg.2)	38
Gambar 3.16	<i>Pseudocode</i> Fungsi DFS Program Pengujian JST	39
Gambar 3.17	<i>Pseudocode</i> Fungsi Cek Digit Program Pengujian JST (Bg.1)	40
Gambar 3.18	<i>Pseudocode</i> Fungsi <i>Load Model</i> Program Pengujian JST (Bg.1)	40
Gambar 3.19	<i>Pseudocode</i> Fungsi <i>Load Model</i> Program Pengujian JST (Bg.2)	41
Gambar 3.20	<i>Pseudocode</i> Fungsi <i>Load Model</i> Program Pengujian JST (Bg.3)	42
Gambar 3.21	<i>Pseudocode</i> Fungsi Fungsi Aktivasi <i>Identity</i>	42
Gambar 3.22	<i>Pseudocode</i> Fungsi Fungsi Aktivasi <i>Sigmoid</i>	42
Gambar 3.23	<i>Pseudocode</i> Fungsi Fungsi Aktivasi <i>Tanh</i>	43
Gambar 3.24	<i>Pseudocode</i> Fungsi Fungsi Aktivasi <i>ReLU</i>	43
Gambar 3.25	<i>Pseudocode</i> Fungsi Prediksi Program Pengujian JST	43
Gambar 3.26	<i>Pseudocode</i> Program Konversi berkas TXT ke PNG	44
Gambar 3.27	<i>Pseudocode</i> Program <i>Scaling</i> Citra (Bg.1)	45
Gambar 3.28	<i>Pseudocode</i> Program <i>Scaling</i> Citra (Bg.2)	46
Gambar 3.29	<i>List</i> yang Berisi Bobot dan Bias Hasil Pelatihan Jaringan Saraf Tiruan	46
Gambar 3.30	Bentuk Umum Inisiasi <i>array</i> bahasa C++	47
Gambar 3.31	<i>Pseudocode</i> Program Inisiasi <i>Array C++</i> (Bg.1)	47
Gambar 3.32	<i>Pseudocode</i> Program Inisiasi <i>Array C++</i> (Bg.2)	48

Gambar 5.1	Grafik Nilai SPOJ untuk Uji Coba <i>Learning Rate</i>	78
Gambar 5.2	Grafik Nilai SPOJ untuk Uji Coba Fungsi Aktivasi	78
Gambar 5.3	Grafik Nilai SPOJ untuk Uji Coba Jumlah <i>Epoch</i>	79
Gambar 5.4	Grafik Nilai SPOJ untuk Uji Coba Fungsi Optimasi	80
Gambar 5.5	Grafik Nilai SPOJ untuk Uji Coba Jumlah <i>Node</i>	80
Gambar 5.6	Grafik Nilai SPOJ untuk Uji Coba <i>Threshold</i>	81
Gambar 5.7	Grafik Nilai SPOJ untuk Uji Coba Kedua Jumlah <i>Epoch</i>	82
Gambar 5.8	Grafik Nilai SPOJ untuk Uji Coba Kedua <i>Learning Rate</i>	83
Gambar 5.9	Hasil Pengiriman Program Pengujian JST ke Situs Penilaian Daring SPOJ Hard Image Recognition (HIR)[1]	85
Gambar 5.10	Peringkat Nilai pada Situs Penilaian Daring SPOJ Hard Image Recognition (HIR)[1]	86
Gambar 6.1	Hasil Prediksi Citra 1 (Salah)	95
Gambar 6.2	Hasil Prediksi Citra 2 (Salah)	96
Gambar 6.3	Hasil Prediksi Citra 3 (Benar)	97
Gambar 6.4	Hasil Prediksi Citra 4 (Benar)	98
Gambar 6.5	Hasil Prediksi Citra 5 (Salah)	99
Gambar 6.6	Hasil Prediksi Citra 6 (Benar)	100
Gambar 6.7	Hasil Prediksi Citra 7 (Benar)	101
Gambar 6.8	Hasil Prediksi Citra 8 (Salah)	102
Gambar 6.9	Hasil Prediksi Citra 9 (Benar)	103
Gambar 6.10	Hasil Prediksi Citra 10 (Benar)	104
Gambar 6.11	Hasil Prediksi Citra 11 (Benar)	105

Gambar 6.12 Hasil Prediksi Citra 12 (Salah)	106
Gambar 6.13 Hasil Prediksi Citra 13 (Benar)	107

DAFTAR TABEL

Tabel 3.1	Daftar Fungsi Tahap Praproses Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST	27
Tabel 3.2	Daftar Variabel Tahap Praproses pada Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST	28
Tabel 3.3	Daftar Fungsi Tahap Persiapan Masukan JST	29
Tabel 3.4	Daftar Variabel Tahap Persiapan Masukan JST	30
Tabel 3.5	Daftar Parameter Arsitektur JST untuk Tahap Uji Coba	31
Tabel 3.6	Daftar Fungsi Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan JST	32
Tabel 3.7	Daftar Variabel Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan JST	32
Tabel 3.8	Daftar Fungsi Tahap Pelatihan JST	34
Tabel 3.9	Daftar Variabel Tahap Pelatihan JST	35
Tabel 4.1	Penjelasan <i>Library</i> Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST	50
Tabel 4.2	Penjelasan Variabel Global Program Pengujian JST	55
Tabel 4.3	Penjelasan Variabel <i>Struct</i> Program Pengujian JST	56
Tabel 5.1	Uji Coba <i>Learning Rate</i> dengan Citra Pelatihan	72
Tabel 5.2	Uji Coba Parameter Fungsi Aktivasi dengan Citra Pelatihan	73
Tabel 5.3	Uji Coba Parameter <i>Epoch</i> dengan Citra Pelatihan	73
Tabel 5.4	Uji Coba Parameter Fungsi Optimasi dengan Citra Pelatihan	74

Tabel 5.5	Uji Coba Parameter Jumlah <i>Node</i> dengan Citra Pelatihan	74
Tabel 5.6	Komponen dan Nilai Uji Coba Kebenaran Lokal Program Pengujian JST	76
Tabel 5.7	Uji Coba <i>Learning Rate</i> pada SPOJ	77
Tabel 5.8	Uji Coba Fungsi Aktivasi pada SPOJ	78
Tabel 5.9	Uji Coba Jumlah <i>Epoch</i> pada SPOJ	79
Tabel 5.10	Uji Coba Fungsi Optimasi pada SPOJ	79
Tabel 5.11	Uji Coba Jumlah <i>Node</i> pada SPOJ	80
Tabel 5.12	Uji Coba <i>Threshold</i> pada SPOJ	81
Tabel 5.13	Uji Coba Kedua Jumlah <i>Epoch</i> pada SPOJ . . .	82
Tabel 5.14	Uji Coba Kedua Jumlah <i>Epoch</i> pada SPOJ . . .	83
Tabel 5.15	Komponen Uji Coba dengan Nilai SPOJ Tertinggi	84
Tabel 6.1	Tabel Contoh Citra Pelatihan JST	93
Tabel 6.2	Tabel Jumlah Citra Pelatihan JST	94

DAFTAR KODE SUMBER

2.1	<i>Struct dgt</i>	19
4.1	<i>Header Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST</i>	50
4.2	Tahap Praproses dan Persiapan Masukan JST (Bg.1)	51
4.3	Tahap Praproses dan Persiapan Masukan JST (Bg.2)	52
4.4	Tahap Pemanggilan Fungsi Pembuatan JST	53
4.5	Tahap Pelatihan JST (Bg.1)	53
4.6	Tahap Pelatihan JST (Bg.2)	54
4.7	<i>Header Program Pengujian JST</i>	54
4.8	Variabel Global Program Pengujian JST	55
4.9	<i>Struct Program Pengujian JST</i>	55
4.10	Fungsi <i>Main</i> Program Pengujian JST (Bg.1)	56
4.11	Fungsi <i>Main</i> Program Pengujian JST (Bg.2)	57
4.12	Fungsi <i>Main</i> Program Pengujian JST (Bg.3)	58
4.13	Fungsi <i>Main</i> Program Pengujian JST	59
4.14	Fungsi Cek Digit Program Pengujian JST	59
4.15	Fungsi <i>Load Model</i> Program Pengujian JST (Bg.1)	60
4.16	Fungsi <i>Load Model</i> Program Pengujian JST (Bg.2)	61
4.17	Fungsi Aktivasi <i>Identity</i>	62
4.18	Fungsi Aktivasi <i>Sigmoid</i>	62
4.19	Fungsi Aktivasi <i>Tanh</i>	62
4.20	Fungsi Aktivasi <i>ReLU</i>	62
4.21	Fungsi Prediksi Program Pengujian JST (Bg.1) . .	62
4.22	Fungsi Prediksi Program Pengujian JST (Bg.2) . .	63
4.23	Program Konversi Berkas TXT ke PNG (Bg.1) . .	63
4.24	Program Konversi Berkas TXT ke PNG (Bg.2) . .	64

4.25 Program Scaling Gambar (Bg.1)	64
4.26 Program Scaling Gambar (Bg.2)	65
4.27 Program Inisiasi Array C++ (Bg.1)	65
4.28 Program Inisiasi Array C++ (Bg.2)	66
4.29 Program Inisiasi Array C++ (Bg.3)	66

BAB I

PENDAHULUAN

Bab ini menjelaskan konteks tugas akhir yang akan dikerjakan, termasuk latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan.

1.1. Latar Belakang

Komputer pada jaman sekarang semakin dirancang untuk memiliki kemampuan yang sama dengan manusia. Pengenalan karakter pada citra adalah salah satu kemampuan komputer yang sedang banyak dikembangkan oleh manusia dan sangat berguna dalam kehidupan manusia, diantaranya adalah untuk membaca nomor polisi kendaraan, membaca tulisan tangan manusia, dan jika ada proses yang ditambahkan, komputer dapat menterjemahkan kalimat dari suatu bahasa ke bahasa lainnya. Hal-hal tersebut dapat membantu pekerjaan manusia.

Topik Tugas Akhir ini mengacu pada permasalahan Online Judge SPOJ dengan kode HIR [1]. Permasalahan ini mengangkat topik pengenalan karakter pada citra biner. Tipe permasalahan ini adalah *challenge*, dimana penilaian akan dilakukan berdasarkan nilai. Semakin banyak program mengenali citra dengan benar, semakin tinggi nilai yang didapat. Pada permasalahan ini terdapat sekumpulan karakter 'X' dan '.' yang merepresentasikan citra biner dari 6 angka yang didapat dari dataset SPOJ. 'X' merepresentasikan *foreground* atau warna hitam dan '.' merepresentasikan *background* atau warna putih dari citra tersebut. Terdapat citra yang mengalami beberapa perlakuan atau transformasi dan terdapat citra yang memiliki derau acak. Dari sekumpulan karakter ini, program yang telah dibuat akan melakukan proses pengenalan karakter sehingga keluaran dari program berupa 6 angka sesuai dengan citra. Hasil tugas akhir ini diharapkan dapat memberi gambaran mengenai al-

goritma untuk menyelesaikan permasalahan di atas secara optimal dan diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.2. Rumusan Masalah

Permasalahan yang akan diselesaikan pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara segmentasi *foreground* pada citra?
2. Bagaimana cara menghilangkan bagian yang bukan termasuk angka pada citra?
3. Bagaimana penentuan metode pengenalan citra yang optimal dengan batasan-batasan yang ada?
4. Bagaimana cara mengevaluasi kinerja terhadap praproses dan metode pengenalan citra yang ditentukan?

1.3. Batasan Masalah

Batasan dari masalah yang akan diselesaikan adalah sebagai berikut:

1. Implementasi dilakukan menggunakan bahasa pemrograman C++ dan Python.
2. Batas maksimum *testcase* adalah 250.
3. Batas maksimum waktu eksekusi program adalah 0,1 detik.
4. Batas maksimum memori yang digunakan program saat dijalankan adalah 1535 MB.
5. Batas maksimum ukuran kode sumber yang dikirim adalah 0,15 MB.
6. Batas minimum tinggi dan lebar citra masukan adalah 10.
7. Batas maksimum tinggi dan lebar citra masukan adalah 250.
8. Dataset yang digunakan adalah dataset pada SPOJ Hard Image Recognition (HIR).

1.4. Tujuan

Tujuan tugas akhir ini adalah sebagai berikut:

1. Melakukan analisis dan mendesain algoritma segmentasi *foreground* pada citra SPOJ Hard Image Recognition (HIR).
2. Melakukan analisis dan mendesain algoritma untuk menghilangkan bagian yang bukan termasuk angka pada citra SPOJ Hard Image Recognition (HIR).
3. Menentukan metode pengenalan citra yang optimal dengan batasan-batasan yang ada.
4. Mengevaluasi kinerja terhadap praproses dan metode pengenalan citra yang ditentukan.

1.5. Manfaat

Manfaat tugas akhir ini adalah sebagai berikut:

1. Membantu memahami penggunaan algoritma yang tepat untuk segmentasi *foreground* pada citra SPOJ Hard Image Recognition (HIR).
2. Membantu memahami penggunaan algoritma yang tepat untuk menghilangkan bagian yang bukan termasuk angka pada citra SPOJ Hard Image Recognition (HIR).
3. Mengetahui metode pengenalan citra yang optimal dengan batasan-batasan yang ada.
4. Mengetahui hasil evaluasi kinerja terhadap praproses dan metode pengenalan citra yang ditentukan.

1.6. Metodologi

Metodologi pelaksanaan yang digunakan pada tugas akhir ini memiliki beberapa tahapan. Tahapan-tahapan tersebut adalah sebagai berikut:

1. Penyusunan proposal tugas akhir

Pada tahapan ini, penulis akan menyusun rencana dan langkah-langkah yang akan dilakukan dalam proses pembuatan tugas akhir.

2. Studi literatur

Pada tahapan ini, penulis mengumpulkan referensi yang diperlukan guna mendukung penggerjaan tugas akhir. Referensi yang digunakan dapat berupa hasil penelitian yang sudah pernah dilakukan, buku, artikel internet, atau sumber lain yang bisa dipertanggungjawabkan.

3. Desain

Pada tahapan ini, penulis melakukan desain rancangan algoritma yang akan digunakan untuk proses pengenalan karakter pada SPOJ Hard Image Recognition (HIR).

4. Implementasi algoritma

Pada tahapan ini, penulis mulai mengembangkan algoritma yang telah didukung oleh hasil rancangan desain pada tahapan sebelumnya. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman C++ dan Python.

5. Pengujian dan evaluasi

Pada tahapan ini, penulis melakukan pengujian dan evaluasi menggunakan dataset SPOJ Hard Image Recognition (HIR) pada sistem penilaian daring SPOJ untuk mengetahui nilai dan performa algoritma yang telah dibangun dengan waktu eksekusi program kurang dari 0,1 detik, memori yang digunakan program saat dijalankan kurang dari 1535 MB, dan ukuran kode sumber yang dikirim kurang dari 0,15 MB.

6. Penyusunan buku

Pada tahapan ini, penulis menyusun laporan yang menjelaskan teori dan metode yang benar-benar digunakan dalam menyelesaikan studi kasus SPOJ Hard Image Recognition (HIR) serta hasil dari implementasi algoritma yang telah dibuat dalam bentuk buku tugas akhir.

1.7. Sistematika Penulisan

Sistematika laporan tugas akhir yang akan digunakan adalah sebagai berikut:

1. Bab 1 : PENDAHULUAN

Bab ini menjelaskan konteks tugas akhir yang akan dikerjakan, termasuk latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan.

2. Bab 2 : DASAR TEORI

Bab ini menjelaskan dasar teori yang akan digunakan dalam proses penggerjaan tugas akhir.

3. Bab 3 : DESAIN

Bab ini menjelaskan desain algoritma yang akan dibangun berdasarkan dasar teori yang dijelaskan pada bab 2.

4. Bab 4 : IMPLEMENTASI

Bab ini menjelaskan implementasi desain algoritma yang dijelaskan pada bab 3.

5. Bab 5 : PENGUJIAN DAN EVALUASI

Bab ini menjelaskan hasil pengujian dan evaluasi algoritma yang telah diimplementasikan pada bab 4.

6. Bab 6 : PENUTUP

Bab ini berisi kesimpulan yang telah didapat dari hasil pengujian dan evaluasi yang telah dilakukan.

[*Halaman ini sengaja dikosongkan*]

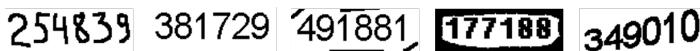
BAB II

DASAR TEORI

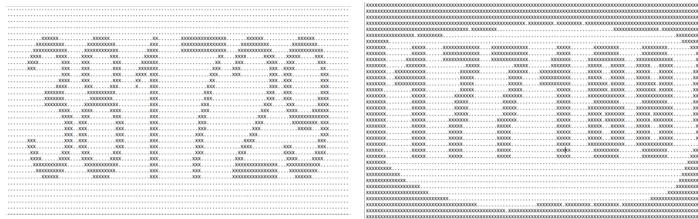
Bab ini menjelaskan dasar teori yang penulis gunakan sebagai landasan penggerjaan tugas akhir. Pertama, bab ini menjelaskan deskripsi permasalahan pada SPOJ Hard Image Recognition (HIR), dilanjutkan dengan menjelaskan beberapa teori yang akan digunakan pada tugas akhir ini.

2.1. Deskripsi Permasalahan

Terdapat kumpulan karakter yang terdiri dari karakter 'X' dan '.' yang merepresentasikan citra hitam dan putih dari 6 angka. Karakter 'X' merepresentasikan *foreground* atau warna hitam dan karakter '.' merepresentasikan *background* atau warna putih. Terdapat citra yang mengalami beberapa perlakuan atau transformasi. Terdapat pula citra yang memiliki derau acak. Program harus dapat mengenali 6 angka sesuai dengan citra yang terdiri dari sekumpulan karakter tersebut. Tipe permasalahan ini adalah *challenge*, dimana penilaian akan dilakukan berdasarkan nilai. Semakin banyak program mengenali citra dengan benar, semakin tinggi nilai yang didapat. Pada situs penilaian daring SPOJ Hard Image Recognition (HIR)[1], terdapat 10 contoh citra masukan yang berupa berkas PNG. Gambar 2.1 merupakan contoh citra masukan yang berupa PNG. Sedangkan pada situs forum diskusi kontes terbuka SPOJ Hard Image Recognition (HIR)[2], terdapat 3 contoh citra masukan tambahan berupa berkas TXT, yang berisi kumpulan karakter 'X' dan '.'. Gambar 2.2 merupakan contoh citra masukan yang berupa berkas TXT.



Gambar 2.1: Contoh Citra Masukan PNG



Gambar 2.2: Contoh Citra Masukan TXT

2.1.1. Parameter Masukan

Parameter masukan pada permasalahan SPOJ Hard Image Recognition (HIR) adalah:

1. t , bilangan bulat yang menentukan banyaknya jumlah *test case*, $t \leq 250$.
2. H , bilangan bulat yang menentukan tinggi dari citra masukan, $10 \leq H \leq 200$.
3. W , bilangan bulat yang menentukan lebar dari citra masukan, $10 \leq W \leq 200$.

2.1.2. Batasan Permasalahan

Batasan pada permasalahan SPOJ Hard Image Recognition (HIR) adalah:

1. Batas *runtime*: 0,1 detik
2. Batas penggunaan memori: 1536 MB
3. Batas ukuran kode sumber yang dikirim: 0,15 MB

2.1.3. Keluaran Permasalahan

Hasil keluaran program adalah 6 angka yang merupakan hasil prediksi pada citra masukan.

2.2. Deskripsi Umum

Subbab ini menjelaskan definisi, deskripsi dan landasan yang akan digunakan pada keseluruhan penyelesaian masalah.

2.2.1. Online Judge

Online judge adalah sistem untuk mengetes program secara *online* dalam kontes pemrograman. Sistem ini juga biasanya memiliki soal-soal pemrograman yang digunakan untuk latihan dari kontes. Sistem dapat mengkompilasi, mengeksekusi, dan mengetes atau membandingkan keluaran dari program yang diunggah dengan keluaran yang seharusnya. Sistem akan mengembalikan hasil kepada pengguna apakah kode program yang diunggah diterima atau tidak. Terdapat banyak contoh *online judge*. Salah satunya adalah *online judge* yang digunakan untuk melakukan uji coba pada tugas akhir ini, yaitu SPOJ (*Sphere Online Judge*).

2.2.2. Pengenalan Pola

Pengenalan pola atau dalam Bahasa Inggris adalah *pattern recognition* mencakup berbagai permasalahan pemrosesan informasi mulai dari pengenalan suara, klasifikasi karakter tulisan tangan, hingga deteksi kesalahan pada mesin[3]. Manusia mengatasi permasalahan tersebut dengan mudah, namun tidak untuk sebuah komputer. Manusia harus membuat program untuk sebuah komputer agar komputer dapat mengenali dan mendeteksi hal-hal tersebut.

Salah satu contoh permasalahan pengenalan pola adalah klasifikasi digit. Citra dari sebuah digit ditangkap oleh sebuah komputer diumpulkan ke sebuah komputasi dan komputer mencari algoritma yang dapat membedakan setiap digitnya. Terdapat 10 kelas yang mewakili digit 0 sampai 9. Tujuan dari klasifikasi ini adalah untuk mengembangkan suatu algoritma yang akan menetapkan citra digit yang diwakili oleh vektor x , ke salah satu dari 10 kelas yang dilambangkan dengan C_k , dimana $k = 1,2,3,4,5,6,7,8,9,10$, maka kelas C_1

menyatakan citra tersebut adalah citra digit 0, kelas C_2 menyatakan citra digit 1, dan seterusnya hingga kelas C_{10} menyatakan citra digit 9. Manusia harus mempersiapkan sejumlah besar contoh citra yang sesuai dengan digit 0 hingga digit 9 yang sudah diklasifikasikan oleh manusia. Citra-citra tersebut akan disebut sebagai suatu set data atau dapat disebut dengan sampel.

2.2.3. Citra Biner

Citra biner adalah citra yang hanya terdiri dari 2 kemungkinan nilai piksel, yaitu 0 dan 1, dimana pada *library* opencv Python 0 adalah hitam dan 1 adalah putih. Citra biner sering disebut dengan citra *black and white*. Citra masukan program merupakan citra biner karena hanya terdiri dari 2 kemungkinan karakter yaitu 'X' dan '.' yang akan direpresentasikan dengan nilai 1 dan 0. Gambar 2.2 adalah contoh citra masukan program C++ pada situs penilaian daring SPOJ Hard Image Recognition (HIR).

2.2.4. Praproses Citra

Praproses citra adalah tahap mempersiapkan citra sebelum melakukan pelatihan dan pengujian. Berkas TXT yang berisi kumpulan karakter 'X' dan '.' pada [2] akan diubah menjadi berkas PNG dengan menggunakan program bantuan. Citra PNG yang memiliki 6 angka akan disegmentasi per angka secara manual yaitu dengan memotong satu per satu angka yang ada. Lalu citra-citra per angka tersebut akan *scaling* dengan menambahkan border untuk mendapatkan variasi citra. *Scaling* juga dilakukan dengan menggunakan program bantuan. Lalu pengubahan warna pada citra per digit dilakukan, warna putih menjadi hitam dan hitam menjadi putih. Setelah itu, *resize* dilakukan pada citra-citra tersebut untuk menjadi citra masukan pelatihan jaringan saraf tiruan pada program Python. Sedangkan persiapan yang akan dilakukan untuk citra masukan pengujian pada program C++ adalah segmentasi, *filtering* atau penapisan

ukuran, dan *resize* atau normalisasi ukuran.

2.2.5. Segmentasi

Segmentasi adalah tahap untuk membagi wilayah masukan menjadi wilayah yang homogen berdasarkan kriteria keserupaan tertentu antara nilai suatu piksel dengan nilai piksel tetangganya. Segmentasi dilakukan untuk mendapatkan wilayah atau sekumpulan karakter yang diduga adalah angka yang akan digunakan dalam proses pengenalan.

```
E:\Cynde - 0511154000074\hard-image-recognition\cpp\tes.exe
Jumlah elemen: 10

Elemen ke-1:
Upleft: 0, 0
Upright: 12, 0
Bottomleft: 0, 8
Bottomright: 12, 8

Elemen ke-2:
Upleft: 6, 5
Upright: 22, 5
Bottomleft: 6, 28
Bottomright: 22, 28

Elemen ke-3:
Upleft: 24, 5
Upright: 39, 5
Bottomleft: 24, 28
Bottomright: 39, 28

Elemen ke-4:
Upleft: 36, 37
Upright: 86, 37
Bottomleft: 36, 39
Bottomright: 86, 39

Elemen ke-5:
Upleft: 44, 5
Upright: 51, 5
Bottomleft: 44, 28
Bottomright: 51, 28

Elemen ke-6:
Upleft: 46, 0
Upright: 82, 0
Bottomleft: 46, 2
Bottomright: 82, 2

Elemen ke-7:
Upleft: 68, 5
Upright: 75, 5
Bottomleft: 60, 28
Bottomright: 75, 28

Elemen ke-8:
Upleft: 78, 5
Upright: 93, 5
Bottomleft: 78, 28
Bottomright: 93, 28

Elemen ke-9:
Upleft: 98, 5
Upright: 105, 5
Bottomleft: 98, 28
Bottomright: 105, 28

Elemen ke-10:
Upleft: 105, 30
Upright: 118, 30
Bottomleft: 105, 39
Bottomright: 118, 39
```

Gambar 2.3: Contoh Hasil Segmentasi pada Program C++

Segmentasi akan dilakukan secara manual untuk citra masukan pelatihan jaringan saraf tiruan pada program pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan dengan bahasa Python. Sedangkan algoritma yang akan digunakan untuk citra masukan pengujian pada program pengujian jaringan saraf tiruan

dengan bahasa C++ adalah *Depth-First Search*. Dengan algoritma ini, program akan mendapatkan koordinat pojok kanan atas, kanan bawah, kiri atas, dan kiri bawah angka dengan koordinat (0,0) ada pada posisi kiri atas. Citra 2.3 adalah contoh hasil segmentasi untuk citra masukan pelatihan jaringan saraf tiruan pada program pengujian jaringan saraf tiruan dengan bahasa C++.

2.2.6. Depth-First Search (DFS)

DFS adalah salah satu algoritma rekursif untuk pencarian jalur yang memanfaatkan *backtracking*. Algoritma ini melakukan pencarian secara mendalam pada semua node dengan terus melakukannya pencarian ke bawah selama memungkinkan. Jika tidak memungkinkan, algoritma ini akan beralih menggunakan *backtracking*[4]. Pada tugas akhir ini, DFS digunakan untuk *pixel traversing*, yaitu mencari piksel-piksel yang berhubungan dengan melakukan pengecekan apakah piksel 8 arah tetangga (kanan, kanan-bawah, bawah, kiri-bawah, kiri, kiri-atas, atas, kanan-atas) memiliki nilai yang sama, yaitu 1.

2.2.7. Penapisan Ukuran

Penapisan ukuran atau *size filtering* adalah tahap untuk menghilangkan derau pada citra biner yang berupa objek-objek hasil segmentasi yang lebar dan tingginya kurang dari *threshold* bawah yang ditentukan atau lebih dari *threshold* atas yang ditentukan[5]. Penapisan ukuran digunakan pada program C++.

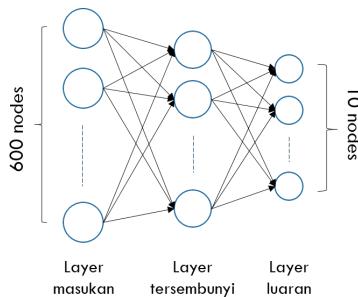
2.2.8. Normalisasi Ukuran

Normalisasi ukuran atau *resize* adalah tahap untuk mengubah ukuran citra sesuai dengan ukuran yang diinginkan. Pada program Python, normalisasi ukuran menggunakan bantuan *library* yang ada yaitu dengan menambahkan *border* pada atas dan kanan angka. Se-

dangkan normalisasi ukuran yang dilakukan pada program C++ adalah dengan menambahkan nilai 0 pada atas dan kanan angka.

2.2.9. Jaringan Saraf Tiruan (JST)

Jaringan saraf tiruan atau dalam Bahasa Inggris disebut dengan *artificial neural network* adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan sistem saraf manusia[6]. JST merupakan salah satu algoritma untuk melakukan pengklasifikasian pada komputer. JST memiliki 3 lapisan, yaitu lapisan masukan, lapisan tersembunyi, dan lapisan luaran. JST bukan merupakan sebuah algoritma, namun sebuah *framework* untuk algoritma-algoritma *machine learning*. Jaringan saraf tiruan memiliki parameter utama yang dapat diubah nilainya, yaitu jumlah layer tersembunyi, fungsi aktivasi, fungsi optimasi, *learning rate*, dan *epoch*.



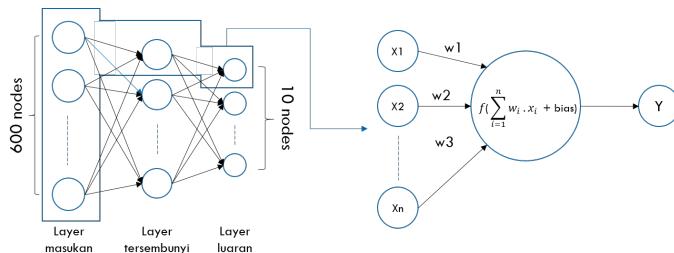
Gambar 2.4: Struktur JST

Di bidang citra, lapisan masukan merupakan fitur-fitur yang dimiliki oleh sebuah citra, yang dapat berupa nilai piksel-piksel citra ataupun fitur lainnya seperti luas, keliling, dan sebagainya. Lapisan tersembunyi adalah lapisan yang terletak antara lapisan masukan dan lapisan luaran yang menerima nilai yang sudah diproses dengan bobot pada masukan dan menghasilkan luaran yang teraktivasi. Lapisan luaran adalah lapisan terakhir yang ada pada jaringan

saraf tiruan yang menghasilkan luaran dari program. Perhitungan luaran dilakukan dengan persamaan (2.1)

$$Y = f\left(\sum_{i=1}^i w_i \cdot x_i\right) + bias \quad (2.1)$$

Dimana Y adalah luaran, $f(.)$ adalah fungsi aktivasi, w_i adalah bobot node ke- i , dan x_i adalah nilai node ke- i .



Gambar 2.5: Ilustrasi Perhitungan Output pada JST

Jaringan saraf tiruan melakukan pelatihan dengan cara:

1. Menghitung output
2. Membandingkan output yang diterima dengan target yang dinginkan
3. Menyesuaikan bobot dan mengulangi proses

Terdapat beberapa parameter arsitektur untuk membuat suatu JST yang nilainya akan diubah-ubah pada tahap uji coba untuk mendapatkan nilai tertinggi pada situs penilaian daring SPOJ. Parameter yang nilainya akan diubah-ubah pada tugas akhir ini adalah:

1. Layer Tersembunyi (Layer yang terletak diantara layer masukan dan layer luaran)
2. Fungsi Aktivasi
3. Fungsi Optimasi

4. *Learning Rate* (Sebuah konstanta positif yang memoderasi derajat perubahan bobot setiap langkah[7])

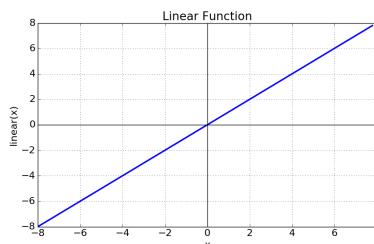
2.2.10. Fungsi Aktivasi

Fungsi aktivasi merubah nilai nonlinear menjadi linear[8]. Sehingga hasil prediksi bukan 1 dan 0, namun terdapat angka diantarnya. Pada sebuah jaringan saraf tiruan, fungsi aktivasi dari sebuah *node* menentukan nilai luaran dari *node* itu sendiri berdasarkan masukan. Nilai luaran yang didapat akan dijadikan masukan untuk *node* selanjutnya[9]. Terdapat 4 nilai fungsi aktivasi yang disediakan oleh *library Scikit-learn*, yaitu *identity*, *sigmoid* atau *logistic*, *tanh*, dan *relu*[10].

2.2.10.1. Identity

Fungsi aktivasi *identity* memiliki nama lain fungsi aktivasi linear, dimana nilai aktivasi proporsional dengan nilai masukannya. Persamaan (2.2) adalah persamaan untuk fungsi aktivasi ini.

$$f(x) = x \quad (2.2)$$

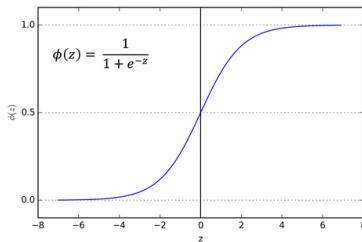


Gambar 2.6: Grafik Fungsi Aktivasi *Identity*[11]

2.2.10.2. Sigmoid

Fungsi aktivasi *sigmoid* memiliki nama lain fungsi aktivasi *logistic*. Persamaan (2.3) adalah persamaan untuk fungsi aktivasi ini.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$



Gambar 2.7: Grafik Fungsi Aktivasi *Sigmoid*[11]

2.2.10.3. Tanh

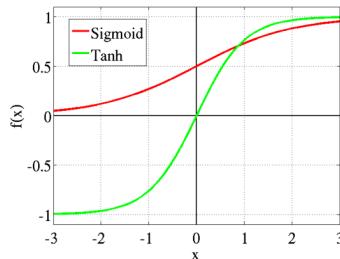
Fungsi aktivasi *tanh* mirip dengan *sigmoid*, namun memiliki gradien yang lebih curam. Persamaan (2.4) adalah persamaan untuk fungsi aktivasi ini.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2.4)$$

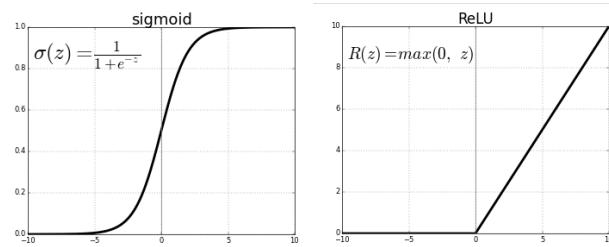
2.2.10.4. ReLu

Fungsi aktivasi *relu* menghasilkan luaran 0 jika nilai masukan adalah angka negatif dan nilai masukan itu sendiri jika nilai masukan adalah angka nol atau positif. Persamaan (2.5) adalah persamaan untuk fungsi aktivasi ini.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2.5)$$



Gambar 2.8: Grafik Fungsi Aktivasi *Tanh*[11]



Gambar 2.9: Grafik Fungsi Aktivasi *ReLU*[11]

2.2.11. Fungsi Optimasi

Fungsi optimasi adalah fungsi untuk mengoptimalkan bobot jaringan saraf tiruan dengan meminimalisasi *error* dari hasil prediksi terhadap output yang diinginkan. Terdapat 3 nilai yang disediakan oleh *library Scikit-learn*, yaitu *lbfgs*, *sgd*, dan *adam*.

2.2.11.1. Limited-memory BFGS (LBFGS)

LBFGS adalah salah satu fungsi optimasi pada JST yang menggunakan salah satu metode *quasi-Newton* yaitu *Broyden–Fletcher–Goldfarb–Shanno* (BFGS)[12].

2.2.11.2. Stochastic Gradient Descent (SGD)

SGD adalah salah satu fungsi optimasi pada JST. Operasi yang dilakukan SGD hanya mengurangi bobot awal dengan sebagian nilai dari nilai gradien yang sudah kita dapat. Nilai sebagian disini diwakili oleh parameter bernama learning rate, seperti yang terlihat pada persamaan 2.6 dan 2.7[13].

$$w_{j+1} = w_j - \alpha \frac{\delta}{\delta w_j} J(W, b) \quad (2.6)$$

$$b_{j+1} = b_j - \alpha \frac{\delta}{\delta b_j} J(W, b) \quad (2.7)$$

2.2.11.3. Adaptive Moment Estimation (Adam)

Adam adalah salah satu fungsi optimasi pada JST yang menggunakan *adaptive learning rate* untuk tiap bobot dan biasnya. Berbeda dengan SGD yang menggunakan *learning rate* yang sama setiap saat, *learning rate* pada fungsi optimasi Adam beradaptasi saat pelatihan. Rumus gradien yang digunakan dapat dilihat pada persamaan 2.8. Lalu gradien akan digunakan untuk memperbaharui bobot dan bias dengan menggunakan rumus pada persamaan 2.9 dan 2.10[14].

$$m_j = \beta_1 \cdot m_{j-1} + (1 - \beta_1) \cdot g_j \quad (2.8)$$

$$s_j = \beta_2 \cdot s_{j-1} + (1 - \beta_2) \cdot (g_j)^2 \quad (2.9)$$

$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j} + \epsilon} m_j \quad (2.10)$$

2.2.12. Struct

Struct merupakan suatu struktur data yang menggabungkan sekumpulan data yang memiliki tipe data yang berbeda-beda. Dalam penerapan permasalahan SPOJ Hard Image Recognition (HIR), *struct* digunakan untuk menyimpan informasi koordinat piksel pojok kiri atas, pojok kanan atas, pojok kiri bawah, pojok kanan bawah

dan id sebuah elemen yang tersegmentasi pada program C++. Kode sumber 2.1 adalah struktur *struct* yang digunakan pada implementasi.

```
1 struct dgt{  
2     int id, xmin, xmax, ymin, ymax;  
3 };
```

Kode Sumber 2.1: *Struct dgt*

[*Halaman ini sengaja dikosongkan*]

BAB III

DESAIN

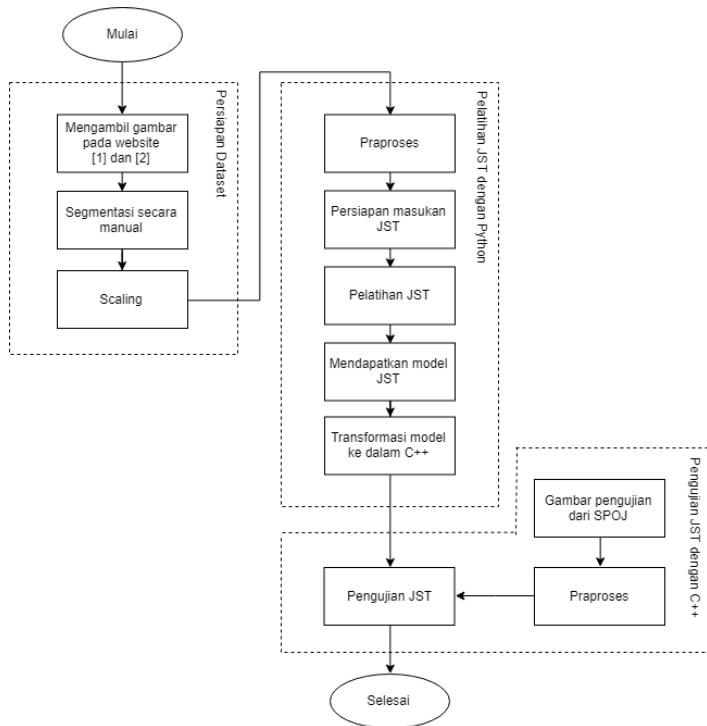
Bab ini menjelaskan desain algoritma yang akan digunakan untuk menyelesaikan permasalahan.

3.1. Deskripsi Umum Sistem

Permasalahan SPOJ Hard Image Recognition (HIR) dapat diselesaikan dengan metode heuristik. Secara umum, penyelesaian permasalahan ini menggunakan jaringan saraf tiruan atau *artificial neural network*. Pada [15], JST digunakan untuk mengenali angka dengan dataset MNIST[16]. Penyelesaian akan dilakukan dengan menggunakan 2 program, yaitu dengan bahasa pemrograman Python untuk pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan dan C++ untuk pengujian jaringan saraf tiruan dengan citra-citra masukan dari SPOJ. Gambar 3.1 merupakan diagram penyelesaian secara umum permasalahan SPOJ Hard Image Recognition (HIR). Selain penelitian sebelumnya[15] yang menggunakan JST untuk mengenali dataset MNIST[16], penggunaan JST pada tugas akhir ini juga didasari oleh karakteristik JST itu sendiri, yaitu:

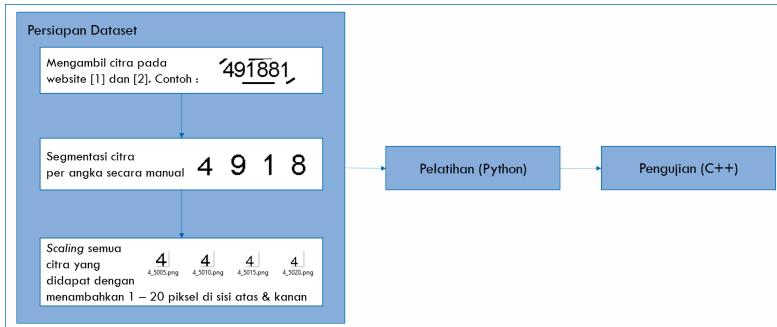
1. *Generalization Capability*: Jaringan dapat mengklasifikasi data suatu kelas yang tidak pernah dilatihkan sebelumnya.
2. *Parametric Models*: Ukuran dari jaringan tidak berubah-ubah seperti SVM karena jumlah *node*, baik pada layer masukan, layer tersembunyi, maupun layer luaran ditentukan di awal.

Langkah awal yang akan dilakukan adalah mempersiapkan citra pelatihan yang dapat diambil dari [1] dan [2] yang berjumlah 13 citra. Pada [1], terdapat 10 citra dalam bentuk PNG, sedangkan pada [2], terdapat 3 citra yang berupa berkas TXT yang berisi kumpulan karakter 'X' dan '.'. Maka dari itu, perlu dilakukan konversi menjadi PNG dengan menggunakan program tambahan pada



Gambar 3.1: Diagram Umum Penyelesaian

bagian 3.4.1. Satu citra terdiri dari 6 angka sehingga perlu dilakukan pemotongan per angka secara manual. citra-citra yang didapat di-*scaling* dengan cara menambahkan 1 sampai 20 piksel pada sisi atas dan kanan untuk mendapatkan variasi citra per angka menggunakan program tambahan pada bagian 3.4.2. Hal ini dilakukan karena angka dari contoh citra pada [1] dan [2] memiliki ukuran yang berbeda-beda sehingga langkah untuk mendapatkan variasi citra pada dataset pelatihan harus dilakukan. Proses persiapan dataset dapat dilihat pada gambar 3.2. Jumlah citra pada dataset yang akan digunakan pada tahap pelatihan jaringan saraf tiruan adalah 1.596 citra.

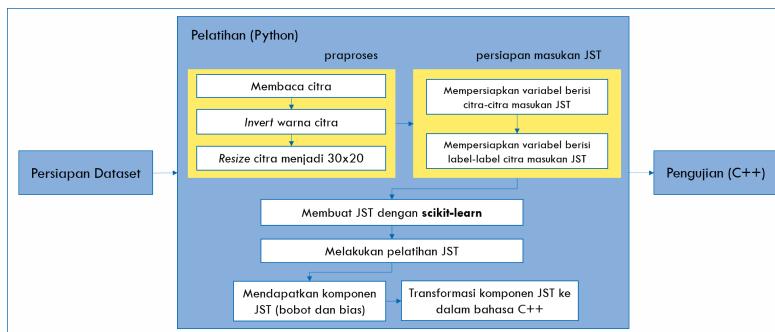


Gambar 3.2: Proses Persiapan Dataset

Pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan dilakukan dengan menggunakan bahasa pemrograman Python dengan menggunakan *library Scikit-learn* [10]. Proses pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan dapat dilihat pada gambar 3.3. Program yang akan dibuat memiliki langkah:

1. Membaca citra dalam bentuk PNG.
2. *Invert* warna citra. Hal ini dilakukan karena pada *library opencv*, piksel berwarna putih bernilai 1 dan hitam bernilai 0.
3. *Resize* citra. Tinggi angka terbesar pada citra yang ada pada [1] dan [2] adalah 30 dan lebar adalah 20. Maka dari itu dilakukan *resize* dengan ukuran 30 untuk tinggi dan 20 untuk lebar.
4. Mempersiapkan variabel berisi citra-citra masukan jaringan saraf tiruan.
5. Mempersiapkan variabel berisi label citra-citra masukan jaringan saraf tiruan. Label citra merupakan angka 0 sampai 9 sesuai dengan citra itu sendiri yang dapat diambil dari nama berkas citra.

6. Membuat jaringan saraf tiruan. Lapisan masukan pada jaringan memiliki jumlah node yang sama dengan jumlah piksel pada citra per angka dan berisi nilai-nilai piksel tersebut yaitu 600. Sedangkan lapisan luaran memiliki jumlah 10 node yang merepresentasikan angka 0 hingga 9 dan berisi nilai dengan tipe data *double*.
7. Melakukan pelatihan.
8. Membuat inisiasi array dalam bahasa C++ untuk bobot dan bias hasil pelatihan JST yang selanjutnya akan digunakan untuk melakukan pengujian jaringan dalam sistem penilaian daring SPOJ[1].



Gambar 3.3: Proses Pelatihan Jaringan Saraf Tiruan

Pengujian jaringan dilakukan dalam sistem penilaian daring SPOJ[1] dengan dataset pada sistem penilaian itu sendiri. Proses pengujian jaringan saraf tiruan dapat dilihat pada gambar 3.4. Program yang akan dikirim adalah program yang menggunakan bahasa pemrograman C++ dengan langkah:

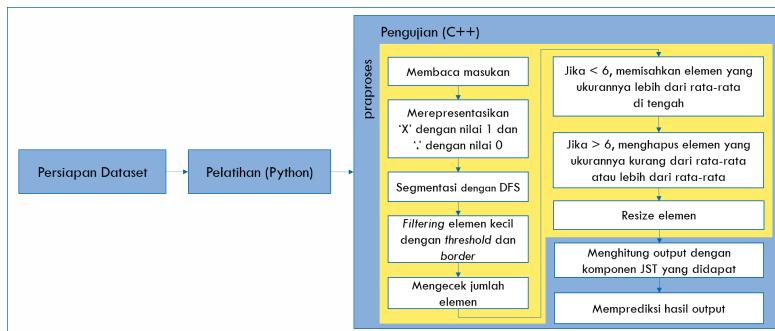
1. Membaca masukan citra dari 6 angka yang berupa sekumpulan karakter 'X' dan '.'.
2. Merepresentasikan karakter 'X' dengan piksel bernilai 1 dan karakter '.' menjadi piksel bernilai 0.

3. Melakukan segmentasi.
4. Melakukan *filtering* untuk derau kecil dengan *threshold* yang ditentukan melalui uji coba dan *border*. Suatu elemen dianggap sebagai *border* jika lebarnya sama dengan lebar citra masukan atau tingginya sama dengan tinggi citra masukan.
5. Mengecek apakah jumlah angka kurang dari 6 atau terdapat angka yang tersambung. Jika ya, membagi elemen yang memiliki lebar atau tinggi yang lebih dari rata-rata.
6. Mengecek apakah jumlah angka lebih dari 6 atau terdapat derau yang diprediksi sebagai angka. Jika ya, menghapus elemen yang lebar atau tingginya kurang dari *threshold* bawah dan lebih dari *threshold* atas yang ditentukan.
7. *Resize* elemen dengan menambahkan nilai 0 pada atas dan atau bawah hingga memiliki tinggi 30 dan lebar 20.
8. Menghitung luaran dengan menggunakan bobot dan bias yang didapat dari program Python dengan menggunakan persamaan 2.1.
9. Memasukkan nilai luaran ke dalam fungsi aktivasi yang dipilih. Nilai yang didapat terakhir akan berupa *array* dengan indeks 0 hingga 9 yang berisi nilai dengan tipe data *double*. Indeks yang nilai arraynya tertinggi adalah prediksi dari citra tersebut.

Filtering derau kecil dan *border* dilakukan terpisah dengan elemen yang lebar atau tingginya lebih dari rata-rata karena elemen yang terlalu kecil atau elemen yang terlalu besar akan mempengaruhi perhitungan rata-rata itu sendiri. Maka dari itu, perlu dihilangkan terlebih dahulu sebelum menghitung rata-rata tinggi dan lebar angka.

Selain 2 program di atas, terdapat program tambahan yang dibuat dalam bahasa Python untuk membantu proses utama, yaitu:

1. Program yang mengubah sekumpulan karakter 'X' dan '.' menjadi citra dengan tipe *png* untuk membantu menyiapkan



Gambar 3.4: Proses Pengujian Jaringan Saraf Tiruan

berkas pada [2] menjadi citra untuk pelatihan jaringan saraf tiruan pada bagian 3.4.1.

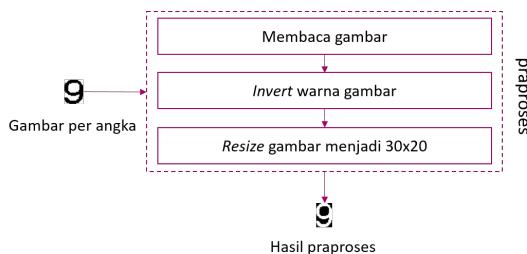
2. Program yang menambah piksel pada citra (sama seperti *scaling*) untuk membantu penambahan variasi citra pelatihan jaringan saraf tiruan pada bagian 3.4.2.
3. Program untuk membuat *array* dalam bahasa C++ yang berisi bobot dan bias yang didapat dari proses pelatihan jaringan saraf tiruan untuk mempermudah pemasukan nilai-nilai yang banyak ke dalam program C++ pada bagian 3.4.3.

3.2. Desain Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Subbab ini menjelaskan desain program pemanggilan fungsi pembuatan jaringan saraf tiruan dan melatih jaringan saraf tiruan yang menggunakan bahasa Python. Terdapat *pseudocode* yang akan menjelaskan tiap proses yang merupakan potongan-potongan dari keseluruhan program.

3.2.1. Desain Praproses

Langkah pertama adalah praproses atau proses mempersiapkan citra sebelum melakukan pelatihan jaringan. Masukan dari praproses adalah citra per angka yang sudah disiapkan pada tahap sebelumnya. Keluaran dari praproses adalah citra biner yang memiliki tinggi 30 dan lebar 20. Gambaran dari praproses dapat dilihat pada gambar 3.5. Fungsi dan variabel yang digunakan pada tahap praproses pada program Python dapat dilihat pada tabel 3.1 dan 3.2. Sedangkan *pseudocode* tahap praproses pada program Python dapat dilihat pada gambar 3.6.



Gambar 3.5: Praproses pada Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST

Tabel 3.1: Daftar Fungsi Tahap Praproses Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST

No	Nama Fungsi	Penjelasan
1	imread	Fungsi untuk membaca citra masukan
2	cvtColor	Fungsi untuk mengubah citra masukan menjadi citra <i>grayscale</i>
3	threshold	Fungsi untuk melakukan <i>thresholding</i>
4	resize	Fungsi untuk mengubah dimensi citra
5	empty	Fungsi untuk membuat <i>array</i> yang kosong

Tabel 3.2: Daftar Variabel Tahap Praproses pada Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST

No	Nama Variabel	Tipe	Penjelasan
1	img	uint8	Array 2D yang berisi nilai piksel citra masukan per angka
2	img_gray	uint8	Array 2D yang berisi nilai piksel citra per angka setelah diubah menjadi citra <i>grayscale</i>
3	img_th	uint8	Array 2D yang berisi nilai piksel citra per angka setelah dilakukan binerisasi menggunakan <i>thresholding</i> dan perubahan dimensi citra

```

1   img <- cv2.imread(file)
2   img_gray <- cv2.cvtColor(img,
      cv2.COLOR_BGR2GRAY)
3   ret, img_th <- cv2.threshold(img_gray, 0, 1,
      cv2.THRESH_BINARY_INV)
4   IF(img_th.shape[0] != 30 OR img_th.shape[1] !=
      20):
5       img_th <- cv2.resize(img_th, (20, 30),
          interpolation <- cv2.INTER_CUBIC)
6   ENDIF

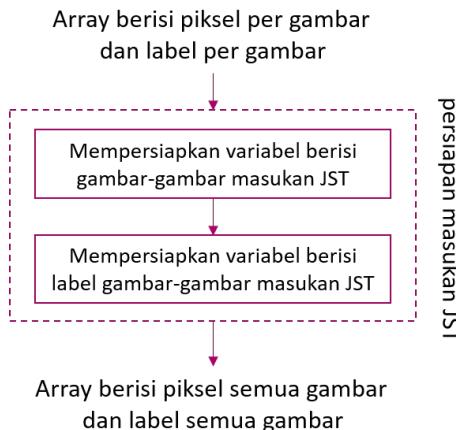
```

Gambar 3.6: *Pseudocode* Tahap Praproses pada Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST

3.2.2. Desain Persiapan Masukan Jaringan Saraf Tiruan

Sebelum memasuki tahap pemanggilan fungsi pembuatan jaringan saraf tiruan, persiapan masukan yang dilakukan adalah mempersiapkan variabel berisi citra dan label citra. Gambaran dari tahap ini dapat dilihat pada gambar 3.7. Fungsi dan variabel yang diguna-

kan pada tahap persiapan masukan jaringan saraf tiruan pada program Python dapat dilihat pada tabel 3.3 dan 3.4. Sedangkan *pseudocode* tahap persiapan masukan jaringan saraf tiruan pada program Python dapat dilihat pada gambar 3.8.



Gambar 3.7: Persiapan Masukan JST

Tabel 3.3: Daftar Fungsi Tahap Persiapan Masukan JST

No	Nama Fungsi	Penjelasan
1	append	Fungsi untuk menambahkan elemen di akhir <i>list</i>
2	ravel	Fungsi untuk membuat <i>array</i> 2 dimensi menjadi 1 dimensi
3	tolist	Fungsi untuk mengubah <i>array</i> menjadi <i>list</i>
4	os.path.splitext	Fungsi untuk membagi lokasi penyimpanan citra menjadi folder dan nama berkas citra
5	split	Fungsi untuk membagi sebuah <i>string</i>

Tabel 3.4: Daftar Variabel Tahap Persiapan Masukan JST

No	Nama Variabel	Tipe	Penjelasan
1	images	float64	Array 2D yang berisi nilai piksel semua citra masukan per angka
2	labels	float64	Array 2D yang berisi semua label citra masukan per angka
3	img_th	uint8	Array 2D yang berisi nilai piksel citra per angka setelah dilakukan binerisasi menggunakan <i>thresholding</i> dan perubahan dimensi citra
4	file	string	Nama berkas citra

```

1   images <- np.append(images,
                      np.array([img_th.ravel().tolist()], float),
                      axis=0)
2   labels <- np.append(labels,
                          np.array([array_class[int(os.path.splitext
(file)[0].split('_')[0])]], float), axis=0)

```

Gambar 3.8: *Pseudocode* Tahap Persiapan Masukan JST

3.2.3. Desain Pemanggilan Fungsi Pembuatan Jaringan Saraf Tiruan

Langkah selanjutnya adalah pemanggilan fungsi pembuatan jaringan saraf tiruan menggunakan *library Scikit-learn* yaitu fungsi *MLPClassifier*. Jumlah *node* layer masukan adalah sejumlah piksel pada citra. Citra memiliki tinggi 30 dan lebar 20 sehingga jumlah piksel ada 600. Maka dari itu, jumlah *node* layer masukan adalah 600. Sedangkan jumlah *node* layer luaran adalah 10 karena terdapat

10 angka yang akan dijadikan label, yaitu angka 0 sampai 9. Nilai dari parameter arsitektur jaringan saraf tiruan didapat dari hasil uji coba. Parameter yang digunakan pada tahap uji coba adalah dapat dilihat pada tabel 3.5. Sedangkan untuk parameter lain, menggunakan nilai *default*.

Tabel 3.5: Daftar Parameter Arsitektur JST untuk Tahap Uji Coba

No	Nama	Nama Parameter pada Fungsi
1	Jumlah <i>node</i> pada layer tersembunyi	hidden_layer_sizes
2	Fungsi aktivasi	activation
3	Fungsi optimasi / <i>solver</i>	solver
4	Learning rate	learning_rate_init

Keluaran tahap ini adalah jaringan dengan parameter arsitektur yang sudah ditentukan di atas. Gambaran dari tahap ini dapat dilihat pada gambar 3.9. Fungsi dan variabel yang digunakan pada tahap pemanggilan fungsi pembuatan jaringan saraf tiruan pada program Python dapat dilihat pada tabel 3.6 dan 3.7. Sedangkan *pseudocode* tahap pemanggilan fungsi pembuatan jaringan saraf tiruan pada program bahasa Python dapat dilihat pada gambar 3.10.



Gambar 3.9: Pemanggilan Fungsi Pembuatan dan Pelatihan JST

Tabel 3.6: Daftar Fungsi Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan JST

No	Nama Fungsi	Penjelasan
1	MLPClassifier	Fungsi untuk membuat jaringan saraf tiruan pada <i>library Scikit-learn</i>

Tabel 3.7: Daftar Variabel Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan JST

No	Nama Variabel	Tipe	Penjelasan
1	mlp	network	Variabel yang berisi model dari jaringan saraf tiruan

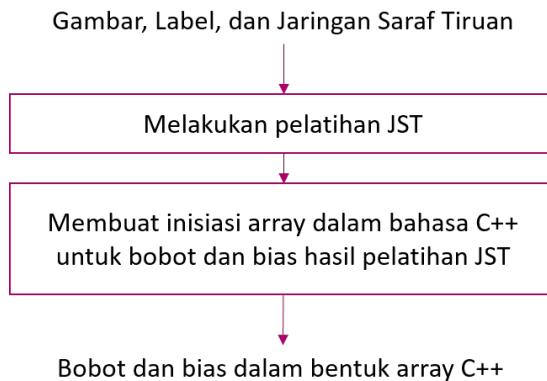
```
1 mlp <- MLPClassifier(hidden_layer_sizes=(40),
max_iter=1, alpha=1e-4,
activation='logistic', solver='sgd',
verbose=10, tol=1e-4, random_state=1,
learning_rate_init=.01, warm_start=True)
```

Gambar 3.10: *Pseudocode* Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan JST

3.2.4. Desain Pelatihan Jaringan Saraf Tiruan

Langkah selanjutnya adalah pelatihan JST menggunakan jaringan yang sudah dibuat pada tahap sebelumnya. Terdapat variabel *epoch* yang nilainya didapat dari hasil uji coba. Setelah melakukan pelatihan, bobot dan bias yang didapat dibulatkan menjadi angka yang memiliki 2 angka di belakang koma dan disimpan dalam berkas CSV. Setelah itu, dilakukan pembuatan inisiasi *array* dalam bahasa C++ untuk bobot dan bias tersebut menggunakan program

tambahan pada bagian 3.4.3. Keluaran tahap ini adalah bobot dan bias jaringan dalam bentuk array bahasa C++. Gambaran dari tahap ini dapat dilihat pada gambar 3.11. Fungsi dan variabel yang digunakan pada tahap pelatihan jaringan saraf tiruan pada program Python dapat dilihat pada tabel 3.8 dan 3.9. Sedangkan *pseudocode* pelatihan jaringan saraf tiruan dapat dilihat pada gambar 3.12 dan 3.13.



Gambar 3.11: Pelatihan JST

Tabel 3.8: Daftar Fungsi Tahap Pelatihan JST

No	Nama Fungsi	Penjelasan
1	fit	Fungsi untuk melatih jaringan saraf tiruan pada <i>library Scikit-learn</i>
2	coefs_	Fungsi untuk mendapatkan bobot jaringan saraf tiruan pada <i>library Scikit-learn</i>
3	intercepts_	Fungsi untuk mendapatkan bias jaringan saraf tiruan pada <i>library Scikit-learn</i>
4	round	Fungsi untuk melakukan pembulatan bobot dan bias
5	append	Fungsi untuk menambahkan elemen di akhir <i>list</i>
6	savetxt	Fungsi untuk menyimpan variabel dalam bentuk CSV

Tabel 3.9: Daftar Variabel Tahap Pelatihan JST

No	Nama Variabel	Tipe	Penjelasan
1	mlp	network	Variabel yang berisi model dari jaringan saraf tiruan
2	epoch	int	Variabel yang berisi jumlah epoch yang digunakan untuk pelatihan
3	weights	list	Variabel yang berisi bobot jaringan
4	biases	list	Variabel yang berisi bias jaringan
5	rounded_weights	list	Variabel yang berisi bobot jaringan yang sudah dibulatkan
6	rounded_biases	list	Variabel yang berisi bias jaringan yang sudah dibulatkan

```

1 epoch = 12000
2 FOR i in range(epoch):
3     mlp.fit(images, labels)
4 ENDFOR
5 weights <- mlp.coefs_
6 biases <- mlp.intercepts_
7 rounded_weights <- []
8 FOR j in weights:
9     j <- np.round(j,2)
10    rounded_weights.append(j)
11 ENDFOR
12 rounded_biases <- []

```

Gambar 3.12: *Pseudocode* Tahap Pelatihan JST (Bg.1)

```

1   FOR j in biases:
2       j <- np.round(j,2)
3       rounded_biases.append(j)
4   ENDFOR
5   np.savetxt("rounded_weights_1.csv",
6             rounded_weights[0], fmt='%.2f',
7             delimiter=",")
6   np.savetxt("rounded_weights_2.csv",
7             rounded_weights[1], fmt='%.2f',
8             delimiter=",")
7   np.savetxt("rounded_biases_1.csv",
8             rounded_biases[0], fmt='%.2f',
9             delimiter=",")
8   np.savetxt("rounded_biases_2.csv",
9             rounded_biases[1], fmt='%.2f',
10            delimiter=",")
```

Gambar 3.13: *Pseudocode* Tahap Pelatihan JST (Bg.2)

3.3. Desain Pengujian Jaringan Saraf Tiruan

Subbab ini menjelaskan desain fungsi yang akan digunakan dalam program pengujian jaringan saraf tiruan yang menggunakan bahasa C++ untuk menguji jaringan yang akan dikirim ke situs penilaian daring SPOJ.

3.3.1. Desain Fungsi Main

Fungsi ini menerima semua masukan yang diperlukan dari pengguna berupa sebuah bilangan yang menyatakan jumlah kasus uji sistem, tinggi dan lebar citra, dan kumpulan karakter 'X' dan '.' dan mengeluarkan hasil prediksi citra masukan. Fungsi ini juga mengecek apakah jumlah elemen hasil segmentasi kurang dari 6 (karena ada angka yang tersambung) dan apakah jumlah elemen hasil segmentasi lebih dari 6 dan mengatasi hal tersebut. *Pseudocode* fungsi main dapat dilihat pada citra 3.14 dan 3.15.

```
MAIN()
1  tc = Input()
2  for t = 0 to tc-1
3      h, w = Input()
4      dgt_ctr = 0
5      for i = 0 to h-1
6          for j = 0 to w-1
7              pic[i][j] = Input()
8      for i = 0 to h-1
9          for j = 0 to w-1
10         v[i][j] = 0.0;
11     meanw, meanh = 0.0
12     for j = 0 to w-1
13         for i = 0 to h-1
14             if !v[i][j] and pic[i][j] = 'X'
15                 candidate.xmin = 10000
16                 candidate.xmax = -1
17                 candidate.ymin = 10000
18                 candidate.ymax = -1
19                 DFS(i, j, h, w)
20                 if(checkDigit(h, w))
21                     dgt_ctr = dgt_ctr + 1
22                     candidate.id = dgt_ctr
23                     meanh = meanh + candidate.ymax -
24                         candidate.ymin + 1
25                     meanw = meanw + candidate.xmax -
26                         candidate.xmin + 1
27                     candidates.push_back(candidate)
28                 else
29                     continue
30             meanh = meanh/candidates.size()
31             meanw = meanw/candidates.size()
32             if candidates.size() < 6
33                 for i = 0 to candidates.size()-1
```

Gambar 3.14: *Pseudocode* Fungsi Main Program Pengujian JST (Bg.1)

```

32         if candidates[i].xmax -
            candidates[i].xmin > meanw
33             dgt newdgt
34             it = candidates.begin() + i + 1
35             newdgt.id = candidates.size() + 5
36             newdgt.xmax = candidates[i].xmax
37             newdgt.ymin = candidates[i].ymin
38             newdgt.ymax = candidates[i].ymax
39             candidates[i].xmax = candidates[i].xmax
                  - meanw
40             newdgt.xmin = candidates[i].xmax + 1
41             candidates.insert(it, newdgt)
42             digit = candidates
43         else if candidates.size() > 6
44             for i = 0 to candidates.size()-1
45                 if candidates[i].xmax -
                     candidates[i].xmin + 1 < meanw - 7 or
                     candidates[i].xmax -
                     candidates[i].xmin + 1 > meanw + 7 or
                     candidates[i].ymax -
                     candidates[i].ymin + 1 < meanh - 7 or
                     candidates[i].ymax -
                     candidates[i].ymin + 1 > meanh +7
46                 continue
47                 digit.push_back(candidates[i])
48             else
49                 digit = candidates
50             for i = 0 to digit.size()-1
51                 loadModel(i)
52                 output ans
53                 candidates.erase(candidates.begin(),
                     candidates.end())
54             digit.erase(digit.begin(), digit.end())
55             ans.erase(ans.begin, ans.end())

```

Gambar 3.15: *Pseudocode Fungsi Main Program Pengujian JST (Bg.2)*

3.3.2. Desain Fungsi DFS

Fungsi ini melakukan rekursi untuk proses segmentasi yaitu dengan mengecek apakah piksel 8 arah tetangga memiliki nilai yang sama, yaitu 1. *Pseudocode* fungsi DFS dapat dilihat pada citra 3.16.

```

1  DFS(i, j, h, w)
2  if i < 0 or i = h return
3  if j < 0 or j = w return
4  if v[i][j] or pic[i][j] != 'X' return
5  dx = {0, +1, 0, -1, -1, +1, +1, -1}
6  dy = {+1, 0, -1, 0, +1, -1, +1, -1}
7  v[i][j] = 1.0
8  candidate.xmin = min(candidate.xmin, j)
9  candidate.ymin = min(candidate.ymin, i)
10 candidate.xmax = max(candidate.xmax, j)
11 candidate.ymax = max(candidate.ymax, i)
12 for direction = 0 to 7
13     DFS(i + dy[direction], j + dx[direction], h,
w)

```

Gambar 3.16: *Pseudocode* Fungsi DFS Program Pengujian JST

3.3.3. Desain Fungsi Cek Digit

Fungsi ini mengecek apakah elemen adalah derau kecil. Sebuah elemen dikatakan derau kecil jika tinggi atau lebarnya kurang dari 3. Angka tersebut didapat dari hasil uji coba. Fungsi ini juga mengecek apakah elemen adalah *border*. Sebuah elemen dikatakan *border* jika lebarnya sama dengan lebar citra masukan atau tingginya sama dengan tinggi citra masukan. *Pseudocode* fungsi cek digit dapat dilihat pada citra 3.17.

```

checkDigit(h, w)
1   if candidate.ymax - candidate.ymin + 1 < 4 or
        candidate.xmax - candidate.xmin + 1 < 4
2     return false
3   else if candidate.xmax - candidate.xmin + 1
        == w or candidate.ymax - candidate.ymin +
        1 == h
4     return false;
5   else
6     return true

```

Gambar 3.17: *Pseudocode* Fungsi Cek Digit Program Pengujian JST (Bg.1)

3.3.4. Desain Fungsi Load Model

Fungsi ini melakukan pengujian jaringan saraf tiruan yang sebelumnya dilatih dengan program bahasa Python. Pada fungsi ini terdapat *array* bobot dan bias jaringan. Pengujian dilakukan dengan menghitung nilai *node* pada layer tersembunyi dan *node* pada layer luaran dengan rumus 2.1 dan menggunakan bobot dan bias jaringan. Sebelum melakukan pengujian, fungsi ini mengubah ukuran elemen yang akan dihitung nilai *node-nodenya* menjadi memiliki tinggi 30 dan lebar 20 sesuai dengan ukuran citra pada dataset pelatihan. Perubahan elemen dilakukan dengan menambahkan nilai 0 pada sisi atas dan kanan elemen. *Pseudocode* fungsi loadModel dapat dilihat pada citra 3.18, 3.19 dan 3.20.

```

loadModel(ctr)
1   resized[35][25] = {0.0}
2   a = digit[ctr].xmin

```

Gambar 3.18: *Pseudocode* Fungsi *Load Model* Program Pengujian JST (Bg.1)

```

3   b = digit[ctr].ymax
4   for i = 29 to 0
5     for j = 0 to 19
6       if j > digit[ctr].xmax - digit[ctr].xmin or
          (29-i) > digit[ctr].ymax -
             digit[ctr].ymin
7         resized[i][j] = 0.0
8       else
9         resized[i][j] = v[b][a]
10      a = a + 1
11    a = digit[ctr].xmin
12    b = b - 1
13    w_1[605][size_hdn_lyr+5] = bobot dari layer
        masukan ke layer tersembunyi
14    w_2[size_hdn_lyr+5][15] = bobot dari layer
        tersembunyi ke layer luaran
15    b_1[size_hdn_lyr+5] = bias layer tersembunyi
16    b_2[15] = bias layer luaran
17    vector<double> hdn_lyr1, output
18    double tmp, activated
19    int wi, wj = 0, bi = 0
20    for h = 0 to size_hdn_lyr-1
21      tmp = 0.0
22      wi = 0
23      for i = 0 to 29
24        for j = 0 to 19
25          tmp = tmp + (resized[i][j] *
             w_1[wi][wj])
26      wi = wi + 1
27      activated = actFunc(tmp + b_1[bi])
28      hdn_lyr1.push_back(activated)
29      wj = wj + 1
30      bi = bi + 1
31      wj = 0
32      bi = 0

```

Gambar 3.19: *Pseudocode Fungsi Load Model Program Pengujian JST (Bg.2)*

```

33     for h = 0 to 9
34         tmp = 0.0
35         wi = 0
36         for i = 0 to size_hdn_lyr-1
37             tmp = tmp + (hdn_lyr1[i] * w_2[wi][wj])
38             wi = wi + 1
39         activated = actFunc(tmp + b_2[bi])
40         output.push_back(activated)
41         wj = wj + 1
42         bi = bi + 1
43     predict(output)

```

Gambar 3.20: *Pseudocode* Fungsi *Load Model* Program Pengujian JST (Bg.3)

3.3.5. Desain Fungsi Fungsi Aktivasi

Fungsi ini melakukan aktivasi untuk nilai setiap *node*. Fungsi aktivasi merupakan salah satu parameter yang digunakan pada uji coba. Terdapat 4 fungsi aktivasi, yaitu *identity*, *logistic* atau *sigmoid*, *tanh*, dan *reLu*. *Pseudocode* tiap fungsi aktivasi ini dapat dilihat pada citra 3.21, 3.22, 3.23, dan 3.24.

```

actFunc(v)
1   return (v)

```

Gambar 3.21: *Pseudocode* Fungsi Fungsi Aktivasi *Identity*

```

actFunc(v)
1   return (1/(1 + pow(e, (-v))))

```

Gambar 3.22: *Pseudocode* Fungsi Fungsi Aktivasi *Sigmoid*

```

1 actFunc(v)
  return ((2/(1 + pow(e, (-2v))))-1)

```

Gambar 3.23: *Pseudocode Fungsi Fungsi Aktivasi Tanh*

```

1 actFunc(v)
  return (max(0, v))

```

Gambar 3.24: *Pseudocode Fungsi Fungsi Aktivasi ReLu*

3.3.6. Desain Fungsi Prediksi

Fungsi ini melakukan prediksi terhadap nilai pada *array node* luaran yang sebelumnya sudah dihitung pada bagian 3.3.4. Prediksi dilakukan dengan mencari indeks yang memiliki nilai tertinggi dari *array node* luaran. Indeks tersebut merupakan angka hasil prediksi dari proses pengujian kumpulan karakter yang tersegmentasi. *Pseudocode* fungsi ini dapat dilihat pada citra 3.25.

```

predict(output)
1 int dgt_prediction
2 double tmp = 0.0
3 for i = 0 to output.size()-1
4   if output[i] > tmp
5     tmp = output[i]
6     dgt_prediction = i
7 ans.push_back(dgt_prediction)

```

Gambar 3.25: *Pseudocode Fungsi Prediksi Program Pengujian JST*

3.4. Desain Program Tambahan

Subbab ini menjelaskan desain program tambahan yang dibuat dalam bahasa Python untuk membantu proses pelatihan dan pengujian.

3.4.1. Desain Program Konversi Berkas TXT ke PNG

Program ini adalah program tambahan yang dibuat dengan menggunakan bahasa Python untuk membantu mempersiapkan dataset pelatihan. Contoh citra pada [2] adalah berupa berkas TXT yang berisi kumpulan karakter 'X' dan '..'. Sedangkan masukan untuk jaringan harus berupa PNG. Maka dari itu, perlu dilakukan konversi dari berkas TXT ke berkas PNG. *Pseudocode* program ini dapat dilihat pada citra 3.26.

```
1  img <- np.empty((20, 40))
2  input namainput
3  input namaoutput
4  inp <- open(namainput, 'r')
5  x <- 0
6  FOR line in inp:
7      y <- 0
8      FOR l in line:
9          IF l = '.':
10              img[x][y]=255
11          ELSEIF l = 'X':
12              img[x][y]=1
13          ENDIF
14          y +=1
15      ENDFOR
16      x+=1
17  ENDFOR
18  cv2.imwrite(namaoutput + ".png", img)
```

Gambar 3.26: *Pseudocode* Program Konversi berkas TXT ke PNG

3.4.2. Desain Program Scaling Citra

Program ini adalah program tambahan yang dibuat dengan menggunakan bahasa Python untuk membantu menambah variasi citra pelatihan pada tahap persiapan dataset. Penambahan variasi citra pelatihan dilakukan dengan melakukan *scaling* citra dengan cara menambahkan piksel pada citra sesuai dengan penjelasan pada bagian 3.1. *Pseudocode* program ini dapat dilihat pada citra 3.27 dan 3.28.

```

1   FOR digit in range(1,10):
2       os.chdir(str(digit))
3       path <- os.getcwd()
4       i <- 5000
5       FOR f in listdir(path):
6           img <- cv2.imread(f)
7           FOR x in range(0,21):
8               ret,thresh <-
9                   cv2.threshold(img,127,255,cv2.
10                      THRESH_BINARY_INV)
11                  constant <-
12                      cv2.copyMakeBorder(thresh,x,0,0,x,cv2.
13                         BORDER_CONSTANT,255)
14                     constant <- misc.imresize(constant, (30,
15                         24))
16                     constant <- cv2.cvtColor(constant,
17                         cv2.COLOR_BGR2GRAY)
18                     ret,thresh1 <-
19                         cv2.threshold(constant,127,255,cv2.
20                             THRESH_BINARY_INV)
21                             cv2.imwrite(str(digit) + "_" + str(i) +
22                               ".png", thresh1)
23                             i += 1

```

Gambar 3.27: *Pseudocode* Program *Scaling* Citra (Bg.1)

```

1      ENDFOR
2      ENDFOR
3      os.chdir("../")
4      ENDFOR

```

Gambar 3.28: *Pseudocode Program Scaling Citra (Bg.2)*

3.4.3. Desain Program Transformasi Komponen JST ke dalam C++

Bobot dan bias jaringan saraf tiruan yang didapat setelah pelatihan dilakukan adalah berupa *list* yang dapat dilihat pada citra 3.29. Program ini adalah program tambahan yang dibuat dengan menggunakan bahasa Python untuk membantu melakukan transformasi bobot dan bias pada *list* tersebut ke dalam bahasa C++ dalam bentuk *array*. Bentuk umum *array* pada bahasa C++ dapat dilihat pada citra 3.30. Ukuran layer tersembunyi yang disimpan pada variabel *size_hdn_lyr* didapat dari hasil uji coba. *Pseudocode* program ini dapat dilihat pada citra 3.31 dan 3.32.

weights - List (2 elements)			
Index	Type	Size	Value
0	float64	(600, 20)	array([-0.01095836, 0.03878816, -0.06187729, ..., 0.10243014, ...])
1	float64	(20, 10)	array([-3.30968772, -11.22424519, 4.22354997, ..., -8.58208638, ...])

Gambar 3.29: *List* yang Berisi Bobot dan Bias Hasil Pelatihan Jaringan Saraf Tiruan

```

double w_2[30][15] =
{{ -3.31, -11.22, 4.22, 0.76, -6.71, 4.94, -10.06, -8.58, 15.39, -11.29,
  { 8.64, 1.56, -18.06, 14.55, 11.28, -19.98, -11.73, 4.69, -15.09, 5.21 },
  { 2.93, 18.38, 11.03, -11.94, -5.16, 1.51, -6.14, 2.78, -14.66, -11.85 },
  { -9.09, 18.47, -18.19, 5.96, 10.61, -2.21, -2.38, 10.96, 10.23, -11.42 },
  { 6.82, -13.57, -3.49, -3.98, -0.46, -3.77, -12.06, 22.02, 5.98, -3.46 },
  { 2.17, -6.45, -16.41, 20.58, -8.16, 7.06, -0.07, -4.08, 15.13, -3.68 },
  { -0.26, 17.17, -17.47, -22.03, 4.58, -3.89, 7.69, -12.06, -12.63, 18.12 },
  { -15.91, 6.18, 1.44, 6.39, -12.55, -2.75, -18.44, -0.88, 8.18, 9.54 },
  { -11.25, 9.52, -5.16, 7.31, -10.25, 10.31, 11.68, -12.44, -4.78, -2.96 },
  { 6.78, -17.86, 8.51, -12.98, 5.35, -19.93, 7.52, -7.21, 4.25, 18.58 },
  { -3.08, 2.52, 3.95, 9.36, 10.88, -10.77, -4.83, -12.08, -6.13, -13.18 },
  { 10.21, -6.19, 0.84, -4.37, -1.94, -0.07, 2.38, 15.28, -18.25, -1.17 },
  { -16.83, 21.82, -1.07, 9.34, -16.13, -5.04, -6.23, 8.86, -19.95, -18.96 },
  { 6.01, -9.94, -3.18, -18.28, -0.64, 12.04, 11.65, 17.98, -13.68, -15.65 },
  { -8.03, 6.93, 6.93, -10.03, -6.82, -16.72, 17.58, 0.98, -12.49, -5.16 },
  { -1.92, 6.33, 10.93, 0.35, 16.48, 7.54, -28.66, -5.92, -0.46, -7.72 },
  { -0.88, -3.95, -4.27, -6.35, -15.99, -13.91, 10.26, -9.16, -10.66, 8.12 },
  { -0.50, -17.75, -10.76, 1.37, -22.93, -6.88, -3.33, 5.56, -10.45, 21.88 },
  { -22.38, 6.86, -3.68, -9.76, -0.16, -8.95, 12.91, -0.51, 9.41, -8.94 },
  { 4.25, -17.64, -11.71, -9.67, 12.42, 12.88, -10.76, -13.54, 8.01, 2.29 }};

```

Gambar 3.30: Bentuk Umum Inisiasi array bahasa C++

```

1  input size_hdn_lyr
2  a <- "double b_1[int(hiddenlayer1size) + 5] = {"
3  with open('rounded_biases_1.csv', newline='') as csvfile:
4      spamreader <- csv.reader(csvfile, delimiter=',', quotechar='|')
5      FOR row in spamreader:
6          x <- (', '.join(row))
7          a <- a + x + ","
8      ENDFOR
9      a <- a + "}";
10     file <- open("rounded_biases_1.txt", "w")
11     file.write(a)
12     a <- "double b_2[15] = {"
13     with open('rounded_biases_2.csv', newline='') as csvfile:
14         spamreader <- csv.reader(csvfile, delimiter=',', quotechar='|')
15         FOR row in spamreader:
16             x <- (', '.join(row))
17             a <- a + x + ","
18         ENDFOR
19         a <- a + "}";

```

Gambar 3.31: Pseudocode Program Inisiasi Array C++ (Bg.1)

```
20     file <- open("rounded_biases_2.txt","w")
21     file.write(a)
22     a <- "double w_1[605][int(hiddenlayer1size) +
      5] = {""
23     with open('rounded_weights_1.csv', newline='') as csvfile:
24         spamreader <- csv.reader(csvfile,
25             delimiter=',', quotechar='|')
26         a <- a + "{ "
27         FOR row in spamreader:
28             x <- ('', '.join(row))
29             a <- a + x
30             a <- a + "},," + "\n" + "{ "
31         ENDFOR
32         a <- a + "}";
33     file <- open("rounded_weights_1.txt","w")
34     file.write(a)
35     a <- "double w_2[int(hiddenlayer1size) + 5][15] =
      {""
36     with open('rounded_weights_2.csv', newline='') as csvfile:
37         spamreader <- csv.reader(csvfile,
38             delimiter=',', quotechar='|')
39         a <- a + "{ "
40         FOR row in spamreader:
41             x <- ('', '.join(row))
42             a <- a + x
43             a <- a + "},," + "\n" + "{ "
44         ENDFOR
45         a <- a + "}";
46     file <- open("rounded_weights_2.txt","w")
47     file.write(a)
```

Gambar 3.32: *Pseudocode* Program Inisiasi Array C++ (Bg.2)

BAB IV

IMPLEMENTASI

4.1. Lingkungan implementasi

Lingkungan implementasi dan pengembangan yang dilakukan adalah sebagai berikut.

4.1.1. Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

1. Perangkat Keras
 - (a) Processor Intel® Core™ i3-4150 CPU @ 3.50GHz, 3.5GHz
 - (b) Random Access Memory 4GB
2. Perangkat Lunak
 - (a) Sistem Operasi Windows 10 Pro 64-bit
 - (b) IDE Spyder 3.2.6
 - (c) Bahasa Pemrograman Python 3.6
 - (d) MinGW 64-bit

4.1.2. Program Pengujian Jaringan Saraf Tiruan

1. Perangkat Keras
 - (a) Processor Intel® Core™ i3-4150 CPU @ 3.50GHz, 3.5GHz
 - (b) Random Access Memory 4GB
2. Perangkat Lunak
 - (a) Sistem Operasi Windows 10 Pro 64-bit
 - (b) IDE Dev-C++ 5.11
 - (c) Bahasa Pemrograman C++
 - (d) MinGW 64-bit

4.2. Implementasi Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Subbab ini menjelaskan implementasi program pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan. Program ini merupakan program yang menggunakan bahasa Python.

4.2.1. Penggunaan Library

Program ini menggunakan beberapa *library* seperti yang ditunjukkan pada kode sumber 4.1. Penjelasan fungsi dari *library* untuk tugas akhir ini ditunjukkan pada tabel 4.1.

```

1 from sklearn.neural_network import MLPClassifier
2 import os
3 import cv2
4 from os import listdir
5 import numpy as np

```

Kode Sumber 4.1: *Header* Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST

Tabel 4.1: Penjelasan *Library* Program Pemanggilan Fungsi Pembuatan dan Pelatihan JST

No	Nama <i>Library</i>	Penjelasan
1	MLPClassifier	Untuk memanggil fungsi pembuatan dan pelatihan JST
2	os	Untuk masuk ke dalam suatu folder
3	cv2	Untuk melakukan operasi yang berhubungan dengan c
4	listdir	Untuk mendapatkan semua berkas citra yang ada pada satu direktori
5	numpy	Untuk komputasi ilmiah yang berhubungan dengan <i>array</i> dan matriks

4.2.2. Implementasi Praproses dan Persiapan Masukan Jaringan Saraf Tiruan

Subbab ini menjabarkan implementasi tahap praproses dan persiapan masukan jaringan saraf tiruan yang dijelaskan pada bagian 3.2.1 dan 3.2.2. Implementasi tahap praproses meliputi pembacaan data masukan berupa citra berwarna, mengubahnya menjadi citra *grayscale*, melakukan *invert* warna citra dengan menggunakan fungsi *thresholding*, dan *resize* citra menjadi memiliki tinggi 30 dan lebar 20. Luas citra masukan dari dataset bervariasi. Untuk itu, *resize* dilakukan apabila citra masukan tidak memiliki luas dari 30 x 20 piksel. Program akan melakukan tahap-tahap tersebut pada semua citra yang ada pada lokasi penyimpanan pada program. Setelah itu, program akan mempersiapkan masukan jaringan saraf tiruan dengan menambahkan hasil praproses citra masukan ke variabel *images* dan labelnya ke variabel *labels*. Implementasi tahap ini dapat dilihat pada kode sumber 4.2 dan 4.3.

```

1 images = np.empty((0,600), float)
2 labels = np.empty((0,10), float)
3 class_0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4 class_1 = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
5 class_2 = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
6 class_3 = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
7 class_4 = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
8 class_5 = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
9 class_6 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
10 class_7 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
11 class_8 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
12 class_9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
13 array_class = [class_0, class_1, class_2,
                 class_3, class_4, class_5, class_6, class_7,
                 class_8, class_9]
14 os.chdir("new_SPOJ_dataset")

```

Kode Sumber 4.2: Tahap Praproses dan Persiapan Masukan JST (Bg.1)

```
15 path = os.getcwd()
16 for file in listdir(path):
17     img = cv2.imread(file)
18     img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
19     ret, img_th = cv2.threshold(img_gray, 0, 1,
20         cv2.THRESH_BINARY_INV)
21     if(img_th.shape[0] != 30 or img_th.shape[1] !=
22         20):
23         img_th = cv2.resize(img_th, (20, 30),
24             interpolation = cv2.INTER_CUBIC)
25         images = np.append(images,
26             np.array([img_th.ravel().tolist()], float),
27             axis=0)
28     labels = np.append(labels,
29         np.array([array_class[int(os.path.splitext(
30             file)[0].split('_')[0])]], float), axis=0)
31 os.chdir("../")
32 os.chdir("dataset_SPOJ_reinforcement")
33 path = os.getcwd()
34 for file in listdir(path):
35     img = cv2.imread(file)
36     img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
37     ret, img_th = cv2.threshold(img_gray, 0, 1,
38         cv2.THRESH_BINARY_INV)
39     if(img_th.shape[0] != 30 or img_th.shape[1] !=
40         20):
41         img_th = cv2.resize(img_th, (20, 30),
42             interpolation = cv2.INTER_CUBIC)
43         images = np.append(images,
44             np.array([img_th.ravel().tolist()], float),
45             axis=0)
46     labels = np.append(labels,
47         np.array([array_class[int(os.path.splitext(
48             file)[0].split('_')[0])]], float), axis=0)
49 os.chdir("../")
```

Kode Sumber 4.3: Tahap Praproses dan Persiapan Masukan JST
(Bg.2)

4.2.3. Implementasi Pemanggilan Fungsi Pembuatan Jaringan Saraf Tiruan

Subbab ini menjabarkan implementasi tahap pemanggilan fungsi pembuatan jaringan saraf tiruan yang dijelaskan pada bagian 3.2.3. Implementasi tahap ini dapat dilihat pada kode sumber 4.4.

```

1 mlp = MLPClassifier(hidden_layer_sizes=(40),
                      max_iter=1, alpha=1e-4, activation='logistic',
2 solver='sgd', verbose=10, tol=1e-4,
                      random_state=1,
3 learning_rate_init=.01, warm_start=True)

```

Kode Sumber 4.4: Tahap Pemanggilan Fungsi Pembuatan JST

4.2.4. Implementasi Pelatihan Jaringan Saraf Tiruan

Subbab ini menjabarkan implementasi tahap pelatihan jaringan saraf tiruan yang dijelaskan pada bagian 3.2.4. Implmentasi tahap ini dapat dilihat pada kode sumber 4.5 dan 4.6. Baris 4 sampai 17 melakukan penyimpanan bobot dan bias yang didapat dari hasil pelatihan jaringan dalam bentuk berkas CSV.

```

1 epoch = 12000
2 for i in range(epoch):
3     mlp.fit(images, labels)
4 weights = mlp.coefs_
5 biases = mlp.intercepts_
6 rounded_weights = []
7 for j in weights:
8     j = np.round(j,2)
9     rounded_weights.append(j)
10 rounded_biases = []
11 for j in biases:
12     j = np.round(j,2)
13     rounded_biases.append(j)

```

Kode Sumber 4.5: Tahap Pelatihan JST (Bg.1)

```

14 np.savetxt("rounded_weights_1.csv",
    rounded_weights[0], fmt='%.2f', delimiter=",")
15 np.savetxt("rounded_weights_2.csv",
    rounded_weights[1], fmt='%.2f', delimiter=",")
16 np.savetxt("rounded_biases_1.csv",
    rounded_biases[0], fmt='%.2f', delimiter=",")
17 np.savetxt("rounded_biases_2.csv",
    rounded_biases[1], fmt='%.2f', delimiter=",")

```

Kode Sumber 4.6: Tahap Pelatihan JST (Bg.2)

4.3. Implementasi Program Pengujian Jaringan Saraf Tiruan

Subbab ini menjelaskan implementasi program pengujian jaringan saraf tiruan. Program ini merupakan program yang menggunakan bahasa C++ yang akan dikirim ke situs penilaian daring SPOJ.

4.3.1. Penggunaan Library, Variabel Global, dan Struct

Program ini menggunakan beberapa *library*, variabel global dan tipe data *struct* seperti yang ditunjukkan pada kode sumber 4.7, 4.8, dan 4.9. Penjelasan fungsi dari variabel global yang digunakan ditunjukkan pada tabel 4.2. *Struct* pada kode sumber 4.9 memiliki 5 variabel yang dimiliki oleh sebuah elemen yang tersegmentasi. Penjelasan mengenai variabel pada *struct* dijelaskan pada tabel 4.3.

```

1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 using namespace std;

```

Kode Sumber 4.7: Header Program Pengujian JST

```

1 const double e = exp(1.0);
2 char pic[255][255];
3 double v[255][255] = {0.0};
4 dgt candidate;
5 vector<dgt> candidates;
6 vector<dgt> digit;
7 vector<int> ans;
8 int dgt_ctr;

```

Kode Sumber 4.8: Variabel Global Program Pengujian JST

Tabel 4.2: Penjelasan Variabel Global Program Pengujian JST

No	Nama	Penjelasan
1	e	Menyimpan nilai eksponen dari angka 1
2	pic	Menyimpan citra masukan yang berupa karakter 'X' dan '.'
3	v	Menyimpan citra masukan yang nilainya sudah diubah berupa angka 1 dan 0
4	candidate	Menyimpan elemen yang sudah tersegmentasi
5	candidates	Menyimpan elemen-elemen yang bukan derau kecil dan <i>border</i>
6	digit	Menyimpan elemen-elemen yang sudah melalui tahap praproses dan akan diprediksi
7	ans	Menyimpan hasil prediksi angka
8	dgt_ctr	Menyimpan jumlah elemen sekarang yang akan dijadikan id dari suatu elemen

```

1 struct dgt{
2   int id, xmin, xmax, ymin, ymax;
3 };

```

Kode Sumber 4.9: Struct Program Pengujian JST

Tabel 4.3: Penjelasan Variabel *Struct* Program Pengujian JST

No	Nama	Penjelasan
1	id	Menyimpan id elemen
2	xmin	Menyimpan indeks kolom paling kiri dari elemen
3	xmax	Menyimpan indeks kolom paling kanan dari elemen
4	ymin	Menyimpan indeks baris paling atas dari elemen
5	ymax	Menyimpan indeks baris paling bawah dari elemen

4.3.2. Implementasi Fungsi Main

Subbab ini menjabarkan implementasi fungsi *main* yang dijelaskan pada bagian 3.3.1. Implementasi fungsi ini dapat dilihat pada kode sumber 4.10, 4.11, 4.12.

```

int main()
1 cin.sync_with_stdio(0); cin.tie(0);
2 int tc, h, w;
3 cin >> tc;
4 for (int t = 0; t < tc; t++) {
5     cin >> h >> w;
6     id = 0;
7     dgt_ctr = 0;
8     for (int i = 0; i < h; i++) {
9         cin >> pic[i];
10    }
11    for (int i = 0; i < h; i++) {
12        for(int j = 0; j < w; j++) {
13            v[i][j] = 0.0;
14        }
15    }
16    float meanw = 0.0;
17    float meanh = 0.0;
18    for(int j = 0; j < w; j++) {

```

Kode Sumber 4.10: Fungsi *Main* Program Pengujian JST (Bg.1)

```

1   for(int i = 0; i < h; i ++){  

2       if (!v[i][j] && pic[i][j] == 'X') {  

3           id += 1;  

4           candidate.xmin = 10000;  

5           candidate xmax = -1;  

6           candidate.ymax = 10000;  

7           candidate.ymin = -1;  

8           DFS(i, j, h, w);  

9           if(checkDigit()){  

10              dgt_ctrl += 1;  

11              candidate.id = dgt_ctrl;  

12              meanh = meanh + candidate.ymax -  

13                  candidate.ymin + 1;  

14              meanw = meanw + candidate xmax -  

15                  candidate.xmin + 1;  

16              candidates.push_back(candidate);  

17          }  

18      }  

19      meanh = meanh/candidates.size();  

20      meanw = meanw/candidates.size();  

21      if(candidates.size() < 6){  

22          for(int i = 0; i < candidates.size(); i ++){  

23              if(candidates[i].xmax - candidates[i].xmin  

24                  > meanw){  

25                  dgt newdgt;  

26                  auto it = candidates.begin() + i + 1;  

27                  newdgt.id = candidates.size() + 5;  

28                  newdgt xmax = candidates[i].xmax;  

29                  newdgt ymin = candidates[i].ymin;  

30                  newdgt ymax = candidates[i].ymax;  

31                  candidates[i].xmax = candidates[i].xmax -  

32                      meanw;  

33                  newdgt xmin = candidates[i].xmax + 1;  

34                  candidates.insert(it, newdgt);  

35          }

```

Kode Sumber 4.11: Fungsi *Main* Program Pengujian JST (Bg.2)

```

    }
1   digit = candidates;
2 }
3 else if(candidates.size() > 6){
4   for(int i = 0; i < candidates.size(); i ++){
5     if(candidates[i].xmax - candidates[i].xmin
6       + 1 < meanw - 7 || candidates[i].xmax -
7         candidates[i].xmin + 1 > meanw + 7 ||
8         candidates[i].ymax - candidates[i].ymin
9           + 1 < meanh - 7 || candidates[i].ymax -
10          candidates[i].ymin + 1 > meanh + 7) {
11        continue;
12      }
13      digit.push_back(candidates[i]);
14    }
15  }
16 else{
17   digit = candidates;
18 }
19 for(int i = 0; i < digit.size(); i ++){
20   loadModel(i);
21 }
22 for(int i = 0; i < ans.size(); i ++){
23   printf("%d", ans[i]);
24 }
25
26 printf("\n");
27 candidates.erase
28   (candidates.begin(), candidates.end());
29 digit.erase (digit.begin(), digit.end());
30 ans.erase(ans.begin(), ans.end());
31 }
32 }
```

Kode Sumber 4.12: Fungsi *Main* Program Pengujian JST (Bg.3)

4.3.3. Implementasi Fungsi DFS

Subbab ini menjabarkan implementasi fungsi DFS yang dijelaskan pada bagian 3.3.2. Fungsi akan mengecek ke 8 arah tetangga yaitu bawah, kanan, atas, kiri, bawah-kiri, atas-kanan, bawah-

kanan, dan atas-kiri. Pengecekan dilakukan dengan menelusuri 1 arah terlebih dahulu dengan rekursi. Fungsi kembali jika baris atau kolom kurang dari nol atau lebih dari ukuran citra masukan. Implementasi fungsi ini dapat dilihat pada kode sumber 4.13.

```

void DFS(int i, int j, int h, int w)
1 if (i < 0 || i == h) return;
2 if (j < 0 || j == w) return;
3 if (v[i][j] || pic[i][j] != 'X') return;
4 const int dx[] = {0, +1, 0, -1, -1, +1, +1, -1};
5 const int dy[] = {+1, 0, -1, 0, +1, -1, +1, -1};
6 v[i][j] = 1.0;
7 candidate.xmin = min(candidate.xmin, j);
8 candidate.ymin = min(candidate.ymin, i);
9 candidate xmax = max(candidate xmax, j);
10 candidate ymax = max(candidate ymax, i);
11 for (int direction = 0; direction < 8; direction
    ++){
12     DFS(i + dy[direction], j + dx[direction], h, w);
13 }

```

Kode Sumber 4.13: Fungsi *Main* Program Pengujian JST

4.3.4. Implementasi Fungsi Cek Digit

Subbab ini menjabarkan implementasi fungsi cek digit yang dijelaskan pada bagian 3.3.3. Implementasi fungsi ini dapat dilihat pada kode sumber 4.14.

```

bool checkDigit()
1 if(candidate.ymax - candidate.ymin + 1 < 4 ||
    candidate xmax - candidate xmin + 1 < 4)
    return false;
2 else if(candidate xmax - candidate xmin + 1 == w
    || candidate ymax - candidate ymin + 1 == h)
    return false;
3 return true;

```

Kode Sumber 4.14: Fungsi Cek Digit Program Pengujian JST

4.3.5. Implementasi Fungsi Load Model

Subbab ini menjabarkan implementasi fungsi *load model* yang dijelaskan pada bagian 3.3.4. Implementasi fungsi ini dapat dilihat pada kode sumber 4.15 dan 4.16. Ukuran layer tersembunyi yang disimpan pada variabel *size_hdn_lyr* didapat dari hasil uji coba. *Array* bobot dan bias jaringan saraf tiruan disimpan pada variabel *w_1*, *w_2*, *b_1*, dan *b_2*. Nilai bobot dan bias yang didapat dari hasil uji coba sangat banyak. Maka dari itu, untuk nilai keempat variabel tersebut ada pada lampiran.

```

void loadModel(int ctr)
1 int size_hdn_lyr;
2 double resized[35][25] = {0.0};
3 int a = digit[ctr].xmin;
4 int b = digit[ctr].ymax;
5 for(int i = 29; i >= 0; i --){
6   for(int j = 0; j < 20; j ++){
7     if(j > (digit[ctr].xmax - digit[ctr].xmin) ||
       (29 - i) > (digit[ctr].ymax -
                     digit[ctr].ymin)) resized[i][j] = 0.0;
8     else resized[i][j] = v[b][a];
9     a += 1;
10    }
11    a = digit[ctr].xmin;
12    b -= 1;
13  }
14 double w_1[605][size_hdn_lyr+5];
15 double w_2[size_hdn_lyr+5][15];
16 double b_1[size_hdn_lyr+5];
17 double b_2[15];
18 vector<double> hdn_lyr1, output;
19 double tmp, activated;
20 int wi, wj = 0, bi = 0;
21 for(int h = 0; h < size_hdn_lyr; h ++){
22   tmp = 0.0;
23   wi = 0;
```

Kode Sumber 4.15: Fungsi *Load Model* Program Pengujian JST (Bg.1)

```

1   for(int i = 0; i < 30 ; i ++){  

2       for(int j = 0; j < 20; j ++){  

3           tmp = tmp + (resized[i][j] * w_1[wi][wj]);  

4           wi += 1;  

5       }  

6       activated = actFunc(tmp + b_1[bi]);  

7       hdn_lyr1.push_back(activated);  

8       wj += 1;  

9       bi += 1;  

10  }  

11 wj = 0, bi = 0;  

12 for(int h = 0; h < 10; h ++){  

13     tmp = 0.0;  

14     wi = 0;  

15     for(int i = 0; i < size_hdn_lyr; i ++){  

16         tmp = tmp + (hdn_lyr1[i] * w_2[wi][wj]);  

17         wi += 1;  

18     }  

19     activated = actFunc(tmp + b_2[bi]);  

20     output.push_back(activated);  

21     wj += 1;  

22     bi += 1;  

23 }  

24 predict(output);

```

Kode Sumber 4.16: Fungsi *Load Model* Program Pengujian JST (Bg.2)

4.3.6. Implementasi Fungsi Fungsi Aktivasi

Subbab ini menjabarkan implementasi fungsi fungsi aktivasi yang dijelaskan pada bagian 3.3.5. Terdapat 4 fungsi aktivasi yang akan digunakan pada tahap uji coba, yaitu *identity*, *logistic* atau *sigmoid*, *tanh*, dan *reLu*. Implementasi keempat fungsi aktivasi dapat dilihat pada kode sumber 4.17, 4.18, 4.19, 4.20.

```

1   double actFunc(double v)
    return (v);

```

Kode Sumber 4.17: Fungsi Aktivasi *Identity*

```

1   double actFunc(double v)
    return (1/(1 + pow(e, (-v))));
```

Kode Sumber 4.18: Fungsi Aktivasi *Sigmoid*

```

1   double actFunc(double v)
    return ((2/(1 + pow(e, (-2*v))))-1);
```

Kode Sumber 4.19: Fungsi Aktivasi *Tanh*

```

1   double actFunc(double v)
    return (max(0,v));
```

Kode Sumber 4.20: Fungsi Aktivasi *ReLU*

4.3.7. Implementasi Fungsi Prediksi

Subbab ini menjabarkan implementasi fungsi prediksi yang dijelaskan pada bagian 3.3.6. Implementasi fungsi ini dapat dilihat pada kode sumber 4.21 dan 4.22.

```

void predict(vector<double> output)
1   int dgt_prediction;
2   double tmp = 0.0;
3   for(int i = 0; i < output.size(); i ++){
```

Kode Sumber 4.21: Fungsi Prediksi Program Pengujian JST (Bg.1)

```

4     if (output[i] > tmp) {
5         tmp = output[i];
6         dgt_prediction = i;
7     }
8 }
9 ans.push_back(dgt_prediction);

```

Kode Sumber 4.22: Fungsi Prediksi Program Pengujian JST (Bg.2)

4.4. Implementasi Program Tambahan

Subbab ini menjabarkan implementasi program tambahan yang dijelaskan pada bagian 3.4. *Library* yang digunakan pada program ini adalah *numpy*, dan *cv2*. Program tambahan dibuat dengan menggunakan bahasa Python yang digunakan untuk membantu proses pelatihan dan pengujian.

4.4.1. Implementasi Program Konversi Berkas TXT ke PNG

Subbab ini menjabarkan implementasi program konversi berkas TXT ke PNG yang dijelaskan pada bagian 3.4.1. Implementasi program ini dapat dilihat pada kode sumber 4.23 dan 4.24.

```

1 import numpy as np
2 import cv2
3 namainput = "hir001.txt"
4 namaoutput = "hir001.png"
5 img = np.empty((20, 40))
6 inp = open(namainput, 'r')
7 x = 0
8 for line in inp:
9     y = 0
10    for l in line:
11        if l == '.':
12            img[x][y]=255

```

Kode Sumber 4.23: Program Konversi Berkas TXT ke PNG (Bg.1)

```

13     elif l == 'X':
14         img[x][y]=1
15         y +=1
16         x+=1
17     cv2.imwrite(namaoutput, img);

```

Kode Sumber 4.24: Program Konversi Berkas TXT ke PNG (Bg.2)

4.4.2. Implementasi Program Scaling Gambar

Subbab ini menjabarkan implementasi program *scaling* pada yang dijelaskan pada bagian 3.4.2. *Library* yang digunakan pada program ini adalah *os*, *cv2*, *listdir*, dan *misc*. Implementasi program ini dapat dilihat pada kode sumber 4.25 dan 4.26. Program membaca berkas yang ada pada lokasi penyimpanan yang ditentukan pada program dan melakukan *scaling* pada citra satu per satu dengan menambahkan piksel sesuai dengan penjelasan pada bagian 3.1.

```

1  import os
2  import cv2
3  from os import listdir
4  from scipy import misc
5  for digit in range(1,10):
6      os.chdir(str(digit))
7      path = os.getcwd()
8      i = 5000
9      for f in listdir(path):
10         print (digit, f)
11         img = cv2.imread(f)
12         for x in range(0,21):
13             ret,thresh =
14                 cv2.threshold(img,127,255,cv2.
15                             THRESH_BINARY_INV)
16             constant=
17                 cv2.copyMakeBorder(thresh,x,0,0,x,cv2.
18                             BORDER_CONSTANT,255)

```

Kode Sumber 4.25: Program Scaling Gambar (Bg.1)

```

15         constant=
16             cv2.copyMakeBorder(thresh,x,0,0,x,cv2.
17                 BORDER_CONSTANT,255)
18         constant = misc.imresize(constant, (30,
19             24))
20         constant = cv2.cvtColor(constant,
21             cv2.COLOR_BGR2GRAY)
22         ret,thresh1 =
23             cv2.threshold(constant,127,255,cv2.
24                 THRESH_BINARY_INV)
25         cv2.imwrite(str(digit) + "_" + str(i) +
26             ".png", thresh1)
27         i += 1
28     os.chdir("../")

```

Kode Sumber 4.26: Program Scaling Gambar (Bg.2)

4.4.3. Implementasi Program Inisiasi Array C++

Subbab ini menjabarkan implementasi program membuat inisiasi *array* C++ untuk bobot dan bias jaringan saraf tiruan hasil pelatihan yang dijelaskan pada bagian 3.4.3. Ukuran layer tersembunyi yang disimpan pada variabel *size_hdn_lyr* didapat dari hasil uji coba. *Library* yang digunakan untuk program ini adalah *csv*. *Library* tersebut digunakan untuk membaca berkas *csv*. Implementasi program ini dapat dilihat pada kode sumber 4.27 dan 4.28, dan 4.29.

```

1   import csv
2   size_hdn_lyr = 40
3   a = "double b_1[size_hdn_lyr+5] = {"
4   with open('rounded_biases_1.csv', newline='') as csvfile:
5       spamreader = csv.reader(csvfile,
6                               delimiter=',', quotechar='|')
6       for row in spamreader:

```

Kode Sumber 4.27: Program Inisiasi Array C++ (Bg.1)

```

7      x = (',').join(row))
8      a = a + x + ","
9      a = a + "};"
10     file = open("rounded_biases_1.txt", "w")
11     file.write(a)
12
13     a = "double b_2[15]={"
14     with open('rounded_biases_2.csv', newline='') as csvfile:
15         spamreader = csv.reader(csvfile,
16             delimiter=',', quotechar='|')
16         for row in spamreader:
17             x = (',').join(row))
18             a = a + x + ","
19             a = a + "};"
20     file = open("rounded_biases_2.txt", "w")
21     file.write(a)
22
23     a = "double w_1[605][size_hdn_lyr+5]={"
24     with open('rounded_weights_1.csv', newline='') as csvfile:
25         spamreader = csv.reader(csvfile,
26             delimiter=',', quotechar='|')
26         a = a + "{"
27         for row in spamreader:
28             x = (',').join(row))
29             a = a + x
30             a = a + "},," + "\n" + "{"
31             a = a + "};"
32     file = open("rounded_weights_1.txt", "w")
33     file.write(a)
34
35     a = "double w_2[size_hdn_lyr+5][15]={"
36     with open('rounded_weights_2.csv', newline='') as csvfile:
37         spamreader = csv.reader(csvfile,
38             delimiter=',', quotechar='|')

```

Kode Sumber 4.28: Program Inisiasi Array C++ (Bg.2)

```
38     a = a + "{\n"
39     for row in spamreader:
40         x = (','.join(row))
41         a = a + x
42         a = a + "}, " + "\n" + "{"
43         a = a + "}";
44 file = open("rounded_weights_2.txt", "w")
45 file.write(a)
```

Kode Sumber 4.29: Program Inisiasi Array C++ (Bg.3)

[*Halaman ini sengaja dikosongkan*]

BAB V

UJI COBA DAN EVALUASI

5.1. Lingkungan Uji Coba

Lingkungan uji coba yang dilakukan adalah sebagai berikut.

5.1.1. Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

1. Perangkat Keras
 - (a) Processor Intel® Core™ i3-4150 CPU @ 3.50GHz, 3.5GHz
 - (b) Random Access Memory 4GB
2. Perangkat Lunak
 - (a) Sistem Operasi Windows 10 Pro 64-bit
 - (b) IDE Spyder 3.2.6

5.1.2. Program Pengujian Jaringan Saraf Tiruan

1. Perangkat Keras
 - (a) Processor Intel® Core™ i3-4150 CPU @ 3.50GHz, 3.5GHz
 - (b) Random Access Memory 4GB
2. Perangkat Lunak
 - (a) Sistem Operasi Windows 10 Pro 64-bit

5.2. Uji Coba Program Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Subbab ini menjelaskan pengujian program pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan untuk penyelesaian permasalahan SPOJ Hard Image Recognition (HIR). Uji coba dilakukan untuk membuktikan program dapat berjalan sesuai yang

diharapkan dengan menerapkan algoritma yang ditawarkan.

5.2.1. Uji Coba Kebenaran Tahap Praproses

Subbab ini menguji kesesuaian algoritma metode praproses yang telah dibuat.

5.2.1.1. Data Uji Coba Kebenaran Tahap Praproses

Citra yang digunakan untuk uji coba ini 1.596 citra dari [1] dan [2] yang sudah melalui tahap persiapan dataset yang dijelaskan pada bagian 3.1. Semua citra pada dataset sudah dalam bentuk citra per angka dalam ruang warna RGB dan format PNG. Tidak semua citra memiliki ukuran yang sama. Citra memiliki format karakter berwarna hitam dan *background* berwarna putih.

5.2.1.2. Skenario Uji Coba Kebenaran Tahap Praproses

Uji coba dimulai dengan mengecek ukuran dari tiap citra. Uji kebenaran tahap praproses dilakukan dengan menghitung persentasi citra yang berhasil memiliki ukuran 30x20 dan tidak ada karakter yang terpotong. Tahap praproses dapat dikatakan berhasil jika semua ukuran citra adalah 30x20 dimana 30 adalah tinggi dan 20 adalah lebar dan tidak ada karakter yang terpotong.

5.2.1.3. Evaluasi Kebenaran Uji Coba Tahap Praproses

Evaluasi dilakukan dengan menghitung jumlah citra yang memiliki ukuran 30x20 dan tidak ada karakter yang terpotong. Dengan melihat secara manual keluaran dari tahap praproses seluruh citra, tahap praproses relatif bagus.

5.2.2. Uji Coba Kebenaran Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Subbab ini menguji kesesuaian algoritma metode pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan yang telah dibuat.

5.2.2.1. Data Uji Coba Kebenaran Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Jumlah data yang digunakan pada tahap pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan adalah berupa *array* fitur 1.596 citra dan label yang sudah melalui tahap persiapan masukan jaringan saraf tiruan pada bagian 3.2.2.

5.2.2.2. Skenario Uji Coba Kebenaran Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Tujuan dari uji coba ini adalah menguji apakah jaringan saraf tiruan dapat mengenali citra yang dilatih menggunakan *library Scikit-learn* sebelum bobot dan bias jaringan dimasukkan ke dalam program pengujian JST dan dikirim ke situs penilaian daring SPOJ. Apabila jaringan belum bisa mengenali citra yang dilatih pada tahap pelatihan, kemungkinan besar jaringan tidak akan bisa menge-nali citra pengujian yang ada pada SPOJ Hard Image Recognition (HIR). Uji kebenaran tahap pemanggilan fungsi pembuatan dan pe-latihan jaringan saraf tiruan dilakukan dengan melihat akurasi yang didapat dari hasil pengujian jaringan saraf tiruan dengan citra yang dilatih.

Tahap pemanggilan fungsi pembuatan dan pelatihan jaringan saraf tiruan menggunakan model MLP yang terdiri dari layer masukan, layer tersembunyi, layer luaran, fungsi aktivasi, fungsi optimasi, dan *learning rate*. Parameter lain menggunakan nilai *default* dari fungsi *library*. Jumlah *node* layer masukan adalah 600 sesuai dengan jumlah piksel pada citra, jumlah layer tersembunyi adalah 1,

sedangkan jumlah *node* layer luaran adalah 10 sesuai dengan jumlah label yang tersedia. Kemudian dilakukan uji coba dengan nilai parameter yang dapat dilihat pada tabel 3.5. Pengujian jaringan saraf tiruan dengan citra yang dilatih juga menggunakan *library Scikit-learn*. Dataset yang digunakan sejumlah 1.596 citra.

5.2.2.3. Evaluasi Kebenaran Uji Coba Tahap Pemanggilan Fungsi Pembuatan dan Pelatihan Jaringan Saraf Tiruan

Evaluasi dilakukan dengan melihat akurasi yang dihasilkan saat melakukan pengujian jaringan dengan citra yang dilatih menggunakan parameter yang dijelaskan pada bagian 5.2.2.2 dan *epoch* pelatihan jaringan. Uji coba kebenaran pertama adalah menentukan *learning rate*. Nilai komponen lain yang digunakan adalah fungsi aktivasi *sigmoid*, jumlah *epoch* 12.000, fungsi optimasi *sgd*, dan jumlah *node* pada 1 layer tersembunyi adalah 40. Perubahan *learning rate* tidak menyebabkan perubahan yang signifikan terhadap akurasi pengujian jaringan menggunakan citra yang dilatih. Hasil pengujian nilai parameter *learning rate* dapat dilihat pada tabel 5.1.

Tabel 5.1: Uji Coba *Learning Rate* dengan Citra Pelatihan

Learning Rate	Akurasi(%)
0,005	99,93
0,001	99,93
0,05	100
0,01	100
0,5	100

Uji coba kedua dilakukan terhadap fungsi aktivasi dengan *learning rate* 0,01. Selain *identity*, penggunaan fungsi aktivasi lain tidak menyebabkan perubahan terhadap akurasi pengujian jaringan menggunakan citra yang dilatih. Hasil pengujian fungsi aktivasi

dapat dilihat pada tabel 5.2.

Tabel 5.2: Uji Coba Parameter Fungsi Aktivasi dengan Citra Pelatihan

Fungsi Aktivasi	Akurasi (%)
Identity	100
Sigmoid	100
Tanh	100
ReLu	100

Uji coba ketiga dilakukan terhadap jumlah *epoch* pelatihan jaringan saraf tiruan dengan *learning rate* 0,01 dan fungsi aktivasi *sigmoid*. Perubahan jumlah *epoch* juga tidak menyebabkan perubahan terhadap akurasi pengujian jaringan menggunakan citra yang dilatih. Hasil pengujian jumlah *epoch* dapat dilihat pada tabel 5.3.

Tabel 5.3: Uji Coba Parameter *Epoch* dengan Citra Pelatihan

Epoch	Akurasi (%)
11.000	100
11.500	100
12.000	100
12.500	100
13.000	100

Uji coba keempat dilakukan terhadap *solver* / fungsi optimasi dengan *learning rate* 0,01, fungsi aktivasi *sigmoid*, dan jumlah *epoch* 12.000. Perubahan fungsi optimasi juga tidak menyebabkan perubahan terhadap akurasi pengujian jaringan menggunakan citra yang dilatih. Hasil pengujian fungsi optimasi dapat dilihat pada tabel 5.4.

Tabel 5.4: Uji Coba Parameter Fungsi Optimasi dengan Citra Pelatihan

Fungsi Optimasi	Akurasi (%)
adam	100
lbfgs	100
sgd	100

Uji coba kelima dilakukan terhadap jumlah *node* pada 1 layer tersembunyi dengan *learning rate* 0.01, fungsi aktivasi *sigmoid*, jumlah *epoch* 12.000, dan fungsi optimasi *sgd*. Perubahan jumlah *node* pada 1 layer tersembunyi juga tidak menyebabkan perubahan terhadap akurasi pengujian jaringan menggunakan citra yang dilatih. Hasil pengujian jumlah *node* layer tersembunyi dapat dilihat pada tabel 5.5.

Tabel 5.5: Uji Coba Parameter Jumlah *Node* dengan Citra Pelatihan

Jumlah Node	Akurasi (%)
20	100
25	100
30	100
35	100
40	100

Nilai akurasi tertinggi secara keseluruhan adalah 100%. Sehingga dapat disimpulkan bahwa jaringan merupakan jaringan yang optimal dalam memprediksi angka pada citra pelatihan. Namun, karena banyak perubahan parameter yang tidak berpengaruh secara signifikan terhadap akurasi pengujian jaringan menggunakan citra yang dilatih, maka pengambilan keputusan untuk penggunaan nilai parameter akan dilakukan setelah melakukan uji coba pada situs daring SPOJ dengan mengirim program pengujian jaringan JST pada

permasalahan SPOJ Hard Image Recognition (HIR).

5.3. Uji Coba Program Pengujian Jaringan Saraf Tiruan

Subbab ini menjelaskan pengujian program pengujian JST untuk penyelesaian permasalahan SPOJ Hard Image Recognition (HIR). Uji coba dilakukan untuk mendapatkan nilai tertinggi pada situs penilaian daring SPOJ dengan cara mengirim program ke [1]. Uji coba dilakukan terhadap parameter jaringan saraf tiruan yang sama seperti pada bagian 5.2.2.2 dan *epoch* pelatihan jaringan. Selain itu, *threshold* untuk *filtering* pada program pengujian JST yang telah dijelaskan pada bagian 3.3.3 juga menjadi komponen uji coba.

5.3.1. Uji Coba Kebenaran Lokal

Uji coba lokal dilakukan dengan menjalankan program hanya untuk menguji apakah program dapat berjalan atau tidak ada *error* dan dapat mengeluarkan 6 angka yang merupakan hasil prediksi dari citra.

5.3.1.1. Data Uji Coba Kebenaran Lokal

Data yang digunakan untuk uji coba kebenaran lokal program pengujian JST adalah citra yang didapat dari [2]. Terdapat 13 berkas TXT yang masing-masing berisi kumpulan karakter 'X' dan '.' yang merepresentasikan 13 variasi citra berisi 6 angka.

5.3.1.2. Skenario Uji Coba Kebenaran Lokal

Uji coba dilakukan dengan menjalankan program dan melihat apakah program dapat berjalan atau tidak ada *error* dan dapat mengeluarkan 6 angka hasil prediksi dari citra masukan program. Komponen dan nilai yang digunakan pada uji coba kebenaran lokal program pengujian JST dapat dilihat pada tabel 5.6.

Tabel 5.6: Komponen dan Nilai Uji Coba Kebenaran Lokal Program Pengujian JST

No	Nama Komponen	Nilai
1	<i>Learning rate</i> jaringan	0,01
2	Fungsi aktivasi jaringan	sigmoid
3	Jumlah epoch pelatihan jaringan	12.000
4	Fungsi Optimasi <i>solver</i> jaringan	adam
5	Jumlah <i>node</i> pada 1 layer tersembunyi jaringan	40
6	<i>Threshold</i> untuk <i>filtering</i>	4

5.3.1.3. Evaluasi Kebenaran Uji Coba Lokal

Evaluasi dilakukan dengan melihat nilai yang didapat dari hasil prediksi jaringan pada program pengujian JST secara lokal. Sama seperti pada situs penilaian daring SPOJ permasalahan Hard Image Recognition[1], nilai akan senilai dengan jumlah citra yang benar terdeteksi oleh program. Dengan parameter yang ada pada tabel 5.6, nilai yang didapat adalah 8 atau citra yang hasil prediksinya benar berjumlah 8 citra dari 13 contoh citra masukan pada [1] dan [2].

5.3.2. Uji Coba Kebenaran Situs Penilaian Daring SPOJ

Uji coba pada situs penilaian daring SPOJ dilakukan dengan mengirim program pengujian JST ke situs SPOJ Hard Image Recognition (HIR)[1].

5.3.2.1. Data Uji Coba Kebenaran Situs Penilaian Daring SPOJ

Data yang digunakan pada uji coba ini adalah data yang ada pada situs SPOJ Hard Image Recognition (HIR)[1] sehingga citra-citra yang digunakan untuk pengujian jaringan tidak dapat diketahui.

5.3.2.2. Skenario Uji Coba Kebenaran Situs Penilaian Daring SPOJ

Uji coba dilakukan dengan mengubah-ubah nilai dari komponen yang ada pada tabel 5.6 dan mengirim program pengujian JST yang dijelaskan pada bagian 4.3 ke situs SPOJ Hard Image Recognition (HIR)[1]. Nilai dari komponen yang diambil adalah yang mendapatkan nilai tertinggi.

5.3.2.3. Evaluasi Uji Coba Kebenaran Situs Penilaian Daring SPOJ

Evaluasi dilakukan dengan melihat nilai tertinggi yang didapatkan pada situs SPOJ Hard Image Recognition (HIR)[1]. Uji coba pertama adalah menentukan nilai *learning rate*. Dari hasil uji coba, *learning rate* 0,005; 0,001; dan 0,01 mendapatkan nilai tertinggi pada situs penilaian daring SPOJ yaitu 23. Hasil pengujian nilai parameter *learning rate* dapat dilihat pada tabel 5.7.

Tabel 5.7: Uji Coba *Learning Rate* pada SPOJ

<i>Learning Rate</i>	Nilai SPOJ
0,005	23
0,001	23
0,05	22
0,01	23
0,5	19

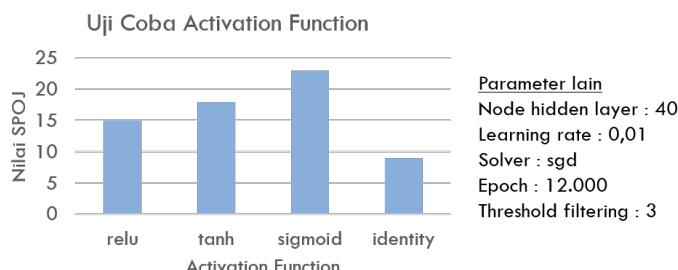
Uji coba kedua dilakukan terhadap fungsi aktivasi. Terdapat 4 fungsi aktivasi yang disediakan oleh *library Scikit-learn* fungsi *MLPClassifier*. Dari hasil uji coba, fungsi aktivasi *sigmoid* mendapatkan nilai tertinggi pada situs penilaian daring SPOJ yaitu 23. Hasil pengujian nilai parameter fungsi aktivasi dapat dilihat pada tabel 5.8.



Gambar 5.1: Grafik Nilai SPOJ untuk Uji Coba *Learning Rate*

Tabel 5.8: Uji Coba Fungsi Aktivasi pada SPOJ

Fungsi Aktivasi	Nilai SPOJ
ReLU	15
Tanh	18
Sigmoid	23
Identity	9



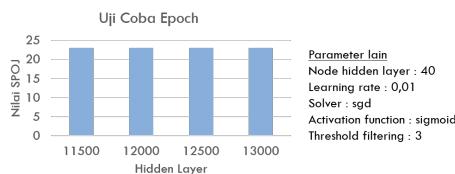
Gambar 5.2: Grafik Nilai SPOJ untuk Uji Coba Fungsi Aktivasi

Uji coba ketiga dilakukan terhadap jumlah *epoch* pelatihan jaringan saraf tiruan. Dari hasil uji coba, nilai tertinggi yang didapat pada situs penilaian daring SPOJ adalah 23. Perubahan jumlah *epoch* menggunakan parameter yang ada pada gambar 5.3 tidak menyebabkan perubahan nilai pada SPOJ. Hasil pengujian jumlah

epoch dapat dilihat pada tabel 5.9.

Tabel 5.9: Uji Coba Jumlah *Epoch* pada SPOJ

Epoch	Nilai SPOJ
11.000	23
11.500	23
12.000	23
12.500	23
13.000	23



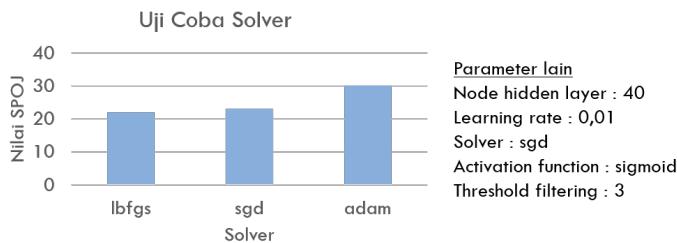
Gambar 5.3: Grafik Nilai SPOJ untuk Uji Coba Jumlah *Epoch*

Uji coba keempat dilakukan terhadap fungsi optimasi. Terdapat 3 fungsi optimasi yang disediakan oleh *library Scikit-learn* fungsi *MLPClassifier*. Dari hasil uji coba, fungsi optimasi *adam* mendapatkan nilai tertinggi pada situs penilaian daring SPOJ yaitu 30. Hasil pengujian fungsi optimasi dapat dilihat pada tabel 5.10.

Tabel 5.10: Uji Coba Fungsi Optimasi pada SPOJ

Fungsi Optimasi	Nilai SPOJ
lbfgs	22
sgd	23
adam	30

Uji coba kelima dilakukan terhadap jumlah *node* pada 1 layer tersembunyi. Dari hasil uji coba, jumlah *node* 40 pada 1 layer

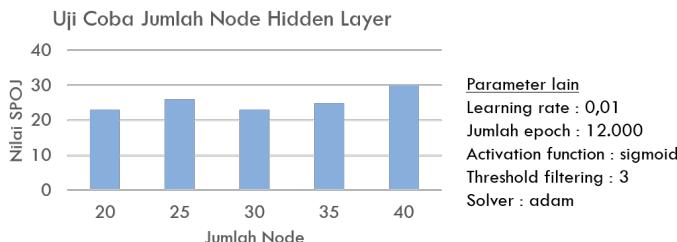


Gambar 5.4: Grafik Nilai SPOJ untuk Uji Coba Fungsi Optimasi

tersembunyi mendapatkan nilai tertinggi pada situs penilaian daring SPOJ yaitu 30. Hasil pengujian jumlah *node* pada 1 layer tersembunyi dapat dilihat pada tabel 5.11.

Tabel 5.11: Uji Coba Jumlah *Node* pada SPOJ

Jumlah <i>Node</i>	Nilai SPOJ
20	23
25	26
30	23
35	25
40	30

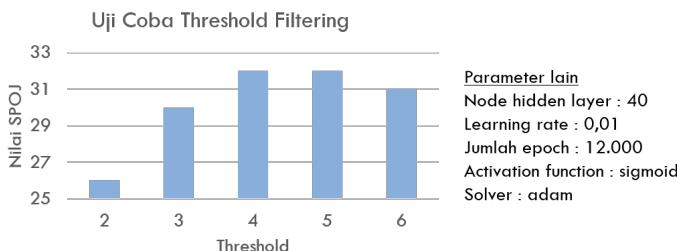


Gambar 5.5: Grafik Nilai SPOJ untuk Uji Coba Jumlah *Node*

Uji coba keenam dilakukan terhadap *threshold* pada tahap *filtering* atau menghilangkan derau kecil yang telah dijelaskan pada bagian 3.3.3. Dari hasil uji coba, *threshold* 4 dan 5 mendapatkan nilai tertinggi pada situs penilaian daring SPOJ yaitu 32. Hasil pengujian *threshold* dapat dilihat pada tabel 5.12.

Tabel 5.12: Uji Coba *Threshold* pada SPOJ

Threshold	Nilai SPOJ
2	26
3	30
4	32
5	32
6	31



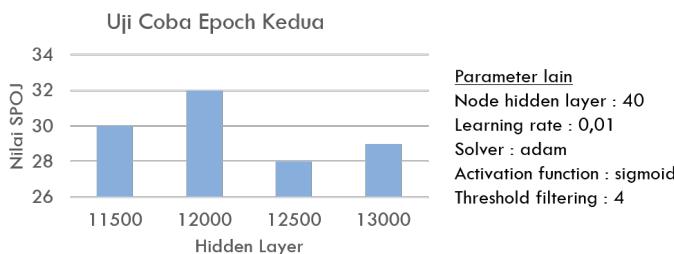
Gambar 5.6: Grafik Nilai SPOJ untuk Uji Coba *Threshold*

Dari hasil uji coba di atas, nilai tertinggi yang berhasil didapat pada situs penilaian daring SPOJ adalah 32. Namun perlu dilakukan uji coba ulang menggunakan fungsi optimasi *adam* dan *threshold filtering* 4 untuk setiap jumlah *epoch*. Karena fungsi optimasi *adam* dan *threshold filtering* 4 mendapatkan nilai tertinggi pada masing-masing uji coba, dan nilai SPOJ tidak berubah saat menggunakan fungsi optimasi *sgd* dan *threshold filtering* 3 pada uji coba jumlah *epoch*. Uji coba untuk parameter jumlah *epoch* yang kedua dilakukan.

kukan dengan menggunakan nilai parameter lain yang dapat dilihat pada gambar 5.7. Hasil pengujian jumlah *epoch* kedua dapat dilihat pada tabel 5.13.

Tabel 5.13: Uji Coba Kedua Jumlah *Epoch* pada SPOJ

<i>Epoch</i>	Nilai pada SPOJ
11.500	30
12.000	32
12.500	23
13.000	23



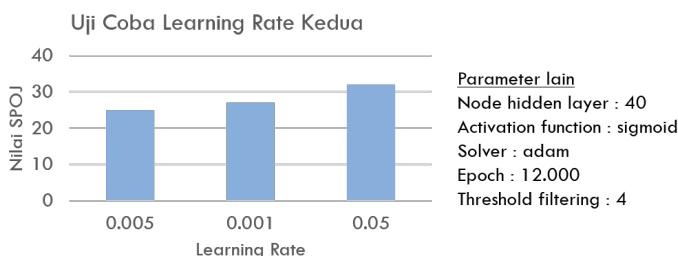
Gambar 5.7: Grafik Nilai SPOJ untuk Uji Coba Kedua Jumlah *Epoch*

Dari uji coba kedua untuk jumlah *epoch* yang sudah dilakukan di atas, nilai tertinggi yang didapat pada situs penilaian daring SPOJ Hard Image Recognition (HIR) adalah tetap 32. Namun, perlu dilakukan uji coba kedua terhadap *learning rate* karena *learning rate* 0,005; 0,001; dan 0,01 pada uji coba sebelumnya mendapatkan nilai yang sama yaitu 23. Uji coba untuk parameter *learning rate* yang kedua dilakukan dengan menggunakan nilai parameter lain yang mendapatkan nilai tertinggi pada masing-masing uji coba. Parameter tersebut yang dapat dilihat pada gambar 5.8. Hasil

pengujian jumlah *epoch* kedua dapat dilihat pada tabel 5.14.

Tabel 5.14: Uji Coba Kedua Jumlah *Epoch* pada SPOJ

Learning Rate	Nilai pada SPOJ
0,005	25
0,001	27
0,01	32



Gambar 5.8: Grafik Nilai SPOJ untuk Uji Coba Kedua *Learning Rate*

Dari uji coba kedua untuk *learning rate* yang sudah dilakukan di atas, nilai tertinggi yang didapat pada situs penilaian daring SPOJ Hard Image Recognition (HIR) adalah tetap 32. Komponen uji coba yang digunakan dan nilainya yang mendapatkan nilai SPOJ tertinggi dapat dilihat pada tabel 5.15. Nilai dan peringkat pada SPOJ dapat dilihat pada gambar 5.9 dan 5.10.

Tabel 5.15: Komponen Uji Coba dengan Nilai SPOJ Tertinggi

No	Nama Komponen	Nilai
1	<i>Learning rate</i> jaringan	0,01
2	Fungsi aktivasi jaringan	sigmoid
3	Jumlah <i>epoch</i> pelatihan jaringan	12.000
4	Fungsi Optimasi <i>solver</i> jaringan	adam
5	Jumlah <i>node</i> pada 1 layer tersembunyi jaringan	40
6	<i>Threshold</i> untuk <i>filtering</i>	4

22940892	■	2018-12-25 11:09:18	Hard Image Recognition	32	edit ideone it	0.07	16M	CPP14
----------	---	------------------------	---------------------------	----	-------------------	------	-----	-------

Gambar 5.9: Hasil Pengiriman Program Pengujian JST ke Situs Penilaian Daring SPOJ Hard Image Recognition (HIR)[1]

Hard Image Recognition statistics & best solutions						
Users accepted	Submissions	Accepted	Wrong Answer	Compile Error	Runtime Error	Time Limit Exceeded
14	219	159	0	10	8	42

Gambar 5.10: Peringkat Nilai pada Situs Penilaian Daring SPOJ Hard Image Recognition (HIR)[1]

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.

6.1. Kesimpulan

Berdasarkan penjabaran di bab-bab sebelumnya, dapat disimpulkan beberapa poin terkait penyelesaian permasalahan SPOJ Hard Image Recognition (HIR).

1. Algoritma untuk segmentasi *foreground* pada citra masukan SPOJ merupakan algoritma yang optimal.
2. Jaringan saraf tiruan yang dibuat merupakan jaringan yang optimal dalam pengujian terhadap dataset citra pelatihan, karena jaringan berhasil mencapai 100% dalam pengujian terhadap dataset citra pelatihan yang dipersiapkan.
3. Sistem yang diajukan secara lokal berhasil memprediksi 8 dari 13 contoh citra masukan dari [1] dan [2] dengan benar.
4. Sistem yang diajukan berhasil memprediksi 32 citra masukan dengan benar pada situs penilaian daring SPOJ Hard Image Recognition (HIR) dengan beberapa batasan dari situs, terutama batas ukuran kode sumber yang dikirim.
5. Sistem yang diajukan mendapatkan nilai tertinggi pada situs penilaian daring SPOJ Hard Image Recognition (HIR)yaitu 32 dengan waktu yang diperlukan adalah 0.07 detik, penggunaan memori adalah 16 MB, dan ukuran kode sumber yang dikirim adalah 139.264 B.
6. Sistem yang mendapatkan nilai tertinggi menggunakan dataset yang terdiri dari 1.569 citra dan memiliki komponen arsi-

tekstur jumlah *node* layer masukan 600, jumlah *node* layer tersembunyi 40, jumlah *epoch* 12.000, *learning rate* 0,01, fungsi aktivasi *sigmoid*, fungsi optimasi *adam*, dan *threshold filtering* 4.

7. Sistem yang diajukan belum bisa memprediksi citra masukan dengan jumlah angka yang tersambung menjadi satu lebih dari 2.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan uji coba yang telah dilakukan.

1. Walaupun nilai yang didapat adalah nilai tertinggi pada sistem penilaian daring SPOJ, tidak menutup kemungkinan penerapan metode atau perspektif lain dapat mendapatkan nilai yang lebih tinggi.
2. Usaha meningkatkan nilai yang didapat pada sistem penilaian daring SPOJ dapat dilakukan dengan penambahan variasi pada citra pelatihan jaringan saraf tiruan.
3. Melakukan pemisahan pada angka yang tersambung menjadi satu lebih dari 2 pada citra masukan program C++.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

- [1] () Hard Image Recognition, [Online]. Available: <https://www.spoj.com/problems/HIR>.
- [2] () Hard Image Recognition Open Contest 2006, [Online]. Available: <http://discuss.spoj.com/t/hard-image-recognition/663>.
- [3] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995, ISBN: 0198538642.
- [4] () Konsep Depth First Search (DFS), [Online]. Available: <https://indonesia.hackerearth.com/konsep-depth-first-search-dfs/>.
- [5] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*. Jan. 1995, ISBN: 978-0-07-032018-5.
- [6] () Jaringan Saraf Tiruan. version 12201279, Wikipedia, Ensiklopedia Bebas, [Online]. Available: https://id.wikipedia.org/w/index.php?title=Jaringan_saraf_tiruan&oldid=12201279.
- [7] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997, ISBN: 0070428077, 9780070428072.
- [8] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.
- [9] Wikipedia contributors, *Activation function — Wikipedia, The Free Encyclopedia*, [Online; accessed 21-December-2018], 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Activation_function&oldid=874013957.

- [10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [11] (). Activation Functions: Neural Networks, [Online]. Available: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>.
- [12] Wikipedia contributors, *Limited-memory BFGS — Wikipedia, The Free Encyclopedia*, [Online; accessed 27-December-2018], 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Limited-memory_BFGS&oldid=873634498.
- [13] ——, *Stochastic gradient descent — Wikipedia, The Free Encyclopedia*, [Online; accessed 27-December-2018], 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stochastic_gradient_descent&oldid=874506665.
- [14] (). Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent, [Online]. Available: <https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>.
- [15] K. T. Islam, G. Mujtaba, R. G. Raj, and H. F. Nweke, “Handwritten digits recognition with artificial neural network”, in *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*, 2017, pp. 1–4. doi: 10.1109/ICE2T.2017.8215993.
- [16] Wikipedia contributors, *MNIST database — Wikipedia, The Free Encyclopedia*, [Online; accessed 15-December-2018], 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=MNIST_database&oldid=869917056.

LAMPIRAN A: Gambar Pelatihan Jaringan Saraf Tiruan

Tabel 6.1: Tabel Contoh Citra Pelatihan JST

No	Kelas	Normal	Scaling
1	0	0	0
2	1	1	1
3	2	2	2
4	3	3	3
5	4	4	4
6	5	5	5
7	6	6	6
8	7	7	7
9	8	8	8
10	9	9	9

Tabel 6.2: Tabel Jumlah Citra Pelatihan JST

No	Kelas	Normal	Scaling	Total
1	0	7	140	147
2	1	9	180	189
3	2	5	100	105
4	3	7	140	147
5	4	7	140	147
6	5	4	80	84
7	6	10	200	210
8	7	6	120	126
9	8	12	240	252
10	9	9	180	189
11	Total	76	1520	1596

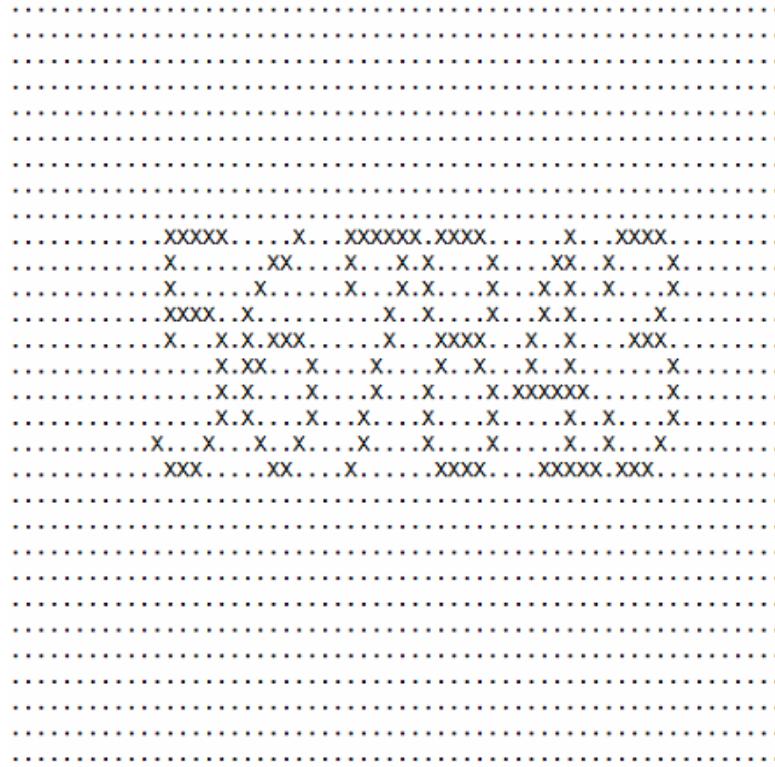
LAMPIRAN B: Hasil Prediksi Uji Coba Lokal Program C++

20 40

000302

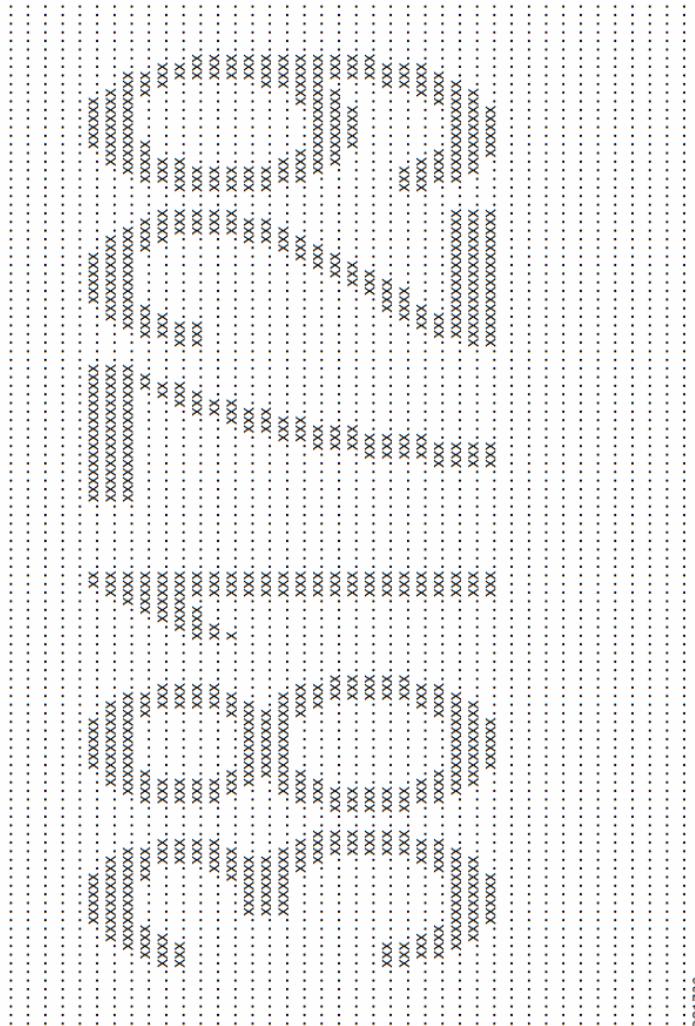
Gambar 6.1: Hasil Prediksi Citra 1 (Salah)

30 60

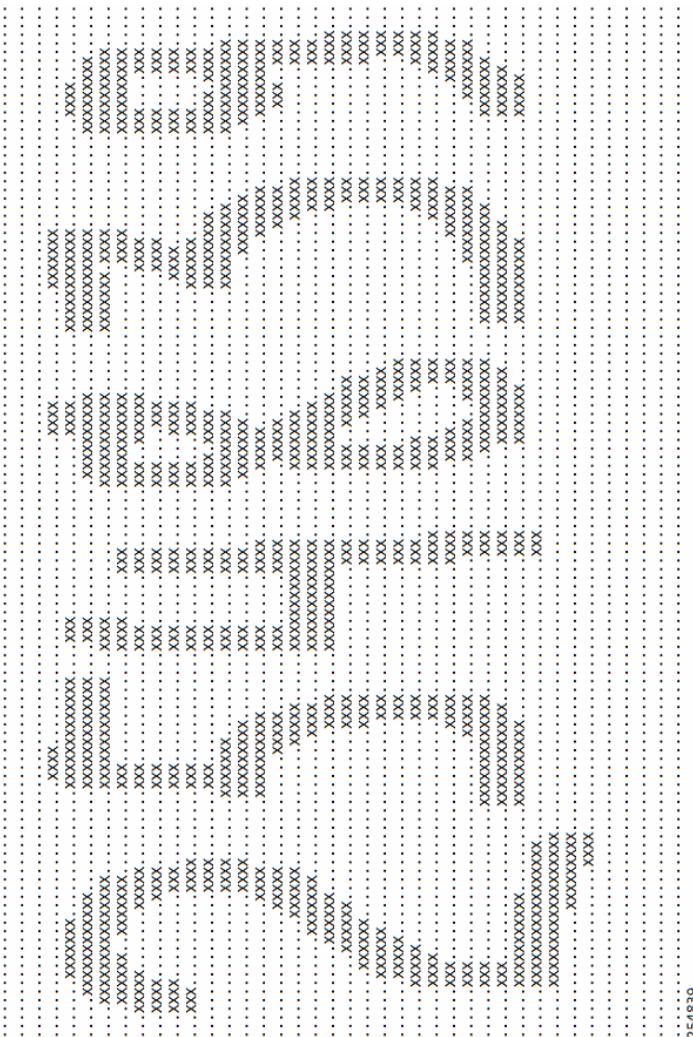


547443

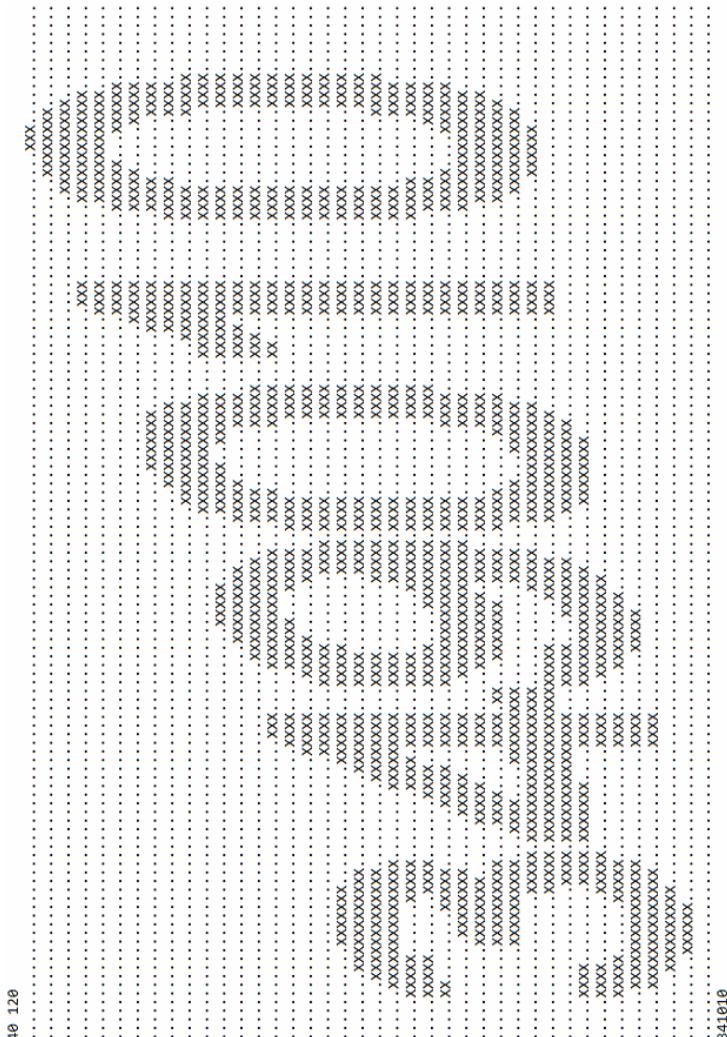
Gambar 6.2: Hasil Prediksi Citra 2 (Salah)



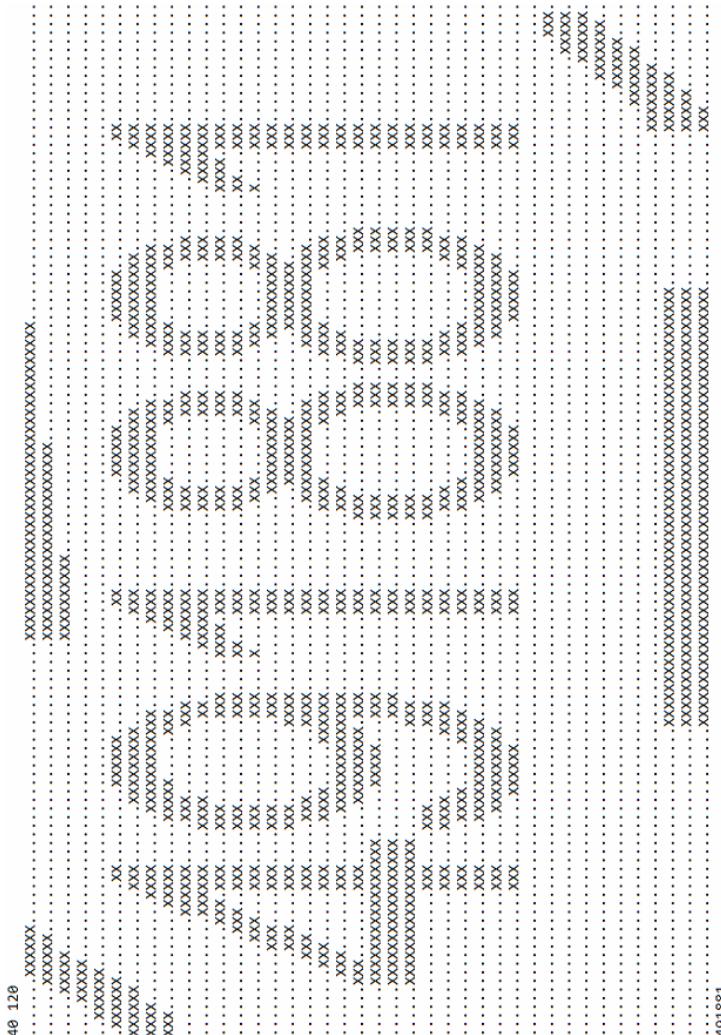
Gambar 6.3: Hasil Prediksi Citra 3 (Benar)



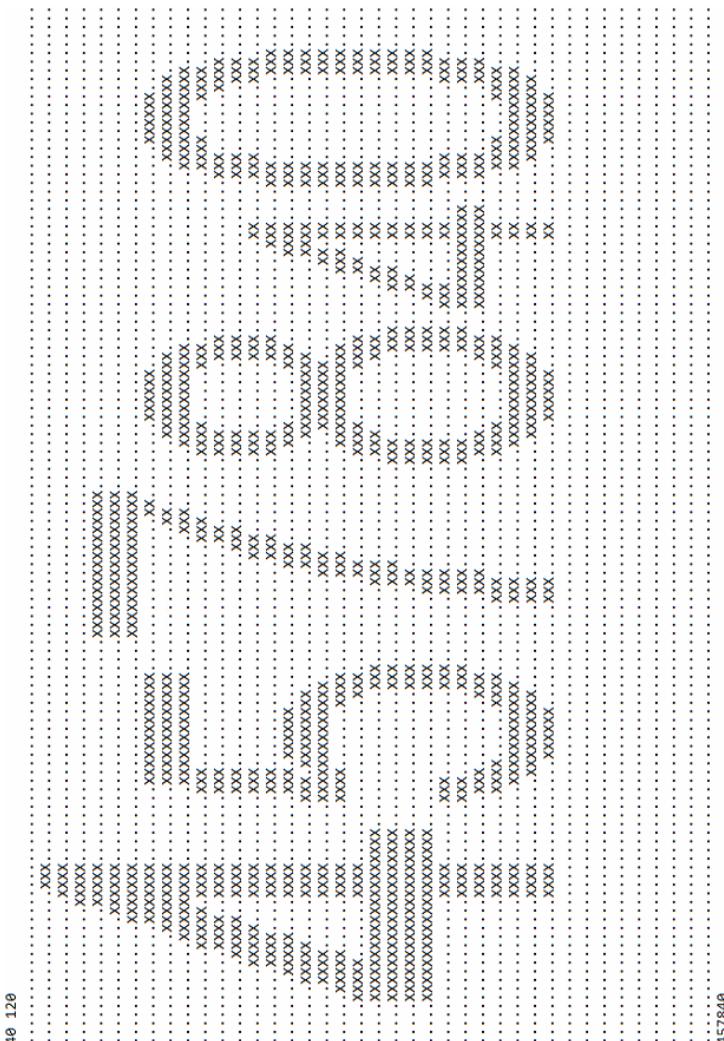
Gambar 6.4: Hasil Prediksi Citra 4 (Benar)



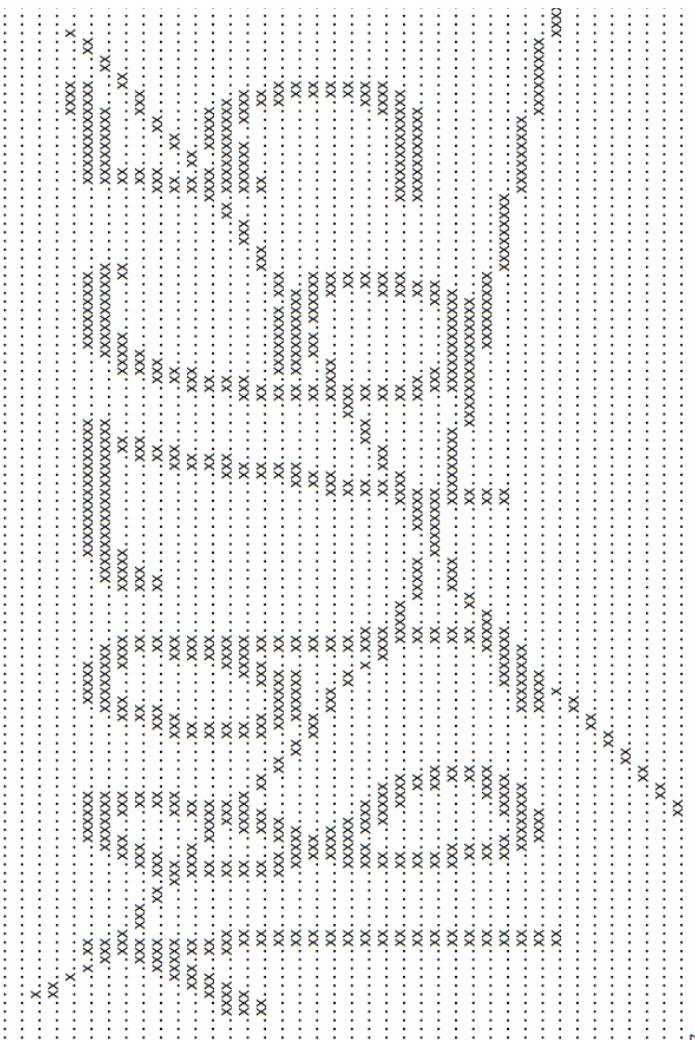
Gambar 6.5: Hasil Prediksi Citra 5 (Salah)



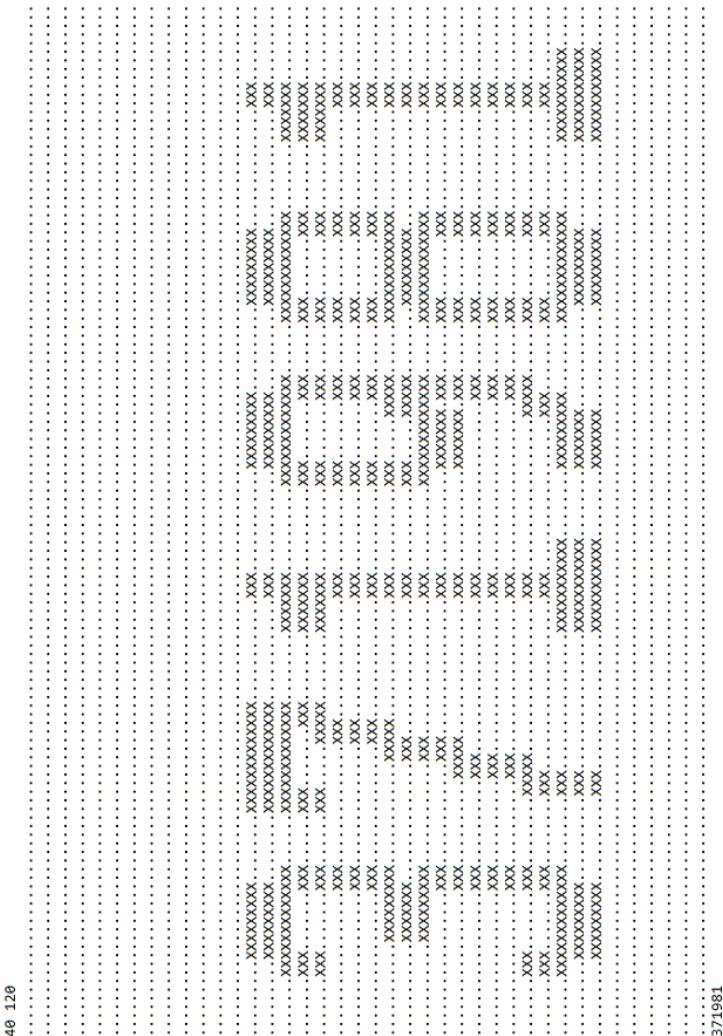
Gambar 6.6: Hasil Prediksi Citra 6 (Benar)



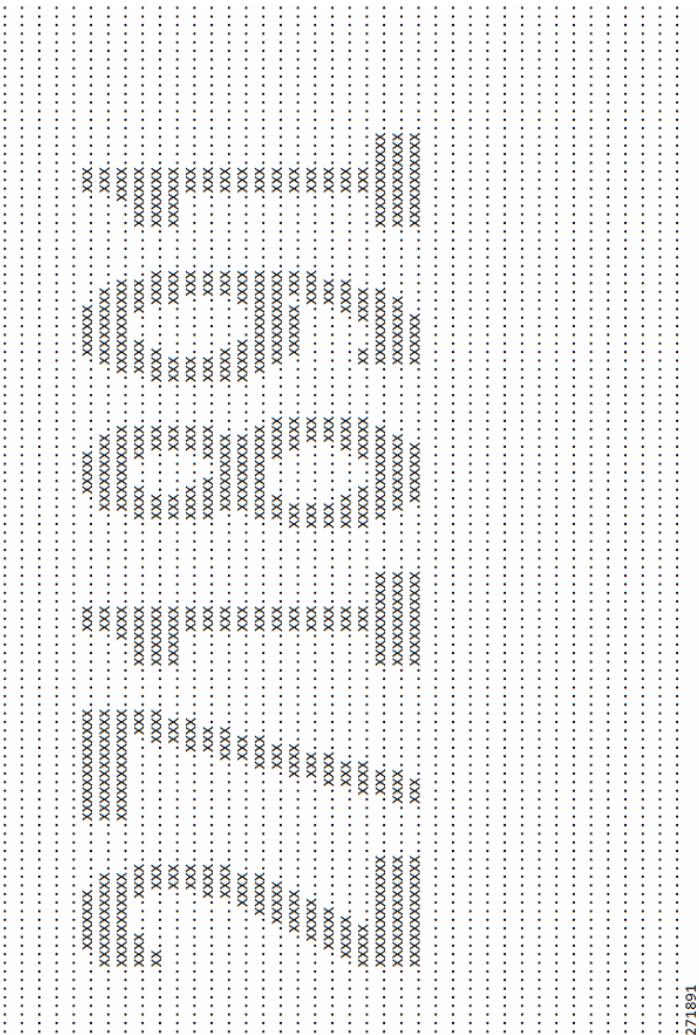
Gambar 6.7: Hasil Prediksi Citra 7 (Benar)



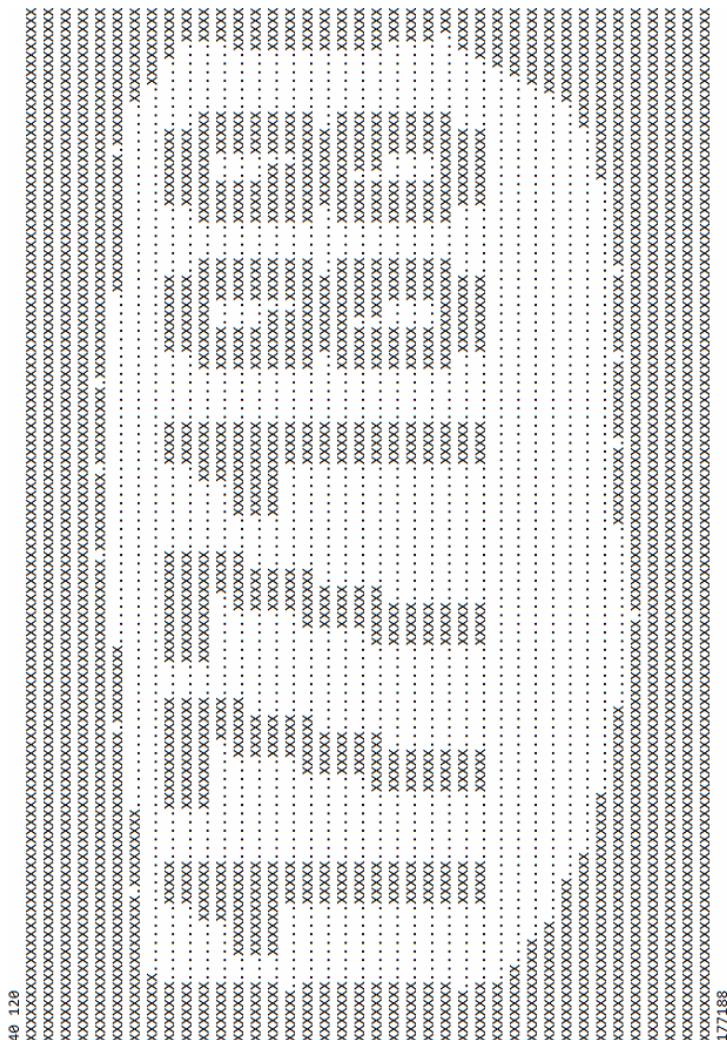
Gambar 6.8: Hasil Prediksi Citra 8 (Salah)



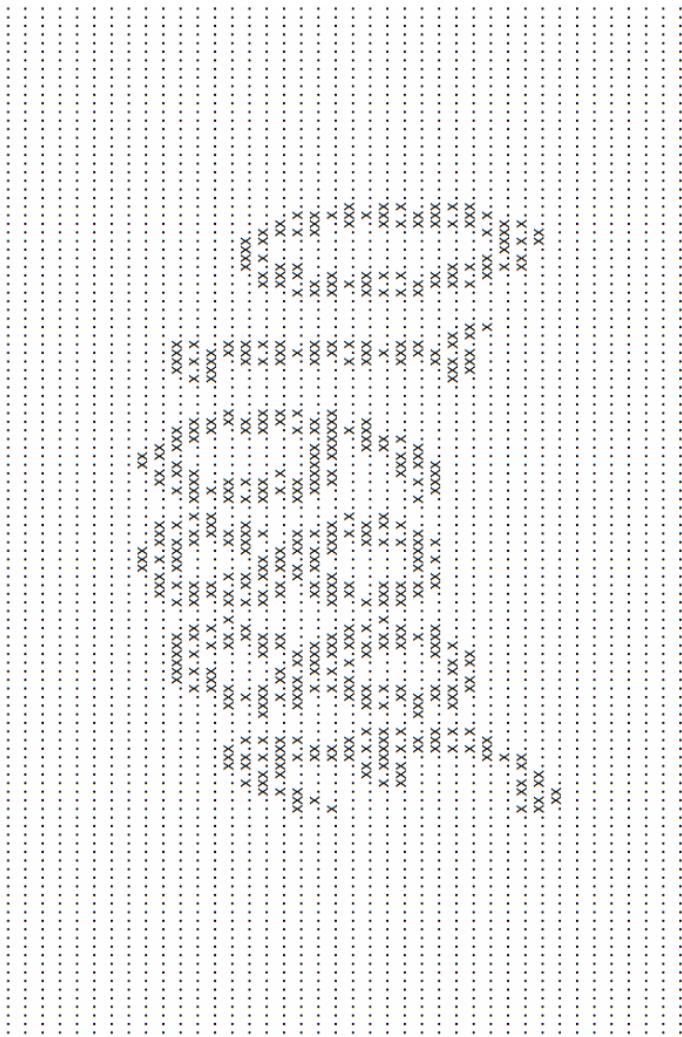
Gambar 6.9: Hasil Prediksi Citra 9 (Benar)



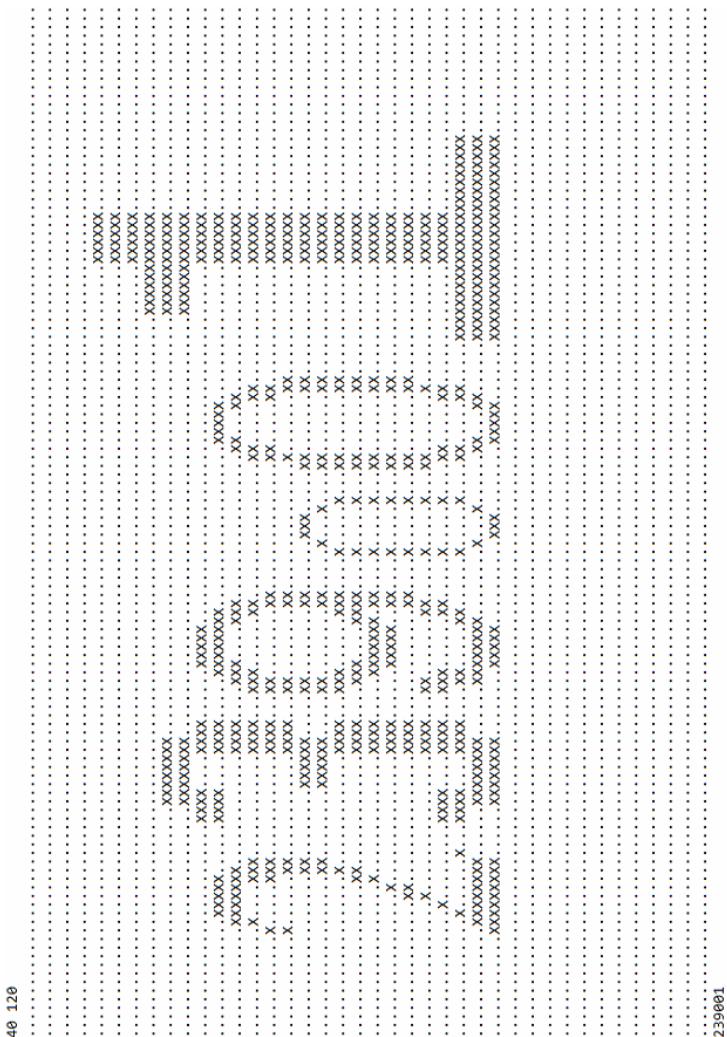
Gambar 6.10: Hasil Prediksi Citra 10 (Benar)



Gambar 6.11: Hasil Prediksi Citra 11 (Benar)



Gambar 6.12: Hasil Prediksi Citra 12 (Salah)



Gambar 6.13: Hasil Prediksi Citra 13 (Benar)

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS



Penulis bernama Cynthia Dewi Tejakusuma, putri keempat dari empat bersaudara yang lahir di Surabaya pada tanggal 19 Juli 1997. Penulis telah menjalani masa pendidikan di Sekolah Dasar Santa Clara Surabaya pada tahun 2003 hingga 2009, Sekolah Menengah Pertama Santa Clara Surabaya pada tahun 2009 hingga 2012, dan Sekolah Menengah Atas St. Louis 1 Surabaya pada tahun 2012 hingga 2015. Pada masa penulisan, penulis sedang menempuh masa studi S1 tahun ketiga di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember.

Selama masa studi S1, penulis memiliki ketertarikan yang dalam mengenai *Artificial Intelligence*, basis data, dan rancang bangun aplikasi *website*. Penulis pernah menjadi asisten dosen pada mata kuliah Sistem Digital, Komputasi Numerik, dan Sistem Basis Data. Dalam mengembangkan kemampuan yang dimiliki, penulis pernah menerima beberapa proyek untuk rancang bangun aplikasi *website*. Selain itu, penulis memiliki pengalaman magang di PT Artajasa Pembayaran Elektronis.

Di luar kesibukan akademik, penulis cukup aktif dalam organisasi baik dari dalam maupun luar jurusan yaitu sebagai staff dari salah satu departemen yang ada. Penulis juga berkontribusi dalam kepanitiaan dalam jurusan yaitu sebagai bendahara selama 2 tahun. Selain itu, penulis pernah membantu kegiatan pelatihan nasional bagi peserta Olimpiade Komputer Indonesia pada tahun 2018 dan 2019. Penulis dapat dihubungi melalui surel di cynthiacyndewi@gmail.com.