MOMENTUM *BACKPROPAGATION* UNTUK KLASIFIKASI FUNGSI SENYAWA AKTIF BERDASARKAN NOTASI SMILES (*SIMPLIFIED MOLECULAR INPUT LINE ENTRY SYSTEM*)

SKRIPSI

Untuk memenuhi sebagai persyaratan
Memperoleh gelar Sarjana Komputer

Disusun oleh:
Nyimas Ayu Widi Indriana
NIM: 145150207111115

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

MOMENTUM *BACKPROPAGATION* UNTUK KLASIFIKASI FUNGSI SENYAWA AKTIF BERDASARKAN NOTASI SMILES (*SIMPLIFIED MOLECULAR INPUT LINE ENTRY SYSTEM*)

SKRIPSI

Untuk memenuhi sebagai presyaratan
Memperoleh gelar Sarjana Komputer

Disusun oleh:
Nyimas Ayu Widi Indriana
NIM: 145150207111115

Skripsi ini telah diuji dan dinyatakan lulus pada
07 Desember 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dian Eka Ratnawati, S.Si, M.Kom	Syaiful Anam, S.Si.,M.T.,Ph.D
NIP. 19730619 200212 2 001	NIP. 19780115 200212 1 003

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat2 dan Pasal 70).

Malang, 07 Desember 2018

Nyimas Ayu Widi Indriana

NIM. 145150207111115

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang selalu melimpahkan rahmat dan karunianya kepada penulis sehingga penulis dapat menyelesaikan skripsi dengan judul "MOMENTUM *BACKPROPAGATIO*N UNTUK KLASIFIKASI FUNGSI SENYAWA AKTIF BERDASARKAN NOTASI SMILES (*SIMPLIFIED MOLECULAR INPUT LINE ENTRY SYSTEM*)" sebagai syarat dalam memperoleh gelar sarjana pada Program Studi Teknik Informatika, Fakultas Ilmu Komuter, Universitas Brawijaya.

Dalam proses penyusunan skripsi penulis mendapatkan bantuan berupa moral maupun materil dari banyak pihak. Maka dari itu, penulis ingin mengucapkan rasa terima kasih sebanyak-banyaknya kepada:

1. Ibu Dian Eka Ratnawati, S.Si, M.Kom selaku dosen pembimbing I yang telah memberikan arahan, masukan ilmu dan bimbingan sehingga penulis dapat menyelesaikan skripsi ini.
2. Bapak Syaiful Anam, S.Si, MT., Ph.D selaku dosen pembimbing II yang juga turut memberikan arahan, masukan dan bimbingan sehingga penulis dapat menyelesaikan skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
4. Bapak Agus Widodo, S.T, M.Sc selaku Ketua Program Studi Teknik Informatika.
5. Kedua orang tua tercinta dan saudara-saudara penulis yang selalu mendoakan, memberikan dukungan moril mau pun materil sampai skripsi ini terselesaikan.
6. Teman-teman seperjuangan dalam menyusun skripsi Suhhy Ramzini, Muhammad Iskandar A.R, R. Rizky Widdie, Nul Khilmiyatul, Sherly Witanto dan Yunita Dwi Afianti yang telah saling membantu dan berdiskusi dalam penyelesaian skripsi ini.
7. Teman-teman Paguyuban KSE UB yang telah memberikan semangat, motivasi, dan saran sehingga skripsi ini dapat terselesaikan.
8. Teman-teman Informatika FILKOM UB 2014 yang telah memberikan kesan dan pesan selama penulis menempuh perkuliahan di Fakultas Ilmu Komputer Universitas Brawijaya.
9. Seluruh bapak ibu dosen serta staff Fakultas Ilmu Komputer atas segala bantuan dan ilmu yang diajarkan selama penulis menempuh studi di Fakultas Ilmu Komputer.

Semoga skripsi ini dapat bermanfaat bagi pembaca dan dapat berguna untuk pengembangan penelitian selanjutnya. Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dan masih banyak kekurangan disebabkan oleh

keterbatasan kemampuan serta pengalaman penulis. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan oleh penulis.

Malang, 07 Desember 2018

Penulis,

Nyimas Ayu widi Indriana

ABSTRAK

Nyimas Ayu Widi Indriana. 2018. Momentum *Backpropagation* untuk Klasifikasi Fungsi Senyawa Aktif Berdasarkan Notasi SMILES (*Simplified Molecular Input Line Entry System*), skripsi Program Informatika / Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing: Dian Eka Ratnawati, S.Si, M.Kom dan Syaiful Anam, S.Si.,MT., Ph.D

Senyawa aktif merupakan senyawa yang dapat dimanfaatkan untuk pembuatan obat-obatan tertentu dan sangat penting dalam bidang kesehatan. Proses klasifikasi fungsi senyawa aktif merupakan hal yang paling utama dalam proses pembuatan obat-obatan. Setelah tahap klasifikasi fungsi senyawa aktif dilakukan, maka dapat dilanjutkan pada tahap selanjutnya yaitu, proses pembuatan dan pengujian obat-obatan. Proses pembuatan dan pengujian obat-obatan ini membutuhkan biaya dan waktu yang cukup tinggi. Hal ini lah yang menjadi kendala utama bagi para ahi medis untuk melakukan proses pembuatan obat-obatan tertentu. Dengan memanfaatkan teknologi yang sudah berkembang saat ini dapat dibuat suatu sistem untuk melakukan proses klasifikasi senyawa aktif, sehingga kinerja para ahli medis dalam pembuatan obat-obatan tertentu dapat lebih cepat. Proses klasifikasi fungsi senyawa aktif dapat dilakukan dengan menggunakan komputer dan memanfaatkan notasi *Simplified Molecular Input Line Entry System* (SMILES). Penggunaan notasi SMILES memungkinkan suatu senyawa dapat diproses oleh komputer. Metode momentum *Backpropagation* dapat digunakan untuk melakukan proses klasifikasi dengan baik. Berdasarkan program yang sudah dibuat, dilakukan 5 macam pengujian dengan menggunakan jumlah data latih sebanyak 522 dan data uji sebanyak 131 menghasilkan akurasi terbaik sebesar 70,99% dengan *learning rate* sebesar 0,00001, *max epoch* sebesar 100, momentum sebesar 0,1 dan *neuron hidden layer* sebanyak 4.

Kata kunci: SMILES, *backpropagation,* momentum *backpropagation*

ABSTRACT

Nyimas Ayu Widi Indriana. 2018. Momentum Backpropagation for Classification Functions of Active Compounds Based on SMILES Notation (Simplified Molecular Input Line Entry System), Thesis Program Informatics / Computer Science, Faculty of Computer Science, Universitas Brawijaya.

Promotor: **Dian Eka Ratnawati, S.Si, M.Kom dan Syaiful Anam, S.Si.,MT., Ph.D**

Classification of active compounds is the most important thing in making medicines. After classifying the active compound, it is continued with the process of making and testing drugs that require a variety of tools. The cost of making and testing these drugs requires a high cost and time. This is a major obstacle for medical experts to make certain medicines. By utilizing current technology, a system can be made to classification process of active compounds, so the performance of medical experts for making certain drugs can be faster. The classification process can be done by using a computer and utilizing the SMILES notation. SMILES notation allows a compound to be processed by a computer. The momentum Backpropagation method can be used to perform the classification process properly. Based on the program that has been made, there are 4 types of testing using 522 training data and 131 test data producing, the best accuracy of 70,99% with a learning rate of 0,00001, max epoch of 100, momentum of 0,25 and hidden layer neurons of 4.

Keyword: *SMILES, backpropagation, momentum backpropagation*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Senyawa merupakan suatu zat yang tersusun dari dua atau lebih unsur yang berbeda dan membentuk suatu ikatan kimia. Senyawa dibagi dua macam yaitu senyawa aktif dan senyawa tidak aktif. Senyawa aktif merupakan senyawa yang dapat memicu terjadinya aktivitas biologi dalam organisme hidup atau sering kita sebut dengan istilah biokimia (Sumardjo, 2006), sedangkan senyawa tidak aktif memiliki peran sebagai pendukung suatu reaksi kimia. Kedua senyawa ini saling melengkapi dalam unsur kimia sehari-hari. Senyawa aktif dapat berfungsi sebagai farmakologi dan banyak digunakan untuk pembuatan obat dalam bidang kesehatan. Obat merupakan salah satu jenis senyawa aktif yang sangat dibutuhkan dalam kehidupan manusia untuk menyembuhkan berbagai macam penyakit.

Pembuatan obat harus sesuai dengan kriteria yang sudah ditetapkan oleh Badan Pengawas Obat dan Makanan (BPOM) dan sudah melalui berbagai macam pengujian. Pembuatan dan pengujian obat dilakukan oleh para ahli di bidang kesehatan yang membutuhkan berbagai macam alat serta laboratorium sehingga dapat mendukung proses pengujian berbagai macam senyawa agar dapat mengetahui fungsi senyawa aktif untuk pembuatan obat tertentu. Namun, pembuatan dan pengujian obat ini membutuhkan waktu dan biaya yang tidak sedikit sehingga dapat menghambat kinerja para ahli kesehatan, selain itu banyaknya jenis senyawa aktif dapat menimbulkan terjadinya kesalahan pada proses pengelompokan senyawa aktif. Oleh karena itu, klasifikasi fungsi senyawa aktif perlu dilakukan untuk memudahkan para ahli kesehatan dalam pembuatan obat, sehingga waktu yang dibutuhkan menjadi lebih singkat serta hasil yang didapatkan dalam klasifikasi senyawa aktif lebih akurat.

Klasifikasi dilakukan untuk mengelompokan suatu objek berdasarkan atribut-atributnya (Susilowati, Sabariah, & Gozali, 2015). Dengan memanfaatkan perkembangan teknologi, proses klasifikasi suatu senyawa aktif dapat menggunakan notasi *Simplified Molecular Input Line Entry System* (SMILES). SMILES merupakan notasi kimia modern yang dapat digunakan untuk melakukan pemrosesan senyawa kimia menggunakan komputer. Penulisan notasi SMILES dilakukan dengan menggunakan karakter *American Standart Code for Information Interchange* (ASCII), karakter inilah yang memungkinkan suatu senyawa dapat di proses oleh komputer (Junaedi, 2011).

Proses klasifikasi dapat dilakukan dengan menggunakan metode *Backpropagation* yang terdapat pada Jaringan Syaraf Tiruan. Metode *Backropagation* baik digunakan untuk melakukan klasifikasi pada pola tertentu. Beberapa penelitian yang sudah dilakukan sebelumnya menggunakan algoritme *Backpropagation* dengan judul, “Klasifikasi Varietas Cabai Berdasarkan Morfologi

Daun Menggunakan *Backpropagation Neural Network*" menghasilkan rata-rata akurasi 97,92 % dengan pengujian *K-fold cross validation* menggunakan parameter *k* = 3, *learning rate 0,2, neuron hidden laye* sebanyak 5 dan *max epoch* sebanyak 100.000 (Syaban & Harjoko, 2016), serta penelitian lain dengan judul, "Klasifikasi Pola Sidik Jari Menggunakan Jaringan Syaraf Tiruan *Backpropagation* untuk Analisa Karakteristik Seseorang" menghasilkan akurasi sebesar 83% dengan *max epoch* sebanyak 1000, *learning rate* 0,5 dan jumlah *neuron hidden* sebanyak 160 (Setiawan & Agung, 2016). Berdasarkan kedua penelitian diatas nilai *epoch* yang digunakan untuk mencapai konvergensi sistem terlalu banyak, sehingga waktu konputasi yang dibutuhkan menjadi semakin lama. Oleh karena itu, pada penelitian ini metode *Backpropagation* ditambahkan dengan momentum. Penambahan momentum berguna untuk meningkatkan kecepatan konvergensi sistem, sehingga waktu komputasi yang dibutuhkan menjadi lebih sedikit (Maharani,2009).

Penelitian yang telah dilakukan sebelumnya menggunakan metode Momentum *Backpropagation* yang dijadikan acuan pada penelitian ini berjudul "Klasifikasi Data menggunakan JST *Backpropagation* Momentum dengan *Adaptive Learning Rate"*. Hasil yang didapatkan dari penelitian ini menunjukkan bahwa konstanta momentum dan *Adatif Learning Rate* dapat mempercepat kecepatan konvergensi sistem, selain itu juga berpengaruh terhadap nilai keakuratan dengan akurasi 96 % (Maharani, 2009).

Berdasarkan uraian diatas, maka penelitian ini diberi judul "**MOMENTUM *BACKPROPAGATION* UNTUK KLASIFIKASI FUNGSI SENYAWA AKTIF BERDASARKAN NOTASI SMILES (*SIMPLIFIED MOLECULAR INPUT LINE ENTRY SYSTEM*)"**.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah dari penelitian momentum *backpropagation* untuk klasifikasi menggunakan notasi SMILES, maka rumusan masalah pada penelitian ini adalah:

1. Bagaimana penerapan metode momentum *Backpropagation* dalam proses klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES?
2. Bagaimana tingkat akurasi penggunaaan metode Momentum *Backpropagation* dalam klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES?

1.3 Tujuan

Tujuan dari penelitian momentum *backpropagation* untuk klasifikasi menggunakan notasi SMILES adalah sebagai berikut:

1. Mengetahui proses penerapan klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES menggunakan metode Momentum *Backpropagation.*

2. Mengetahui tingkat akurasi yang didapatkan dari penggunaan metode Momentum *Backpropagation* dalam klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES.

1.4 Manfaat

Manfaat dari penelitian ini adalah:

1. Dapat membantu dalam melakukan klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES.
2. Dapat merekomendasikan seyawa aktif berdasarkan notasi SMILES menggunakan momentum *backpropagation.*

1.5 Batasan Masalah

Pada penelitian ini penulis membatasi dalam beberapa hal yaitu:

1. Data yang diguakan dalam penelitian ini adalah data fungsi senyawa aktif yang didapat dari *https://pubchem.ncbi.nlm.nih.gov*.
2. Pengolahan data menggunakan metode Momentum *Backpropagation.*
3. Jumlah data yang digunakan dalam penelitian ini berjumlah 653 data.
4. Jumlah kelas yang digunakan sebanyak dua kelas, terdiri dari data kelas kanker dan kelas metabolisme.

1.6 Sistematika Pembahasan

Sistematika pembahasan yang menjadi langkah-langkah dalam proses penyusunan tugas akhir ini yaitu:

BAB I Pendahuluan

Bab ini akan menjelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, Batasan masalah, dan sistematika pembahasan.

BAB II Landasan Kepustakaan

Bab ini penulis akan menjelaskan tentang teori dan metode yang berkaitan dengan klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES menggunakan momentum *backpropagation.*

BAB III Metodologi Penelitian

Bab ini akan menjelaskan mengenai tahapan-tahapan metode pelaksanaan hingga tahap implementasi klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES.

BAB IV Perancangan

Bab ini akan menjelaskan mengenai rancangan scenario dan perhitungan manual dari momentum *backpropagation*.

BAB V Implementasi

Bab ini akan menjelaskan mengenai spesifikasi sistem yang digunakan, dan implementasi program.

BAB VI Pembahasan

Bab ini akan menjelaskan mengenai hasil pengujian sistem serta analisis dari data yang sedang diuji.

BAB VII Penutup

Bab ini akan menjelaskan mengenai kesimpulan dan saran dari tugas akhir penulis. Kesimpulan harus menjawab rumusan masalah dan harus sesuai dengan analisis yang telah di jelaskan pada bab sebelumnya. Saran, merupakan arahan bagi penulis untuk bahan koreksi kesalahan untuk penelitian yang lebih baik untuk selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka dalam penelitian ini akan menjelaskan tentang penelitian yang sudah pernah dilakukan sebelumnya dan akan dijadikan sebagai acuan dalam pengerjaan penelitian ini. Penelitian pertama yang berjudul "Klasifikasi Varietas Cabai Berdasarkan Morfologi Daun Menggunakan *Backpropagation Neural Network*" (Syaban & Harjoko, 2016). Pada penelitian ini data yang digunakan diambil dari data citra daun. Tujuan dari penelitian ini adalah untuk mengklasifikasikan varietas cabai. Hasil penelitian dengan menggunkan pengujian *k-fold cross validation* dengan *k = 3*, *learning rate* sebanyak 0,2*, neuron hidden layer* sebanyak 5*,* dan *max* epoch sebanyak 100000 menghasilkan tingkat akurasi sebesar 97,92 %.

Pada penelitian kedua dengan judul "Klasifikasi Pola Sidik Jari Menggunakan Jaringan Syaraf Tiruan *Backpropagation* untuk Analisa Karakteristik Seseorang" (Setiawan & Agung, 2016). Pada penelitian ini data yang digunakan pada penelitian ini adalah sidik jari anak usia dini. Tujuan dari penelitian ini adalah dapat mendeteksi pola sidik jari sehingga dapat mengetahui karakteristik anak. Hasil penelitian dengan menggunkan *epoch* sebanyak 1000, *learning rate* sebesar 0,5 dan jumlah *neuron* sebanyak 160 menghasilkan tingkat akurasi sebesar 83 %.

Tabel 2.1 Kajian pustaka

No	Pustaka	Objek	Metode	*Output*
1	Kharis Syaban & Agus Harjoko	Citra daun	*Backpropagation*	Klasifikasi varietas cabai
2	Ahmad Fahrudin Setiawan, Alam Katon Agung	Citra sidik jari	*Backpropagation*	Pola sidik jari

2.2 Klasifikasi

Klasifikasi merupakan proses pengelompokan suatu objek berdasarkan atribut-atributnya (Susilowati, Sabariah, & Gozali, 2015). Tujuan dari pengelompokan tersebut adalah untuk menentukan kelas dari suatu objek yang belum diketahui. Terdapat 2 proses dalam klasifikasi, yaitu proses *learning* dan proses *testing.* Proses *learning* pada penelitian ini menggunakan model Jaringan Syaraf Tiruan dengan menggunakan data *training,* sedangkan proses *testing* menggunakan nilai yang sudah di dapatkan pada proses *training* sebelumnya.

2.3 *Preprocessing*

Pemrosesan data merupakan tahap awal dalam proses *preprocessing* yang bertujuan untuk menghilangkan *noise* atau data sehingga data layak untuk digunakan (Andono, Nurtantio, Sutojo, & Muljono, 2017). Tahap *preprocessing* yang digunakan untuk mengambil notasi SMILES pada penelitian ini menggunakan *Regular Expression* (regex). Regex sangat berguna untuk membantu mencari pola dari suatu kalimat, dapat diartikan juga jika regex dapat digunakan untuk mengambil huruf, kata, angka atau simbol yang diinginkan (Muliantara, 2009). Regular Expression (regex) dapat diimplementasi pada banyak bahasa pemrograman. Mulau dari *Perl, PHP, Pyhton, java, JavaScript* maupun *VB* (Haryanto, 2002). Pola umum penggunaan regex dapat dilihat pada Tabel 2.2.

Tabel 2.2 Pola umum penggunaan *regular expression*

Pola	Penjelasan
[0-9]	Cocok untuk pola angka.
[aiueo]	Cocok dengan huruf vokal.
[^aiueo]	Cocok dengan huruf konsonan.
[a-z]	Cocok dengan semua huruf kecil.
[A-Z]	Cocok dengan semua huruf besar.
[a-z0-9]	Cocok dengan semua huruf kecil dan angka.
\	Digunakan untuk mengawali penggunaan regex.
{}	Membuat grup yang terdiri dari beberapa karakter.
$	Cocok dengan kata atau karakter yang berada pada akhir kalimat atau kata.
^	Cocok dengan kata atau karakter yang berada pada awal kalimat atau kata.

2.4 SMILES (*Simplified Molecular Input Line Entry System*)

Simplified Molecular Input Line Entry System (SMILES) merupakan sistem notasi kimia yang didesain untuk memproses informasi kimia secara modern. Perkembangan SMILES diprakarsai oleh David Weininger pada akhir 1980 memanfaatkan konsep grafik dengan node sebagai atom (Junaedi, 2011). Sejak itu SMILES mulai dimodifikasi dan disebar luaskan. SMILES merupakan notasi kimia yang dirancang untuk pemrosesan informasi kimia modern. Sistem notasi SMILES cocok digunakan untuk pemrosesan mesin yang memiliki kecepatan tinggi. Spesifikasi molekul notasi SMILES dilakukan dengan menggunakan karakter ASCII untuk mempresentasikan simbol atom dan ikatan (Junaedi, 2011). Sehingga dapat disimpan dalam variabel *string*. Hal tersebut membuat notasi SMILES lebih mudah

untuk diproses oleh computer dan cenderung memakan memori yang lebih sedikit.

Penulisan SMILES bersifat *casesensitif,* yaitu adanya perbedaan antara huruf besar dan kecil. Penulisan untuk satu atom, maka ditulis dengan huruf kapital. Sedangkan untuk penulisan atom lebih dari satu huruf, maka huruf pertama ditulis dengan menggunakan huruf kapital dan untuk huruf berikutnya ditulis dengan huruf kecil. Terdapat tiga macam ikatan, yaitu ikatan tunggal (-), ikatan rangkap (=), dan ikatan rangkap tiga (≡). Contoh penulisan notasi SMILES dapat dilihat pada Tabel 2.3.

Tabel 2.3 Contoh Notasi SMILES

Rumus Molekul	Notasi SMILES
C10H18N4O5	C(CC(C(=O)O)NC(=O)CCC(=O)O)CN=C(N)N
C9H9NO3	C1=CC=C(C=C1)C(=O)NCC(=O)O
C9H21N3O	CC(=O)NCCCNCCCCN

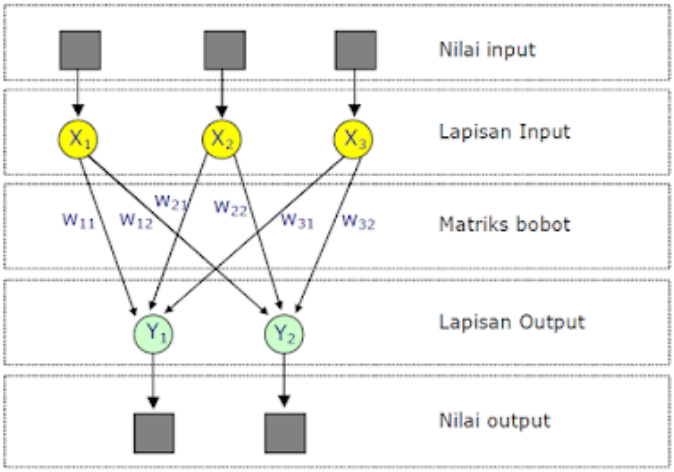
2.5 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (*Artificial neural network*) merupakan model sederhana dari sistem saraf yang ada di tubuh makhluk hidup, memiliki kegunaan yang mirip dengan otak manusia (Muller, Reinhardt, & Strickland, 1995). Jaringan Syaraf Tiruan terdiri dari banyak *neuron*. *Neuron* merupakan unit dimana pemrosesan informasi terjadi. Setiap *neuron* memiliki tugas untuk mentransferkan informasi pada *neuron* yang lainnya. Pada Jaringan Syaraf Tiruan proses pengiriman informasi disebut dengan bobot (Yegnanarayana, 2006).

Pada Jaringan Syaraf Tiruan, beberapa *neuron* dikumpulkan dalam satu *layer* yang disebut dengan *neuron layer*. *Neuron layer* ini berhubungan dengan *layer* sebelum atau *layer* sesudahnya. Informasi pada *neuron layer* akan dikirimkan dari *input layer* hingga ke *output layer* melalui *hidden layer* (Kusumadewi, 2003). Arsitektur Jaringan Syaraf Tiruan ada bermacam-macam, diantaranya (Kusumadewi, 2003).

1. Jaringan dengan lapisan tunggal (*single layer)*

   Jaringan dengan lapisan tunggal ini hanya memiliki satu lapisan saja, yang terdiri dari *input layer* dan *output layer.* Arsitektur jaringan dengan lapisan tunggal (*single layer)* dapat dilihat pada Gambar 2.1.

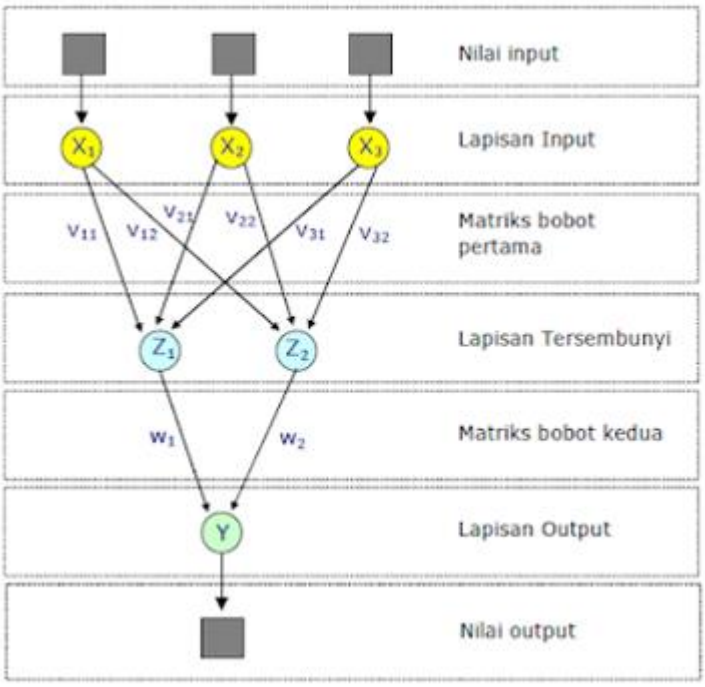


Gambar 2.1 Jaringan dengan lapisan tunggal *(single layer)*

Pada Gambar 1 lapisan *input* memiliki 3 neuron, yaitu X1, X2, dan X3. Sedangkan, pada lapisan *output* memiliki 2 *neuron*, yaitu Y1 dan Y2. *Neuron* pada lapisan *input* dan *neuron* pada lapisan *output* saling berhubungan, hubungan antar kedua lapisan ini ditentukan oleh bobot pada masing-masing *neuron* yang berada pada setiap lapisan.

2. Jaringan dengan banyak lapisan (*multi layer)*

Jaringan dengan banyak lapisan memiliki 1 atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output.* Arsitektur jaringan dengan banyak lapisan (*multi layer*) dapat diihat pada Gambar 2.2



Gambar 2.2 Jaringan dengan banyak lapisan *(multi layer)*

Pada Gambar 2 lapisan *input* memiliki 3 *neuron*, yaitu X1, X2, dan X3. Sedangkan, pada lapisan tersembunyi memiliki 2 *neuron*, yaitu Z1 dan Z2, kemudian lapisan *output* memiliki 1 *neuron*, yaitu Y. *Neuron* yang terdapat pada setiap lapisan saling berhubungan, hubungan antar kedua lapisan ini ditentukan oleh bobot pada masing-masing *neuron* yang berada pada setiap lapisan. Jaringan dengan banyak lapisan dapat digunakan untuk menyelesaikan masalah yang lebih rumit daripada jaringan dengan lapisan tunggal.

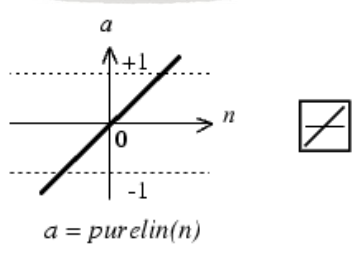
2.6 *Backpropagation*

Backpropagation merupakan algoritme *Supervised Learning* dan merupakan salah satu algoritme dari Jaringan Syaraf Tiruan. Algoritme ini digunakan untuk melatih jaringan sehingga dapat menghasilkan nilai *error* minimum, sehingga memiliki nilai yang seimbang dalam merespon *input* yang berbeda dari data latih sebelumnya. *Backpropagation* bekerja secara *iterative* dengan menggunakan sekumpulan data latih. Dalam setiap *iterative*, bobot relasi dalam jaringan dimodifikasi untuk meminimalkan nilai *Mean Square Error* (MSE) antara nilai prediksi dengan nilai sesungguhnya. Modifikasi relasi Jaringan Syaraf Tiruan tersebut dilakukan dari *output layer* hingga *layer* pertama dari *neuron hidden layer* (Sari, 2016). *Neuron hidden layer* merupakan lapisan tersembunyi yang berfungsi untuk meneruskan atau memutuskan hubungan antara *input layer* dan *output layer,* lapisan ini juga berfungsi untuk meminimalkan nilai *error* sehingga nilai bobot yang baru dapat mendekati target *output* yang diinginkan.

Pada Algoritme *Backpropagation* perubahan nilai bobot terjadi berdasarkan *error output*. Cara untuk mendapatkan nilai *error output* adalah dengan melakukan tahap perambatan maju dan mengaktifkan *neuron* dengan menggunakan fungsi aktivasi *sigmoid* (Salimin, 2015/2016). Fungsi aktivasi inilah yang menentukan nilai bobot pada *neuron* selanjutnya. Terdapat beberapa contoh dari fungsi aktifasi, diantaranya (Santosa & Nugraha, 2007).

1. Linier / Purelin

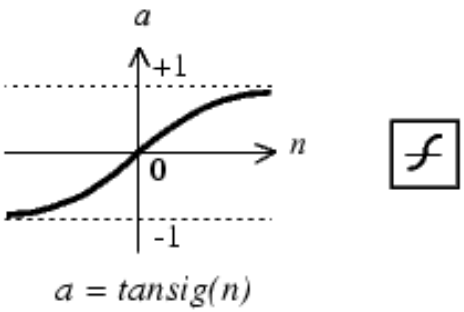
   Fungsi linier ini memiliki fungsi untuk membawa nilai *input ke output* yang sebanding. Fungsi ini dapat dilihat pada Gambar 2.3. Algoritme dari fungsi linier ini adalah $a = n$.



Gambar 2.3 Fungsi Linier / Purelin

2. Tansig

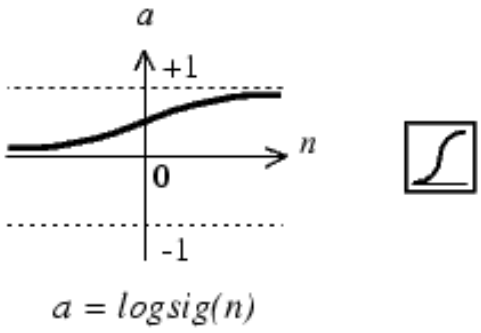
Fungsi tansig merupakan fungsi sigmoid tangen. Fungsi ini akan membawa nilai *input* ke *output* dengan rumus *hyperbolic* tangen sigmoid dengan nilai *output* maksimal 1 dan nilai *output* minimal -1. Algoritme dari fungsi ini adalah $a = tansig\ (n) = 2/\big(1 + exp(-2 * n)\big) - 1$. Fungsi ini dapat dilihat pada Gambar 2.4.



Gambar 2.4 Fungsi Tansig

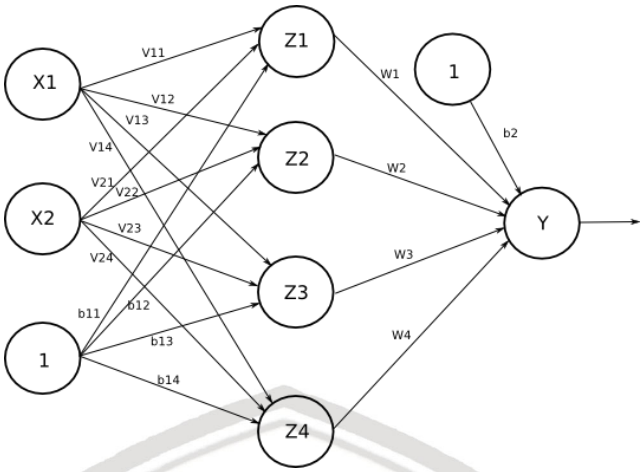
3. Logsig

Fungsi Logsig merupakan fungsi yang membawa nilai *input* ke *output* dengan perhitungan $log - sigmoid$, dengan nilai *output* antara -1 hingga 1. Algoritme dari fungsi ini adalah $a = logsig(n) = 1/\big(1 + exp(-n)\big)$. Fungsi ini dapat dilihat pada Gambar 2.5.



Gambar 2.5 Fungsi Logsig

Arsitektur *backpropagation* dengan menggunakan jaringan lapis banyak (*multi layer*) dapat dilihat pada Gambar 2.1. Pada Gambar 2.1 X1 dan X2 menunjukkan *neuron input*, lalu Z1 sampai Z4 menunjukkan *neuron hidden layer*, sedangkan Y menunjukkan *output layer*. *Neuron input* akan mengirimkan informasi kepada *neuron hidden layer*. Setelah informasi diterima, nilai yang di hasilkan oleh *neuron hidden layer* akan dikirimkan kepada *output layer.*



Gambar 2.6 Arsitektur jaringan *Backpropagation*

2.7 Momentum *Backpropagation*

Momentum *backpropagation* merupakan pengembangan dari algoritme *backpropagation* standar yang didalam pembelajarannya menggunakan nilai konstanta momentum yang memiliki nilai rentang nilai antara 0 sampai 1.

Perhitungan momentum *backpropagation* dapat dilihat pada tahap berikut ini (Andrian & Putra, 2014):

1. Inisialisasi bobot awal V dan W dengan *range* nilai 0 sampai 1.
2. Menentukan nilai maksimum *epoch, learning rate (α)*, momentum, dan *target error.*
3. Lakukan langkah ke- 6 sampai ke- 10 selama kondisi *epoch* < *maksimum epoch* dan *MAPE > Target error*.
4. Lakukan langkah ke-6 sampai ke-10 selama proses *training.*
5. Lakukan langkah ke-6 sampai ke-7 untuk proses pengujian.

Perambatan maju *(Feedforward)*

6. Penjumlahan bobot pada *hidden layer*

   $Z_{\_in(j)} = V_{0j} + \sum_{i=1}^{n} X_i \, . V_{ij}$ ..... (2.1)

   Melakukan operasi aktifasi penjumlahan terbobot dengan fungsi aktifasi sigmoid.

   $Z_j = \frac{1}{1+e^{-Z\_in(j)}}$ ..... (2.2)
7. Menghitung nilai output layer

   $Y_{\_in(k)} = W_{0k} + \sum_{j=1}^{n} Z_j \, . \, W_{jk}$ ..... (2.3)

   Menghitung sinyal *output* dengan fungsi aktifasi:

   $Y_k = Y_{\_in(k)}$ ..... (2.4)

Perambatan mundur (*Backward)*

Perhitungan momentum terletak pada Persamaan 2.6 – 2.7 dan Persamaan 2.10-2.11

8. Menghitung nilai error, berdasarkan nilai error dan nilai target
   $\delta_k = (t_k - Y_k)$ ..... (2.5)
   Menghitung nilai koreksi bobot W_{jk} :
   $\Delta W_{jk}(t) = \alpha\delta_k Z_j + \Delta W_{jk}(t-1)\mu$ ..... (2.6)
   Menghitung nilai koreksi bias W_{0k} :
   $\Delta W_{0k}(t) = \alpha\delta_k + \Delta W_{0k}(t-1)\mu$ ..... (2.7)
9. Penjumlahan delta unit untuk setiap *neuron hidden layer*
   $\delta_{in\ j} = \sum_{k=1}^{m} \delta_k W_{jk}$ ..... (2.8)
   Menghitung nilai error:
   $\delta_j = \delta_{in\ j}(Z_j)(1 - Z_j)$ ..... (2.9)
   Menghitung nilai koreksi bobot V_{ij} :
   $\Delta V_{ij}(t) = \alpha\delta_j X_i + \Delta V_{ij}(t-1)\mu$ ..... (2.10)
   Menghitung nilai koreksi bobot bias V_{0j} :
   $\Delta V_{0j}(t) = \alpha\delta_j + \Delta V_{0j}(t-1)\mu$ ..... (2.11)
10. Perbaikan nilai bobot dan nilai bias
    Perbaikan nilai bobot dan nilai bias pada *output layer*
    $W_{jk}(baru) = W_{jk}(lama) + \Delta W_{jk}$ ..... (2.12)
    $W_{0k}(baru) = W_{0k}(lama) + \Delta W_{0k}$ ..... (2.13)
    Perbaikan nilai bobot dan nilai bias pada *hidden layer*
    $V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij}$ ..... (2.14)
    $V_{0j}(baru) = V_{0j}(lama) + \Delta V_{0j}$ ..... (2.15)

Keterangan:

V = Bobot *input*
V_{0j} = Bias pada *neuron hidden*
W = Bobot *neuron* pada *hidden layer*
W_{0k} = Bias pada *output neuron*
X = Data
Y_k = *Output neuron* k
Z_j = *Neuron hidden* j
i = Data ke-
j = Jumlah *neuron* pada *hidden layer*
k = Jumlah target
μ = Momentum

2.8 *Pyhton*

Guido van Rossum mengenalkan bahasa pemrograman *Pyhton* pertama kali pada tahun 1990 di CWI, Belanda. *Pyhton* merupakan produk *opensource, free,*

dan *multiplatform*. Pemrograman ini juga lebih sederhana, fleksibel dan singkat dibanding dengan Bahasa program yang lain.

2.8.1 Numpy

NumPy merupakan salah satu *package/library* yang berfungsi untuk proses komputasi menggunakan bahasa pemrograman *Pyhton. Beberapa* manfaat dari penggunaan *numPy* adalah sebagai berikut:

- Dapat membangun data *array* multidimensi (*N-dimensional*)
- Mengintegrasikan kode C/C++ dan Fortran
- Memiliki kemampuan untuk menangani masalah komputasi aljabar linier, transformasi *fourier* dan bilangan acak
- Memiliki peranan penting dalam operasi vector dan matriks,

Selain digunakan untuk ilmiah, *NumPy* juga digunakan sebagai tempat data *generic multi-dimensi* yang efisien dan tipe data *numerik* yang dapat didefinisikan (Majster, 1979). Dengan hal ini, *NumPy* memungkinkan untuk terintegrasi dengan mudah dan cepat dengan berbagai macam *database*.

2.9 MAPE

Mean Absolute Percentage Error (MAPE) ditentukan dari rata-rata keseluruhan persentase kesalahan antara data yang diinginkan dengan data hasil peramalan atau prediksi yang kemudian dibagi dengan jumlah keseluruhan data yang digunakan. MAPE dapat dihitung menggunakan rumus sebagai berikut (Makrindakis, et al., 1982):

$$MAPE = \frac{\sum \frac{|X_i - F_i|}{X_i} \times 100\%}{n}$$

Keterangan :

X_i : Target

F_i : Hasil prediksi atau ramalan

n : Jumlah data

BAB 3 METODOLOGI PENELITIAN

Dalam melakukan penelitian, tahapan-tahapan metode dalam meneliti suatu permasalahan harus di jabarkan. Penelitian yang penulis lakukan yaitu klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES merupakan jenis penelitian yang melibatkan pengolahan data.

3.1 Tipe Penelitian

Tipe penelitian yang digunakan pada penelitian ini adalah tipe penelitian nonimplementatif. Teknik pada tipe penelitian ini dapat berupa survey, eksperimen, studi kasus, wawancara, kuisioner, observasi dan sebagainya. Proses utama pada penelitian nonimplementatif berupa penggalian informasi dengan tujuan untuk mendapatkan informasi penting sebagai dasar pengambilan keputusan.

3.2 Studi Literatur

Studi literatur yang digunakan pada penelitian ini berupa jurnal, paper, buku, dan skripsi penelitian terdahulu yang mempunyai keterkaitan dengan metode momentum *Backpropagation* dan *Simplified Molecular Line Entry System* (SMILES). Studi literatur juga berfungsi untuk memperkuat pemahaman penulis, sehingga penelitian ini dapat berjalan sesuai harapan.

3.3 Lokasi Penelitian

Penelitian ini diambil dari situs pemerintah dengan alamat situs https://pubchem.ncbi.nlm.nih.gov, sehingga lokasi khusus yang digunakan pada penelitian ini tidak ada.

3.4 Pengumpulan Data

Pengumpulan data yang dilakukan pada penelitian ini menggunakan teknik sekunder. Data yang dikumpulkan akan digunakan untuk proses manualisasi pada notasi SMILES dengan menggunakan metode momentum *Backpropagation*. Data yang digunakan pada penelitian ini berasal dari situs https://pubchem.ncbi.nlm.nih.gov dan memiliki fungsi *pharmacology* atau dengan kata lain, data yang digunakan termasuk dalam senyawa aktif.

3.5 Perangkat Pendukung

Pada tahap ini dilakukan spesifikasi kebutuhan yang dibutuhkan oleh sistem agar dapat melakukan klasifikasi senyawa sehingga sistem dapat bekerja secara baik. Spesifikasi kebutuhan yang digunakan dalam pembuatan sistem meliputi kebutuhan *hardware,* kebutuhan *software*, kebutuhan data. Setiap kebutuhan tersebut harus dipenuhi untuk melakkan perancangan, implementasi, dan pengujian sistem. Perangkat yang digunakan dalam penelitian ini antara lain:

1. Perangkat Keras, meliputi:
    a. Laptop dengan *processor* Intel Core i3-5005U 2.08Hz
    b. RAM 4 GB
    c. Harddisk 1TB
2. Perangkat Lunak, meliputi:
    a. Sistem operasi Windows 10 64 bit
    b. Aplikasi Pengolah Kata Microsoft Word
    c. Aplikasi Pengolah data Microsoft Excel
    d. *Python* 3.6
3. Data, meliputi:
    a. Data SMILE sebanyak 653 data dengan 11 fitur.
    b. Data fitur
    c. Kelas Penyakit

3.6 Perancangan Sistem

Pada tahap ini, proses perancangan sistem yang dibuat didasarkan pada studi literatur, pengumpulan data, dan analisis kebutuhan dengan menerapkan metode momentum *Backpropagation* untuk proses klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES. Pengguna akan menginputkan senyawa aktif dan nantinya akan menghasilkan kelompok kelas yang dapat digunakan sebagai obat untuk penyakit tertentu pada manusia. Tahap awal klasifikasi adalah dengan melakukan tahap preprocessing untuk notasi SMILES dengan menggunakan regex untuk mendapatkan panjang senyawa dan jumlah masing-masing atom, kemudian jumlah masing-masing atom ini akan dibagi dengan panjang senyawa atau panjang notasi SMILES. Hasil pembagian inilah yang nantinya akan digunakan sebagai input untuk proses perhitungan momentum *Backpropagation*.

3.7 Implementasi Sistem

Implementasi sistem dilakukan setelah proses perancangan. Pada tahapan ini implementasi sistem berguna untuk melakukan perhitungan manual terhadap

notasi SMILES dengan menggunakan metode momentum *Backpropagation,* selain itu implementasi sistem juga berfungsi untuk membantu proses pembuatan program sehingga program yang dibuat dapat mengklasifikasikan fungsi senyawa aktif menggunakan metode momentum *Backpropagation* sesuai dengan harapan penulis.

3.8 Pengujian dan Analisis Sistem

Pengujian dan analisis sistem yang dilakukan pada penelitian ini meliputi pengujian perhitungan manualisasi notasi SMILES menggunakan metode momentum *Backpropagation* serta dilakukan pengujian terhadap program klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES menggunakan metode momentum *Backpropagation*. Pengujian dan analisis sistem pada penelitian ini bertujuan untuk melihat kinerja sistem yang sudah dikerjakan dengan melakukan pengujian validasi sistem, *houldout validation, learning rate*, momentum, dan *cross validation*. Pengujian tersebut dilakukan untuk mengetahui bagaimana tingkat akurasi sistem yang telah dibuat, serta mengetahui apakah metode momentum *Backpropagation* dapat digunakan untuk proses klasifikasi senyawa aktif berdasarkan notasi SMILES dengan baik.

3.9 Kesimpulan

Kesimpulan akan menjawab rumusan masalah yang sudah di buat berdasarkan pengujian yang telah dilakukan, serta memberikan informasi mengenai kinerja sistem menggunakan metode momentum *Backpropagation* yang telah dibuat. Jika, masih terdapat kesalahan pada penelitian yang sudah dilakukan, maka penelitian selanjutnya dapat melakukan penyempurnaan lebih lanjut.

BAB 4 PERANCANGAN

4.1 Deskripsi Umum Sistem

Sistem yang akan dibuat bertujuan untuk mengetahui penerapan momentum *Backpropagation* pada permasalahan klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES. Fitur yang digunakan sebagai masukan ada 11, yang merupakan jumlah masing-masing elemen atom dan kemudian akan dibagi dengan panjang kode SMILES. Sistem akan mengolah masukan dengan cara pembelajaran. Sistem ini mempunyai 2 proses, yaitu proses pelatihan dan proses pengujian. Pada proses pelatihan dan pengujian masing-masing memerlukan masukan berupa data latih dan data uji. Proses yang akan dijalankan oleh sistem adalah sebagai berikut :

1. Proses Pelatihan

   Pada proses ini data latih digunakan untuk melakukan proses perhitungan *backpropagation* sehingga di dapatkan nilai bobot V, nilai bias V , nilai bias W dan nilai bobot W yang baru. Bobot inilah yang nantinya akan digunakan untuk proses pengujian terhadap data yang akan diuji.

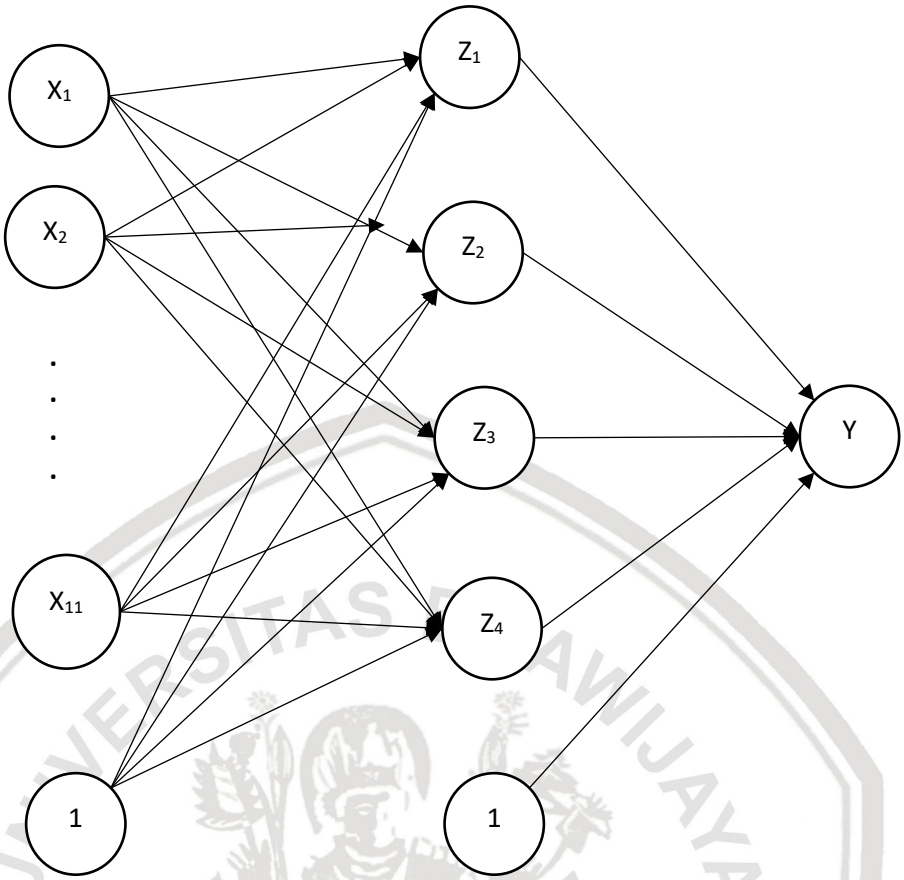
2. Proses Pengujian

   Pada proses ini dilakukan perhitungan *backpropagation* menggunakan bobot V, bias V , bias W dan bobot W yang sudah di dapat dari proses pelatihan sebelumnya.

4.2 Arsitektur Perancangan Sistem

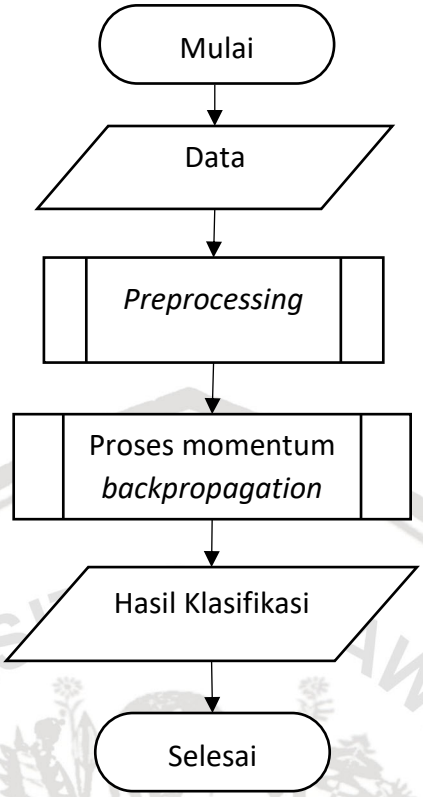
Arsitektur perancangan sistem merupakan gambaran secara umum proses yang terjadi pada sistem klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES menggunakan momentum *Backpropagation*. Arsitektur jaringan pada penelitian ini menggunakan 3 layer utama yang terdiri dari 11 *input layer,* 4 *neuron hidden layer* dan 1 *output layer*. Arsitektur jaringan pada penelitian ini dapat dilihat pada Gambar 4.1.

Klasifikasi dilakukan pada pola, $X_k = \{X_1, X_2, ..., X_n\}$ dalam p kelas, maka akan terdapat p *neuron* pada lapisan *output*. Banyaknya *neuron input layer* disesuaikan dengan masukan pada sistem yang terdiri dari 11 fitur *input* data yaitu atom B, C, N, O, OH, I, P, S, F, Cl, dan Br. Setiap *neuron input layer* terhubung dengan *neuron hidden layer* yang terletak antara *input layer* dan *output layer*. *Hidden neuron* tersebut dihubungkan oleh bobot yang disebut *input weight* dengan nilai *random* antara 0 sampai 1. Kemudian setiap *neuron* pada *hidden layer* terhubung dengan *output layer* yang dihubungkan oleh *output weight*. Semua *neuron hidden* terhubung pada masing-masing *output*.



Gambar 0.1 Arsitektur jaringan

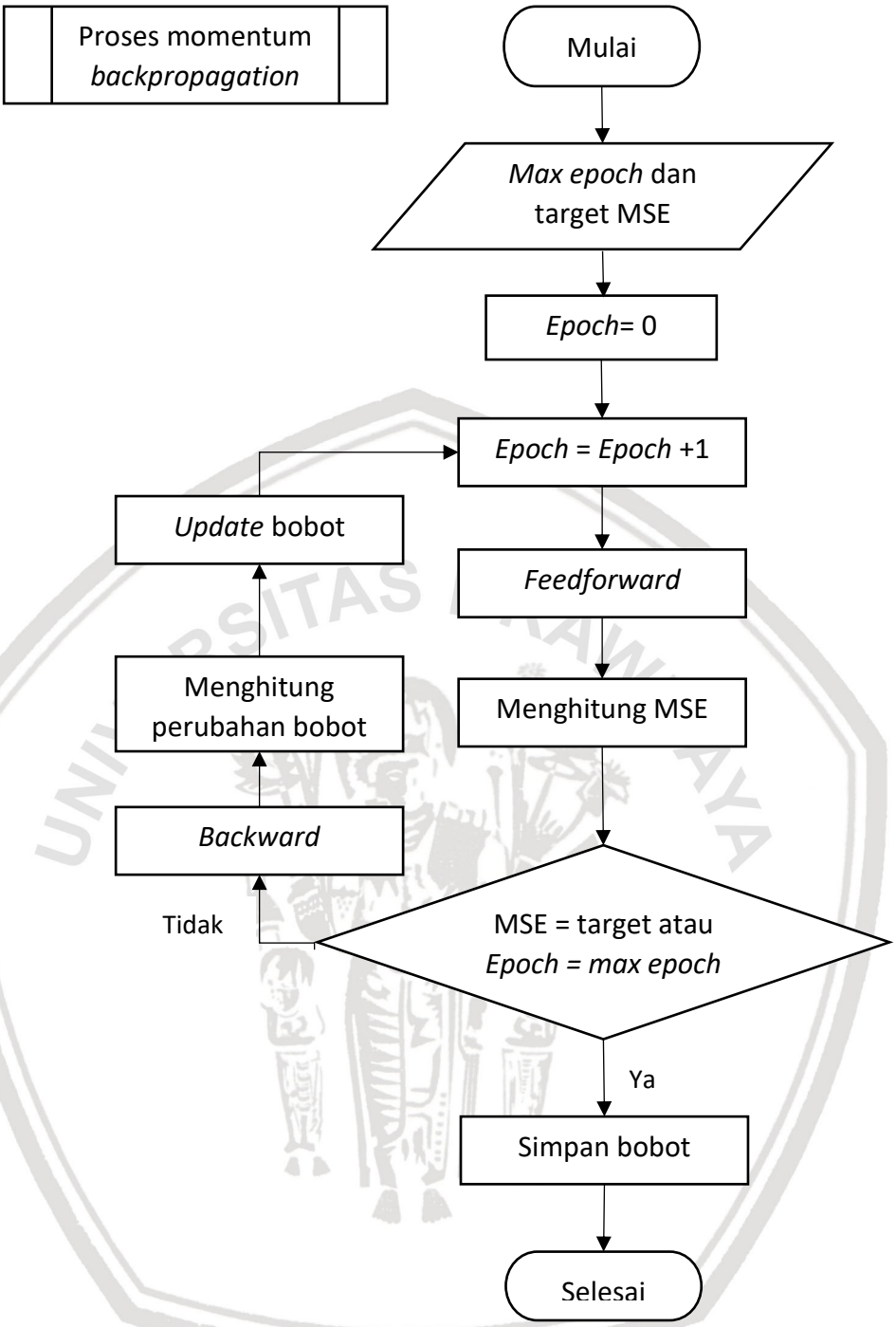
Diagram alir sistem merupakan gambaran dari alur proses sistem yang akan bekerja. Diagram alir sistem pada klasifikasi fungsi senyawa aktif data SMILES ini meliputi *input* data, *training* data, dan *testing* data. Data *training* dan data *testing* di dapatkan dari situs pubchem. Semua data yang digunakan akan melalui tahap *preprocessing* terlebih dahulu yang bertujuan untuk normalisasi fitur yang akan digunakan. Setelah proses *preprocessing* dilakukan, maka selanjutnya dilakukan proses perhitungan momentum *Backpropagation* dengan menggunakan inputan dari fitur yang didapat pada saat tahap *preprocessing,* serta nilai bobot dan bias yang sudah di *random* sebelumnya. Diagram alir sistem ditunjukkan pada Gambar 4.1.



Gambar 0.2 Diagram alir sistem

4.2.1 Proses perhitungan momentum *Backpropagation*

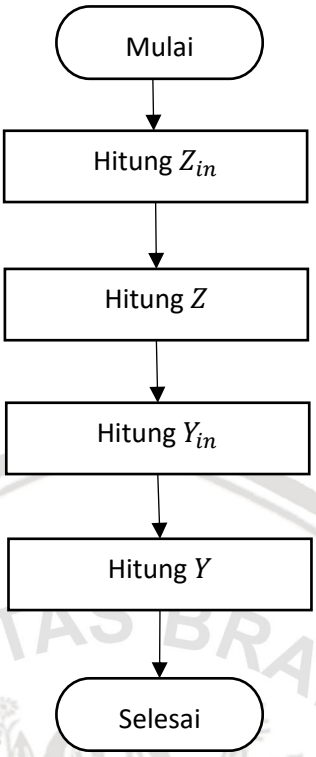
Pada proses ini dilakukan inisialisasi awal untuk bobot V, bobot W dan bobot bias secara *random* dengan range nilai antara 0 sampai 1. Kemudian menentukan *stopping condition*, jika *stopping condition* belum terpenuhi, maka proses perulangan akan terus dilakukan. Proses *feedforward* akan dilakukan pada semua data latih dan dilanjutkan dengan melakukan proses *backward* untuk menghitung nilai koreksi *error* dan nilai *update* bobot. Pada proses *update* bobot akan dilakukan perhitungan penambahan nilai momentum. Jika *stoping condition* sudah terpenuhi, maka nilai bobot V, bobot W dan bobot bias terakhir akan disimpan. Bobot terakhir yang di dapatkan dari proses *training* akan digunakan pada saat proses pengujian. Alur proses perhitungan menggunakan momentum *Backpropagation* ditunjukkan pada Gambar 4.2.



Gambar 0.3 Alur proses perhitungan momentum *Backpropagation*

4.2.2 Alur proses pengujian

Pada proses pengujian ini dilakukan *preprocessing* data untuk semua data uji yang digunakan, kemudian dilakukan perhitungan *feedforward* dengan menggunakan bobot V, bobot W dan bobot bias yang sudah di dapatkan dari proses *training* sebelumnya. Alur proses pengujian ditunjukkan pada Gambar 4.4.



Gambar 0.4 Alur proses pengujian

4.3 Perhitungan Manual

Perhitungan manual merupakan contoh perhitungan dari perancangan sistem yang dibuat dengan tujuan untuk mengetahui kebenaran dari perhitungan sistem. Jumlah data latih yang digunakan dalam perhitungan manual berjumlah 10 data, terdiri dari 5 data kelas kanker dan 5 data kelas metabolisme. Fitur yang digunakan sebanyak 11, yaitu atom O, Oh, B, C, Br, Cl, N, S, F, I, dan P. Masing-masing fitur akan dilakukan perhitungan *preprocessing* dengan cara menghitung masing-masing fitur, kemudian membagi dengan panjang senyawa. Dalam proses perhitungan manual terdapat beberapa langkah yaitu proses *training* dan proses *testing*. Proses *training* bertujuan untuk mendapatkan nilai bobot yang nantinya akan digunakan pada proses *testing.* Fitur dari data *training* yang akan digunakan dan sudah dilakukan *preprocessing* dapat dilihat pada Tabel 4.1.

Tabel 0.1 Fitur data latih

No	Pola Input											Target
	O	Oh	B	C	Br	Cl	N	S	F	I	P	
1.	0,0476190	0,0000000	0	0,3333333	0	0	0,0952381	0	0	0	0	1
2.	0,0857143	0,0571429	0	0,2857143	0	0	0,1142857	0	0	0	0	1
3.	0,0833333	0,0416667	0	0,3750000	0	0	0,0416667	0	0	0	0	1
4.	0,0625000	0,0000000	0	0,5625000	0	0	0,1875000	0	0	0	0	1
5.	0,1428571	0,1428571	0	0,4285714	0	0	0,0000000	0	0	0	0	1
6.	0,0000000	0,0000000	0	0,4285714	0	0	0,2857143	0	0	0	0	2
7.	0,0000000	0,0000000	0	0,5238095	0	0	0,0952381	0	0	0	0	2
8.	0,0000000	0,0000000	0	0,5263158	0	0	0,1052632	0	0	0	0	2
9.	0,0000000	0,0555556	0	0,5555556	0	0	0,1111111	0	0	0	0	2
10.	0,0526316	0,0000000	0	0,5263158	0	0	0,1052632	0	0	0	0	2

Dalam proses perhitungan momentum *Backpropagation* digunakan parameter *learning rate* sebesar 0,00001, maksimum iterasi sebesar 4, jumlah *neuron* pada *hidden layer* sebanyak 4, dan nilai momentum sebesar 0,25. Nilai bobot V, bobot W, dan bobot bias didapat

secara *random* dengan *range* nilai antara 0 sampai 1. *Learning rate* dan momentum bernilai antara 0 sampai 1. Nilai bobot V, bobot W, dan bobot bias dapat dilihat pada Tabel 4.2 dan 4.3.

Tabel 0.2 Nilai bobot V

I / j	1	2	3	4
1.	0,564	0,194	0,199	0,795
2.	0,492	0,481	0,738	0,418
3.	0,422	0,09	0,809	0,16
4.	0,929	0,038	0,66	0,794
5.	0,612	0,556	0,117	0,505
6.	0,274	0,015	0,835	0,304
7.	0,854	0,345	0,053	0,722
8.	0,68	0,402	0,247	0,648
9.	0,31	0,444	0,531	0,958
10.	0,884	0,717	0,032	0,73
11.	0,166	0,09	0,855	0,392
bias	0,25	0,579	0,386	0,199

Tabel 0.3 Nilai bobot W

i / k	1
1.	0,417
2.	0,115
3.	0,517
4.	0,566
Bias	0,025

Perambatan maju *(Feedforward)*

Hal pertama yang dilakukan untuk melakukan proses *feedforward* adalah dengan melakukan penjumlahan setiap *input* pada setiap *neuron hidden layer* dengan menggunakan Persamaan 2.1. Contoh perhitungannya dapat dilihat di bawah ini dan untuk hasil perhitungan dari seluruh data latih yang digunakan dapat dilihat pada Tabel 4.4.

$$Z_{\_in(1,1)} = (1 * 0{,}25) + \big((0{,}0476 * 0{,}564) + (0 * 0{,}492) + (0 * 0{,}422) + (0{,}3333 * 0{,}929) + (0 * 0{,}612) + (0 * 0{,}274) + (0{,}0952 * 0{,}854) + (0 * 0{,}68) + (0 * 0{,}31) + (0 * 0{,}884) + (0 * 0{,}166)\big)$$

$$= 0{,}6679$$

$$Z_{\_in(1,2)} = (1 * 0{,}579) + \big((0{,}0476 * 0{,}194) + (0 * 0{,}481) + (0 * 0{,}09) + (0{,}3333 * 0{,}038) + (0 * 0{,}556) + (0 * 0{,}015) + (0{,}0952 * 0{,}345) + (0 * 0{,}402) + (0 * 0{,}444) + (0 * 0{,}717) + (0 * 0{,}09)\big)$$

$$= 0{,}6337619048$$

$$Z_{\_in(1,3)} = (1 * 0{,}386) + \big((0{,}0476 * 0{,}199) + (0 * 0{,}738) + (0 * 0{,}809) + (0{,}3333 * 0{,}66) + (0 * 0{,}117) + (0 * 0{,}835) + (0{,}0952 * 0{,}53) + (0 * 0{,}247) + (0 * 0{,}531) + (0 * 0{,}032) + (0 * 0{,}855)\big)$$

$$= 0{,}6205238095$$

$$Z_{\_in(1,4)} = (1 * 0{,}199) + \big((0{,}0476 * 0{,}795) + (0 * 0{,}418) + (0 * 0{,}16) + (0{,}3333 * 0{,}794) + (0 * 0{,}505) + (0 * 0{,}304) + (0{,}0952 * 0{,}722) + (0 * 0{,}648) + (0 * 0{,}958) + (0 * 0{,}73) + (0 * 0{,}392)\big)$$

$$= 0{,}5702857143$$

Tabel 0.4 Penjumlahan *input* pada *neuron hidden layer*

i	$Z_{\_in(j)}$			
	$Z_{\_in(i,1)}$	$Z_{\_in(i,2)}$	$Z_{\_in(i,3)}$	$Z_{\_in(i,4)}$
1.	0,6678571429	0,6337619048	0,6205238095	0,5702857143
2.	0,6894856380	0,6733999787	0,6398570468	0,6003998937
3.	0,7014581498	0,6438332823	0,6830414356	0,6104997441
4.	0,9679371757	0,6771874096	0,7796245929	0,8306870474
5.	0,7989994998	0,6917141414	0,8027136446	0,7125707240
6.	0,8921421670	0,6938569432	0,6839991188	0,7455704556
7.	0,8179525735	0,6317619678	0,7367622113	0,6836669636
8.	0,8288435318	0,6353162163	0,7389492667	0,6928967945
9.	0,8883361028	0,6651674898	0,7995591789	0,7435595287
10.	0,8585303472	0,6455275155	0,7494262959	0,7347426240

Setelah menghitung jumlah *input* yang ada pada *neuron hidden layer*, maka dilakukan proses perhitungan fungsi aktivasi sigmoid untuk menghitung nilai *output* dari *neuron hidden layer* menggunakan Persamaan 2.2. Contoh perhitungan fungsi aktivasi sigmoid dapat dilihat dibawah ini dan untuk hasil perhitungan fungsi aktivasi pada seluruh data latih dapat dilihat pada Tabel 4.5.

$$Z_{1,1} = \frac{1}{1+e^{-0,6678571429}} = 0,6610231717$$

$$Z_{1,2} = \frac{1}{1+e^{-0,6337619048}} = 0,6533419726$$

$$Z_{1,3} = \frac{1}{1+e^{-0.6205238095}} = 0,6503376715$$

$$Z_{1,4} = \frac{1}{1 + e^{-0,5702857143}} = 0,6388290996$$

Tabel 0.5 Hasil perhitungan fungsi aktivasi pada *neuron hidden layer*

i	Z_j			
	$Z_{i,1}$	$Z_{i,2}$	$Z_{i,3}$	$Z_{i,4}$
1.	0,6610231717	0,6533419726	0,6503376715	0,6388290996
2.	0,6658524946	0,6622640528	0,6547211451	0,6457477903
3.	0,6685109834	0,6556194645	0,6644171759	0,6480547920
4.	0,7247081423	0,6631106675	0,6855991995	0,6965001828
5.	0,6897604238	0,6663481375	0,6905546566	0,6709689468
6.	0,7093320427	0,6668243730	0,6646306740	0,6782127586
7.	0,6938015555	0,6528888756	0,6762874356	0,6645566337
8.	0,6961103452	0,6536939188	0,6767660471	0,6666110230
9.	0,7085466833	0,6604202362	0,6898801773	0,6777737363
10.	0,7023535103	0,6560018920	0,6790536791	0,6758451447

Tahap kedua dalam proses *feedforward* setelah menghitung nilai *output* pada *neuron hidden layer*, adalah melakukan proses perhitungan nilai *output layer* dengan menggunakan Persamaan 2.3. Contoh perhitungan dapat dilihat di bawah ini dan untuk hasil perhitungan nilai *output layer* pada seluruh data latih dapat dilihat pada Tabel 4.6. Proses perhitungan sinyal *output* dapat dilakukan setelah didapatkan nilai *output layer,* kemudian dilakukan perhitungan sinyal *output* menggunakan fungsi aktivasi purelin sesuai dengan Persamaan 2.4. Hasil perhitungan sinyal *output* dapat dilihat pada Tabel 4.7.

$$Y_{\_in(1,1)} = 1 * 0,025 + ((0,661 * 0,417) + (0,6533 * 0,115) + (0,6503 * 0,517) + (0,6388 * 0,566))$$
$$= 1,0735828360$$

$$Y_{\_in(2,1)} = 1 * 0{,}275 + ((0{,}7456 * 0{,}667) + (0{,}7425 * 0{,}365) + (0{,}7361 * 0{,}767) + (0{,}7283 * 0{,}816))$$
$$= 1{,}0828029427$$

Tabel 0.6 Hasil perhitungan *output layer*

i	$Y_{\_in(i,1)}$
1.	1,0735828360
2.	1,0828029427
3.	1,0894632495
4.	1,1521267570
5.	1,1260320731
6.	1,1249406119
7.	1,1151826628
8.	1,1176790007
9.	1,1367698206
10.	1,1270227248

Setelah menghitung setelah menghitung nilai pada *output layer*, maka dilakukan proses perhitungan fungsi aktivasi untuk menghitung nilai keluaran dari *output layer* menggunakan Persamaan 2.4. Hasil perhitungan fungsi aktivasi pada seluruh data latih dapat dilihat pada Tabel 4.7.

Tabel 0.7 Hasil perhitungan fungsi aktivasi pada *output layer*

i	$Y_{(i,1)}$
1.	1,0735828360
2.	1,0828029427

i	$Y_{(i,1)}$
3.	1,0894632495
4.	1,1521267570
5.	1,1260320731
6.	1,1249406119
7.	1,1151826628
8.	1,1176790007
9.	1,1367698206
10.	1,1270227248

Perambatan mundur (*Backward)*

Jika *stopping condition* belum terpenuhi maka, akan dilakukan proses *backward*. Tahap awal proses *backforward* adalah menghitung nilai *error* berdasarkan nilai target dengan menggunakan Persamaan 2.5. Contoh perhitungan nilai *error* berdasarkan nilai target dapat dilihat di bawah ini dan untuk hasil perhitungan nilai *error* berdasarkan nilai target pada seluruh data latih dapat dilihat pada Tabel 4.8.

$$\delta_{1,1} = (1 - 1{,}0735828360)$$
$$= -0{,}0735828360$$
$$\delta_{2,1} = (1 - 1{,}0828029427)$$
$$= -0{,}0828029427$$

Tabel 0.8 Hasil perhitungan nilai *error* berdasarkan nilai target

i	$\delta_{i,1}$
1.	-0,0735828360
2.	-0,0828029427

i	$\delta_{i,1}$
3.	-0,0894632495
4.	-0,1521267570
5.	-0,1260320731
6.	0,8750593881
7.	0,8848173372
8.	0,8823209993
9.	0,8632301794
10.	0,8729772752

Setelah menghitung nilai *error* berdasarkan nilai target, kemudian dilakukan proses perhitungan untuk menghitung nilai koreksi bobot W_{jk} dan koreksi bobot bias W_{0k} dengan penggunakan Persamaan 2.6 dan Persamaan 2.7. Contoh perhitungan koreksi bobot W_{jk} dan koreksi bobot bias W_{0k} pada data kedua dapat dilihat di bawah. Pada proses perhitungan koreksi bobot W_{jk} dan koreksi bobot bias W_{0k} ditambahkan perhitungan momentum untuk data kedua hingga seterusnya. Hasil perhitungan koreksi bobot W_{jk} dan koreksi bobot bias W_{0k} untuk seluruh data latih dapat dilihat pada Tabel 4.9 dan Tabel 4.10.

$$\Delta W_{1,1} = (0{,}00001 * -0{,}0735828360 * 0{,}0058128431)$$

$$= -0{,}0000004864$$

$$\Delta W_{2,1} = (0{,}00001 * -0{,}0735828360 * 0{,}0074244587) + (0{,}25 * -0{,}0000004864)$$

$$= -0{,}0000004807$$

$$\Delta W_{3,1} = (0{,}00001 * -0{,}0735828360 * \ 0{,}0087207723) + (0{,}25 * -0{,}0000004807)$$

$$= -0{,}0000004785$$

$$\Delta W_{4,1} = (0{,}00001 * -0{,}0735828360 * 0{,}0266665209) + (0{,}25 * -0{,}0000004785)$$

$$= -0{,}0000004701$$

Tabel 0.9 Hasil perhitungan koreksi bobot W_{jk}

i	W11	W21	W31	W41
1.	-0,0000004864	-0,0000004807	-0,0000004785	-0,0000004701
2.	-0,0000006729	-0,0000006686	-0,0000006618	-0,0000006522
3.	-0,0000007663	-0,0000007537	-0,0000007598	-0,0000007428
4.	-0,0000012941	-0,0000011972	-0,0000012329	-0,0000012453
5.	-0,0000011928	-0,0000011391	-0,0000011786	-0,0000011570
6.	0,0000059089	0,0000055503	0,0000055213	0,0000056455
7.	0,0000076161	0,0000071645	0,0000073642	0,0000072915
8.	0,0000080460	0,0000075588	0,0000078123	0,0000077045
9.	0,0000081279	0,0000075906	0,0000079083	0,0000077769
10.	0,0000081634	0,0000076244	0,0000079051	0,0000078442

$$\Delta W_{1,1} = (0{,}00001 * -0{,}0735828360)$$

$$= -0{,}0000007358$$

$$\Delta W_{2,1} = (0{,}00001 * -0{,}0828029427) + (0{,}25 * -0{,}0000007358)$$

$$= -0{,}0000010120$$

Tabel 0.10 Hasil perhitungan koreksi bobot bias W_{0k}

i	WB1
1.	-0,0000007358
2.	-0,0000010120
3.	-0,0000011476
4.	-0,0000018082
5.	-0,0000017124
6.	0,0000083225
7.	0,0000109288
8.	0,0000115554
9.	0,0000115212
10.	0,0000116101

Tahap kedua dalam proses *backward* adalah menghitung delta unit untuk setiap *neuron hidden layer* dengan menggunakan Persamaan 2.8. Contoh perhitungan delta unit untuk setiap *neuron hidden layer* dapat dilihat dibawah ini dan untuk hasil perhitungan delta unit untuk setiap *neuron hidden layer* pada seluruh data latih dapat dilihat pada Tabel 4.11.

$\delta_{in\,1,1} = -0{,}0735828360 * 0{,}417 = -0{,}0306840426$

$\delta_{in\,1,2} = -0{,}0735828360 * 0.115 = -0{,}0084620261$

$\delta_{in\,1,3} = -0{,}0735828360 * 0{,}517 = -0{,}0380423262$

$\delta_{in\,1,4} = -0{,}0735828360 * 0{,}566 = -0{,}0416478852$

Tabel 0.11 Hasil perhitungan delta unit untuk setiap *neuron hidden layer*

i	$\delta_{in\,1}$	$\delta_{in\,2}$	$\delta_{in\,3}$	$\delta_{in\,4}$
1.	-0,0306840426	-0,0084620261	-0,0380423262	-0,0416478852

i	$\delta_{in\,1}$	$\delta_{in\,2}$	$\delta_{in\,3}$	$\delta_{in\,4}$
2.	-0,0345287868	-0,0095222986	-0,0428090818	-0,0468664266
3.	-0,0373060713	-0,0102881709	-0,0462523980	-0,0506360988
4.	-0,0634365647	-0,0174942876	-0,0786492443	-0,0861034607
5.	-0,0525549687	-0,0144932977	-0,0651581869	-0,0713337614
6.	0,3648959036	0,1006281200	0,4524019307	0,4952798795
7.	0,3689701536	0,1017551538	0,4574516336	0,5008078323
8.	0,3679358968	0,1014743930	0,4561675215	0,4994013350
9.	0,3599817964	0,0992853119	0,4463041478	0,4886024162
10.	0,3640535981	0,1004130107	0,4513504598	0,4941262211

$$\delta_{1,1} = -0{,}0306840426 * 0{,}661 * (1 - 0{,}661) = -0{,}0068754206$$

$$\delta_{1,2} = -0{,}0084620261 * 0{,}6616 * (1 - 0{,}6616) = -0{,}0019165325$$

$$\delta_{1,3} = -0{,}0380423262 * 0{,}6503 * (1 - 0{,}6503) = -0{,}0086507711$$

$$\delta_{1,4} = -0{,}0416478852 * 0{,}6388 * (1 - 0{,}6388) = -0{,}0096092700$$

Tabel 0.12 Hasil perhitungan nilai *error* pada *neuron hidden layer*

i	δ_1	δ_2	δ_3	δ_4
1.	-0,0068754206	-0,0019165325	-0,0086507711	-0,0096092700
2.	-0,0076824116	-0,0021298561	-0,0096774796	-0,0107210504
3.	-0,0082671764	-0,0023228898	-0,0103127581	-0,0115490702
4.	-0,0126559912	-0,0039081348	-0,0169530756	-0,0182012106
5.	-0,0112462893	-0,0032222702	-0,0139235828	-0,0157483273

i	δ_1	δ_2	δ_3	δ_4
6.	0,0752342724	0,0223565120	0,1008389160	0,1080899862
7.	0,0783843726	0,0230602609	0,1001465651	0,1116406400
8.	0,0778334291	0,0229715883	0,0997883626	0,1109873357
9.	0,0743392219	0,0222662550	0,0954847722	0,1067090570
10.	0,0761065175	0,0226595424	0,0983672199	0,1082524240

Proses perhitungan delta V dapat di lihat dibawah ini dan untuk hasil perhitungan delta V pada semua data latih dapat dilihat pada Tabel 4.13 sampai dengan Tabel 4.16.

$$\Delta V_{1,1} = (0{,}00001 * -0{,}0068754206 * 0{,}0476)$$
$$= -0{,}0000000033$$
$$\Delta V_{2,1} = (0{,}5 * -0{,}0068754206 * 0{,}0476) + (0{,}25 * -0{,}0000000033)$$
$$= -0{,}0000000074$$

Tabel 0.13 Hasil perhitungan koreksi bobot V pada *neuron hidden layer* 1

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	-0,0000000033	0,0000000000	0	-0,0000000229	0	0	-0,0000000065	0	0	0	0
2.	-0,0000000074	-0,0000000044	0	-0,0000000277	0	0	-0,0000000104	0	0	0	0
3.	-0,0000000087	-0,0000000045	0	-0,0000000379	0	0	-0,0000000060	0	0	0	0
4.	-0,0000000101	-0,0000000011	0	-0,0000000807	0	0	-0,0000000252	0	0	0	0
5.	-0,0000000186	-0,0000000164	0	-0,0000000684	0	0	-0,0000000063	0	0	0	0
6.	-0,0000000046	-0,0000000041	0	0,0000003053	0	0	0,0000002134	0	0	0	0
7.	-0,0000000012	-0,0000000010	0	0,0000004869	0	0	0,0000001280	0	0	0	0

i	O	OH	B	C	Br	Cl	N	S	F	I	P
8.	-0,0000000003	-0,0000000003	0	0,0000005314	0	0	0,0000001139	0	0	0	0
9.	-0,0000000001	0,0000000412	0	0,0000005458	0	0	0,0000001111	0	0	0	0
10.	0,0000000400	0,0000000103	0	0,0000005370	0	0	0,0000001079	0	0	0	0

Tabel 0.14 Hasil perhitungan koreksi bobot V pada *neuron hidden layer* 2

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	-0,0000000009	0,0000000000	0	-0,0000000064	0	0	-0,0000000018	0	0	0	0
2.	-0,0000000021	-0,0000000012	0	-0,0000000077	0	0	-0,0000000029	0	0	0	0
3.	-0,0000000024	-0,0000000013	0	-0,0000000106	0	0	-0,0000000017	0	0	0	0
4.	-0,0000000031	-0,0000000003	0	-0,0000000246	0	0	-0,0000000078	0	0	0	0
5.	-0,0000000054	-0,0000000047	0	-0,0000000200	0	0	-0,0000000019	0	0	0	0
6.	-0,0000000013	-0,0000000012	0	0,0000000908	0	0	0,0000000634	0	0	0	0
7.	-0,0000000003	-0,0000000003	0	0,0000001435	0	0	0,0000000378	0	0	0	0
8.	-0,0000000001	-0,0000000001	0	0,0000001568	0	0	0,0000000336	0	0	0	0
9.	0,0000000000	0,0000000124	0	0,0000001629	0	0	0,0000000331	0	0	0	0
10.	0,0000000119	0,0000000031	0	0,0000001600	0	0	0,0000000321	0	0	0	0

Tabel 0.15 Hasil perhitungan koreksi bobot V pada *neuron hidden layer* 3

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	-0,0000000041	0,0000000000	0	-0,0000000288	0	0	-0,0000000082	0	0	0	0
2.	-0,0000000093	-0,0000000055	0	-0,0000000349	0	0	-0,0000000131	0	0	0	0
3.	-0,0000000109	-0,0000000057	0	-0,0000000474	0	0	-0,0000000076	0	0	0	0

i	O	OH	B	C	Br	Cl	N	S	F	I	P
4.	-0,0000000133	-0,0000000014	0	-0,0000001072	0	0	-0,0000000337	0	0	0	0
5.	-0,0000000232	-0,0000000202	0	-0,0000000865	0	0	-0,0000000084	0	0	0	0
6.	-0,0000000058	-0,0000000051	0	0,0000004105	0	0	0,0000002860	0	0	0	0
7.	-0,0000000015	-0,0000000013	0	0,0000006272	0	0	0,0000001669	0	0	0	0
8.	-0,0000000004	-0,0000000003	0	0,0000006820	0	0	0,0000001468	0	0	0	0
9.	-0,0000000001	0,0000000530	0	0,0000007010	0	0	0,0000001428	0	0	0	0
10.	0,0000000517	0,0000000132	0	0,0000006930	0	0	0,0000001392	0	0	0	0

Tabel 0.16 Hasil perhitungan koreksi bobot V pada *neuron hidden layer* 4

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	-0,0000000046	0,0000000000	0	-0,0000000320	0	0	-0,0000000092	0	0	0	0
2.	-0,0000000103	-0,0000000061	0	-0,0000000386	0	0	-0,0000000145	0	0	0	0
3.	-0,0000000122	-0,0000000063	0	-0,0000000530	0	0	-0,0000000084	0	0	0	0
4.	-0,0000000144	-0,0000000016	0	-0,0000001156	0	0	-0,0000000362	0	0	0	0
5.	-0,0000000261	-0,0000000229	0	-0,0000000964	0	0	-0,0000000091	0	0	0	0
6.	-0,0000000065	-0,0000000057	0	0,0000004391	0	0	0,0000003066	0	0	0	0
7.	-0,0000000016	-0,0000000014	0	0,0000006946	0	0	0,0000001830	0	0	0	0
8.	-0,0000000004	-0,0000000004	0	0,0000007578	0	0	0,0000001626	0	0	0	0
9.	-0,0000000001	0,0000000592	0	0,0000007823	0	0	0,0000001592	0	0	0	0
10.	0,0000000569	0,0000000148	0	0,0000007653	0	0	0,0000001538	0	0	0	0

$$\Delta V_{1.1} = (0{,}00001 * -0{,}0068754206)$$
$$= -0{,}0000000688$$

$$\Delta V_{2.1} = (0{,}00001 * -0{,}0076824116) + (0{,}25 * -0{,}0000000688)$$
$$= -0{,}0000000940$$

Tabel 0.17 Hasil perhitungan koreksi bobot bias V_{0j}

i	V_{01}	V_{02}	V_{03}	V_{04}
1.	-0.0000000688	-0.0000000192	-0.0000000865	-0.0000000961
2.	-0.0000000940	-0.0000000261	-0.0000001184	-0.0000001312
3.	-0.0000001062	-0.0000000298	-0.0000001327	-0.0000001483
4.	-0.0000001531	-0.0000000465	-0.0000002027	-0.0000002191
5.	-0.0000001507	-0.0000000439	-0.0000001899	-0.0000002123
6.	0.0000007147	0.0000002126	0.0000009609	0.0000010278
7.	0.0000009625	0.0000002838	0.0000012417	0.0000013734
8.	0.0000010190	0.0000003007	0.0000013083	0.0000014532
9.	0.0000009981	0.0000002978	0.0000012819	0.0000014304
10.	0.0000010106	0.0000003011	0.0000013042	0.0000014401

Tahap ketiga dalam proses *backward* adalah *update* nilai bobot dan nilai bias. Nilai bobot dan nilai bias pertama yang akan di *update* adalah nilai bobot dan nilai bias pada *output layer* dengan menggunakan Persamaan 2.12 dan Persamaan 2.13. Contoh perhitungan *update* nilai bobot dan nilai bias pada *output layer* dapat dilihat di bawah ini dan untuk hasil perhitungan nilai bobot dan nilai bias pada *output layer* untuk seluruh data latih dapat dilihat pada Tabel 4.18 dan Tabel 4.19.

$$W_{1,1}(baru) = 0{,}417 + -0{,}0000004864 = 0{,}4169995136$$
$$W_{2,1}(baru) = 0{,}115 + -0{,}0000004807 = 0{,}1149995193$$
$$W_{3,1}(baru) = 0{,}517 + -0{,}0000004785 = 0{,}5169995215$$

$W_{4,1}(baru) = 0{,}566 + -0{,}0000004701 = 0{,}5659995299$

Tabel 0.18 Hasil perhitungan *update* bobot W_{jk} pada *output layer*

i	W_{11}	W_{21}	W_{31}	W_{41}
1.	0,4169995136	0,1149995193	0,5169995215	0,5659995299
2.	0,4169988407	0,1149988507	0,5169988597	0,5659988777
3.	0,4169980743	0,1149980970	0,5169980999	0,5659981349
4.	0,4169967803	0,1149968998	0,5169968669	0,5659968896
5.	0,4169955875	0,1149957607	0,5169956884	0,5659957327
6.	0,4170014963	0,1150013110	0,5170012096	0,5660013782
7.	0,4170091124	0,1150084755	0,5170085739	0,5660086697
8.	0,4170171584	0,1150160343	0,5170163862	0,5660163742
9.	0,4170252863	0,1150236249	0,5170242945	0,5660241511
10.	0,4170334496	0,1150312494	0,5170321996	0,5660319953

$W_{0,1}(baru) = 0{,}25 + -0{,}0000007358 = 0{,}0249992642$
$W_{0,2}(baru) = 0{,}0249992642 + -0{,}0000010120 = 0{,}0249982522$
$W_{0,3}(baru) = 0{,}0249982522 + -0{,}0000011476 = 0{,}0249971046$

Tabel 0.19 Hasil perhitungan *update* bobot bias W_{0k} pada *output layer*

i	W_{01}
1.	0,0249992642
2.	0,0249982522
3.	0,0249971046
4.	0,0249952964
5.	0,0249935840
6.	0,0250019065
7.	0,0250128353
8.	0,0250243907
9.	0,0250359119
10.	0,0250475219

$V_{1,1}(baru) = 0{,}564 + -0{,}0000000033 = 0{,}56400$

$V_{1,2}(baru) = 0{,}194 + -0{,}0000000009 = 0{,}19400$

Tabel 0.20 Hasil perhitungan *update* bobot V pada *neuron hidden layer* 1

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
2.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
3.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
4.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
5.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
6.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
7.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
8.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600

i	O	OH	B	C	Br	Cl	N	S	F	I	P
9.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0,31000	0,88400	0,16600
10.	0,56400	0,49200	0,42200	0,92900	0,61200	0,27400	0,85400	0,68000	0.31000	0,88400	0,16600

Tabel 0.21 Hasil perhitungan *update* bobot V pada *neuron hidden layer* 2

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
2.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
3.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
4.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
5.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
6.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
7.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
8.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
9.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000
10.	0,19400	0,48100	0,09000	0,03800	0,55600	0,01500	0,34500	0,40200	0,44400	0,71700	0,09000

Tabel 0.22 Hasil perhitungan *update* bobot V pada *neuron hidden layer* 3

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
2.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
3.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
4.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
5.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
6.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500

i	O	OH	B	C	Br	Cl	N	S	F	I	P
7.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
8.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
9.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500
10.	0,19900	0,73800	0,80900	0,66000	0,11700	0,83500	0,05300	0,24700	0,53100	0,03200	0,85500

Tabel 0.23 Hasil perhitungan *update* bobot V pada *neuron hidden layer* 4

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
2.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
3.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
4.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
5.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
6.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
7.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
8.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
9.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200
10.	0,79500	0,41800	0,16000	0,79400	0,50500	0,30400	0,72200	0,64800	0,95800	0,73000	0,39200

$$V_{0,1}(baru) = 0{,}25 + -0{,}0000000688 = 0{,}25000$$

$$V_{0,2}(baru) = 0{,}579 + -0{,}0000000192 = 0{,}57900$$

$$V_{0,3}(baru) = 0{,}386 + -0{,}0000000865 = 0{,}38600$$

$$V_{0,4}(baru) = 0{,}199 + -0{,}0000000961 = 0{,}1989999$$

Tabel 0.24 Hasil perhitungan *update* bobot bias V

i	V_{01}	V_{02}	V_{03}	V_{04}
1.	0,25000	0,57900	0,38600	0,1989999
2.	0,25000	0,57900	0,38600	0,1989999
3.	0,25000	0,57900	0,38600	0,1989999
4.	0,25000	0,57900	0,38600	0,1989999
5.	0,25000	0,57900	0,38600	0,1990000
6.	0,25000	0,57900	0,38600	0,1989998
7.	0,25000	0,57900	0,38600	0,1989996
8.	0,25000	0,57900	0,38600	0,1989994
9.	0,25000	0,57900	0,38600	0,1989992
10.	0,25000	0,57900	0,38600	0,1989990

Proses pelatihan dilakukan hingga mencapai *epoch* maksimal atau batas nilai MSE yang di inginkan. Setelah *stopping condition* sudah terpenuhi, maka akan didapatkan nilai bobot V, bobot W, bias V, dan bias W. Bobot dan bias inilah yang nantinya akan digunakan untuk melakukan proses pengujian. Pada proses pengujian manualisasi akan digunakan 6 data uji yang terdiri dari 2 kelas, yaitu kelas metabolisme dan kelas kanker. Masing – masing kelas ada 3 data uji. Fitur data uji dapat dilihat pada Tabel 4.25.

Tabel 0.25 Fitur data uji

i	O	OH	B	C	Br	Cl	N	S	F	I	P
1.	0,0000	0,0000	0,0000	0,5294	0,0000	0,0000	0,1176	0,0000	0,0000	0,0000	0,0000
2.	0,0000	0,0667	0,0000	0,2000	0,0000	0,0000	0,0667	0,0000	0,0000	0,0000	0,0000
3.	0,0345	0,0345	0,0000	0,4828	0,0000	0,0690	0,0345	0,0000	0,0000	0,0000	0,0000
4.	0,0833	0,1667	0,0000	0,3333	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
5.	0,0000	0,0345	0,0000	0,4483	0,0000	0,0345	0,0345	0,0000	0,0000	0,0000	0,0000
6.	0,0323	0,0323	0,0000	0,4194	0,0000	0,0323	0,0323	0,0000	0,0000	0,0000	0,0000

Pada proses pengujian dilakukan perhitungan untuk menghitung nilai bobot *input* pada *hidden layer*, fungsi aktivasi dari bobot *input* ke *hidden layer*, menghitung nilai *output layer*, dan menghitung keluaran dari *output layer*. Hasil perhitungan semua data uji dapat dilihat pada Tabel 4.26.

Tabel 0.26 Hasil Perhitungan data uji

i	$Z_{\_in(j)}$				Z_j				$Y_{\_in(i,1)}$	Y	Target
	$Z_{\_in(i,1)}$	$Z_{\_in(i,2)}$	$Z_{\_in(i,3)}$	$Z_{\_in(i,4)}$	$Z_{i,1}$	$Z_{i,2}$	$Z_{i,3}$	$Z_{i,4}$			
1.	0,842	0,640	0,742	0,704	0,699	0,655	0,677	0,669	1,121	1,121	1
2.	0,526	0,642	0,571	0,434	0,628	0,655	0,639	0,607	1,036	1,036	1
3.	0,783	0,634	0,796	0,670	0,686	0,653	0,689	0,662	1,117	1,117	1
4.	0,689	0,688	0,746	0,600	0,666	0,666	0,678	0,646	1,095	1,095	2
5.	0,722	0,625	0,738	0,605	0,673	0,651	0,677	0,647	1,096	1,096	2
6.	0,710	0,628	0,722	0,604	0,670	0,652	0,673	0,647	1,093	1,093	2

Berdasarkan Tabel 4.26 dari enam data latih yang telah diuji, hanya terdapat 3 data latih yang mempunyai nilai sama dengan target yang diinginkan. Perhitungan nilai akurasi dapat dihitung sebagai berikut:

$$\frac{3}{6} \times 100\% = 50\%$$

4.4 Perancangan Pengujian

Perancangan pengujian digunakan untuk mengetahui hasil dari *scenario* pengujian sehingga dapat mengetahui hasil dan kualitas dari sistem yang telah di buat.

4.4.1 Pengujian validasi program

Pengujian validitas program dilakukan untuk mengetahui apakah hasil keluaran sistem sama dengan hasil perhitungan manual yang sudah dilakukan. Pada pengujian ini data yang digunakan merupakan data yang sama dengan data pada perhitungan manual. Jika hasil perhitungan manual sama dengan hasil perhitungan sistem, maka program dapat dikatakan valid.

4.4.2 Pengujian *houldout validation*

Pengujian *houldout validation* dilakukan untuk mengetahui tingkat akurasi sistem terhadap jumlah data latih dan data uji yang memiliki jumlah yang berbeda. Pada penelitian ini data yang digunakan memiliki total data yang beerbeda pada masing-masing kelas. Tingkat akurasi tertinggi nantinya akan digunakan untuk proses pengujian selanjutnya.

4.4.3 Pengujian *learning rate*

Proses pengujian *learning rate* dilakukan untuk mengetahui pengaruh *learning rate* terhadap nilai akurasi data dengan menggunakan data pada pengujian sebelumnya.

4.4.4 Pengujian momentum

Proses pengujian momentum dilakukan untuk mengetahui pengaruh nilai momentum terhadap nilai akurasi data dengan menggunakan data pada pengujian *houldout validation*.

4.4.5 Pengujian *k*-fold *cross validation*

Pengujian *cross validation* dilakukan untuk mengetahui tingkat akurasi sistem pada jumlah data yang sama. Pada pengujian ini data yang digunakan akan dibagi menjadi 5 kelompok data, masing-masing kelompok data memiliki jumlah yang sama.

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi sistem pada program momentum *Backpropagation* dan analisis kode program momentum *Backpropagation*.

5.1 Implementasi Program

5.1.1 Implementasi *preprocessing data*

Pada proses *preprocessing* data yang akan dilakukan yaitu menghitung masing-masing fitur kemudian fitur yang sudah didapatkan dibagi dengan panjang senyawa yang ada. Kode program untuk proses *preprocessing* data ditunjukkan pada Tabel 5.1.

Tabel 0.1 Kode program *preprocessing* data

```
1   class preprocessing():
2
3       def setRantai(self,rantai):
4           self.senyawa = rantai
5
6       def getO(self):
7           data = self.senyawa
8           data1 = re.sub('[A-Za-z]O$','',data)
9           data1 = re.sub('O[0-9]$','', data1)
10          data1 = re.sub('O$','', data1)
11          data1 = re.sub('=O$','', data1)
12          hilang = re.sub('[A-Za-z]O','', data1)
13          find_O = re.findall('=O',hilang)
14          find_O+= re.findall('O[A-Za-z]',hilang)
15          find_O+= re.findall('[0-9]O',hilang)
16          find_O+= re.findall('[A-Za-z]O',data1)
17          jumlah_O=len(find_O)
18          return jumlah_O
19
20      def getOH(self):
21          data = self.senyawa
22          data1 = re.sub('[A-Za-z]O$','',data)
23          data1 = re.sub('O[0-9]$','', data1)
24          data1 = re.sub('O$','', data1)
25          data1 = re.sub('=O$','', data1)
26          hilang = re.sub('=O','', data1)
27          hilang = re.sub('[A-Za-z]O','',hilang)
28          hilang = re.sub('O[A-Za-z]','',hilang)
29          hilang = re.sub('[0-9]O','',hilang)
30          find_oh = re.findall('O',hilang)
31          find_oh += re.findall('O[0-9]$', data)
32          find_oh += re.findall('[A-Za-z]O$', data)
33          find_oh += re.findall('O$', data)
34          jumlah_OH = len(find_oh)
35          return jumlah_OH
36
37      def getBr(self):
38          data = self.senyawa
39          find_Br = re.findall('Br', data)
40          jumlah_Br= len(find_Br)
```

```
41          return jumlah_Br
42
43      def getB(self):
44          data = self.senyawa
45          data = re.sub('Br','', data)
46          find_B = re.findall('B',data)
47          jumlah_B = len(find_B)
48          return jumlah_B
49
50      def getCl(self):
51          data = self.senyawa
52          find_Cl = re.findall('Cl', data)
53          jumlah_Cl= len(find_Cl)
54          return jumlah_Cl
55
56      def getC(self):
57          data = self.senyawa
58          data = re.sub('Cl','', data)
59          find_C = re.findall('C',data)
60          jumlah_C = len(find_C)
61          return jumlah_C
62
63      def getAll(self):
64          find = ['N','S','F','I','P']#
65          jumlah =[]
66          jumlah.append(self.getO())
67          jumlah.append(self.getOH())
68          jumlah.append(self.getB())
69          jumlah.append(self.getC())
70          jumlah.append(self.getBr())
71          jumlah.append(self.getCl())
72          for i in range(len(find)):
73              jum = re.findall(find[i],self.senyawa)
74              jumlah.append(len(jum))
75          return jumlah
76
77      def getPanjang(self):
78          data= re.sub('\d','',self.senyawa)
79          data = re.sub('Cl','l',data)
80          data = re.sub('Br','r',data)
81          return len(data)
82
83      def getAllFiturNormalisasi(self, fitur,panjang):
84          fiturNorm = fitur
85          for i in range(len(fitur)):
86              for j in range(len(fitur[0])):
87                  fiturNorm[i][j] = fitur[i][j]/panjang[i]
88          return fiturNorm
```

Penjelasan kode program *preprocessing* data:

Baris 6-18 proses pengambilan jumlah atom O

Baris 20-35 proses pengambilan jumlah atom Oh

Baris 37-41 proses pengambilan jumlah atom Br

Baris 43-48 proses pengambilan jumlah atom B

Baris 50-54 proses pengambilan jumlah atom Cl

Baris 56-61 proses pengambilan jumlah atom C

Baris 63-75 untuk mengambil jumlah masing-masing atom

Baris 77-81 untuk menghitung panjang senyawa

Baris 83-88 proses perhitungan normalisasi untuk semua atom dengan membagi jumlah masing-masing atom dengan panjang senyawa

5.1.2 Implementasi momentum *Backpropagation*

Pada proses perhitungan momentum *Backpropagation,* masing-masing fitur yang sudah didapatkan akan dijadikan sebagai inputan.

Tabel 0.2 Kode program momentum *Backpropagation*

```
1   class MomentumBackprop():
2       def bobotawal(self,n, m):
3           bobot = []
4           for i in range(m):
5               bobotrand = []
6               for j in range(n):
7                   ran = random.uniform(0,1)
8                   bobotrand.append(ran)
9               bobot.append(bobotrand)
10          return np.array(bobot)
11
12      def bias(self,n):
13          bias = np.zeros(n,np.float16)
14          for i in range(n):
15              bias[i] = random.uniform(0,1)
16          return bias
17
18      def hitZin(self,bobot,bias,data):
19          zin=[]
20          for j in range(len(bobot[0])):
21              Sum = 0
22              for i in range (len(data)):
23                  Sum += data[i]*bobot[i][j]
24              zin.append(bias[j]+ Sum)
25          return zin
26
27      def hitZ(self,Zin):
28          Z = []
29          for j in range(len(Zin)):
30              temp = 1/(1+ math.exp(-Zin[j]))
31              Z.append(temp)
32          return Z
33
34      def hitYin(self,data,bobot,bias):
35          yin=[]
36          for k in range(len(bobot[0])):
37              y = 0
38              for j in range(len(data)):
39                  y += data[j]*bobot[j][k]
40              yin.append(bias[k]+y)
41          return yin
42
43      def hitY(self,Yin):
```

```
44          Y=[]
45          for k in range(len(Yin)):
46              temp = Yin[k]
47              Y.append(temp)
48          return Y
49
50      def deltak(self,target, y):
51          error = np.zeros((len(y)),np.float32)
52          for k in range(len(y)):
53              error[k] = (target - y[k])
54          return error
55
56      def deltaW(self,alfa, deltak, mom, z,delW):
57          dw = np.zeros((len(z), len(deltak)),np.float32)
58          for j in range(len(z)):
59              for k in range(len(deltak)):
60                  dw[j][k] = (alfa * deltak[k] * z[j]) +
    (mom*delW[j][k])
61          return dw
62
63      def deltabiasW(self, alfa, err,mom,dbias):
64          db = np.zeros((len(err)),np.float32)
65          for k in range (len(err)):
66              db[k] = (alfa*err[k])+(mom*dbias[k])
67          return db
68
69      def delta_net(self,deltak, W,n): #delta unit
70          deltain = np.zeros(n,np.float32)
71          for j in range(len(W)):
72              for k in range(len(deltak)):
73                  temp = deltak[k] * W[j][k] #THIS
74              deltain[j] = temp
75          return deltain
76
77      def delta(self,delta_net, Z):
78          deltaj = np.zeros((len(delta_net)),np.float32)
79          for j in range(len(delta_net)):
80              deltaj[j] = delta_net[j]*Z[j]*(1-Z[j])
81          return deltaj
82
83      def deltaV(self,alfa,fitur, delta, mom, dv):
84          deltav = np.zeros((len(fitur),
    len(delta)),np.float32)
85          for i in range(len(fitur)):
86              for j in range(len(delta)):
87                  deltav[i][j] = (alfa*delta[j]*fitur[i])
    +(mom*dv[i][j])
88          return deltav
89
90      def deltabiasV(self, alfa, err,mom,dbias):
91          db = np.zeros((len(err)),np.float32)
92          for k in range (len(err)):
93              db[k] = (alfa*err[k])+(mom*dbias[k])
94          return db
95
96      def updateBobot(self,bobot, delta):
97          for i in range(len(bobot)):
98              for j in range(len(bobot[0])):
99                  bobot[i][j] = bobot[i][j] + delta[i][j]
```

```
100             return bobot
101
102         def updateBias(self,bias, delta):
103             for i in range(len(bias)):
104                 bias[i] = bias[i] + delta[i]
105             return bias
106
107         def MAPE(self,y,t):
108             jml = np.zeros(len(y),np.float32)
109             for i in range(len(y)):
110                 jml[i] = jml[i]+((abs(t[i]-y[i])/t[i]))
111             jumlah = 0
112             for i in range(len(jml)):
113                 jumlah = jumlah+jml[i]
114             result= jumlah/len(jml)
115             return result
```

Penjelasan kode program momentum *Backpropagation*:

Baris 1-10 untuk menghitung nilai bobot awal secara *random* dengan *range* nilai antara 0-1

Baris 12-16 untuk menghitung nilai bobot bias awal secara *random* dengan *range* nilai antara 0-1

Baris 18-25 untuk menghitung nilai Z_{in}

Baris 27-32 untuk menghitung nilai Z

Baris 34-41 untuk menghitung nilai Y_{in}

Baris 43-48 untuk menghitung nilai Y

Baris 50-54 untuk menghitung nilai δ_k

Baris 56-61 untuk menghitung nilai koreksi bobot W dengan penambahan nilai momentum

Baris 63-67 untuk menghitung nilai koreksi bias W dengan penambahan nilai momentum

Baris 69-75 untuk menghitung nilai delta unit

Baris 77-81 untuk menghitung nilai *error* dari *neuron hidden layer*

Baris 83-88 untuk menghitung nilai koreksi bobot V dengan penambahan nilai momentum

Baris 90-94 untuk menghitung nilai koreksi bias V dengan penambahan nilai momentum

Baris 96-100 untuk menghitung nilai *update* bobot

Baris 102-105 untuk menghitung nilai *update* bias

Baris 107-115 untuk menghitung nilai *error* pada setiap data

5.1.3 Implementasi *running* program

Pada proses *running program*, semua method yang sudah dibuat pada kelas-kelas sebelumnya akan dipanggil.

Tabel 0.3 Kode *running program*

```
1   def bacaData(file):#baca kelas
2       f= open(file,'r', encoding='utf-8-sig')
3       read= csv.reader(f)
4       hasil = []
5       for i in read:
6           for j in i:
7               hasil.append(j)
8       return hasil
9
10  def convertKelaas(data):
11      hasil = np.zeros(len(data))
12      for i in range(len(data)):
13          hasil[i] = float(data[i])
14      return hasil
15
16  def PreprocessingData(senyawa):
17      pre = preprocessing()
18      fitur =[]
19      panjangsenyawa=[]
20      for i in range(len(senyawa)):
21          pre.setRantai(senyawa[i])
22          fitur.append(pre.getAll())
23          panjangsenyawa.append(pre.getPanjang())
24      fitur =
    pre.getAllFiturNormalisasi(fitur,panjangsenyawa)
25      return fitur
26
27  def backprop(data, target,hidden):
28      bp = MomentumBackprop()
29      bias_V = bp.bias(hidden)
30      bobot_V = bp.bobotawal(hidden,11)
31      bobot_W = bp.bobotawal(1,hidden)
32      bias_W = bp.bias(1)
33      alfa= 0.00001
34      momentum = 0.25
35      delta_w = [np.zeros(1)] * hidden #THIS
36      delta_biasW = np.zeros(1) #THIS
37      delta_v = np.zeros([11, hidden]) #THIS
38      delta_biasV = np.zeros(hidden) #THIS
39      mape = 100
40      epoch = 0
41      while  epoch < 100 and mape > 0.2:
42          y_mp = []
43          for i in range(len(data)):
44              zin = bp.hitZin(bobot_V,bias_V,data[i])
45              z = bp.hitZ(zin)
46              yin = bp.hitYin(z,bobot_W,bias_W)
47              y_mp.append(bp.hitY(yin))
48              y = bp.hitY(yin)
49              delta_k = bp.deltak(target[i],y)
50              delta_w=
    bp.deltaW(alfa,delta_k,momentum,z,delta_w)
51              delta_biasW=
    bp.deltabiasW(alfa,bias_W,momentum,delta_biasW)
52              delta_in =
    bp.delta_net(delta_k,bobot_W,hidden)
53              delta = bp.delta(delta_in,z)
```

```
54              delta_v =
    bp.deltaV(alfa,data[i],delta,momentum,delta_v)
55              delta_biasV =
    bp.deltabiasV(alfa,bias_V,momentum,delta_biasV) #THIS
56              bobot_V = bp.updateBobot(bobot_V, delta_v)
57              bobot_W = bp.updateBobot(bobot_W, delta_w)
58              bias_W = bp.updateBias(bias_W,delta_biasW)
59              bias_V = bp.updateBias(bias_V, delta_biasV)
60          mape = bp.MAPE(y_mp,target)
61          epoch = epoch+1
62          print('nilai mape:', mape,'\n nilai epoch:',
63  epoch)
64      return bobot_V,bias_V,bobot_W,bias_W
65
66  def ujidata(data,bobotV,bobotW,biasV,biasW):
67      bp = MomentumBackprop()
68      y = []
        for i in range(len(data)):
69          zin = bp.hitZin(bobotV,biasV,data[i])
70          z = bp.hitZ(zin)
71          yin = bp.hitYin(z,bobotW,biasW)
72          y.append(bp.hitY(yin))
73      return y
74
75  def akurasi(y, ct):
76      total = 0
77      for i in range(len(y)):
78          if round(y[i][0])==ct[i]:
79              total +=1
80      akurasi = total/len(y)*100
81      return akurasi
82
83  a = bacaData('Skanker.csv')
84  fitur = PreprocessingData(a)
85  kls = bacaData('Kkanker.csv')
86  kelas = convertKelaas(kls)
87  bobot_V,bias_V,bobot_W,bias_W = backprop(fitur,kelas,4)
88  b= bacaData('Usenyawa.csv')
89  fitur_uji = PreprocessingData(b)
90  y = ujidata(fitur_uji,bobot_V,bobot_W,bias_V,bias_W)
91  print('Hasil ')
92  for i in range(len(y)):
93      for j in range(len(y[0])):
94          print(i+1,':',round(y[i][j]))
95  c = bacaData('kuji.csv')
96  ct = convertKelaas(c)
97  print('Akurasi : ', akurasi(y, ct))
```

Penjelasan kode program *running*:

Baris 1-8 membaca isi *file* dengan format file .CSV kemudian menyimpannya pada variabel hasil

Baris 10-14 merubah tipe data *integer* pada isi *file* menjadi tipe data *float*

Baris 16-25 melakukan tahap *preprocessing* data

Baris 27-64 melakukan tahap perhitungan momentum *Backpropagation* untuk data latih

Baris 66-73 melakukan tahap perhitungan momentum *Backpropagation* untuk data uji

Baris 75-81 untuk melakukan proses perhitungan akurasi sistem

Baris 83 inisialisasi proses membaca *file* senyawa untuk data latih

Baris 84 inisialisasi untuk proses *preprocessing* dari *file* yang sudah dibaca sebelumnya

Baris 85 inisialisasi proses membaca *file* kelas senyawa untuk data latih

Baris 86 inisialisasi proses untuk mengubah data dari tipe data *integer* menjadi tipe data *float*

Baris 87 proses perhitungan data latih menggunakan momentum *Backpropagation* dengan parameter yang sudah ditentukan

Baris 88 inisialisasi proses membaca *file* senyawa untuk data uji

Baris 89 inisialisasi untuk proses preprocessing dari *file* yang sudah dibaca sebelumnya

Baris 90 proses perhitungan data uji menggunakan momentum *Backpropagation* dengan parameter yang sudah ditentukan

Baris 91-94 untuk mencetak hasil prediksi

Baris 95-97 untuk menghitung akurasi

BAB 6 PEMBAHASAN

Pada bab ini akan dilakukan proses pengujian sistem. Proses ini bertujuan untuk mengetahui seberapa besar nilai akurasi dari program yang sudah dibuat. Setelah melakukan proses pengujian, maka hasilnya akan dianalisis. *Scenario* pengujian yang dilakukan pada bab ini yaitu, pengujian untuk mengetahui nilai akurasi program dengan nilai akurasi perhitungan manual, pengujian pengaruh jumlah data latih dan data uji terhadap nilai akurasi, pengujian pengaruh nilai *learning rate* terhadap nilai akurasi, pengujian pengaruh nilai momentum terhadap nilai akurasi, dan pengujian *cross validation*.

6.1 Pengujian Validitas Program

Pada pengujian ini data yang digunakan sebanyak 16 data, terdiri dari 10 data training dan 6 data uji. Data training terdiri dari 5 data kanker dan 5 data metabolisme. Proses pengujian dilakukan dengan menggunakan *learning rate* sebesar 0,00001, jumlah *neuron hidden layer* sebanyak 4, momentum sebesar 0,25 dan *max epoch* sebanyak 4. Pengujian ini bertujuan untuk mengetahui apakah *output* dari sistem sudah sama dengan hasil perhitungan manual yang sudah dilakukan. Hasil pengujian validitas program dapat dilihat pada Tabel 6.1.

Tabel 0.1 Hasil pengujian validitas program

Jenis Pengujian	Akurasi
Manualisasi	50 %
Sistem	50 %

Tabel 6.1 menunjukkan bahwa penggunaan data yang sama pada program dan pada perhitungan manual menghasilkan nilai akurasi sebesar 50 %. Hal ini menandakan bahwa nilai *output* dari sistem sama dengan nilai *output* pada manualisasi.

6.2 Pengujian *Holdout Validation*

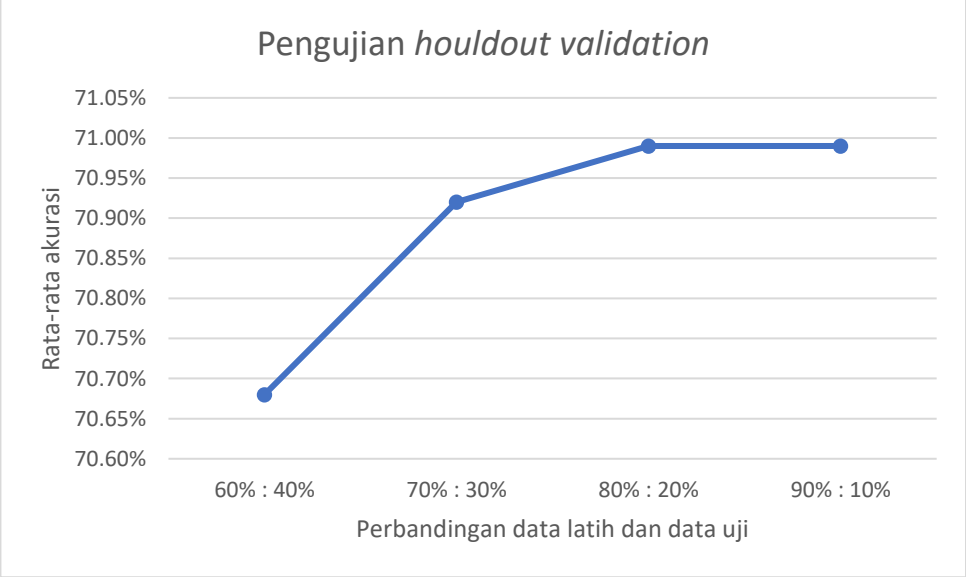
Pada pengujian ini data yang digunakan sebanyak 653 data, terdiri dari 189 data kanker dan 464 data 1 metabolisme. Data yang digunakan di bagi menjadi data latih dan data uji. Persentase perbandingan data latih dan data uji untuk setiap data metabolisme dan kanker menggunakan perbandingan *holdout validation*.

Proses pengujian ini menggunakan *learning rate*, jumlah *neuron hidden layer*, momentum dan *max epoch* adalah 0,0001, 4, 0,25, 50. Pengujian ini bertujuan untuk mengetahui perbandingan data latih dan data uji yang menghasilkan akurasi terbesar. Hasil pengujiannya dapat dilihat pada Tabel 6.2.

Tabel 0.2 Hasil pengujian pengaruh jumlah data latih dan data uji

Data latih	Data uji	Percobaan					Rata-rata akurasi (%)
		1	2	3	4	5	
60 %	40 %	70,99 %	70,99 %	70,22 %	70,99 %	70,22 %	70,68 %
70 %	30 %	70,92 %	70,92 %	70,92 %	70,92 %	70,92 %	70,92 %
80 %	20 %	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %
90 %	10 %	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %

Tabel 6.1 menunjukkan bahwa jumlah data latih dan data uji dapat mempengaruhi tingkat akurasi. Semakin besar atau semakin banyak data latih yang digunakan, maka semakin besar pula tingkat akurasinya. Sedangkan semakin kecil data latih yang digunakan akan semakin kecil pula tingkat akurasinya. Hal ini di sebabkan karena dengan menambahkan jumlah data latih, maka dapat membantu dalam proses pembelajaran pada metode momentum *Backpropagation* sehingga mampu melakukan proses klasifikasi dengan baik. Grafik hasil pengujian pengaruh jumlah data latih dan data uji dapat dilihat pada Gambar 6.1.



Gambar 0.1 Hasil pengujian *houldout validation*

Tabel 6.1 menunjukkan bahwa hasil proses pengujian pengaruh jumlah data latih dan data uji, menghasilkan nilai akurasi yang sama sebesar 70,99 % pada perbandingan data latih dan data uji sebesar 80 % : 20 % dan 90 % : 10 %. Hal ini menunjukkan bahwa perbandingan minimal untuk perbandingan data latih dan

data uji adalah 80 % : 20 %. Perbandingan data latih dan data uji inilah yang akan digunakan pada pengujian selanjutnya.

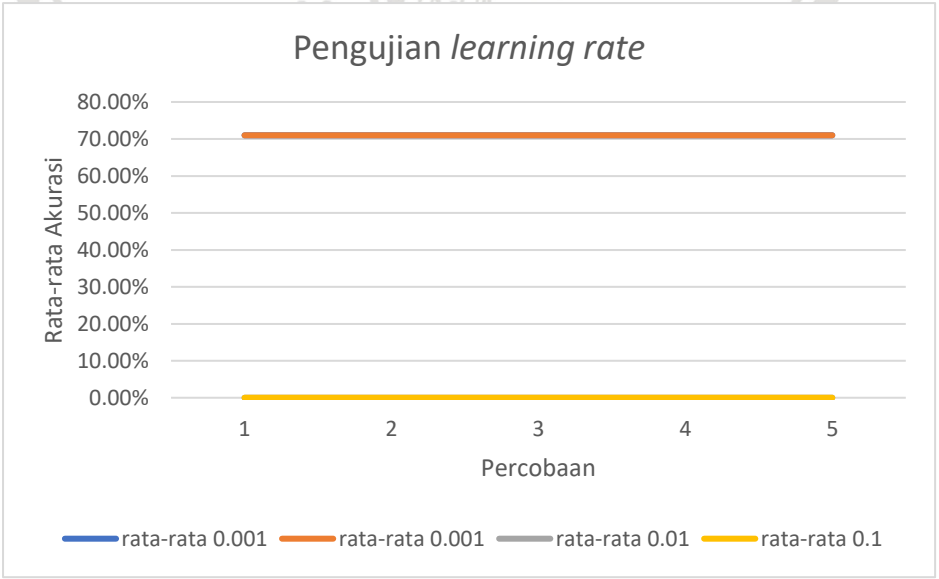
6.3 Pengujian *Learning Rate*

Pada pengujian ini menggunakan total data sebanyak 653 data yang terdiri dari data latih sebanyak 522 data dan data uji sebanyak 131 data. Data latih terdiri dari 371 data metabolisme dan 151 data kanker. Sedangkan data uji terdiri dari 3 data metabolisme dan 38 data kanker. Pengujian dilakukan dengan menggunakan jumlah *neuron hidden layer* sebanyak 6, momentum sebesar 0,25, dan *max epoch* sebanyak 100. Pengujian ini bertujuan untuk mengetahui pengaruh *learning rate* terhadap jumlah data yang digunakan. Hasil pengujian dapat dilihat pada Tabel 6.3.

Tabel 0.3 Hasil pengujian pengaruh *learning rate*

Nilai α	Percobaan					Rata-rata akurasi (%)
	1	2	3	4	5	
0,1	Nan	Nan	Nan	Nan	Nan	Nan
0,01	Nan	Nan	Nan	Nan	Nan	Nan
0,0001	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %
0,00001	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %	70,99 %

Tabel 6.2 menunjukkan bahwa nilai *learning rate* berpengaruh terhadap nilai akurasi. Hal ini dikarenakan semakin kecil nilai *learning* rate yang digunakan, maka hasil akurasi akan semakin baik. *Learning rate* 0,1 dan 0,01 menghasilkan hasil akurasi berupa nilai Nan, hal ini dikarenakan semakin besar nilai *learning rate* yang digunakan maka, nilai *error* yang dihasilkan akan semakin besar pula sehingga



neural network tidak bisa mencapai hasil yang konvergen. Grafik dari proses pengujian *learning rate* dapat dilihat pada Gambar 6.2.

Gambar 0.2 Hasil pengujian *learning rate*

Berdasarkan hasil dari pengujian pengaruh *learning rate* yang sudah dilakukan, maka pada pengujian ini dipilih nilai *learning rate* sebesar 0,00001 untuk proses pengujian berikutnya.

6.4 Pengujian Momentum

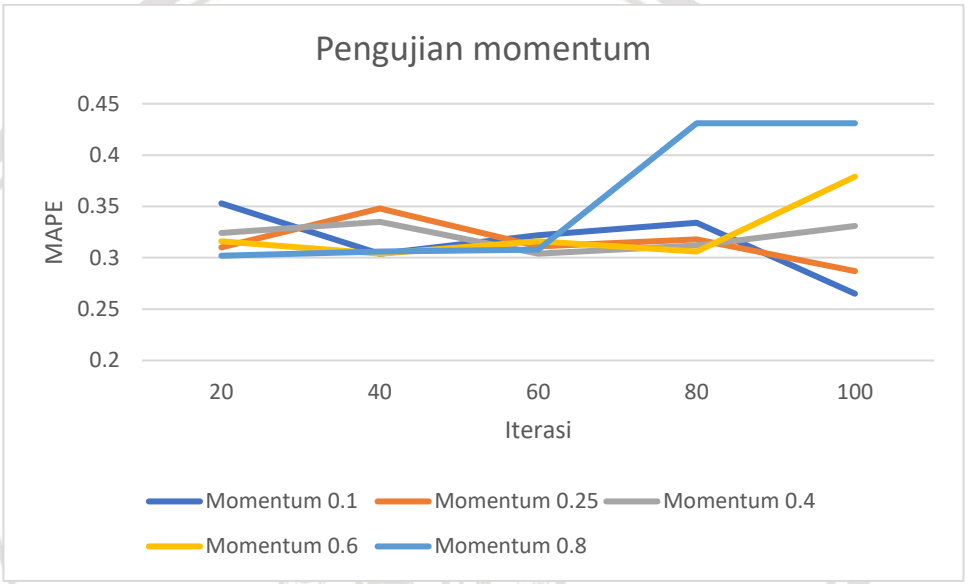
Pada pengujian ini menggunakan total data sebanyak 653 data yang terdiri dari data latih sebanyak 522 data dan data uji sebanyak 131 data. Data latih terdiri dari 371 data metabolisme dan 151 data kanker. Sedangkan data uji terdiri dari 4 data metabolisme dan 38 data kanker. Pengujian dilakukan dengan menggunakan jumlah *neuron hidden layer* sebanyak 4, dan *learning rate* sebesar 0,00001. Pengujian ini bertujuan untuk mengetahui pengaruh momentum terhadap nilai MAPE. Hasil pengujian dapat dilihat pada Tabel 6.4.

Tabel 0.4 Hasil pengujian pengaruh momentum

Momentum	Iterasi	Percobaan					Rata-rata
		1	2	3	4	5	
0,1	20	0,353	0,317	0,395	0,348	0,349	0,353
	40	0,303	0,295	0,319	0,300	0,301	0,304
	60	0,1	0,353	0,303	0,301	0,332	0,309
	80	0,316	0,405	0,310	0,306	0,342	0,334
	100	0,313	0,335	0,312	0,312	0,305	0,265
0,25	20	0,325	0,310	0,303	0,303	0,323	0,310
	40	0,319	0,305	0,317	0,450	0,293	0,348
	60	0,319	0,330	0,301	0,295	0,331	0,311
	80	0,323	0,327	0,323	0,298	0,304	0,318
	100	0,300	0,294	0,305	0,306	0,302	0,287
0,4	20	0,302	0,340	0,348	0,306	0,295	0,324
	40	0,317	0,314	0,305	0,403	0,323	0,335
	60	0,303	0,308	0,299	0,307	0,307	0,304
	80	0,312	0,309	0,318	0,304	0,318	0,312
	100	0,305	0,309	0,309	0,316	0,297	0,331
0,6	20	0,340	0,306	0,303	0,316	0,321	0,316
	40	0,293	0,300	0,326	0,298	0,303	0,304
	60	0,335	0,306	0,312	0,310	0,316	0,316
	80	0,308	0,302	0,310	0,302	0,307	0,306
	100	0,306	0,303	0,307	0,303	0,303	0,379
0,8	20	0,300	0,302	0,307	0,300	0,322	0,302
	40	0,306	0,303	0,305	0,311	0,304	0,306
	60	0,308	0,309	0,310	0,305	0,309	0,308

	80	0,308	0,308	0,306	0,308	0,308	0,431
	100	0,308	0,308	0,309	0,308	0,308	0,431

Tabel 6.4 menunjukkan bahwa perubahan nilai momentum menghasilkan nilai MAPE yang bervariasi untuk setiap iterasi. Hasil penelitian menunjukkan jika nilai momentum kecil maka, iterasi yang dibutuhkan akan semakin banyak tetapi nilai kesalahan yang dihasikan semakin kecil sedangkan jika menggunakan nilai momentum besar walaupun menggunakan iterasi besar nilai kesalahannya tidak bisa mengecil. Hal ini terjadi karena tingkat kemiripan data pada kelas yang sama kecil. Fitur yang digunakan juga belum mewakili setiap kelas, sehingga metode momentum *Backpropagation* sulit membedakan pola kelas yang digunakan. Grafik dari pengujian momentum dapat dilihat pada Gambar 6.3.



Gambar 0.3 Hasil pengujian momentum

Berdasarkan hasil dari pengujian pengaruh momentum, dipilih nilai momentum dengan nilai MAPE minimum. Sehingga, pada pengujian ini dipilih nilai momentum sebesar 0,1 untuk digunakan pada proses pengujian selanjutnya.

6.5 Pengujian *K-Fold Validation*

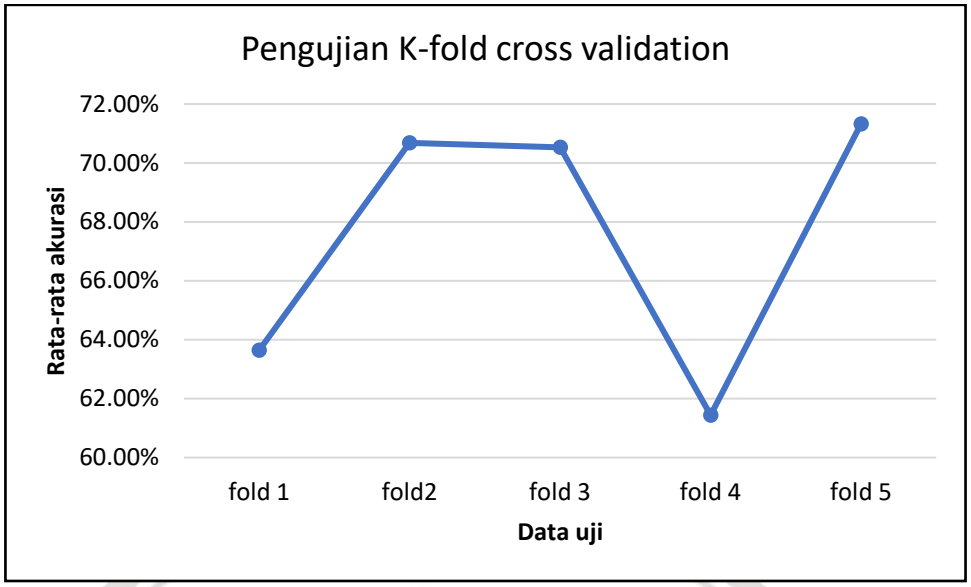
Pada pengujian ini jumlah data yang digunakan sebanyak 653 data yang terdiri dari 189 data kanker dan 464 data metabolisme. Nilai K yang digunakan sebanyak 5 dan akan dibagi menjadi 5 kelompok data. Setiap kelompok data memiliki jumlah data yang sama. Kelompok-kelompok data ini nantinya akan dilakukan proses pengujian sebanyak 5 kali. Jika data uji menggunakan kelompok data 1 maka kelompok data yang lain akan menjadi data *training*. Proses ini terus berulang hingga semua kelompok data menjadi data uji.

Pada pengujian ini menggunakan *learning rate*, jumlah *neuron hidden layer*, momentum, dan *max epoch* adalah 0,00001, 4, 0,1, 100. Pengujian ini bertujuan untuk menguji keakuratan sistem. Hasil pengujian *K-fold cross validation* dapat dilihat pada Tabel 6.5.

Tabel 0.5 Hasil pengujian menggunakan *K-Fold cross validation*

Data latih	Data uji	Percobaan					Rata-rata akurasi (%)
		1	2	3	4	5	
Fold 2, *fold* 3, *fold* 4, *fold* 5	*Fold* 1	70,22 %	70,99 %	70,99 %	35,11 %	70,99 %	63,64 %
Fold 1, *fold* 3, *fold* 4, *fold* 5	*Fold* 2	70,99 %	70,22 %	70,99 %	70,99 %	70,22 %	70,68 %
Fold 1, *fold* 2, *fold* 4, *fold* 5	*Fold* 3	70,22 %	70,99 %	70,99 %	69,46 %	70,99 %	70,53 %
Fold 1, *fold* 2, *fold* 3, *fold* 5	*Fold* 4	70,99 %	70,99 %	70,99 %	70,99 %	28,24 %	61,44 %
Fold 1, *fold* 2, *fold* 3, *fold* 4	*Fold* 5	71,32 %	71,32 %	71,32 %	71,32 %	71,32 %	71,32 %
Rata-rata		70,74 %	70,70 %	71,06 %	63,57 %	62,35 %	67,32 %

Tabel 6.3 menunjukkan bahwa hasil akurasi yang didapat bervariasi. Hasil rata-rata akurasi sebesar 67,32% sedangkan nilai akurasi terkecil terjadi saat menggunakan data uji fold 4 dengan nilai akurasi 61,44% dan hasil akurasi tertinggi terjadi saat menggunakan data uji *fold* 5 yang menghasilkan nilai akurasi 71,32%. Metode momentum *Backpropagation* lebih banyak mengenali data kelas metabolisme dari pada data kelas kanker. Hal ini terjadi karena jumlah data kelas metabolisme lebih banyak dari pada jumlah data kelas kanker, sehingga metode momentum *Backpropagation* tidak dapat melakukan pembelajaran dengan baik terhadap pola kelas kanker. Grafik dari pengujian 5 *fold cross validation* dapat dilihat pada Gambar 6.4.



Gambar 0.4 Hasil pengujian *K-fold cross validation*

Berdasarkan pengujian *K-fold cross validation* dapat disimpulkan bahwa dengan melakukan pertukaran data latih menjadi data uji maka nilai akurasi yang didapatkan juga akan beragam karena adanya perbedaan pola pada notasi SMILES yang dipelajari oleh metode momentum *Backpropagation.*

BAB 7 PENUTUP

Berdasarkan penelitian yang sudah dilakukan, maka diperoleh kesimpulan dan saran yang akan dijelaskan pada bab ini. Kesimpulan dan saran dalam penelitian ini nantinya dapat digunakan untuk masukan pada penelitian selanjutnya.

7.1 Kesimpulan

Kesimpulan yang diperoleh berdasarkan penelitian klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES menggunakan metode momentum *Backpropagation* adalah:

1. Penerapan metode momentum *Backpropagation* dalam melakukan klasifikasi fungsi senyawa aktif berdasarkan notasi SMILES dengan menggunakan *preprocessing* untuk mendapatkan fitur, fitur yang digunakan sebanyak 11 meliputi atom B, C, N, O, P, S, F, Cl, Br, I, OH. Setelah semua fitur didapatkan, maka dilakukan proses pelatihan dengan menggunakan metode momentum *Backpropagation* sehingga didapatkan hasil klasifikasi. Perbedaan antara metode *Backpropagation* biasa dan metode momentum *Backpropagation* terletak pada proses perhitungan delta bobot dan bias. Pada perhitungan delta bobot dan bias yang terdapat dalam metode momentum *Backpropagation* dilakukan penambahan perkalian antara delta bobot dan bias sebelumnya dengan momentum.
2. Terdapat 5 pengujian pada penelitian ini, diantaranya.
   - Pengujian validasi program. Pada pengujian ini hasil akurasi yang didapatkan pada program sama dengan hasil akurasi pada perhitungan manual.
   - Pengujian *houldout validation.* Pada pengujian ini menghasilkan nilai akurasi tertinggi 70,99 % dengan menggunakan perbandingan data latih 80 % dan data uji 20 %.
   - Pengujian *learning rate*. Pada pengujian ini menghasilkan nilai akurasi sebesar 70,99 %.
   - Pengujian momentum. Pada pengujian ini menghasilkan nilai MAPE minimum sebesar 0,265 dengan nilai momentum sebesar 0,1 dan jumlah iterasi sebanyak 100.
   - Pengujian *K-fold cross validation.* Pada pengujian ini menghasilkan nilai rata-rata akurasi sebesar 67,32 %.

   Berdasarkan pengujian yang telah dilakukan. Hasil akurasi terbaik sebesar 70,99 % menggunakan parameter *learning rate* sebesar 0,00001*, max epoch* sebesar 100*, neuron hidden layer* sebesar 4, dan momentum sebesar 0,1.

7.2 Saran

Berdasarkan penelitian yang dilakukan tentang momentum *Backpropagation* dalam melakukan klasifikasi senyawa aktif berdasarkan notasi SMILES, saran yang dapat diberikan untuk penelitian selanjutnya adalah untuk melakukan penambahkan jumlah data dan fitur yang digunakan sehingga dapat melakukan pembelajaran dan pengenalan pola suatu kelas dengan lebih baik.

DAFTAR PUSTAKA

Adfa, M. (2005). Survei etnobotani, studi senyawa flavonoid dan uji brine shrimp beberapa tumbuhan obat tradisional suku serawai di propinsi bengkulu. *Gradien*, 43-50.

Andono, Nurtantio, P., Sutojo, & Muljono. (2017). *Pengolahan citra digital.* Semarang: Andi.

Andrian, Y., & Putra, P. H. (2014). Analisis penambahan momentum pada proses predksi curah hujan kota medan menggunakan metode backpropagation neural network. *Seminar nasional informatika*, 165-172.

Bisri, H., Bustomi, M. A., & Purwanti, E. (2013). Klasifikasi citra paru-paru dengan ekstraksi fitur histogram dan jaringan syaraf tiruan backpropagation. *Sains dan Seni POMITS*, B-68 - B-71.

Haryanto, S. (2002). *Kuasai regex hari ini juga.* Jakarta: Masterweb Media.

Junaedi, H. (2011). Penggambaran rantai karbon dengan menggunakan simplified molecular input line system(SMILES). *IdeaTech*, 219-226.

Kusumadewi, S. (2003). *Artificial intelligence.* Yogyakarta: Graha ilmu.

Maharani, W. (2009). Klasifikasi Data Menggunakan JST Backpropagation Momentum dengan Adaptive Learning Rate. *Seminar Nasional Informatika*, D-25 - D-31.

Majster, M. E. (1979). Data type, abstract data types and their specification problem. *Theoretical computer sciense*, 89-127.

Makrindakis, S., Hibon, M., A, A., R, F., R, C., & R, W. (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of forecasting*, 111-153.

Muliantara. (2009). *Penerapan regular expression dalam melindungi alamat email dari spam robot pada konten wordpress.* Denpasar: Universitas Udayana.

Muller, B., Reinhardt, J., & Strickland, M. T. (1995). Neural networks : an introduction. *Springer*.

Salimin, J. (2015/2016). Aplikasi graf pada artificial neural network dan backpropagation algorithm. *Matematika Diskrit*.

Santosa, B., & Nugraha, R. (2007). Peramalan dengan menggunakan Artificial Neural Network dan Support Vector Regression. *Prosiding Seminar Nasional Managemen Teknologi V*, 1-7.

Sari, Y. (2016). Optimasi conjugate gradient pada algoritma backpropagation neural netword untuk prediksi kurs time series. *Gema aktualita*, 86-90.

Setiawan, A. F., & Agung, A. K. (2016). Klasifikasi pola sidik jari menggunakan jaringan syaraf tiruan backpropagation untuk analisa karakteristik seseorang. *Jurnal Ilmiah dan Teknik Informatika*, 50-55.

Sumardjo, D. (2006). *Pengantar kimia : buku panduan kuliah mahasiswa kedokteran dan program strata I fakultas bioeksakta.* Jakarta: EGC.

Susilowati, E., Sabariah, M. K., & Gozali, A. A. (2015). Implementasi Metode Support Vector Machine untk Melakukan Klasifikasi Kemacetan Lalu Lintas Pada Twitter. *e-Proceeding of Engineering*, 1478.

Syaban, K., & Harjoko, A. (2016). Klasifikasi varietas cabai berdasarkan morfologi daun menggunakan backpropagation neural network. *IJCCS*, 161-172.

Yegnanarayana, B. (2006). *Artificial neural networks.* New Delhi: Prentice-hall of india private limited.