PREDIKSI PRODUKTIVITAS PADI MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Gandhi Ramadhona
NIM: 135150201111256

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

PENGESAHAN

PREDIKSI PRODUKTIVITAS PADI MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Gandhi Ramadhona
NIM: 135150201111256

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Budi Darma Setiawan, S.Kom., M.Cs.	Dr.Eng. Fitra A. Bachtiar, S.T., M.Eng.
NIP: 198410152014041002	NIK: 2012018406281001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 197105182003121001

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 ayat 2 dan Pasal 70).

Malang, 27 Juli 2018

Gandhi Ramadhona

NIM: 135150201111256

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT, karena dengan rahmat dan karunia-Nya penulis dapat menyelesaikan tugas akhir yang berjudul **"Prediksi Produktivitas Padi Menggunakan Jaringan Syaraf Tiruan Backpropagation"** dengan baik.

Shalawat dan salam selalu penulis haturkan untuk baginda besar Baginda Muhammad SAW, yang dengan semangat juangnya terus memberikan tauladan untuk tetap bersemangat dan tidak berputus asa dengan rahmat-Nya. Maka tanpa itu semua, niscaya penulis tidak dapat menyelesaikan tugas akhir dengan baik.

Penulisan dan penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu diantaranya:

1. Kedua orang tua, Bapak Eldison & Ibu Anyta yang memberikan dukungan, motivasi, do'a, serta kasih sayang selama ini.
2. Bapak Budi Darma Setiawan, S.Kom., M.Cs. selaku dosen pembimbing I yang telah banyak memberikan ilmu dan saran dalam menyelesaikan tugas akhir ini.
3. Bapak Dr.Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng. selaku dosen pembimbing II yang telah banyak memberikan ilmu dan saran dalam menyelesaikan tugas akhir ini.
4. Ibu Lailil Muflikhah, S.Kom., M.Cs. selaku dosen pembimbing akademik pertama yang selalu memberikan kritik dan saran serta bimbingan dalam pelaksanaan kuliah.
5. Ibu Ratih Kartika Dewi, S.T., M.Kom. selaku dosen pembimbing akademik kedua yang selalu memberikan kritik dan saran serta bimbingan dalam pelaksanaan kuliah.
6. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
7. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
8. Mita Febriani yang tidak pernah lelah memberikan dukungan, saran, do'a, dan mendorong penulis untuk terus semangat dalam menyelesaikan tugas akhir ini.
9. Teman – teman seperjuangan TIF – N yang selalu membantu, mendukung, mendo'akan, dan memberikan saran dalam menyelesaikan tugas akhir ini.
10. Keluarga BIOS terima kasih telah memberikan warna – warni selama masa kuliah dan terima kasih atas dukungan, semangat, dan do'anya dalam menyelesaikan tugas akhir ini.

11. Teman – teman kosan KKD5A terima kasih atas segala bantuan dan dukungannya selama masa kuliah ini.
12. Teman – teman Agency Red Model yang selalu mendukung dalam menyelesaikan tugas akhir ini.
13. Semua teman – teman FILKOM, khususnya Informatika 2013 terima kasih atas segala bantuan dan dukungannya selama ini.
14. Segenap dosen dan karyawan FILKOM Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
15. Semua pihak yang tidak dapat disebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga jasa dan amal baiknya mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa FILKOM Universitas Brawijaya.

Malang, 21 Februari 2018

Penulis

gandhiramadhona@gmail.com

ABSTRAK

Ramadhona Gandhi. 2018. Prediksi Produktivitas Padi Menggunakan Jaringan Syaraf Tiruan Backpropagation. Fakultas Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Budi Darma Setiawan, S.Kom., M.Cs. dan Dr.Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng.

Beras sangat penting bagi manusia, terutama masyarakat ASEAN. Salah satu negara yang membudidayakan beras adalah Indonesia. Pada tahun 2015, Indonesia merupakan negara ASEAN yang menduduki peringkat ketiga tertinggi dalam hal penghasil beras terbesar di dunia. Meski demikian Indonesia masih perlu mengimpor beras tiap tahunnya karena tingginya permintaan dan konsumsi perkapita masyarakat Indonesia. Selain itu perbedaan hasil panen ditiap daerah mengakibatkan kelangkaan beras karena tidak optimalnya teknik pertanian yang digunakan. Pada penelitian ini mengimplementasikan metode jaringan syaraf tiruan *backpropagation* untuk meramalkan hasil produktivitas padi. Dalam implementasinya, data dinormalisasi menggunakan *min – max normalization* dan inisialisasi bobot menggunakan *Nguyen – Widrow*. Berdasarkan hasil pengujian parameter untuk metode *backpropagation*, hasil RMSE yang paling minimum yakni 8.6918 dengan nilai parameter *learning rate* = 0.8, *hidden layer* = 3, *hidden neuron* = 4 dengan jumlah *epoch* 10000 terhadap 135 data latih dan 13 data uji. Berdasarkan hasil pengujian 5 *fold cross validation* terhadap kestabilan pengujian data mendapatkan nilai rata – rata RMSE sebesar 8.2126.

Kata kunci: prediksi, produktivitas padi, jaringan syaraf tiruan, *backpropagation*.

ABSTRACT

Ramadhona Gandhi. 2018. *Predicting the Productivity of Rice Using Artificial Neural Network Backpropagation*. Faculty of computer Science, University of Brawijaya, Malang. Advisor: Budi Darma Setiawan, S.Kom., M.Cs. and Dr.Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng.

Rice is very important for human beings, especially to the ASEAN community. Indonesia is one of the ASEAN countries that cultivate rice. In 2015, Indonesia ranked as the third-highest in terms of the world's largest rice producer. However, Indonesia still have to import rice every year due to its high demand and to fulfil Indonesian's per-capita consumption. The other reason is the different amount of harvest on each areas resulting in a scarcity of rice because the country can not be able to optimize the farming techniques that are used. This research use the methods of backpropagation neural network to predict the results of the rice productivity. In its implementation, the data is normalized using the min – max normalization and weighting initialization using Nguyen – Widrow. Based on the results of testing the parameters for the method of backpropagation, shows the most minimum RMSE i.e. 8.6918 with parameter values learning rate = 0.8, hidden layer neurons, hidden = 3 = 4 with the number of epoch 10000 against 135 training and 13 test data. Based on result of 5 fold cross validation against the stability testing data gets an average RMSE of 8.2126.

Keyword: *predicting, rice productivity, articial neural network, backpropagation*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Salah satu makanan pokok adalah beras. Beras merupakan bagian bulir padi yang telah dipisah dari sekam. Beras sangat penting bagi manusia, terutama masyarakat ASEAN. Faktanya mayoritas masyarakat ASEAN mengkonsumsi beras dan ASEAN merupakan benua penghasil beras terbanyak di dunia (Riaddy, 2015). Di wilayah ASEAN beras sangat cocok dibudidayakan karena memiliki iklim yang hangat dan curah hujan yang tinggi.

Salah satu negara yang membudidayakan beras adalah Indonesia. Pada tahun 2015, Indonesia merupakan negara ASEAN yang menduduki peringkat ketiga tertinggi dalam hal penghasil beras terbesar di dunia (Riaddy, 2015). Jawa Timur, Jawa Barat, Jawa Tengah, Sulawsi Selatan, dan Sumatera Selatan merupakan 5 provinsi penghasil produksi beras tertinggi di Indonesia (Subagya, et al., 2015). Walaupun Indonesia menduduki peringkat ketiga tertinggi dalam hal penghasil beras terbesar di dunia, Indonesia masih perlu mengimpor beras tiap tahunnya. Keadaan itu disebabkan karena tingginya permintaan dan konsumsi perkapita masyarakat Indonesia (150 kilogram/orang/tahun). Selain itu perbedaan hasil panen ditiap daerah mengakibatkan kelangkaan beras karena tidak optimalnya teknik – teknik pertanian yang digunakan (Cox, et al., 2017). Sehingga pemerintah dituntut untuk terus meningkatkan hasil produktivitas padi guna memenuhi kebutuhan masyarakat Indonesia yang mayoritasnya mengkonsumsi beras.

Meningkatnya hasil produktivitas padi disuatu wilayah dikarenakan beberapa faktor pendukung diantaranya letak geografis dan iklim. Letak geografis dan iklim Indonesia sangat cocok dan mendukung dalam budidaya tanaman padi. Curah hujan yang tinggi serta udara yang hangat sangat cocok untuk melakukan bercocok tanam. Faktor – faktor lain yang mempengaruhi produksi padi adalah curah hujan, kecepatan angin, dan suhu. Perubahan iklim seperti suhu, curah hujan, dan kecepatan angin dapat mempengaruhi hasil dan kualitas dari pertanian (Hidayati, Aldrian, Sucahyono, Abdurrahim, Surtiari, & Yogaswara, 2017). Untuk mendapatkan hasil produksi padi yang optimal, faktor lain yang mempengaruhinya adalah penyinaran matahari. Sinar matahari sangat berpengaruh pada fase awal penanaman bibit padi karena dapat meningkatkan jumlah isi gabah sehingga hasil produktivitas padi menjadi optimal atau meningkat (Amrullah, Sopandie, Sugianta, & Junaedi, 2014). Tidak optimalnya hasil produktivitas padi disebabkan oleh hama dan penyakit tanaman. Hama dan penyakit tamanaman itu muncul ketika perubahan suhu dan kelembaban udara, sehingga kelembaban menjadi faktor penting untuk meningkatkan hasil produktivitas padi (Ruminta, 2016).

Salah satu penelitian dalam meningkatkan hasil produtivitas padi adalah penelitian (Amrullah, Sopandie, Sugianta, & Junaedi, 2014). Dewasa ini mengkaji tentang “Meningkatkan Produktivitas Tanaman Padi melalui Pemberian Nano

Silika". Pada penelitian ini melihat pengaruh unsur hara *silika* (Si) dalam dalam ukuran nano yang disolasi dari sekam padi terhadap pertumbuhan, respon morfologi, dan fisiologi serta produktivitas. Berdasarkan hasil penelitian dapat disimpulakn pemberian nano *silika koloid* 20 ppm dan 30 ppm secara umum memberikan pengaruh yang terbaik pada pertumbuhan, respon morfologi, fisiologi dan produktivitas tanaman padi kecuali pada jumlah stomata.

Pada saat ini banyak sekali penelitian yang memanfaatkan teknologi untuk memudahkan pekerjaan manusia. Salah satunya adalah Jaringan Syaraf Tiruan (JST). JST merupakan salah satu cabang dari ilmu kecerdasan buatan yang bermodelkan jaringan syaraf pada manusia (Suhartanto, Dewi, & Muflikhah, 2017). Banyak metode yang ada dalam JST salah satunya *Backpropagation*. *Backpropagation* baik dalam masalah peramalan. Itu terbukti pada penelitian yang dilakukan oleh Dewi & Muslikh (2013) yang membandingkan metode *Backpropagation Neural Network* (BPNN) dengan *Adaptive Neuro Fuzzy Inference* (ANFIS) terhadap prediksi cuaca. Penelitian ini membuktikan bahwa BPNN sangat baik digunakan untuk peramalan yang menghasilkan prediksi cuaca dengan akurasi sebesar 89,56% dari data Karangploso dan 94,90% dari data Banyuwangi. Sedangkan dengan metode ANFIS hanya menghasilkan prediksi cuaca dengan akurasi sebesar 76,93% untuk data Karangploso dan 44,38% untuk data Banyuwangi.

Pada pelatihannya *backpropagation* sering menghasilkan nilai yang jauh dari konvergen, artinya nilai yang didapat jauh dari target yang diinginkan. Untuk mendapatkan nilai yang konvergen (nilai yang diinginkan), perlu mendapatkan hasil parameter optimal yaitu: *learning rate*, *hidden layer*, *neuron* pada *hidden layer*, dan *epoch*. Pada penelitian Rahmat, et al. (2006) penentuan *learning rate* sangat berpengaruh terhadap cepat-lambatnya proses *learning*, *epoch* yang dibutuhkan, dan nilai yang diinginkan (konvergen). Besarnya nilai *learning rate* akan membuat proses learning semakin cepat namun, nilai yang dihasilkan tidak stabil dan mengakibatkan nilai akurasi menurun tajam pada awal *epoch* atau naik turun tidak terkendali (Ramadhiat, Tritoasmoro, & Wijayanto, 2016). Selain itu, penambahan jumlah *layer* sering kali membuat pelatihan menjadi lebih mudah. Namun terkadang semakin besar arsitektur jaringan maka akan menjadi semakin kompleks. *Hidden layer* memiliki *node* (*neuron*) yang mana *node* – *node* tersebut saling terhubung dengan *hidden layer* dengan *hidden layer*, *input layer* ke *hidden layer*, dan *hidden layer* ke *output layer*. Dalam penentuan jumlah *node* pada *hidden layer*, belum ada formula khusus yang bisa menemukan atau menentukan jumlah *neuron* pada *hidden layer* yang optimal (Ramadhiat, Tritoasmoro, & Wijayanto, 2016). Selain itu faktor optimasi inisialisasi bobot sangat mempengaruhi proses pelatihan pada *backpropagation*. Pada penelitian Zamani, et al (2012) tentang "implementasi algoritma genetika pada struktur *backpropagation neural network* untuk klasifikasi kanker payudara" menyimpulkan bahwa inisialisasi bobot awal menggunakan metode *Nguyen Widrow* terbukti menghasilkan rata – rata akurasi lebih baik dari pada inisialisasi bobot awal secara acak.

Berdasarkan penjabaran referensi dan permasalahan penelitian, penulis mengusulkan sebuah penelitian yang berjudul "Peramalan Hasil Produktivitas Padi Menggunakan Jaringan Syaraf Tiruan *Backpropagation*".

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah yang akan dikaji adalah sebagai berikut:

1. Bagaimana pengaruh perbedaan *learning rate* yang digunakan pada prediksi produktivitas padi?
2. Bagaimana pengaruh variasi jumlah *hidden layer* pada proses prediksi produktivitas padi?
3. Bagaimana pengaruh variasi *hidden neuron* pada *hidden layer* terhadap proses prediksi produktivitas padi?
4. Bagaimana pengaruh perubahan data latih dan data uji terhadap hasil prediksi produktivitas padi?

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Melihat dan mengetahui pengaruh *learning rate* serta nilai optimal terhadap proses prediksi produktivitas padi.
2. Melihat dan mengetahui pengaruh jumlah *hidden layer* serta variasi optimal pada proses prediksi produktivitas padi.
3. Melihat dan mengetahui pengaruh *hidden neuron* serta variasi optimal pada proses prediksi produktivitas padi.
4. Melihat pengaruh perubahan data latih dan data uji terhadap hasil prediksi produktivitas padi.

1.4 Manfaat

Hasil penelitian ini dapat digunakan oleh organisasi atau instansi yang terkait mengenai pertanian untuk meramalkan hasil produktivitas padi sehingga dapat bermanfaat oleh masyarakat khususnya petani yang ada di Indonesia.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Menggunakan data dari Badan Pusat Statistik (BPS) yang diperoleh dari website *bps.go.id*.
2. Data yang digunakan hanya terbatas dari tahun 2011 – 2015.

1.6 Sistematika Pembahasan

Sistematika penulisan diuraikan dengan tujuan untuk memberi gambaran, uraian, dan penyusunan tugas akhir secara garis besar.

BAB I: PENDAHULUAN

Berisi tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika pembahasan.

BAB II: LANDASAN KEPUSTAKAAN

Menguraikan teori – teori (objek, metode, algoritma) yang digunakan dan menjadi acuan/referensi dalam penelitian tugas akhir.

BAB III: METODOLOGI PENELITIAN

Berisi tentang bagaimana menganalisis sumber pengetahuan, membahas tentang perancangan sistem, dan analisa tabel pengujian.

BAB IV: PERANCANGAN

Berisi tentang proses dan langkah – langkah implementasi sistem.

BAB V: IMPLEMENTASI

Berisi tentang spesifikasi sistem dalam implementasi dan implementasi algoritma yang digunakan dalam penelitian tugas akhir.

BAB VI: PENGUJIAN DAN ANALISIS

Berisi tentang hasil pengujian beserta analisis yang telah diterapkan pada sistem.

BAB VII: PENUTUP

Berisi tentang kesimpulan dan saran untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Indonesia merupakan salah satu penghasil beras terbesar di dunia. Walau demikian, hampir tiap tahun Indonesia masih tetap mengimpor beras guna menjaga tingkat cadangan beras nasional. Memiliki jumlah penduduk yang tinggi serta permintaan yang sangat banyak, mengharuskan pemerintah untuk menjaga hasil produktivitas padi.

Penelitian yang dilakukan oleh Asnawi (2014) mengkaji tentang produktivitas padi. Peningkatan produktivitas padi pada saat ini dilakukan dengan pendekatan pengolahan tanaman (padi) terpadu (PTT). Kegiatan penelitian menggunakan 180 unit sampel untuk dibandingan hasil produktivitas padi dan pendapatan petani. Terdiri dari 60 unit lokasi SLPTT LL VUB(Sekolah Lapangan Tanaman Terpadu Varietas Luas Lahan Unggul Baru), 60 unit lokasi SLPTT LL non VUB, dan 60 unit lokasi non SLPTT. Hasil penelitiannya adalah produtivitas rata-rata padi sawah pada lokasi SLPTT LL VUB lebih tinggi dari lokasi SLPTT LL non VUB dan non SLPTT. Penggunaan varietas unggul pada lokasi SLPTT LL VUB meningkatkan produktivitas sebesar 8,85% dibandingkan dengan SLPTT LL non VUB dan 47,13% dibandingkan dengan lokasi non SLPTT. Melalui penerapan VUB pada lokasi SLPTT LL VUB mampu meningkatkan pendapatan petani sebesar 29,07% sampai 76,12%.

Penelitian yang dilakukan oleh Rachman, et al (2018) mengkaji tentang peralaman menggunakan jaringan syaraf tiruan *backpropagation*. Penelitian tersebut menggunakan data produksi gula pada PG Candi Baru Sidoarjo. Arsitektur JST *backpropagation* yang diusulkan 4 *neuron* pada *input layer*, 3 *hidden layer*, dan 1 *output layer* yang menghasilkan nilai MAPE sebesar 16.98% dengan nilai iterasi maksimum 800 dan nilai *learning rate* 0.4.

Pada penelitian Suhartanto, et al (2017) *backpropagation* digunakan untuk mendiagnosis penyakit pada kulit anak. Peneliti mengangkat masalah bawah setiap penyakit memerlukan jenis pengobatan yang berbeda sehingga diperlukannya jaringan syaraf tiruan *backpropagation* untuk mengetahui data lampau dalam mendianosis penyakit kulit pada anak. Data masukkan yang digunakan adalah data biner 0 dan 1 yang diperoleh dari gejala penyakit yang berjumlah 19 dimana nilai akan menjadi 1 jika mengalami gejala dan 0 bila tidak mengalami gejala. Fungsi aktivasi yang digunakan adalah *sigmoid biner*. Dari hasil pengujian penulis didapat bahwa, *hidden neuron* optimal sebanyak 4, *learning rate* 0.4 dan *epoch* maksimum 300000. Hasil rata-rata akurasi dari penelitian adalah 87,22%.

Pada penelitian Chamidah, et al (2012), peneliti meneliti pengaruh Normalisasi Data pada Jaringan Syaraf Tiruan *Backpropagation Gradient Descent Adaptive Gain* (BPGDAG) untuk Klasifikasi. Data peneltiian yang digunakan dalam penelitian ini menggunakan data kanker payudara yang terbagi/diklasifikasikan ke dalam kanker jinak dan kanker ganas. Penelitian ini

menganalisa beberapa metode normalisasi yang nantinya akan mentranformasikan data ke dalam rang 0 dan 1. Beberapa metode normalisasi yang digunakan untuk dibandingkan nilai akurasinya terhadap data klasifikasi kanker payudara adalah *decimal scaling*, *sigmoid, softmax*, *min-max normalization*, *statistical column*, dan *z-core*. Dari hasil penelitian didapatkan bahwa metode normalisasi dengan *min-max normalization* memiliki rata-rata *epoch* paling rendah untuk mencapai konvergensi. Sedangkan untuk perbandingan rata-rata akurasi metode *min-max normalization* juga memiliki akurasi paling tinggi untuk kasus klafisikasi kanker payudara dengan nilai rata-rata 96,86%.

Pada penelitian Zamani, et al (2012) yang berjudul "Implementasi Algoritma Genetika pada Struktur *Backpropagation Neural Network* untuk Klasifikasi Kanker Payudara" mencoba meningkatkan akurasi dari penelitian sebelumnya. Dengan mengkombinasikan jaringan syaraf tiruan dan algoritma genetika menghasilkan nilai rata – rata sebesar 97,00% untuk studi kasus deteksi kanker payudara. Dalam penelitian ini inisialisasi bobot awal menggunakan nguyen widrow sangat baik digunakan.

Pada penelitian Kholis (2015) dengan judul "Analisis Variasi Parameter *Backpropagation Artificial Neural Network* terhadap Pengenalan Pola Data Iris". Dewasa ini menggunakan inisialisasi bobot awal menggunakan *Nguyen-Widrow* pada *backpropagation*. Dari hasil penelitian didapat variasi *alpha* terbaik adalah 0,7 dan koefesien momentum terbaik adalah 0,7 yang menghasilkan rata – rata sebesar 96,6667. Sehingga, dapat dikatakan metode ANN *backpropagation* pada kasus pengenalan pola sangat baik.

Dari penelitian yang sudah dijabarkan dapat disimpulkan yakni, hasil produktivitas padi masih sangat rentan terhadap parameter-parameter yang mempengaruhi dan perlu tingkatkan dan metode *backpropagation* sangat handal dalam studi kasus yang bersifat klasifikasi, masih belum ada yang menerapkan JST *backpropagation* untuk per prediksi amalan produktivitas padi, dan perlu melakukan pengujian – pengujian parameter untuk mendapatkan hasil akurasi yang baik. Dari hasil kesimpulan tersebut peneliti akan mencoba membuat suatu prediksi produktivitas padi menggunakan *backpropagation*.

2.2 Padi

Padi erat hubungannya dengan manusia karena setengah dari penduduk dunia mengkonsumsi nasi dari olahan tanaman padi. Hal itu karena padi mengandung karbohidrat yang meningkatkan energi dalam tubuh. Selain mengandung energi yang tinggi padi juga memiliki kandungan serat rendah yang dapat mengobati dan mencegah gangguan pencernaan (Yana, 2017).

Padi memiliki banyak jenis, salah satunya adalah beras merah. Varietas jenis padi ini sangat diunggulkan terhadap kondisi cuaca yang tidak menentu kerena dapat tumbuh di tanah yang berbatu dan tahan akan penyakit tanaman

(Hidayati, Aldrian, Sucahyono, Abdurrahim, Surtiari, & Yogaswara, 2017). Beberapa faktor yang mempengaruhi hasil produktivitas padi yaitu:

Tekanan Udara

Tekanan udara adalah massa yang timbul akibat adanya gravitasi. Karena adanya gravitasi, udara ditarik ke permukaan bumi sehingga menghasilkan berat. Berat inilah yang mempengaruhi segala bentuk aktivitas udara, baik di daratan tinggi, maupun daratan rendah. Pada penelitian Haryanti (2010) mengatakan tekanan udara sangat mempengaruhi kegiatan transpirasi tanaman. Selain mempengaruhi tanaman, tekanan udara juga mempengaruhi faktor angin (Setiawan, 2009).

Kelembaban

Kelembaban merupakan konsentrasi uap air di udara. Kelembaban udara berpengaruh terhadap penguapan pada permukaan tanah dan pengupan pada daun. Meningkatnya lanju transpirasi disebebabkan karena rendahnya kelembaban udara yang mengakibatkan meningkatnya pertumbuhan nutrisi pada tanaman. Sebaliknya, rendahnya laju transipirasi disebabkan tingginya kelembaban udara yang mengakibatkan terhambatnya pertumbuhan nutrisi tanaman. Selain itu, suhu yang berubah – ubah dapat meningkatkan populasi hama dan penyakit tanaman (Ruminta, 2016).

Suhu Rata – rata

Salah satu faktor yang mempengaruhi pertumbuhan dan perkembangan tumbuhan adalah suhu. Pengaruh suhu pada tanaman membantu meningkatkan laju metabolisme, fotosintesi, respirasi, dan transpirasi tumbuhan. Suhu yang tinggi merupakan salah satu isu perubahan iklim yang menyebabkan peningkatan laju transpirasi dan respirasi serta penuaan dini dengan hasil yang rendah (Yuliawan & Handoko, 2012).

Hubungan metabolisme terhadap kecepatan reaksi enzim tumbuhan sangat erat. Laju metabolisme tumbuhan akan stabil jika berada di suhu yang optimum. Suhu optimum dapat mempengaruhi sistem enzim tetap stabil diwaktu yang lama. Dinginnya suhu membuat sistem enzim pada tumbuhan tetap stabil namun tidak berfungsi. Namun, sistem enzim pada tumbuhan akan rusak jika terpengaruh pada suhu tinggi (Setiawan, 2009).

Curah Hujan

Hujan merupakan salah satu faktor yang mempengaruhi hasil produktivitas. Penyebab utama penurunan hasil panen karena berkurangnya intensitas hujan (Hidayati, Aldrian, Sucahyono, Abdurrahim, Surtiari, & Yogaswara, 2017). Hujan memliki dampak yang baik dan buruk bagi tumbuhan. Rusaknya tanaman diakibatkan karena tingginya curah hujan yang mengakibatkan kelebihan air. Selain itu curah yang sangat tinggi menyebabkan fotosintesis tanaman rendah (Setiawan, 2009). Itu terjadi karena kurangnya sinar matahari karena tertutup oleh awan.

Penyinaran Matahari

Tanaman sangat erat kaitannya dengan sinar matahari. Tanaman yang mendapatkan sinar matahari yang cukup akan sangat mudah untuk melakukan proses fotosintesis. Menurut Setiawan (2009) mengatakan peran cahaya matahari sangat penting untuk pertumbuhan dan perkembangan tanaman karena cahaya digunakan tanaman untuk berfotosintesis dan juga sebagai pemicu modulator respon morfogenesis.

Kecepatan Angin

Untuk perkembangbiakan tumbuhan membutuhkan faktor angin. Peranan angin untuk perkembangbiakan tumbuhan adalah untuk penyebaran spora (Setiawan, 2009). Selain itu, angin juga mempengaruhi temperatur dan kelembaban tanah.

Luas Lahan Produksi

Luas lahan produksi padi disetiap wilayah provinsi di Indonesia berbeda – beda. Semakin luas lahan padi maka hasil produksi juga akan meningkat. Pada penelitian yang dilakukan oleh Wahed (2015) tentang "Pengaruh Luas Lahan, Produksi, Ketahanan Pangan dan Harga Gabah Terhadap Kesejahteraan Petani Padi di Kabupaten Pasuruan" mengatakan variabel luas lahan mempunyai pengaruh positif terhadap variabel nilai tukar petani. Pada peneltian Santoso (2015) tentang "Pengaruh Luas Lahan dan Pupuk Bersubsidi Terhadap Produksi Padi Nasional" menyimpulkan bahwa faktor – faktor yang mempengaruhi produksi padi adalah luas lahan sawah, realisasi pupuk urea bersubsidi, realisasi pupuk SP-36 bersubsidi, dan realisasi pupuk ZA bersubsidi.

2.3 Peramalan

Peramalan adalah suatu pengambilan keputusan yang didasari dengan peristiwa masa lalu untuk memprediksi hasil masa depan. Selain itu peramalan juga bisa diartikan sebagai penggunaan data lampau untuk menentukan tres masa depan (Pakaja, Naba, & Purwanto, 2012).

Jenis – jenis peramalan (*forecasting*) berdasarkan horizon waktu dibagi menjadi tiga jenis yaitu (Herjanto, 2008):

1. Peramalan jangka panjang (*long-range forecast*), peramalan ini lebih mencakup ke waktu yang lebih panjang atau lama(1 – 2 tahun). Contoh peramalan ini yaitu simpanan jangka panjang, membangun sesuatu untuk kebutuhan perusahaan, perencanaan untuk kegiatan penelitian dan perkembangan (litbang) dan lain sebagainya.
2. Peramalan jangka menengah (*medium- range forecast*), peramalan ini mencakup waktu yang lebih pendek dari peramalan jangka panjang yaitu sekitar 2 bulan sampai dengan 1 tahun. Contoh peramalan ini banyak digunakan untuk perencanaan produksi dan sebagainya.

3. Peramalan jangka pendek (*short- range forecast*), peramalan ini adalah peramalan yang memakan waktu dekat misalnya memperhatikan target produksi harian suatu perusahaan.

2.4 Min – Max Normalization

Secara garis beras normalisasi adalah bentuk perubahan data menjadi nilai yang selaras. Salah satu normalisasi adalah *min – max normalization*. *Min – max normalization* merupakan strategi normalisasi yang secara linear mengubah x ke $y = (x - min)/(max - min)$, dimana *min* dan *max* adalah nilai minimum dan maksimum dalam x, dimana x adalah himpunan nilai yang diamati dari x. Dengan demikian nilai himpunan x akan berada pada interval 0 – 1 (Fitri, Setyawati, & S, 2013).

Min-max Normalization:

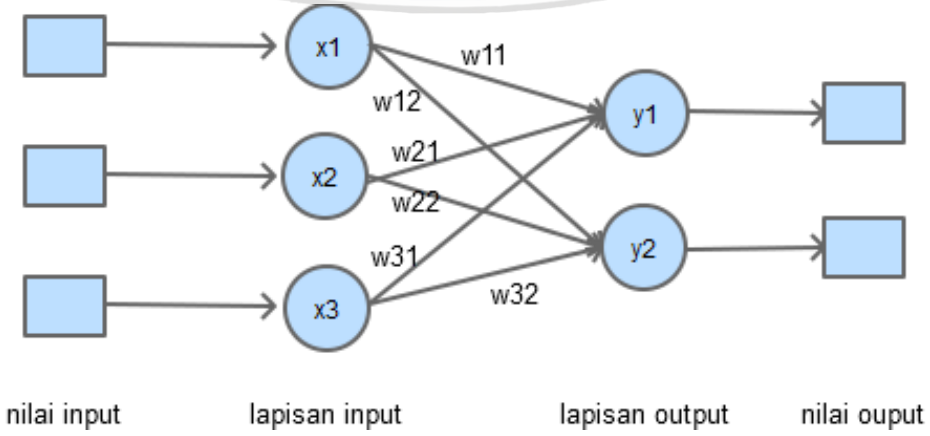
$$S' = \frac{s - \min(s_k)}{\max(s_k) - \min(s_k)} \quad (2\text{-}1)$$

2.5 Jaringan Syaraf Tiruan (*Artificial Neural Network*)

Jaringan syaraf tiruan (JST) adalah serangkaian algoritma yang mencoba untuk mengidentifikasi hubungan yang mendasarinya dalam serangkaian data dengan menggunakan proses yang meniru cara otak manusia beroperasi. JST memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai penelitian. Arsitektur tersebut antara lain (Lesnussa, Latuconsina, & Persulessy, 2015):

1. Jaringan Lapisan Tunggal (*Single Layer Network*)

   Terdari dari 2 lapisan yaitu lapisan *input* dan lapisan *output*. Setiap node atau *neuron* yang ada pada lapisan *input* saling terhubung satu sama lain terhadap lapisan *output*. Jaringan ini hanya menerima *input* pada lapisan *input* kemudian diproses menjadi *output* tanpa harus melalui lapisan tersembunyi. Contoh algoritma yang menggunakan jaringan ini adalah *perceptron*.

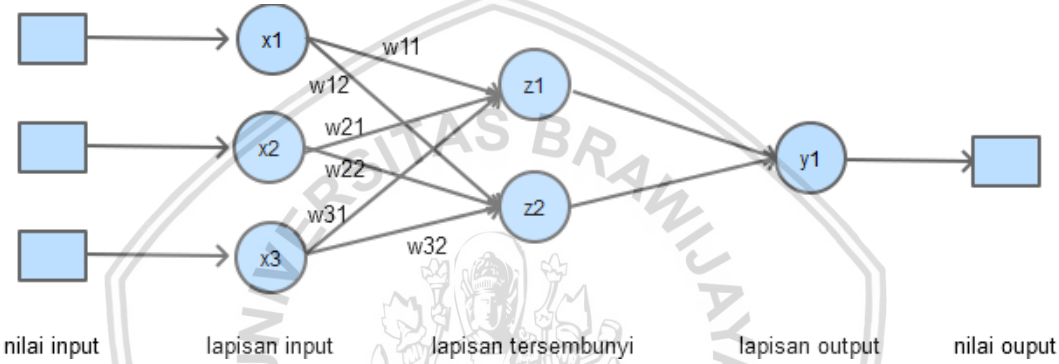


Gambar 2.1 *Single Layer Network*

Sumber: (Lesnussa, Latuconsina, & Persulessy, 2015)

2. Jaringan Lapisan Banyak (*Multilayer Network*)

   Berbeda dengan jaringan lapis tunggal, pada jaringan lapis banyak memiliki 3 lapisan yakni lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Karena memiliki lapisan yang kompleks dari jaringan yang lain, jaringan ini dapat menyelesaikan masalah yang kompleks seperti peramalan, klasifikasi, *recognetion*, dan lain sebagainya. Namun, pada jaringan lapis banyak lebih cenderung memakan waktu banyak dalam waktu prosesnya. Contoh algoritma yang menggunakan jaringan ini adalah *backpropagation*.

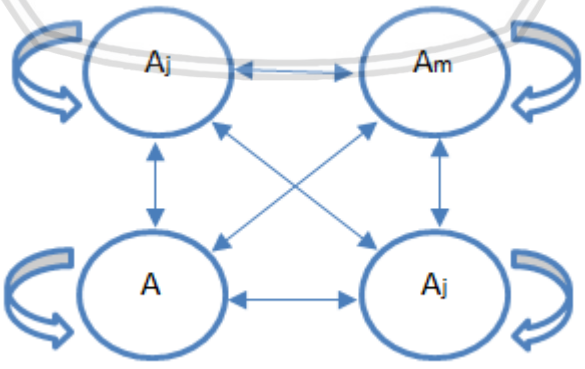


Gambar 2.2 *Multilayer Network*

Sumber: (Lesnussa, Latuconsina, & Persulessy, 2015)

3. Jarigan Lapisan Kompetitif (*Competitive Layer*)

   Lapisan kompetitif adalah aturan yang didasarkan pada gagasan bahwa hanya satu neuron dari iterasi tertentu dalam lapisan tertentu yang akan diaktifkan pada suatu waktu. Contoh jaringan ini adalah LVQ.

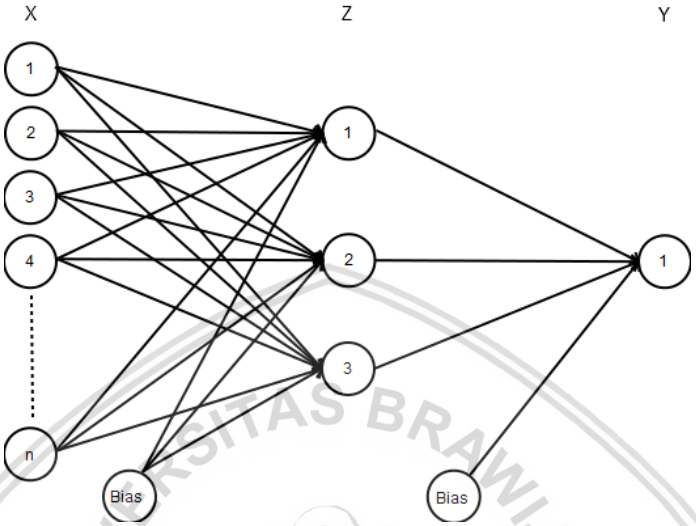


Gambar 2.3 *Competitive Layer*

Sumber: (Lesnussa, Latuconsina, & Persulessy, 2015)

2.5.2 *Backpropagation*

Jaringan Syaraf Tiruan (JST) disusun dengan struktur dan fungsi otak manusia sebagai model untuk ditiru. Pada sebuah jaringan syaraf tiruan terdapat sejumlah *neuron*. Satu *neuron* bisa terhubung ke banyak *neuron* lain, dan setiap koneksi (*link*) tersebut mempunyai bobot (*weight*) (Purwaningsih, 2016).



Gambar 2.4 Arsitektur *Backpropagation*

Sumber: (Dewi & Muslikh, 2013)

Salah satu model yang terdapat pada JST adalah *Backpropagation*. Model ini sering menggunakan *supervised learning* untuk menyelesaikan masalah yang rumit dan dilatih menggunakan metode pembelajaran (Matondang, 2013). *Backpropagation* memiliki proses pembelajaran maju dan perbaikan kesalahan secara mundur. Model jaringan ini sering digunakan untuk proses prediksi, pengenalan dan peramalan (Dewi & Muslikh, 2013). Arsitektur *backpropagation* disajikan pada Gambar 2.4.

2.5.3 Fungsi Aktifasi

Pada JST keluaran suatu neuron menggunakan fungsi aktivasi yang berdasarkan *output* dari pengkombinasi linier. Lebih tepatnya, fungsi ini digunakan untuk mengaktifkan atau menonaktifkan *neuron*. Beberapa jenis – jenis dari fungsi aktivasi (Wiranti, 2013):

1. *Hard Limit* (Tangga Biner)
   Biasa digunakan oleh lapisan tunggal (*Single Layer Network*) dengan masalah yang bersifat linier. Fungsi ini berupa bilangan biner (0 atau 1). Berikut ini persamaan dari fungsi *hard limit*:

$$y = \begin{cases} 0, & jika\ x \leq 0 \\ 1, & jika\ x > 0 \end{cases} \quad (2\text{-}2)$$

2. *Symetric Hard Limit* (Bipolar)
   Fungsi ini pengembagan dari fungsi tangga biner, yang memiliki *output* antara -1, 0, dan 1.

Berikut ini persamaan dari fungsi s*ymetric hard limit*:

$$y = \begin{cases} 1, \ jika\ x > 0 \\ 0, \ jika\ x = 0 \\ -1, \ jika\ x < 0 \end{cases} \quad (2\text{-}3)$$

3. *Threshold*
   Fungsi ini hampir menyerupai fungsi aktivasi tangga biner, namun pada fungsi ini menambahkan nilai *threshold* (θ).
   Berikut ini persamaan dari fungsi *threshold*:

$$y = \begin{cases} 0, \ jika\ x < \theta \\ 1, \ jika\ \ x \geq \theta \end{cases} \quad (2\text{-}4)$$

4. Bipolar *Threshold*
   Fungsi ini pengembagan dari fungsi *threshold*, yang memiliki *output* antara -1 atau 1.
   Berikut ini persamaan dari fungsi bipolar *threshold*:

$$y = \begin{cases} 1, \ jika\ \geq \theta \\ -1, \ jika\ \ x < \theta \end{cases} \quad (2\text{-}5)$$

5. *Linier* (Identitas)
   Fungsi aktifasi ini memiliki keluaran *output* yang sama dengan nilai masukkannya.
   Berikut ini persamaan dari fungsi linier:

$$y = x \quad (2\text{-}6)$$

6. *Sigmoid biner*
   Fungsi ini memiliki *range* nilai antara 0 sampai 1.
   Berikut ini persamaan dari fungsi *sigmoid biner*:

$$y = \ f(x) = \frac{1}{1+e^{-x}} \quad (2\text{-}7)$$

7. *Sigmoid* Bipolar (*Symetric Sigmoid*)
   Fungsi ini hampir menyerupai fungsi *sigmoid biner*, namun pada fungsi ini memiliki *range* keluaran antara 1 sampai -1.
   Berikut ini persamaan dari fungsi *sigmoid* bipolar:

$$y = \ f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2\text{-}8)$$

2.5.4 Tingkat Pembelajaran (*Learning Rate*)

Tingkat pembelajaran atau bisa juga disebut *learning rate* adalah parameter yang mengkontrol laju pembelajaran untuk menyesuaikan bobot dalam model. Semakin rendah nilainya, semakin lambat model untuk melakukan pembelajaran, namun tidak akan melewatkan minima lokal (sampai pada akurasi terbaik). Tetapi jikat nilai tingkat pembelajaran terlalu tinggi proses pembelajaran jaringan akan semakin cepat namun tingkat kepekaan pada model sangat kurang, sehingga hasil yang diperoleh jauh dari yang diharapkan.

Umumnya, nilai tingkat pembelajaran yang digunakan antara 0 sampai 1 (Brian, 2016).

2.5.5 Langkah – langkah *Backpropagation*

Langkah – langkah algoritma *backpropagation* menurut (Andrian & Wayahdi, 2014):

Langkah 0: Melakukan inisialisasi bobot random, menetapkan *epoch*, target *error*, dan tingkat pembelajaran (*learning rate*).

Langkah 1: Inisialisasi kondisi berhenti. Proses pembelajaran berhenti ketika kondisi berhenti telah terpenuhi. Kondisi berhenti biasa menggunakan iterasi (*epoch* <= *epoch* maksimum) atau minimum *error*.

Langkah 2: Untuk setiap data *training*, lakukan step 3 – 9.

FASE I : *Feed forward*

Langkah 3: Setiap unit *input* $(X_i, i = 1, 2, 3, \dots, n)$ menerima sinyal *input* X_i dan menyebarkan sinyal tersebut pada unit *hidden layer*.

Langkah 4: Setiap *hidden* unit $(Z_j, j = 1, 2, 3, \dots, m)$ menjumlahkan sinyal *input* terbobot dan biasnya.

$$Z_{\_inj} = v_{jo} + \sum_{i=1}^{n} x_i v_{ji} \quad (2\text{-}9)$$

Dengan:

$Z_{\_inj}$ = total sinyal yang diterima pada *hidden* unit

X_i = nilai masukkan pada unit

V_{ji} = bobot antara *input* dan *hidden* unit

Hitung sinyal *output* dengan menggunakan fungsi aktivasi yang telah ditentukan.

$$Z_j = f = \left(Z_{net_j}\right) = \frac{1}{1+e^{-z\_net_j}} \quad (2\text{-}10)$$

Dengan:

Z_j = keluaran dari nilai $Z_{\_inj}$

Lalu sinyal *output* dikirimkan keseluruh unit pada unit *output*.

Langkah 5: Setiap *output* unit $Y_k(k = 1, 2, 3 \dots, p)$ menerima sinyal *output* dari *hidden* unit dan menjumlahkan sinyal *output* yang terbobot beserta biasnya.

$$Y_{ink} = W_{ko} + \sum_{j=1}^{p} Z_j W_{kj} \quad (2\text{-}11)$$

Dengan:

Y_{ink} = total sinyal yang diterima pada *output* unit

Z_j = nilai masukan pada *hidden* unit

W_{kj} = bobot antara *hidden* dan *output* unit

Hitung sinyal *output* dengan menggunakan fungsi aktivasi yang telah ditentukan.

$$Y_{out} = f = (Y_{ink}) = \frac{1}{\alpha + e^{-y\_in_k}} \quad (2\text{-}12)$$

Dengan:

Y_{out} = keluaran dari nilai Y_{ink}

Lalu sinyal *output* dikirimkan keseluruh unit *layer* atasnya.

FASE II : *Feed Backward*

Langkah 6: Setiap unit *output* $Y_{out}(k = 1, 2, 3, \dots, p)$ menerima target (*desired output*) yang sesuai dengan *input* data *training* untuk menghitung kesalahan *error* pada *output* unit.

$$\delta_k = (t_k - y_{out}) y_{out}(1 - y_{out}) \quad (2\text{-}13)$$

Dengan :

δ_k = faktor koreksi *error* pada *output* unit

Y_{out} = keluaran pada *output* unit

Kemudian hitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor *error* δ_k digunakan untuk mengkoreksi nilai *error* pada bobot antara *hidden* dan *output* unit (ΔW_{kj}) yang nantinya digunakan untuk memperbaharui bobot W_{kj}.

$$\Delta W_{kj} = \alpha \delta_k Z_j \quad (2\text{-}14)$$

Dengan :

ΔW_{kj} = faktor koreksi *error* pada bobot W_{kj}

α = tingkat pembelajaran (*learning rate*)

Kemudian faktor koreksi *error* δ_k dikirimkan ke *layer* unit yang ada diatasnya.

Langkah 7: Setiap unit tersembunyi Z_j $(j = 1, 2, 3, \dots, m)$:

Menjumlahkan *input delta* yang sudah terbobot

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{kj} \quad (2\text{-}15)$$

Dengan :

δ_in_j = *delta input*

Kemudian hasil dari *delta input* (δ_in_j) dikalikan dengan turunan fungsi aktivasi yang telah ditentukan dan menghasilkan faktor koreksi *error* δ_j.

$$\delta_j = \delta_{in_j} z_j Z(1 - z_j) \quad (2\text{-}16)$$

Dengan :

$\delta_j =$ *delta output*

Hitung koreksi *error* (ΔV_{ji}) yang nantinya akan digunakan untuk memperbaharui V_{ji}.

$$\Delta V_{ji} = \alpha \delta_j x_i \quad (2\text{-}17)$$

Dengan :

$\Delta V_{ji} =$ faktor koreksi *error* pada bobot V_{ji}

FASE III : Perubahan Bobot dan Bias

Langkah 8: Setiap unit *output* Y_k $(k = 1, 2, 3, \dots, p)$ memperbaharui bobot beserta biasnya.

$$W_{jk}(baru) = W_{jk}(lama) + \Delta W_{jk} \quad (2\text{-}18)$$

Setiap unit tersembunyi Z_j $(j = 1, 2, 3, \dots, p)$ memperbaharui bobot beserta biasnya.

$$V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij} \quad (2\text{-}19)$$

Langkah 9: Tes kondisi berhenti.

2.6 Algoritma Nguyen – Widrow

Algoritma Nguyen – Widrow biasa digunakan untuk menginisialisasi bobot pada model JST untuk mengurangi waktu pembelajaran. Berikut ini langkah – langkah dalam inisialisasi bobot *Nguyen Widrow* (Nguyen & Widrow, 1990):

1. Inisialisasi semua bobot dengan *range* (-0,5) – 0,5.
2. Hitung nilai $\left|\left|V_{ji}\right|\right|$. Persamaan $\left|\left|V_{ji}\right|\right|$ dapat dilihat pada persamaan 2-20.

   $$\left|\left|V_{ji}\right|\right| = \sqrt{V_1^2 j + V_2^2 j + \cdots + V_n^2 j} \quad (2\text{-}20)$$

3. Hitung faktor skala β. Persamaan β dapat dilihat pada persamaan 2-21.

   $$\beta = 0{,}7(p)^{1/n} = 0{,}7\sqrt[n]{p} \quad (2\text{-}21)$$

4. Hitung nilai V_{ij}. Persamaan V_{ij} dapat dilihat pada persamaan 2-22.

   $$V_{ij} = \frac{\beta_{ij}(lama)}{\left|\left|V_{ji}\right|\right|} \quad (2\text{-}22)$$

5. Nilai bias diperoleh dari bilangan acak faktor skala β dengan *range* $-\beta - \beta$. Persamaan bias dapat dilihat dari persamaan 2-23.

   $$V_{oj} = bilangan\ acak\ antara - \beta\ dan\ \beta \quad (2\text{-}23)$$

2.7 K – Fold Cross Validation

Pengujian algoritma dengan menggunakan *croos validation* merupakan pengujian dengan mengevaluasi dan membandingkan *learning* algoritma dengan membagi data menjadi 2, data latih dan data uji. Selain itu, *cross validation* menentukan keakuratan suatu algoritma karena menggunakan seluruh basis elemen data untuk digunakan dalam pembelajaran maupun pengujian (Riska, Cahyani, & Rosadi, 2015).

2.8 Akurasi Hasil Pengujian

Akurasi hasil pengujian dihitung guna mengetahui seberapa tepat sistem dalam bekerja. Pada penelitian ini hasil akurasi dihitung menggunakan RMSE. RMSE disini menyatakan ukuran besar kesalahan dari suatu model prakiraan. RMSE disini menghasilkan seberapa besar hasil kesalahan prakiraan suatu model.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2} \quad (2\text{-}25)$$

Dengan:

$A_t =$ nilai target t

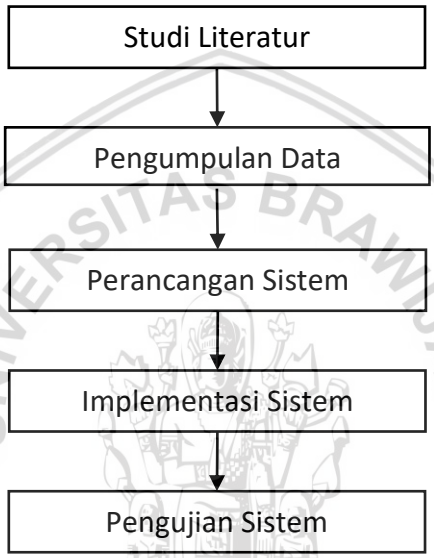
$F_t =$ hasil prediksi t

$n =$ jumlah data

BAB 3 METODOLOGI

Pada bab 3 ini terdapat proses dari pembangunan sistem dalam penelitian "Prediksi Produktivitas Padi Menggunakan Jaringan Syaraf Tiruan *Backpropagation*".

Terdapat beberapa tahap dalam metodologi penelitian, diantaranya studi literatur, pengumpulan data, perancangan sistem, implementasi sistem, dan pengujian sistem. Langkah – langkah pengerjaan penelitian digambarkan pada diagram alir seperti pada Gambar 3.1:



Gambar 3.1 Diagram Alir Metode Penelitian

Berdasarkan Gambar 3.1, langkah – langkah dapat diuraikan sebagai berikut:

1. Studi literatur dikumpulkan untuk mempelajari metode, objek, dan algoritma yang akan digunakan dalam tahap pengerjaan tugas akhir ini.
2. Mengumpulkan data terkait dengan objek yang diteliti pada tugas akhir ini.
3. Melakukan perancangan beserta langkah – langkah dari metode yang digunakan berdasarkan data objek yang diteliti pada tugas akhir ini.
4. Implementasi dilakukan sesuai dengan perancangan yang telah dilakukan pada langkah sebelumnya.
5. Melakukan dan memahami pengujian sistem terhadap implementasi yang telah dilakukan ditahap sebelumnya.

3.1 Studi Literatur

Studi literatur penelitian bertujuan untuk mempelajari konsep yang akan digunakan dalam penelitian ini. Sumber ilmu pengetahuan dari penelitian ini

terdiri dari data – data yang mempengaruhi produktivitas padi yang diperoleh dari berbagai sumber seperti buka, jurnal, penelitian sebelumnya, *e-book*, artikel, dan informasi yang ada di internet. Beberapa hal yang mendukung penelitian ini yaitu:

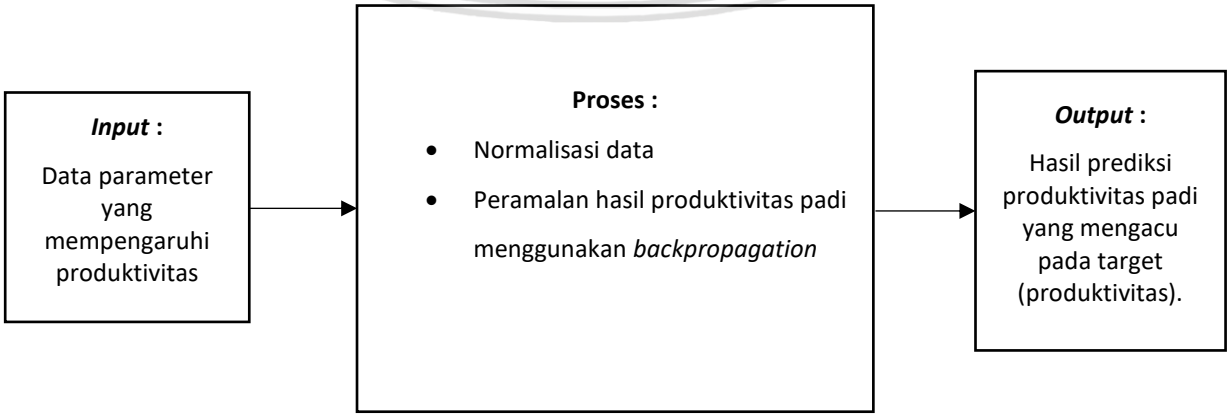
1. Faktor eksternal yang mempengaruhi produktivitas padi yang digunakan untuk prediksi produktivitas padi.
2. Pemahaman tentang konsep Jaringan Syaraf Tiruan *Backpropagation*.
3. Pemahaman tentang fungsi aktivasi, normalisasi dan denormalisasi data, dan algoritma *nguyen – widrow* untuk pembangkitan boobt.
4. Pemehaman tentang bahasa Java untuk megimplementasi algoritma dan metode yang digunakan.

3.2 Pengumpulan Data

Tahap pengumpulan data merupakan tahap dalam mengumpulkan data yang diperlukan untuk melakukan penelitan ini. Data didapatkan dengan cara pendekatan sekunder yang diperoleh dari website Badan Pusat Statistik (*bps.go.id*). Jenis data yang digunakan dari tahun 2011 sampai dengan tahun 2015 yaitu data berkala (*time series*). Data yang digunakan berjumlah 148 dengan parameter sebagai berikut: tekanan udara, kelembaban, suru rata-rata, curah hujan, cahaya (penyinaran matahari), kecepatan angin, luas wilayah produksi, dan produktivitas sebagai target.

3.3 Perancangan Sistem

Perancangan sistem bertujuan untuk memaparkan tahapan atau langkah – langkah dalam membangun pemodelan sistem prediksi produktivitas padi menggunakan jaringan syaraf tiruan *backpropagation*. Rancangan langkah kerja meliputi model dan arsitektur dari sistem, perancangan manual dari sistem, dan acuan pengujian sistem pada perancangan sistem. Gambaran proses dari model perancangan sistem tunjukkan pada Gambar 3.2:



Gambar 3.2 Diagram Blok Model Perancangan Sistem

Tiga proses utama pada perancangan sistem yaitu:

1. ***Input***

   Input merupakan data masukkan berupa parameter-parameter yaitu, parameter cahaya (penyinaran matahari), tekanan udara, parameter curah hujan, parameter kelembaban, kecepatan angin, parameter luas wilayah produksi, parameter suhu rata-rata, dan parameter produktivitas sebagai target.

2. **Proses**

   Proses diawali dengan normalisasi data menggunakan *min – max normalization*. Normalisasi data digunakan untuk meningkatkan konvergensi data. Setelah proses normalisasi data selesai, maka dilanjutkan ke proses peramalan menggunakan *backpropagation*. Pada *backpropagation* akan dilakukan *feedforward*, *backforward*, dan pembaruan bobot dengan inisialisasi bobot awal menggunakan *Nguyen – Widrow*. Pada proses training menggukan 135 data dan pada proses testing menggunakan 13 data.

3. ***Output***

   Keluaran dari sistem berupa hasil prediksi produktivitas padi yang dihitung menggunakan RMSE untuk melihat nilai akurasi pada sistem.

3.4 Implementasi Sistem

Tahap implementasi sistem merupakan tahapan dimana membangun sistem sesuai dengan perancangan yang telah dilakukan sebelumnya. Langkah-langkah implementasi sistem, yaitu:

1. Implementasi metode *backpropagation* untuk melakukan prediksi produktivitas padi.
2. *Output* yang diperoleh berupa hasil prediksi produktivitas padi.

3.5 Pengujian Sistem

Pada tahapan ini, pengujian akurasi dilakukan dengan menggunakan *Root Mean Squared Error* (RMSE). Berikut persamaan untuk RMSE (Herawati, 2013)*:*

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2} \quad (3\text{-}1)$$

Dengan:

$A_t =$ nilai target

$F_t =$ hasil prediksi

$n =$ jumlah data

3.5.1 Pengujian dan Analisis

Beberapa pengujian dilakukan untuk mengetahui keberhasilan sistem terhadap *input*, model, dan arsitektur jaringan menggunakan algoritma *backpropagation*. Pada penelitian ini pengujian terdiri dari: pengujian *learning rate*, pengujian *hidden layer*, pengujian *hidden neuron*, dan pengujian *k-fold cross validation*. Pengujian yang akan dilakukan sebagai berikut:

1. Pengujian *learning rate* dilakukan untuk mengetahui *learning rate* yang optimal yang cocok digunakan dalam model. Tabel 3.1 merupakan tabel yang akan digunakan dalam pengujian perbedaan *learning rate* (α).

Tabel 3.1 Pengujian *Learning Rate*

Learning Rate	***Prosentase Data ,epoch, neuron, hidden layer***					***Rata-rata RMSE***
	RMSE					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	

2. Pengujian untuk mengetahui nilai *hidden layer* optimal terhadap nilai *learning rate* yang didapatkan pada pengujian sebelumnya. Tabel 3.2 merupakan tabel yang akan digunakan dalam pengujian optimal *hidden layer*.

Tabel 3.2 Pengujian *Hidden Layer*

Hidden Layer	***prosentase data, learning rate, neuron, epoch***					***Rata-rata RMSE***
	RMSE					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	

3. Pengujian jumlah *hidden neuron* yang optimal pada *hidden layer*. Tabel 3.3 merupakan tabel yang akan digunakan untuk pengujian *hidden neuron*.

Tabel 3.3 Pengujian *Neuron*

Neuron	***prosentase data, learning rate, hidden layer, epoch***					***Rata-rata RMSE***
	RMSE					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	

4. Pengujian *K-Fold Cross Validation*. Tabel 3.4 merupakan tabel yang akan digunakan untuk pengujian *K-Fold Cross Validation*.

Tabel 3.4 Pengujian *K-Fold Cross Validation*

Learning Rate = 0.8, Hidden Layer = 3, Neuron = 4, Epoch = 10000			
Nomor	**Data Latih**	**Data Uji**	**Hasil RMSE**
Percobaan 1			
Percobaan 2			
Percobaan 3			
Percobaan 4			
Percobaan 5			
Rata - rata RMSE			

BAB 4 PERANCANGAN

4.1 Perhitungan *Backpropagation*

Dalam mengimplementasikan jaringan syaraf tiruan *backpropagation* untuk prediksi produktivitas padi ini, terdapat beberapa parameter yang dapat diubah – ubah, yaitu *learning rate*, *hidden layer*, *hidden neuron*, dan porsentase data. Arsitektur jaringan yang digunakan dalam penelitian ini adalah *multilayer perceptron*. Terdiri dari tiga jenis *layer*, yaitu *input layer*, *hidden layer*, dan *output layer*.

4.1.1 *Feed Forward*

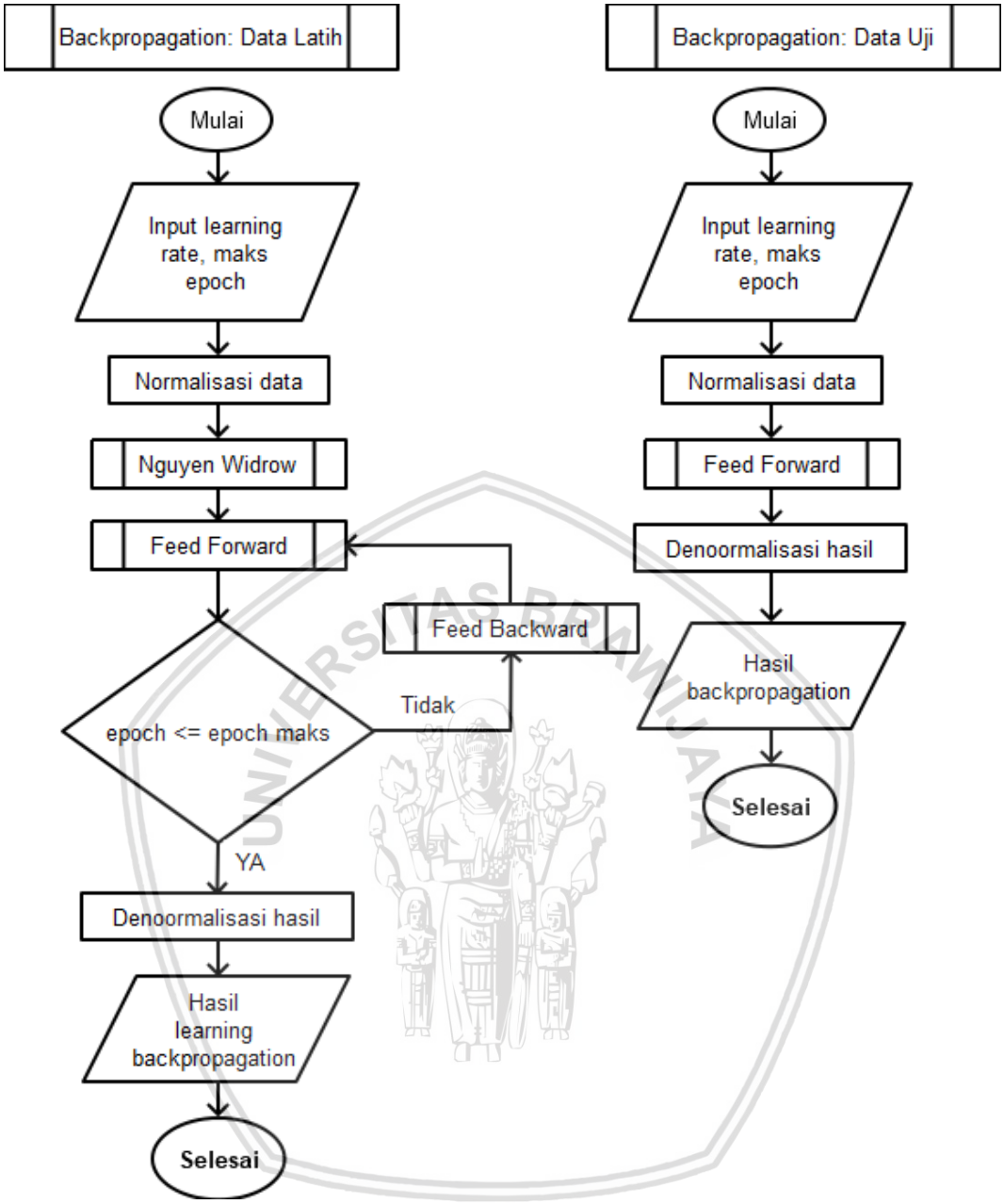
Fase *feed forward* merupakan fase aliran maju dari *input layer* ke *output layer*. Pada fase ini seluruh nilai *input* dan bobot akan dialirkan ke tiap – tiap *neuron* yang saling terhubung satu sama lain. Pada masing – masing *neuron* juga akan melakukan proses untuk mendapatkan nilai keluaran dari *neuron*.

4.1.2 *Feed Backward*

Pada fase *feed backward* akan melakukan proses koreksi bobot dan proses perbaikan bobot yang dilakukan mundur dari *output layer* menuju *input layer*. Sebelum melakukan koreksi bobot, pada fase ini melakukan proses untuk mencari faktor kesalahan pada hasil *output layer*. Kesalahan tersebut dihasilkan berdasarkan faktor pengurangan dan perkalian antara target *input* dengan keluaran dari *output layer*. Faktor kesalahan keluaran atau yang biasa disebut *delta error* juga dikirimkan ke *layer* diatasnya untuk mencari nilai *delta input* dan *delta output* yang nantinya digunakan untuk memperbaharui bobot.

4.1.3 Diagram Alir

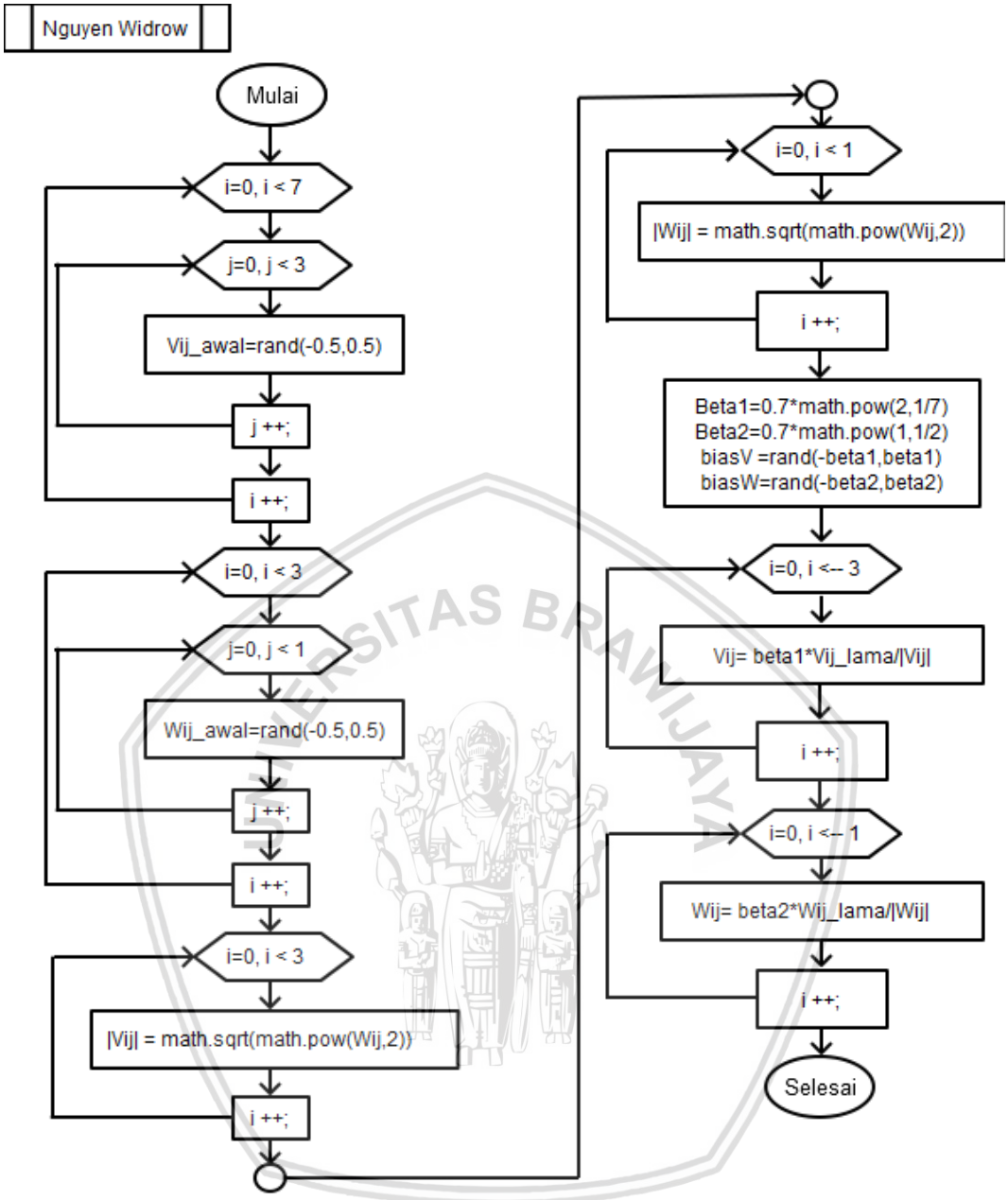
Diagram alir adalah suatu bagan dengan simbol – simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program. Fungsinya adalah untuk memudahkan dalam mengimplementasikan tahap – tahap algoritma pada bab implementasi. Gambar 4.1 merupakan gambar Diagram alir sistem.



Gambar 4.1 Diagram Alir Sistem

4.1.3.1 Nguyen – Widrow

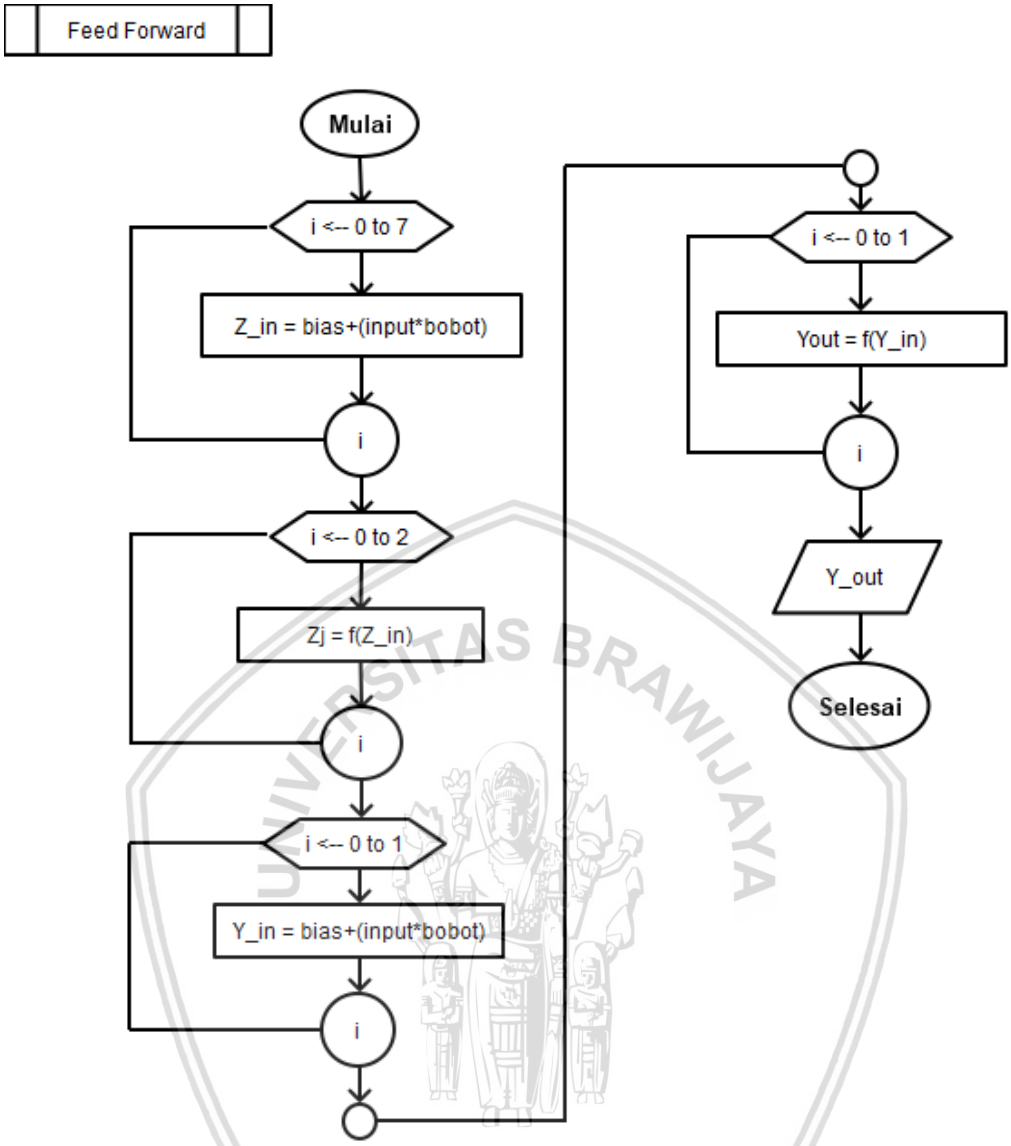
Pada penelitian ini, pembangkitan atau inisialisasi bobot awal menggunakan algoritma *nguyen – widrow*. Algoritma ini menginisialisasi seluruh bobot dan bias, baik bobot antara *input* ke *hidden*, dari *hidden* ke *hidden*, maupun dari *hidden* ke *output*. Proses pembangkitan bobot *nguyen – widrow* disajikan pada Gambar 4.2.



Gambar 4.2 Diagram Alir *Nguyen - Widrow*

4.1.3.2 *Feed Forward*

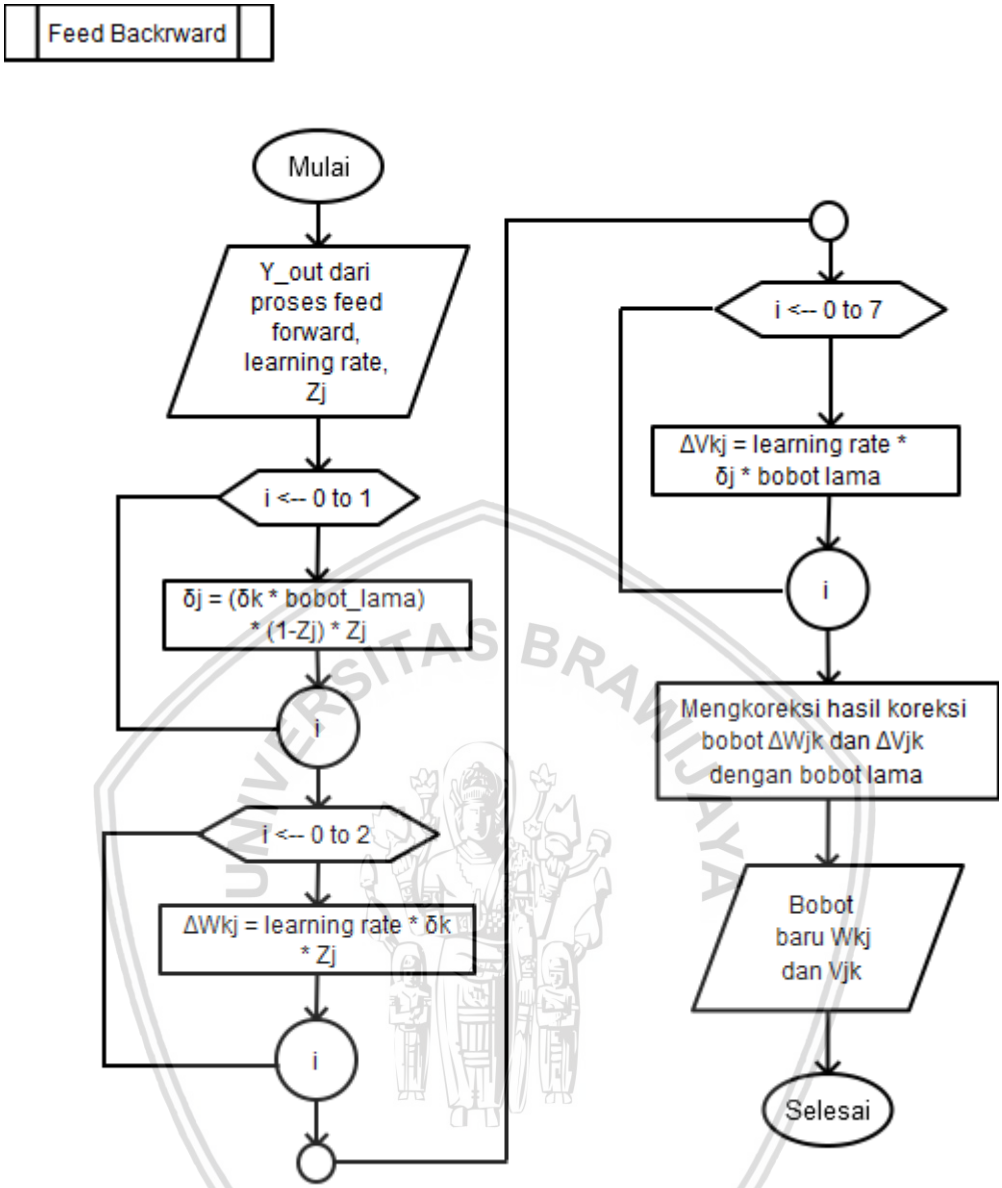
Fase awal dalam proses *learning backpropagation* adalah proses *feed forward*. *Feed forward* merupakan perhitungan aliran maju dari *input* menuju *output*. Proses dari *feed forward* disajikan pada Gambar 4.3.



Gambar 4.3 Diagram Alir *Feed Forward*

4.1.3.3 *Feed Backward*

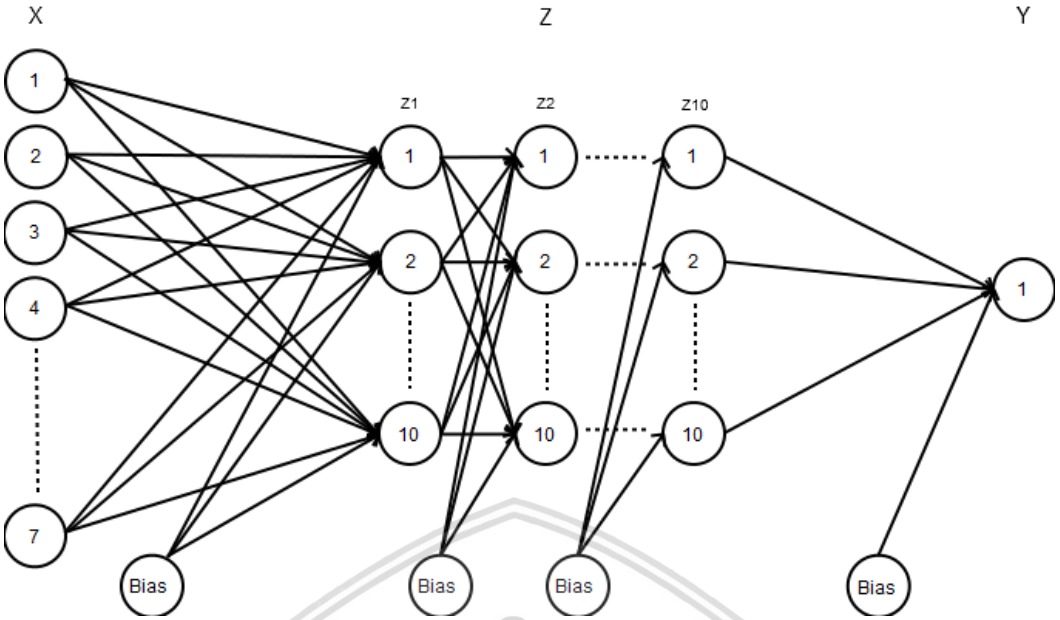
Fase selanjutnya setelah fase *feed forward* adalah fase *feed backward*. Perhitungan *feed backward* hampir menyerupai *feed forward* namun, pada fase *feed backward* perhitungan dimulai dari belakang (*output*) menuju ke depan (*input*). Proses dari *feed backward* disajikan pada Gambar 4.4.



Gambar 4.4 Diagram Alir *Feed Backward*

4.2 Perhitungan Manual

Perhitungan manual menggambarkan perancangan sistem dalam penelitian ini. Perhitungan manual ini nantinya akan menjadi pedoman dalam mengimplementasikan metode dan algoritma yang digunakan dalam implementasi. Pada perhitungan manual ini menggunakan 3 data latih dan 1 data uji dengan 7 node pada input layer yang merupakan parameter – parameter yang mempengaruhi hasil produktivitas padi seperti tekanan udara, penyinaran matahari, curah hujan, kecepatan angin, kelembaban, luas wilayah panen, dan suhu rata-rata. Serta 3 node pada hidden layer dan 1 node pada output layer. Struktur jaringan backpropagation yang digunakan dalam perhitungan manual ini disajikan pada Gambar 4.5.



Gambar 4.5 Arsitektur *Backpropagation*

Keterangan: X adalah *input layer*, Z adalah *hidden layer*, dan Y adalah *output layer*.

X_1= Tekanan Udara

X_2= Penyinaran Matahari

X_3= Curah Hujan

X_4= Kecepatan Angin

X_5= Kelembaban

X_6= Luas Wilayah Panen

X_7= Suhu Rata-rata

Input yang digunakan pada manualisasi ini adalah sebagai berikut:

- Tingkat pembelajaran (*learning rate)* = 0,3
- Maksimum *epoch* = 1
- Fungsi aktivasi = *sigmoid biner*

Tabel 4.1 Data *Learning* dan Data *Testing*

	Data *Learning*								
Nomor	**Nama**	X_1	X_2	X_3	X_4	X_5	X_6	X_7	**target**
1	Data1	0,94	0,47	0,18	0,25	0,34	0,18	0,73	0,60
2	Data2	0,93	0,31	0,42	0,28	0,16	0,07	0,71	0,36
3	Data3	0,95	0,50	0,40	0,28	0,45	0,07	0,69	0,46
	Data *Testing*								
Nomor	**Nama**	X_1	X_2	X_3	X_4	X_5	X_6	X_7	**target**
1	Data4	0,94	0,50	0,47	0,14	0,53	0,36	0,76	0,52

Berikut ini merupakan tahapan membangkitkan bobot *nguyen – widrow*.

a. Menginisialisasi bobot awal acak dengan *range* (-0,5) – 0,5 seperti pada Tabel 4.2:

Tabel 4.2 Inisialisasi bobot awal acak *input* ke *hidden layer*

V_{ij}	Z_1	Z_2	Z_3
X_1	0,3	-0,3	-0,39
X_2	-0,4	0,1	0,29
X_3	0,3	-0,4	0,20
X_4	0,2	-0,5	-0,13
X_5	0,3	0,1	-0,36
X_6	0,0	-0,3	0,10
X_7	0,3	0,3	0,00
Bias	0,4	0,1	-0,12

Tabel 4.3 Inisialisasi bobot awal acak *hidden* ke *output layer*

W_{ij}	Y_1
Z_1	-0,5
Z_2	0,5
Z_3	0,1
Bias	0,1

b. Menghitung nilai $\left|\left|V_{ij}\right|\right|$ dengan bobot V_{ij} acak sebelumnya.

Contoh perhitungan untuk $\left|\left|V_{i1}\right|\right|$:

$$\left|\left|V_{i1}\right|\right| = \sqrt{v_{11}^2 + v_{21}^2 + v_{31}^2 + v_{41}^2 + v_{51}^2 + v_{61}^2 + v_{71}^2}$$

$= \sqrt{0{,}3^2 + (-0{,}4)^2 + 0{,}3^2 + 0{,}2^2 + 0{,}3^2 + 0{,}0^2 + 0{,}3^2}$

$= 0{,}884$

Proses dilakukan hingga $||V_{i2}||$ dan $||V_{i3}||$ didapatkan. Hasil $\left|\left|V_{ij}\right|\right|$ disajikan pada Tabel 4.4.

Tabel 4.4 Hasil $\left|\left|V_{ij}\right|\right|$

	Z_1	Z_2	Z_3
$\left\|\left\|V_{ij}\right\|\right\|$	0,884	0,947	0,658

c. Menghitung nilai $\left|\left|W_{ij}\right|\right|$ dengan boobt W_{ij} acak pada Tabel 4.3.

$||W_{i1}|| = \sqrt{w_{11}^2 + w_{11}^2 + w_{11}^2}$

$= 0{,}714$

d. Menghitung nilai faktor skala β antara *input layer* dengan *hidden layer*

n adalah jumlah unit *input* = 7 dan p adalah jumlah unit *output* = 3

$\beta_1 = 0{,}7\ \sqrt[n]{p} = 0{,}7\ \sqrt[7]{3} = 0{,}3$

e. Menghitung nilai faktor skala β antara *hidden layer* dengan *ouput layer*

n adalah jumlah unit *input* = 3 dan p adalah jumlah unit *output* = 1

$\beta_2 = 0{,}7\ \sqrt[n]{p} = 0{,}7\ \sqrt[3]{1} = 0{,}2333$

f. Mendapatkan nilai bobot V_{ij} dengan menghitung menggunakan β_1 , $V_{ij}(lama)$, dan $\left|\left|V_{ij}\right|\right|$. Contoh perhitungannya dapat dilihat dibawah ini:

$V_{11} = \frac{\beta_1 V_{11}(lama)}{||V_{i1}||} = \frac{0{,}3 \times 0{,}3}{0{,}884} = 0{,}09$

Tabel 4.5 Bobot Awal V_{ij} *Nguyen – Widrow*

V_{ij}	Z_1	Z_2	Z_3
X_1	0,0943	-0,0915	-0,1779
X_2	-0,1452	0,0294	0,1323
X_3	0,1055	-0,1168	0,0912
X_4	0,0662	-0,1425	-0,0593
X_5	0,1059	0,0402	-0,1642
X_6	0,0003	-0,1086	0,0456
X_7	0,0981	0,1054	0,0000

Proses dilakukan hingga seluruh bobot V_{ij} didapatkan. Hasil bobot awal *nguyen – widrow* disajikan pada tabel 4.5.

Untuk mendapatkan nilai bias, didapatkan dari bilangan acak antara *range* $-\beta_1$ dan β_1 yaitu antara (-0,3) – 0,3. Hasil perhitungan bias disajikan pada Tabel 4.6.

Tabel 4.6 Bias awal *hidden layer* dengan *Nguyen – Widrow*

	Z_1	Z_2	Z_3
Bias	-0,10	0,10	0,23

Menghitung nilai bobot W_{ij} dengan menggunakan bobot acak W_{ij} lama. Di bawah ini merupakan contoh perhitungan bobot W_{11} pada *hidden layer* ke *output layer*:

$$W_{11} = \frac{\beta_2 W_{11}(lama)}{||W_{i1}||} = \frac{0{,}2333 \times (-0{,}5)}{0{,}714} = -0{,}16$$

Proses dilakukan hingga W_{21} dan W_{31} didapatkan. Hasil bobot W_{ij} disajikan pada Tabel 4.7.

Tabel 4.7 Bobot Awal W_{ij} dengan *Nguyen – Widrow*

W_{ij}	Y_1
Z_1	-0,16
Z_2	0,16
Z_3	0,03

Untuk mendapatkan nilai bias, didapatkan dari bilangan acak antara *range* $-\beta_2$ dan β_2 yaitu antara (-0,2333) – 0,2333. Hasil perhitungan bias disajikan pada Tabel 4.8.

Tabel 4.8 Bias awal *output layer* dengan *Nguyen – Widrow*

	Y_1
Bias	0,1

Epoch 1

Data1

1. Mendapatkan nilai Z_{in_j} ditiap *neuron* pada *hidden layer* pertama. Proses dilakukan dengan cara mengalikan *input* X_i pada Data1 dengan bobot V_{ij} kemudian dijumlahkan dengan bias disetiap *neuron*.

   Contoh perhitungan Z_{in_j} Data1 dapat dilihat pada perhitungan Z_{in_1}:

$$Z_{in\_1} = bias + (X_1 \times V_{11}) + (X_2 \times V_{21}) + (X_3 \times V_{31}) + (X_4 \times V_{41}) + (X_5 \times V_{51}) + (X_6 \times V_{61}) + (X_7 \times V_{71})$$

$$= (-0{,}10) + (0{,}94 \times 0{,}0943) + \big(0{,}47 \times (-0{,}152)\big) + (0{,}18 \times 0{,}1055) + (0{,}25 \times 0{,}0662) + (0{,}34 \times 0{,}1059) + (0{,}18 \times 0{,}0003) + (0{,}73 \times 0{,}0981) = 0{,}06$$

Proses dilakukan hingga seluruh Z_{in} didapatkan. Hasil seluruh Z_{in} disajikan pada Tabel 4.9.

Tabel 4.9 Hasil seluruh Z_{in} Data1, *epoch* 1

Z_{in_1}	Z_{in_2}	Z_{in_3}
0,06	0,04	0,08

2. Menghitung keluaran dari Z_{in} (Z_{out}) disetiap *neuron* pada *hidden layer* menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut contoh perhitungan dari Z_j Data1:

$$Z_1 = \frac{1}{(1 + e^{-Z_{in1}})} = \frac{1}{(1 + e^{0,06})} = 0,52$$

Proses dilakukan hingga seluruh Z_j didapatkan. Hasil seluruh Z_j disajikan pada Tabel 4.10.

Tabel 4.10 Hasil seluruh Z_j Data1, *epoch* 1

Z_1	Z_2	Z_3
0,52	0,51	0,52

3. Mendapatkan nilai Y_{in} dengan cara hasil Z_j sebagai masukkan dikalikan dengan bobot W_{ij} pada Tabel 4.7 kemudian dijumlahkan dengan biasnya.

Perhitugan Y_{in} Data1:

$$Y_{in} = bias + (Z_1 \times W_{11}) + (Z_2 \times W_{21}) + (Z_3 \times W_{31})$$
$$= 0,1 + \big(0,52 \times (-0,16)\big) + (0,51 \times 0,16) + (0,52 \times 0,03)$$
$$= 0,116$$

4. Menghitung keluaran dari Y_{in} (Y_{out}) menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut perhitungan dari Y_{out} Data1:

$$Y_{out} = \frac{1}{(1 + e^{-Y_{in}})} = \frac{1}{(1 + e^{-0,116})} = 0,529$$

5. Menghitung kesalahan *output* unit δ_k. Perhitungan dilakukan menggunakan perkalian antara selisih target dan keluaran Y_{out}.

Perhitungan δ_k pada Data1:

$$\delta_k = (target - Y_{out}) \times (1 - Y_{out}) \times Y_{out}$$
$$= (0,60 - 0,529) \times (1 - 0,529) \times 0,529$$
$$= 0,02$$

6. Menghitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor *error* δ_k digunakan untuk mengkoreksi nilai *error* pada

bobot antara *hidden* dan *output* unit (ΔW_{ij}) yang nantinya digunakan untuk memperbaharui bobot W_{ij}.

Perhitungan ΔW_{ij} Data1:

$$\Delta W_{11} = \alpha \times \delta_k \times Z_1 = 0{,}3 \times 0{,}02 \times 0{,}52$$
$$= 0{,}003$$

Proses dilakukan hingga seluruh koreksi bobot ΔW_{ij} didapatkan. Untuk perhitungan koreksi bias didapatkan dari perkalian *learning rate* dengan kesalahan *output* unit (δ_k).

$$\Delta W_{bias} = \alpha \times \delta_k$$
$$= 0{,}3 \times 0{,}02$$
$$= 0{,}006$$

Hasil perhitungan ΔW_{ij} dan ΔW_{bias} disajikan pada Tabel 4.11.

Tabel 4.11 Hasil seluruh koreksi bobot ΔW_{ij} dan ΔW_{bias} Data1, *epoch* 1

ΔW_{ij}	δ_1
Z_1	0,003
Z_2	0,003
Z_3	0,003
Bias	0,006

7. Menghitung delta *input* δ_{in_j} yang sudah terbobot. Delta *input* δ_{in_j} didapatkan untuk menghitung faktor koreksi *error* δ_j. Proses perhitungan delta *input* δ_{in_j} dilakukan dengan menjumlahkan perkalian antara faktor koreksi *error output* unit δ_k dengan bobot W_{ij} pada Tabel 4.7.

   Berikut contoh perhitungan δ_{in_j} Data1:

   $$\delta_{in_1} = (\delta_k \times W_{11})$$
   $$= 0{,}02 \times (-0{,}16)$$
   $$= -0{,}0030$$

   Perhitungan dilakukan hingga δ_{in_2} dan δ_{in_3} didapatkan seperti pada Tabel 4.12:

Tabel 4.12 Hasil seluruh δ_{in_j} Data1, *epoch* 1

δ_{in_1}	δ_{in_2}	δ_{in_3}
-0,0030	0,0030	0,0006

Contoh perhitungan δ_j Data1:

$$\delta_1 = \delta_{in\_1} \times (1 - Z_1) \times Z_1$$
$$= (-0{,}0030) \times (1 - 0{,}52) \times 0{,}52$$

$= -0{,}0008$

Proses dilakukan hingga seluruh δ_j didapatkan. Hasil seluruh δ_j disajikan pada Tabel 4.13.

Tabel 4.13 Hasil seluruh δ_j Data1, *epoch* 1

δ_1	δ_2	δ_3
-0,0008	0,0008	0,0002

8. Menghitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor koreksi *error* δ_j digunakan untuk mengkoreksi nilai *error* pada bobot antara *hidden* dan *input* unit (ΔV_{ij}) yang nantinya digunakan untuk memperbaharui bobot V_{ij}.

   Berikut contoh perhitungan ΔV_{ij} Data1:

$$\Delta V_{11} = \alpha \times \delta_1 \times X_{11}$$
$$= 0{,}3 \times (-0{,}0008) \times 0{,}94$$
$$= -0{,}0002$$

   Proses dilakukan hingga seluruh koreksi bobot ΔV_{ij} didapatkan. Untuk perhitungan koreksi bias didapatkan dari perkalian *learning rate* dengan faktor koreksi *error* (δ_j).

   Perhitungan ΔV_{bias} Data1:

$$\Delta V_{bias\_1} = \alpha \times \delta_1$$
$$= 0{,}3 \times (-0{,}0008)$$
$$= -0{,}0002$$

   Proses dilakukan hingga seluruh ΔV_{ij} dan ΔV_{bias} didapatkan. Hasil ΔV_{ij} dan ΔV_{bias} disajikan pada Tabel 4.14.

Tabel 4.14 Hasil seluruh koreksi bobot ΔV_{ij} dan ΔV_{bias} Data1, *epoch* 1

ΔV_{ij}	δ_1	δ_2	δ_3
X_1	-0,0002	0,0002	4,560E-05
X_2	-0,0001	0,0001	2,285E-05
X_3	0,0000	0,0000	8,542E-06
X_4	-0,0001	0,0001	1,215E-05
X_5	-0,0001	0,0001	1,639E-05
X_6	0,0000	0,0000	8,570E-06
X_7	-0,0002	0,0002	3,517E-05
Bias	-0,0002	0,0002	4,847E-05

9. Menghitung bobot baru V_{ij} dengan menjumlahkan bobot lama V_{ij} pada Tabel 4.5 dengan koreksi bobot ΔV_{ij} pada Tabel 4.14.

   Contoh perhitungan $V_{ij(baru)}$ Data1:

$$V_{11(baru)} = V_{11(lama)} + \Delta V_{11}$$
$$= 0{,}09 + -0{,}0002$$
$$= 0{,}09$$

Proses dilakukan hingga seluruh $V_{ij(baru)}$ didapatkan. Hasil seluruh $V_{ij(baru)}$ disajikan pada Tabel 4.15.

Tabel 4.15 Hasil seluruh bobot baru V_{ij} Data1, *epoch* 1

ΔV_{ij}	Z_1	Z_2	Z_3
X_1	0,0941	-0,0913	-0,1778
X_2	-0,1453	0,0296	0,1323
X_3	0,1055	-0,1168	0,0912
X_4	0,0661	-0,1424	-0,0593
X_5	0,1058	0,0403	-0,1642
X_6	0,0003	-0,1086	0,0456
X_7	0,0979	0,1056	0,0000
Bias	-0,1002	0,1002	0,2300

10. Menghitung bobot baru W_{ij} dengan menjumlahkan bobot lama W_{ij} pada Tabel 4.7 dengan koreksi bobot ΔW_{ij} pada Tabel 4.11.

    Contoh perhitungan $\Delta W_{ij(baru)}$ Data1:

$$W_{11(baru)} = W_{11(lama)} + \Delta W_{11}$$
$$= (-0{,}16) + 0{,}004$$
$$= -0{,}16$$

Proses dilakukan hingga seluruh $W_{ij(baru)}$ didapatkan. Hasil seluruh $W_{ij(baru)}$ disajikan pada Tabel 4.16.

Tabel 4.16 Hasil seluruh bobot baru W_{ij} Data1, *epoch* 1

W_{ij}	Y_1
Z_1	-0,16
Z_2	0,17
Z_3	0,04
Bias	0,11

Selanjutnya proses dilanjutkan pada epoch 1 Data2.

Data2

Pada Data2, perhitungan akan dilakukan menggunakan hasil bobot dan bias baru dari data pertama.

1. Mendapatkan nilai Z_{in_j} ditiap *neuron* pada *hidden layer* pertama. Proses dilakukan dengan cara mengalikan *input* X_i pada Data2 dengan bobot V_{ij} pada Tabel 4.15 kemudian dijumlahkan dengan bias disetiap *neuron*.

   Contoh perhitungan Z_{in_j} Data2 dapat dilihat pada perhitungan Z_{in_1}:

   $$Z_{in\_1} = bias + (X_1 \times V_{11}) + (X_2 \times V_{21}) + (X_3 \times V_{31}) + (X_4 \times V_{41}) + (X_5 \times V_{51}) + (X_6 \times V_{61}) + (X_7 \times V_{71})$$

   $$= (-0{,}10) + (0{,}93 \times 0{,}0941) + \big(0{,}31 \times (-0{,}1453)\big) + (0{,}42 \times 0{,}1055) + (0{,}28 \times 0{,}0661) + (0{,}16 \times 01058) + (0{,}07 \times 0{,}0003) + (0{,}71 \times 0{,}0979) = 0{,}092$$

   Proses dilakukan hingga seluruh Z_{in} didapatkan. Hasil Z_{in} disajikan pada Tabel 4.17.

Tabel 4.17 Hasil selruh Z_{in} Data2, *epoch* 1

Z_{in_1}	Z_{in_2}	Z_{in_3}
0,092	0,009	0,104

2. Menghitung keluaran dari Z_{in} (Z_{out}) disetiap *neuron* pada *hidden layer* menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut contoh perhitungan dari Z_j Data2:

   $$Z_1 = \frac{1}{(1 + e^{-Z_{in1}})} = \frac{1}{(1 + e^{-0{,}092})} = 0{,}523$$

   Proses dilakukan hingga seluruh Z_j didapatkan. Hasil Z_j disajikan pada Tabel 4.18.

Tabel 4.18 Hasil seluruh Z_j Data2, *epoch* 1

Z_1	Z_2	Z_3
0,523	0,502	0,526

3. Mendapatkan nilai Y_{in} dengan cara hasil Z_j sebagai masukkan dikalikan dengan bobot W_{ij} pada Tabel 4.16 kemudian dijumlahkan dengan biasnya.

   Perhitugan Y_{in} Data2:

   $$Y_{in} = bias + (Z_1 \times W_{11}) + (Z_2 \times W_{21}) + (Z_3 \times W_{31})$$

   $$= 0{,}11 + \big(0{,}523 \times (-0{,}16)\big) + (0{,}502 \times 0{,}17) + (0{,}526 \times 0{,}04)$$

   $$= 0{,}124$$

4. Menghitung keluaran dari Y_{in} (Y_{out}) menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut perhitungan dari Y_{out} Data2:

$$Y_{out} = \frac{1}{(1 + e^{-Y_{in}})} = \frac{1}{(1 + e^{-0,124})} = 0,531$$

5. Menghitung kesalahan *output* unit δ_k. Perhitungan dilakukan menggunakan perkalian antara selisih target dan keluaran Y_{out}.

   Perhitungan δ_k pada Data2:

$$\delta_k = (target - Y_{out}) \times (1 - Y_{out}) \times Y_{out}$$
$$= (0,36 - 0,531) \times (1 - 0,531) \times 0,531$$
$$= -0,043$$

6. Menghitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor *error* δ_k digunakan untuk mengkoreksi nilai *error* pada bobot antara *hidden* dan *output* unit (ΔW_{ij}) yang nantinya digunakan untuk memperbaharui bobot W_{ij}.

   Perhitungan ΔW_{ij} Data2:

$$\Delta W_{ij} = \alpha \times \delta_k \times Z_1 = 0,3 \times (-0,043) \times 0,523$$
$$= -0,0068$$

   Proses dilakukan hingga seluruh koreksi bobot ΔW_{ij} didapatkan. Untuk perhitungan koreksi bias didapatkan dari perkalian *learning rate* dengan kesalahan *output* unit (δ_k).

$$\Delta W_{bias} = \alpha \times \delta_k$$
$$= 0,3 \times (-0,043)$$
$$= -0,0130$$

   Hasil perhitungan ΔW_{ij} dan ΔW_{bias} disajikan pada Tabel 4.19.

Tabel 4.19 Hasil seluruh koreksi bobot ΔW_{ij} dan ΔW_{bias} Data2, *epoch* 1

ΔW_{ij}	δ_1
Z_1	-0,0068
Z_2	-0,0065
Z_3	-0,0068
Bias	-0,0130

7. Menghitung delta *input* δ_{in_j} yang sudah terbobot. Delta *input* δ_{in_j} didapatkan untuk menghitung faktor koreksi *error* δ_j. Proses perhitungan delta *input* δ_{in_j} dilakukan dengan menjumlahkan perkalian antara faktor koreksi *error output* unit δ_k dengan bobot W_{ij} pada Tabel 4.16.

Berikut contoh perhitungan δ_{in_j} Data2:

$$\delta_{in_1} = (\delta_k \times W_{11})$$
$$= (-0{,}043) \times (-0{,}16)$$
$$= 0{,}0069$$

Perhitungan dilakukan hingga δ_{in_2} dan δ_{in_3} didapatkan seperti pada Tabel 4.20:

Tabel 4.20 Hasil seluruh δ_{in_j} Data2, *epoch* 1

δ_{in_1}	δ_{in_2}	δ_{in_3}
0,0069	-0,0072	-0,0015

Berikut contoh perhitungan δ_1 Data2:

$$\delta_1 = \delta_{in\_1} \times (1 - Z_1) \times Z_1$$
$$= 0{,}0043 \times (1 - 0{,}523) \times 0{,}523$$
$$= 0{,}0017$$

Proses dilakukan hingga seluruh δ_j didapatkan. Hasil seluruh δ_j disajikan pada Tabel 4.21.

Tabel 4.21 Hasil seluruh δ_j Data2, *epoch* 1

δ_1	δ_2	δ_3
0,0017	-0,0018	-0,0004

8. Menghitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor koreksi *error* δ_j digunakan untuk mengkoreksi nilai *error* pada bobot antara *hidden* dan *input* unit (ΔV_{ij}) yang nantinya digunakan untuk memperbaharui bobot V_{ij}.

   Berikut contoh perhitungan ΔV_{ij} Data2:

$$\Delta V_{11} = \alpha \times \delta_1 \times X_{11}$$
$$= 0{,}3 \times 0{,}0017 \times 0{,}93$$
$$= 0{,}00048$$

   Proses dilakukan hingga seluruh koreksi bobot ΔV_{ij} didapatkan. Untuk perhitungan koreksi bias didapatkan dari perkalian *learning rate* dengan faktor koreksi *error* (δ_j).

   Berikut contoh perhitungan ΔV_{bias} Data2:

$$\Delta V_{bias\_1} = \alpha \times \delta_1$$
$$= 0{,}3 \times 0{,}0017$$
$$= 0{,}00052$$

   Proses dilakukan hingga seluruh ΔV_{ij} dan ΔV_{bias} didapatkan. Hasil ΔV_{ij} dan ΔV_{bias} disajikan pada Tabel 4.22.

Tabel 4.22 Hasil seluruh koreksi bobot ΔV_{ij} dan ΔV_{bias} Data2, *epoch* 1

ΔV_{ij}	δ_1	δ_2	δ_3
X_1	0,00048	-0,00050	-1,072E-04
X_2	0,00016	-0,00017	-3,586E-05
X_3	0,00022	-0,00023	-4,876E-05
X_4	0,00014	-0,00015	-3,180E-05
X_5	0,00008	-0,00009	-1,860E-05
X_6	0,00003	-0,00004	-7,740E-06
X_7	0,00037	-0,00038	-8,109E-05
Bias	0,00052	-0,00054	-1,149E-04

9. Menghitung bobot baru V_{ij} dengan menjumlahkan bobot lama V_{ij} pada Tabel 4.15 dengan koreksi bobot ΔV_{ij} pada Tabel 4.22.

   Berikut contoh perhitungan $V_{ij(baru)}$ Data2:

$$V_{11(baru)} = V_{11(lama)} + \Delta V_{11}$$
$$= 0{,}09 + 0{,}00048$$
$$= 0{,}09$$

   Proses dilakukan hingga seluruh $V_{ij(baru)}$ didapatkan. Hasil seluruh $V_{ij(baru)}$ disajikan pada Tabel 4.23.

Tabel 4.23 Hasil seluruh bobot baru V_{ij} Data2, *epoch* 1

V_{ij}	Z_1	Z_2	Z_3
X_1	0,0946	-0,0918	-0,1779
X_2	-0,1452	0,0294	0,1322
X_3	0,1057	-0,1170	0,0912
X_4	0,0663	-0,1426	-0,0593
X_5	0,1059	0,0402	-0,1642
X_6	0,0003	-0,1086	0,0456
X_7	0,0983	0,1052	0,0000
Bias	-0,0997	0,0997	0,2299

10. Menghitung bobot baru W_{ij} dengan menjumlahkan bobot lama W_{ij} pada Tabel 4.16 dengan koreksi bobot ΔW_{ij} pada Tabel 4.19.

    Berikut contoh perhitungan $\Delta W_{ij(baru)}$ Data2:

$$W_{11(baru)} = W_{11(lama)} + \Delta W_{11}$$
$$= (-0{,}16) + (-0{,}0039)$$

$= -0{,}17$

Proses dilakukan hingga seluruh $W_{ij(baru)}$ didapatkan. Hasil seluruh $W_{ij(baru)}$ disajikan pada Tabel 4.24.

Tabel 4.24 Hasil seluruh bobot baru W_{ij} Data2, *epoch* 1

W_{ij}	Y_1
Z_1	-0,17
Z_2	0,16
Z_3	0,03
Bias	0,09

Selanjutnya proses dilanjutkan pada epoch 1 Data3.

Data3

Pada Data3, perhitungan akan dilakukan menggunakan hasil bobot dan bias baru dari data ke-dua.

1. Mendapatkan nilai Z_{in_j} ditiap *neuron* pada *hidden layer* pertama. Proses dilakukan dengan cara mengalikan *input* X_i pada Data3 dengan bobot V_{ij} pada Tabel 4.23 kemudian dijumlahkan dengan bias disetiap *neuron*.

   Contoh perhitungan Z_{in_j} Data3 dapat dilihat pada perhitungan Z_{in_1}:

   $$Z_{in_1} = bias + (X_1 \times V_{11}) + (X_2 \times V_{21}) + (X_3 \times V_{31}) + (X_4 \times V_{41}) + (X_5 \times V_{51}) + (X_6 \times V_{61}) + (X_7 \times V_{71})$$

   $$= (-0{,}10) + (0{,}95 \times 0{,}0946) + \big(0{,}50 \times (-0{,}1452)\big) + (0{,}40 \times 0{,}1057) + (0{,}28 \times 0{,}0663) + (0{,}45 \times 0{,}1059) + (0{,}07 \times 0{,}0003) + (0{,}69 \times 0{,}0983)$$

   $$= 0{,}093$$

   Proses dilakukan hingga seluruh Z_{in} didapatkan. Hasil Z_{in} disajikan pada Tabel 4.25.

Tabel 4.25 Hasil seluruh Z_{in} Data3, *epoch* 1

Z_{in_1}	Z_{in_2}	Z_{in_3}
0,093	0,023	0,077

2. Menghitung keluaran dari Z_{in} (Z_{out}) disetiap *neuron* pada *hidden layer* menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut contoh perhitungan dari Z_j Data3:

   $$Z_1 = \frac{1}{(1 + e^{-Z_{in1}})} = \frac{1}{(1 + e^{-0{,}093})} = 0{,}523$$

   Proses dilakukan hingga seluruh Z_j didapatkan. Hasil Z_j disajikan pada Tabel 4.26.

Tabel 4.26 Hasil seluruh Z_j Data3, *epoch* 1

Z_1	Z_2	Z_3
0,523	0,506	0,519

3. Mendapatkan nilai Y_{in} dengan cara hasil Z_j sebagai masukkan dikalikan dengan bobot W_{ij} pada Tabel 4.24 kemudian dijumlahkan dengan biasnya.

   Perhitugan Y_{in} Data3:

   $$Y_{in} = bias + (Z_1 \times W_{11}) + (Z_2 \times W_{21}) + (Z_3 \times W_{31})$$
   $$= 0{,}10 + \big(0{,}523 \times (-0{,}16)\big) + (0{,}523 \times 0{,}16) + (0{,}523 \times 0{,}03)$$
   $$= 0{,}101$$

4. Menghitung keluaran dari Y_{in} (Y_{out}) menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut perhitungan dari Y_{out} Data3:

   $$Y_{out} = \frac{1}{(1 + e^{-Y_{in}})} = \frac{1}{(1 + e^{-0{,}122})} = 0{,}525$$

5. Menghitung kesalahan *output* unit δ_k. Perhitungan dilakukan menggunakan perkalian antara selisih target dan keluaran Y_{out}.

   Perhitungan δ_k pada Data3:

   $$\delta_k = (target - Y_{out}) \times (1 - Y_{out}) \times Y_{out}$$
   $$= (0{,}46 - 0{,}525) \times (1 - 0{,}525) \times 0{,}525$$
   $$= -0{,}0153$$

6. Menghitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor *error* δ_k digunakan untuk mengkoreksi nilai *error* pada bobot antara *hidden* dan *output* unit (ΔW_{ij}) yang nantinya digunakan untuk memperbaharui bobot W_{ij}.

   Perhitungan ΔW_{ij} Data3:

   $$\Delta W_{ij} = \alpha \times \delta_k \times Z_1 = 0{,}3 \times (-0{,}0153) \times 0{,}523$$
   $$= -0{,}002405$$

   Proses dilakukan hingga seluruh koreksi bobot ΔW_{ij} didapatkan. Untuk perhitungan koreksi bias didapatkan dari perkalian *learning rate* dengan kesalahan *output* unit (δ_k).

   $$\Delta W_{bias} = \alpha \times \delta_k$$
   $$= 0{,}3 \times (-0{,}0153)$$
   $$= -0{,}004596$$

   Hasil perhitungan ΔW_{ij} dan ΔW_{bias} disajikan pada Tabel 4.27.

Tabel 4.27 Hasil seluruh koreksi bobot ΔW_{ij} dan ΔW_{bias} Data3, *epoch* 1

ΔW_{ij}	δ_1
Z_1	-0,002405
Z_2	-0,002324
Z_3	-0,002386
Bias	-0,004596

7. Menghitung delta *input* δ_{in_j} yang sudah terbobot. Delta *input* δ_{in_j} didapatkan untuk menghitung faktor koreksi *error* δ_j. Proses perhitungan delta *input* δ_{in_j} dilakukan dengan menjumlahkan perkalian antara faktor koreksi *error output* unit δ_k dengan bobot W_{ij} pada Tabel 4.24.

   Berikut contoh perhitungan δ_{in_j} Data3:

   $$\delta_{in_1} = (\delta_k \times W_{11})$$
   $$= (-0{,}0153) \times (-0{,}16)$$
   $$= 0{,}002563$$

   Perhitungan dilakukan hingga δ_{in_2} dan δ_{in_3} didapatkan seperti pada Tabel 4.28:

Tabel 4.28 Hasil seluruh δ_{in_j} Data3, *epoch* 1

δ_{in_1}	δ_{in_2}	δ_{in_3}
0,002563	-0,002447	0,000441

   Berikut contoh perhitungan δ_j Data3:

   $$\delta_1 = \delta_{in\_1} \times (1 - Z_1) \times Z_1$$
   $$= 0{,}002563 \times (1 - 0{,}523) \times 0{,}523$$
   $$= 6{,}393\text{E} - 04$$

   Proses dilakukan hingga seluruh δ_j didapatkan. Hasil seluruh δ_j disajikan pada Tabel 4.29.

Tabel 4.29 Hasil seluruh δ_j Data3, *epoch* 1

δ_1	δ_2	δ_3
6,393E-04	-6,116E-04	-1,100E-04

8. Menghitung faktor koreksi bobot (yang akan digunakan untuk memperbaiki bobot baru). Faktor koreksi *error* δ_j digunakan untuk mengkoreksi nilai *error* pada bobot antara *hidden* dan *input* unit (ΔV_{ij}) yang nantinya digunakan untuk memperbaharui bobot V_{ij}.

   Berikut contoh perhitungan ΔV_{ij} Data3:

   $$\Delta V_{11} = \alpha \times \delta_1 \times X_{11}$$
   $$= 0{,}3 \times (6{,}393\text{E} - 05) \times 0{,}95$$

$= 1{,}8E-04$

Proses dilakukan hingga seluruh koreksi bobot ΔV_{ij} didapatkan. Untuk perhitungan koreksi bias didapatkan dari perkalian *learning rate* dengan faktor koreksi *error* (δ_j).

Berikut contoh perhitungan ΔV_{bias} Data3:

$$\Delta V_{bias\_1} = \alpha \times \delta_1$$
$$= 0{,}3 \times (6{,}393E-05)$$
$$= 1{,}9E-04$$

Proses dilakukan hingga seluruh ΔV_{ij} dan ΔV_{bias} didapatkan. Hasil ΔV_{ij} dan ΔV_{bias} disajikan pada Tabel 4.30.

Tabel 4.30 Hasil seluruh koreksi bobot ΔV_{ij} dan ΔV_{bias} Data3, *epoch* 1

ΔV_{ij}	δ_1	δ_2	δ_3
X_1	1,8E-04	-1,7E-04	-3,1E-05
X_2	9,6E-05	-9,2E-05	-1,7E-05
X_3	7,7E-05	-7,4E-05	-1,3E-05
X_4	5,4E-05	-5,2E-05	-9,3E-06
X_5	8,6E-05	-8,3E-05	-1,5E-05
X_6	1,4E-05	-1,3E-05	-2,4E-06
X_7	1,3E-04	-1,3E-04	-2,3E-05
Bias	1,9E-04	-1,8E-04	-3,3E-05

9. Menghitung bobot baru V_{ij} dengan menjumlahkan bobot lama V_{ij} pada Tabel 4.23 dengan koreksi bobot ΔV_{ij} pada Tabel 4.30.

   Berikut contoh perhitungan $V_{ij(baru)}$ Data3:

$$V_{11(baru)} = V_{11(lama)} + \Delta V_{11}$$
$$= 0{,}09 + (1{,}8E-04)$$
$$= 0{,}09$$

Proses dilakukan hingga seluruh $V_{ij(baru)}$ didapatkan. Hasil seluruh $V_{ij(baru)}$ disajikan pada Tabel 4.31.

Tabel 4.31 Hasil seluruh bobot baru V_{ij} Data3, *epoch* 1

V_{ij}	Z_1	Z_2	Z_3
X_1	0,094790	-0,091974	-0,177962
X_2	-0,145088	0,029295	0,132228
X_3	0,105793	-0,117105	0,091159
X_4	0,066313	-0,142636	-0,059318
X_5	0,105969	0,040122	-0,164202
X_6	0,000348	-0,108620	0,045605

X_7	0,098403	0,105103	-0,000071
Bias	-0,099517	0,099506	0,229898

10. Menghitung bobot baru W_{ij} dengan menjumlahkan bobot lama W_{ij} pada Tabel 4.24 dengan koreksi bobot ΔW_{ij} pada Tabel 4.27.

    Berikut contoh perhitungan $\Delta W_{ij(baru)}$ Data3:

$$W_{11(baru)} = W_{11(lama)} + \Delta W_{11}$$
$$= (-0{,}16) + (-0{,}002405)$$
$$= -0{,}17$$

    Proses dilakukan hingga seluruh $W_{ij(baru)}$ didapatkan. Hasil seluruh $W_{ij(baru)}$ disajikan pada Tabel 4.32.

Tabel 4.32 Hasil seluruh bobot baru W_{ij} Data3, *epoch* 1

W_{ij}	Y_1
Z_1	-0,17
Z_1	0,16
Z_1	0,03
Bias	0,09

Pehitungan Data *Testing*

Selanjutnya dilakukan proses *testing* dengan data *testing* pada Tabel 4.1. Proses testing dilakukan sampai mendapatkan nilai Y_{out} (hanya sampai fase *feed forward*) dengan menggunakan bobot terakhir yang didapat dari proses *learning* sebelumnya pada *epoch* 1 Data3.

Tabel 4.33 Data *testing*

Nomor	Nama	X_1	X_2	X_3	X_4	X_5	X_6	X_7	target
1	Data4	0,94	0,50	0,47	0,14	0,53	0,36	0,76	0,52

1. Mendapatkan nilai Z_{in_j} ditiap *neuron* pada *hidden layer* pertama. Proses dilakukan dengan cara mengalikan *input* X_i pada Data *testing* dengan bobot V_{ij} dari hasil *learning* pada Tabel 4.31 kemudian dijumlahkan dengan bias disetiap *neuron*.

   Contoh perhitungan Z_{in_j} Data *testing* dapat dilihat pada perhitungan Z_{in_1}:

$$Z_{in\_1} = bias + (X_1 \times V_{11}) + (X_2 \times V_{21}) + (X_3 \times V_{31}) + (X_4 \times V_{41}) + (X_5 \times V_{51}) + (X_6 \times V_{61}) + (X_7 \times V_{71})$$
$$= (-0{,}10) + (0{,}94 \times 0{,}094790) + \big(0{,}50 \times (-0{,}145088)\big) + (0{,}47 \times 0{,}105793) + (0{,}14 \times 0{,}066313) + (0{,}53 \times 0{,}105969) + (0{,}36 \times 0{,}000348) + (0{,}76 \times 0{,}098403)$$
$$= 0{,}107$$

Proses dilakukan hingga seluruh Z_{in} didapatkan. Hasil Z_{in} disajikan pada Tabel 4.34.

Tabel 4.34 Hasil seluruh Z_{in} Data *Testing*

Z_{in_1}	Z_{in_2}	Z_{in_3}
0,107	0,016	0,092

2. Menghitung keluaran dari Z_{in} (Z_{out}) disetiap *neuron* pada *hidden layer* menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut contoh perhitungan dari Z_j Data *testing*:

$$Z_j = \frac{1}{(1 + e^{-Z_{in1}})} = \frac{1}{(1 + e^{-0,107})} = 0,527$$

Proses dilakukan hingga seluruh Z_j didapatkan. Hasil Z_j disajikan pada Tabel 4.35.

Tabel 4.35 Hasil seluruh Z_j Data *Testing*

Z_1	Z_2	Z_3
0,527	0,504	0,523

3. Mendapatkan nilai Y_{in} dengan cara hasil Z_j sebagai masukkan dikalikan dengan bobot W_{ij} pada Tabel 4.32 kemudian dijumlahkan dengan biasnya.

Perhitugan Y_{in} Data *testing*:

$$\begin{aligned} Y_{in} &= bias + (Z_1 \times W_{11}) + (Z_1 \times W_{21}) + (Z_1 \times W_{31}) \\ &= 0,10 + \big(0,527 \times (-0,16)\big) + (0,527 \times 0,16) + (0,527 \times 0,03) \\ &= 0,092 \end{aligned}$$

4. Menghitung keluaran dari Y_{in} (Y_{out}) menggunakan fungsi aktivasi yang ditentukan. Pada perhitungan ini fungsi aktivasi yang digunakan adalah *sigmoid* biner. Berikut perhitungan dari Y_{out} Data *testing*:

$$Y_{out} = \frac{1}{(1 + e^{-Y_{in}})} = \frac{1}{(1 + e^{-0,122})} = 0,523$$

Ketika hasil dari Y_{out} didapatkan, maka *output* tersebut akan didenormalisasi sehingga akan menghasilkan data asli hasil produktivitas padi.

Perhitungan denormalisasi:

$$\begin{aligned} X_y &= y_{out} \times (X_{max} - X_{min}) \times X_{min} \\ &= 0,523 \times (62,14 - 22,85) \times 22,85 \\ &= 43,40 \end{aligned}$$

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang sistem yang telah dibuat sesuai dengan rancangan. Adapun isi dari bab ini yaitu spesifikasi sistem, batasan implementasi, dan implementasi algoritma.

5.1 Spesifikasi Sistem

Spesifikasi sistem bertujuan untuk mengetahui spesifikasi perangkat yang digunakan dalam penelitian ini, baik perangkat keras maupun perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Tabel 5.1 menjelaskan Spesifikasi perangkat keras.

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	NVIDIA Geforce GT 630M
Memori (RAM)	4 GB DDR3 Memory
Hardisk	750 GB HDD

5.1.2 Spesifikasi Perangkat Lunak

Tabel 5.2 menjelaskan tentang Spesifikasi perangkat lunak.

Tabel 5.2 Spesifikasi Perangkat Lunak

Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows 10 64-bit
Bahasa Pemrograman	Java
Tools Pemrograman	Netbeans 7.3.1

5.2 Batasan Implementasi

Batasan implementasi bertujuan untuk menjelaskan fungsi yang dapat dilakukan oleh pengguna dalam menggunakan sistem.

Batasan – batasan pada sistem yaitu:

a. Masukkan yang diterima oleh sistem ialah:
   i. Untuk Pengujian Sistem
      - Jumlah *neuron* pada *Hidden layer*
      - Jumlah *hidden layer*
      - Maksimum *epoch*

- Nilai *learning rate* dengan rentang 0 – 1
- Prosentase data yang digunakan

ii. Untuk masukkan pengguna:

- Nilai parameter *input* yaitu tekanan udara, penyinaran matahari, curah hujan, kecepatan angin, kelembaban, luas lahan, suhu rata-rata.

b. *Output* yang dihasilkan berupa hasil prediksi produktivitas padi.

5.3 Implementasi Algoritma

5.3.1 Algoritma Inisialisasi Bobot dengan *Nguyen – Widrow*

Pada sistem ini, bobot awal ditentukan secara acak menggunakan algoritma *Nguyen – Widrow* dengan rentang *range* 0.5 – (-0.5). Bobot acak ini digunakan untuk proses pembangkitan bobot *Nguyen – Widrow*. Pada *Source Code* 5.1 menjelaskan bagaimana pembangkitan bobot menggunakan algoritma *Nguyen – Widrow*.

Source Code* 5.1 Bobot Awal *Nguyen – Widrow

```
1   // pembangkitan bobot menggunakan nguyen-widrow
2       public double[][] getNguyenWidrow(double[][] hasil) {
3           // random range [-0.5, 0.5]
4           for (int i = 0; i < hasil.length; i++) {
5               for (int j = 0; j < hasil[i].length; j++) {
6                   hasil[i][j] = Math.random() *(1) - 0.5;
7               }
8           }
9           // hitung total_v
10          double total_v = 0;
11          for (int i = 0; i < hasil.length; i++) {
12              for (int j = 0; j < hasil[0].length; j++) {
13                  total_v += Math.pow(hasil[i][j], 2);
14              }
15          }
16          total_v = Math.sqrt(total_v);
17          // hitung beta
18          int p = input[0].length;
19          int n = neuron;
20          double beta = 0.7*Math.pow(p, 1/n);
21          // hitung nilai bobot akhir
22          for (int i = 0; i < hasil.length; i++) {
23              for (int j = 0; j < hasil[0].length; j++) {
24                  hasil[i][j] = (beta * hasil[i][j]) / total_v;
25              }
26          }
27          return hasil;
28      }
```

Penjelasan *Source Code* 5.1:

Baris 3 – 8 merupakan perulangan untuk mendapatkan nilai random *nguyen – widrow* (-0,5 – 0,5) sebanyak variabel hasil (v = bobot input).

Baris 11 – 15 merupakan perulangan untuk perhitungan penjumlahan dan pemangkatan seluruh nilai bobot v yang di simpan pada variabel *total_v*.

Baris 16 – 28 merupakan hasil akhir nilai inisialisasi bobot *nguyen – widrow* untuk v dengan menggunakan rumus dari *nguyen – widrow*.

5.3.2 Algoritma *Feed Forward*

Feed forward merupakan fase pertama dalam metode *backpropagation*. Proses *feed forward* diawali dengan menghitung nilai masukkan dari *input layer* ke *hidden layer*. Setelah hasil masukkan pada *hidden layer*, akan dilakukan proses keluaran pada *hidden layer* dengan menggunakan fungsi aktivasi. Selanjutnya proses dilanjutkan hingga ke *output layer*.

5.3.2.1 Algoritma perhitungan *input layer* ke *hidden layer*

Proses dalam penjumlahan dari perkalian bobot dengan *input* selanjutnya ditambahkan dengan bias ditiap – tiap *neuron*. S*ource code* algoritma perhitungan *input layer* ke *hidden layer* dapat dilhat pada Source Code 5.2.

Source Code* 5.2 *Input* pada *Hidden Layer

```
1  // mengembil nilai unit tersembunyi
2  public double[] getZ_in(double[] b, double[] input, double[][] v) {
3      double[] z_in = new double[v[0].length];
4      for (int i = 0; i < z_in.length; i++) {
5          z_in[i] = b[i]; // bobot v
6          for (int j = 0; j < input.length; j++) {
7              z_in[i] += (input[j] * v[j][i]); //dimensi pertama dan
8  kedua
9           }
10      }
11    return z_in;
12 }
```

Penjelasan *Source Code* 5.2:

Baris 3 – 12 merupakan perhitungan untuk mendapakan nilai *z_in* dimana pada *class public double [] getZ_in* menerima parameter *b[0]* (bias), *input[i]* (data masukkan), dan *v (bobot)*. Di dalam *method* tersebut terdapat perulangan untuk menentukan nilai dimensi pada array (*z_in,* bias*,* dan *v*) dan proses mendapatkan nilai *z_in*.

5.3.2.2 Algoritma perhitungan keluaran tiap – tiap *neuron* pada *hidden layer*

Selanjutnya menghitung hasil keluaran pada tiap – tiap *neuron* menggunakan fungsi aktivasi. S*ource code* algoritma perhitungan keluaran tiap – tiap *neuron* pada *hidden layer* dapat dilihat pada *Source Code* 5.3

Source Code* 5.3 Keluaran tiap – tiap *Neuron pada Hidden Layer

```
1  // fungsi sigmoid untuk hidden layer
2  public double[] getZ(double[] z_in) {
3     double[] z = new double[z_in.length];
4     for (int i = 0; i < z_in.length; i++) {
5          z_in[i] *= -1;
6          z[i] = 1 / (1 + Math.exp(z_in[i]));
7      }
8    return z;
9  }
```

Penjelasan *Source Code* 5.3:

Pada *Source Code* 5.3 merupakan perhitungan *Z_out* menggunakan fungsi aktivasi. Pada implementasi ini menggunakan fungsi aktivasi sigmoid. *Method public double getZ* menerima nilai dari *Z_in*.

5.3.2.3 Algoritma perhitungan *input* pada *hidden layer* ke *output layer*

Setelah hasil keluaran pada lapisan tersembunyi didapatkan selanjutnya akan dilakukan perhitungan *Y_in* dari hasil nilai keluaran pada lapisan terakhir pada lapisan tersembunyi. *Source code* algoritma perhitungan *input* pada *hidden layer* ke *output layer* dapat dilihat pada *Source Code* 5.4.

Source Code* 5.4 *Input* pada *Hidden Layer* ke *Output Layer

```
1   // mengambil nilai output layer
2   public double getY_in(double b, double[] z, double[][] w1, int k) {
3   // bias, z, bobot, indek
4      double y_in = 0;
5      y_in = b;
6      for (int i = 0; i < z.length; i++) {
7           y_in += (z[i] * w1[i][k]); //indeks output hanya 1 indeks
8   yaitu 0. indeks = 0
9        }
10    return y_in;
11  }
```

Penjelasan *Source Code* 5.4:

Pada S*ource Code* 5.4 perhitungan untuk nilai *Y_in* dimana menerima parameter *b1 [j]* (bias untuk *hidden* terakhir ke *output*), nilai *Z*, *w1* (bobot antar *hidden layer*), dan k. Perulangan dilakukan sepanjang nilai *z*.

5.3.2.4 Algoritma perhitungan keluaran pada *output layer*

Setelah hasil perhitungan *input* pada *hidden layer* ke *output layer,* selanjutnya dilakukan perhitungan *output* (*Y_out*). Proses perhitungan didapatkan dari hasil nilai perhitungan *input* pada *hidden layer* ke *output layer* dengan menggunakan fungsi aktivasi. *Source code* algoritma perhitungan keluaran pada *output layer* dapat dilihat pada *Source Code* 5.5.

Source Code* 5.5 Perhitungan Keluaran pada *Output Layer

```
1   // mengambil nilai output layer dengan fungsi aktivasi sigmoid
2       public double getYout(double y_in) {
3           double y = 0;
4           y_in *= -1;
5           y = 1 / (1 + Math.exp(y_in));
6           return y;
7       }
```

Penjelasan S*ource Code* 5.5:

Pada *Source Code* 5.5 merupakan perhitungan *Y_out* dimana menggunakan fungsi aktivasi sigmoid. Karena output hanya satu, maka perulangan hanya dilakukan satu kali.

5.3.3 Algoritma *Feed Backward*

Feed backward merupakan fase kedua dalam metode *backpropagation*. Proses *feed backward* diawali dengan menghitung faktor koreksi nilai *error* pada *output layer*. Selanjutnya faktor koreksi nilai *error* pada *output layer* digunakan untuk mendapatkan hasil faktor koreksi *error* pada bobot *hidden layer* ke *output layer*. Proses dilanjutkan ke tahap selanjutnya hingga mendapatkan bobot baru.

5.3.3.1 Algoritma koreksi nilai *error* pada *output layer*

Proses perhitungan faktor koreksi nilai error pada output layer dengan menggunakan target pada nilai input. Source code algoritma koreksi nilai error pada output layer dapat dilihat pada *Source Code* 5.6.

Source Code* 5.6 Perhitungan *Error* pada *Output Layer

```
1  // hitung nilai eror untuk detal
2    error = (aktual - y_out) * y_out * (1 - y_out);
3    delta_w1 = hitungDelta(error, z[z.length - 1], w1);
4    delta_b1 = hitungDelta(error, b1);...
5  ..
6  .
7  }
```

Penjelasan *Source Code* 5.6:

Baris – 2 merupakan perhitungan untuk mendapatkan nilai *error* pada *output*.

5.3.3.2 Algoritma menghitung koreksi bobot antara *hidden layer* dan *output layer*

Perhitungan *error* pada pada layer ini dilakukan dengan menghitung turunan fungsi aktivasi *sigmoid*. *Source code* algoritma menghitung koreksi bobot antara *hidden layer* dan *output layer* dapat dilihat pada *Source Code* 5.7.

Source Code* 5.7 Koreksi *Bobot* antara *Hidden Layer* dan *Output Layer

```
1  // method untuk menghitung nilai delta
2  public double[][] hitungDelta(double error, double[][] z, double[][]
3  bobot) {
4    double[][] hasil = new double[bobot.length][bobot[0].length]; //
5  bobot untuk melihat panjang detla yang ingin dihitung
6    for (int i = 0; i < hasil.length; i++) {
7         for (int j = 0; j < hasil[0].length; j++) {
8              hasil[i][j] = alpa * z[hidden_layer - 1][i] * error;
9          }
10    }
11   return hasil;
12 }
```

Penjelasan *Source Code* 5.7:

Pada S*ource Code* 5.7 merupakan *method hitungDelta* dimana di dalamnya terdapat perulangan ganda *i* dan *j*. Koreksi bobot didapat dari perkalian nilai *error (delta_output)*, *alpa*, dan nilai *Z* yang disimpan ke dalam parameter hasil.

5.3.3.3 Algoritma perbaikan bobot antar *hidden* dan *hidden* ke *input layer*

Awal proses perhitungan perbaikan bobot antar *hidden layer* melakukan perhitungan untuk mendapatkan koreksi keluaran pada tiap – tiap *neuron*. Selanjutnya proses perbaikan bobot antara *hidden layer* dengan *input layer* menggunakan koreksi keluaran *neuron* pada *layer* terakhir (pertama). *Source code* algoritma perbaikan bobot antar *hidden* dan *hidden* ke *input layer* dapat dilihat pada *Source Code* 5.8.

Source Code* 5.8 Perbaikan Bobot *antar Hidden dan* Hidden *ke Input Layer

```
1  for (int i = 0; i < w1.length; i++) {
2     delta[0][hidden_layer - 1][i] = w1[i][0] * error;
3     delta[1][hidden_layer - 1][i] = delta[0][hidden_layer - 1][i] *
4  z[1][hidden_layer - 1][i] * (1 - z[1][hidden_layer - 1][i]);
5     delta_b[hidden_layer - 1][i] = alpa * delta[1][hidden_layer -
6  1][i];
7    }
8  if (hidden_layer > 1) {
9       for (int i = hidden_layer - 2; i >= 0; i--) {
10           for (int k = neuron - 1; k >= 0; k--) {
11                //menghitung delta_in
12                for (int j = 0; j < neuron; j++) {
13                     delta[0][i][k]   +=   delta[1][i   +   1][j]   *
14 w[i][k][j];
15                     delta_w[i][k][j] = alpa * delta[1][i + 1][j] *
16 z[1][i][k];
17                }
18                delta[1][i][k] = delta[0][i][k] * (z[1][i][k]) * (1 -
19 z[1][i][k]);
20                delta_b[i][k] = alpa * delta[1][i][k];
21             }
22       }
23     //menghitung delta_v
24     for (int i = 0; i < input[0].length; i++) {
25          for (int j = 0; j < neuron; j++) {
26               delta_v[i][j] = alpa * delta[1][0][j] * input[ii][i];
27          }
28       }
29 } else {
30     //menghitung delta_v single hidden_layer
31     for (int i = 0; i < input[0].length; i++) {
32          for (int j = 0; j < neuron; j++) {
33               delta_v[i][j] = alpa * delta[1][0][j] * input[ii][i];
34           }
35       }
36   }
```

Penjelasan S*ource Code* 5.8:

Baris 1 – 7 merupakan perhitungan untuk setiap *neuron* untuk mendapatkan nilai *delta_in*, *delta_out*, beserta *delta_bias* pada *hidden layer* terakhir.

Baris 8 – 29 merupakan kondisi jika *hidden layer* lebih dari 1. Pada baris 9 – 22 merupakan perhitungan nilai *delta_in*, *delta_out*, *delta_bias* pada tiap – tiap *neuron* selain *neuron* pada *hidden layer* terakhir. Pada baris 24 – 28 merupakan perhitungan untuk mendapatkan nilai *delta_v* jika *hidden layer* lebih dari 1.

Baris 29 – 36 merupakan kondisi lain jika *hidden layer* bernilai 1. Pada kondisi ini menghitung nilai *delta_v* jika *hidden layer* bernilai 1.

5.3.3.4 Algoritma bobot baru

Proses perhitungan bobot baru dilakukan dengan menambahkan bobot lama dengan hasil koreksi bobot. Source code algoritma bobot baru dapat dilihat pada Source Code 5.9.

***Source Code* 5.9 Perbaharui Bobot**

```
1         public void update_bobot(double[][] delta_v, double[]
2   delta_b1, double[][][] delta_w, double[][] delta_b, double[][]
3   delta_w1) {
4         // update bobot n_aktual
5         for (int i = 0; i < b.length; i++) {
6             for (int j = 0; j < b[0].length; j++) {
7                 b[i][j] += delta_b[i][j];
8             }
9         }
10        // update bobot b1
11        for (int i = 0; i < b1.length; i++) {
12            b1[i] += delta_b1[i];
13        }
14        // update bobot v
15        for (int i = 0; i < v.length; i++) {
16            for (int j = 0; j < v[0].length; j++) {
17                v[i][j] += delta_v[i][j];
18            }
19        }
20        // update bobot w
21        for (int i = 0; i < delta_w.length; i++) {
22            for (int j = 0; j < delta_w[0].length; j++) {
23                for (int k = 0; k < delta_w[0][0].length; k++) {
24                    w[i][j][k] += delta_w[i][j][k];
25                }
26            }
27        }
28        // update bobot w1
29        for (int i = 0; i < delta_w1.length; i++) {
30            for (int j = 0; j < delta_w1[0].length; j++) {
31                w1[i][j] += delta_w1[i][j];
32            }
33        }
34    }
```

Penjelasan *Source Code* 5.9:

Baris 5 – 9 merupakan *update* bias pada *output layer* dengan menambahkan nilai bias lama dengan nilai koreksi bias yang sudah didapatkan.

Baris 11 – 13 merupakan *update* bias pada *hidden layer* dengan menambahkan nilai bias lama dengan nilai koreksi bias yang sudah didapatkan.

Baris 15 – 19 merupakan *update* bobot *v* (*input*) terhadap *hidden layer* dengan menambahkan nilai *v_lama* dengan *v_baru* yang didapatkan dari koreksi pada nilai *v*.

Baris 21 – 27 merupakan *update* bobot *w* (*hidden*) terhadap *hidden* ke *hidden* dengan menambahkan nilai *w_lama* dengan *w_baru* yang didapatkan dari koreksi pada nilai *w*.

Baris 29 – 33 merupakan *update* bobot *w1* (*ouput*) terhadap *hidden* terakhir ke *output layer* dengan menambahkan nilai *w1_lama* dengan *w1_baru* yang didapatkan dari koreksi pada nilai *w*.

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian dan Analisis

Pada bab ini terdapat 4 jenis pengujian yang akan dilakukan diantaranya yaitu: pengujian *learning rate* yang akan dilakukan pertama, pengujian *hidden layer* yang dilakukan kedua, pengujian *neuron* pada *hidden layer* yang dilakukan ketiga, dan pengujian *epoch* yang terakhir dilakukan. Tiap pengujian akan menggunakan data yang sama yaitu, 148 data. Selanjutnya data akan dibagi menjadi 135 data latih dan 13 data uji.

6.1.1 Pengujian dan Analisis Perbedaan *Learning Rate*

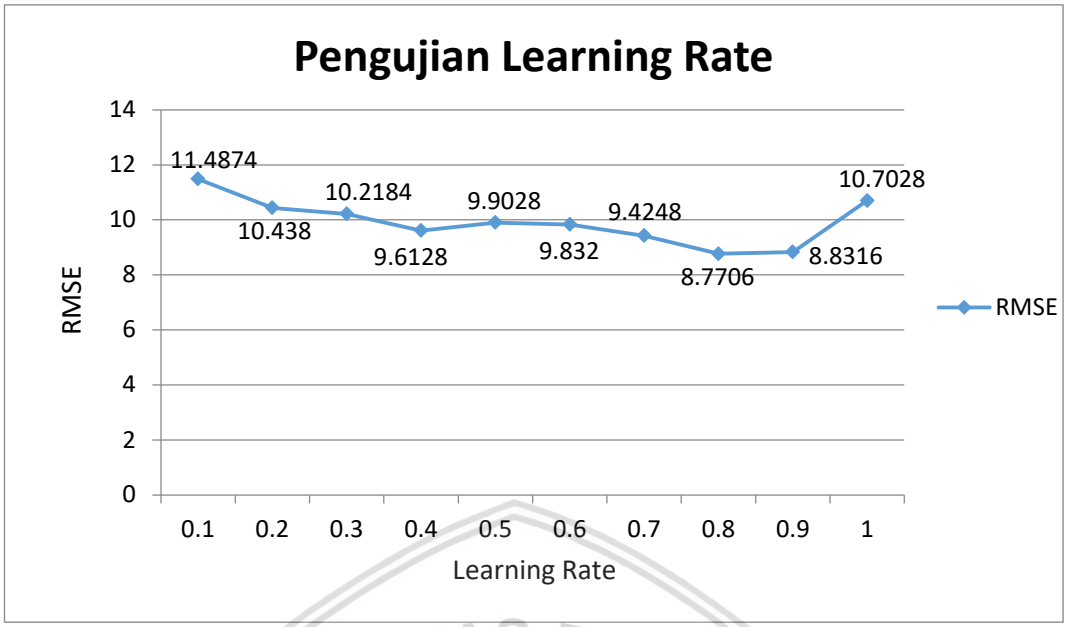
Pengujian pertama merupakan pengujian perbedaan *learning rate*. Pengujian dilakukan dengan membedakan jumlah *input learning rate* yaitu 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, dan 1 dengan jumlah *hidden layer* 3, *hidden neuron* 7, dan maksimum *epoch* 10000. Pengujian dilakukan sebanyak 5 kali yang selanjutnya akan dihitung rata-rata dari nilai RMSE.

Tabel 6.1 Pengujian Perbedaan *Learning Rate*

Learning rate	***Data Latih = 135 & Data Uji = 13, epoch = 10000, neuron = 7, hidden layer = 3***					***Rata-rata RMSE***
	RMSE					
	Percobaan ke 1	**Percobaan ke 2**	**Percobaan ke 3**	**Percobaan ke 4**	**Percobaan ke 5**	
0.1	13.272	10.898	12.027	9.526	11.714	11.4874
0.2	11.385	13.073	5.526	10.503	11.703	10.438
0.3	10.687	10.953	9.699	9.373	10.38	10.2184
0.4	10.834	9.937	9.345	8.95	8.998	9.6128
0.5	11.424	12.044	7.448	9.528	9.07	9.9028
0.6	10.009	11.368	7.382	9.58	10.821	9.832
0.7	9.843	8.194	8.688	10.945	9.454	9.4248
0.8	8.752	7.165	9.459	9.341	9.136	8.7706
0.9	8.532	8.131	7.378	9.82	10.297	8.8316
1	11.86	8.861	11.659	10.388	10.746	10.7028

Dari hasil pengujian pada Tabel 6.1, didapatkan nilai rata-rata RMSE terbaik yaitu 8.7706 yang didapat dari *learning* dengan jumlah nilai *learning rate* adalah 0.8.

Dari hasil rata-rata RMSE disetiap pengujian didapatkan grafik pada Gambar 6.1:



Gambar 6.1 Grafik Pengujian Perbedaan *Learning Rate*

Berdasarkan Gambar 6.1 nilai *learning rate* dengan *input* 0.8 menunjukkan rata-rata RMSE terkecil. Dimana pada *learning rate* 0.8 didapat rata-rata RMSE sebesar 8.7706. Dari 5 kali percobaan, *learning rate* 0.8 menunjukkan nilai RMSE yang terbaik. Setelah dilakukan pengamatan, besarnya nilai RMSE yang dihasilkan disebabkan karena keterbatasan/kurangnya data latih yang digunakan pada penelitian ini sehingga proses pembelajaran pada algoritma *backpropagation* belum sempurna.

6.1.2 Pengujian dan Analisis Perbedaan Jumlah *Hidden Layer*

Pengujian kedua merupakan pengujian banyak *hidden layer*. Pengujian dilakukan dengan membedakan jumlah *input hidden layer* yaitu 1, 2, 3, 4, 5, 6, 7, 8, 9, dan 10 terhadap nilai *learning rate* 0.8 yang telah diuji pada sub-bab sebelumnya, *neuron* pada *hidden layer* 7, dan *epoch* sebanyak 10000.

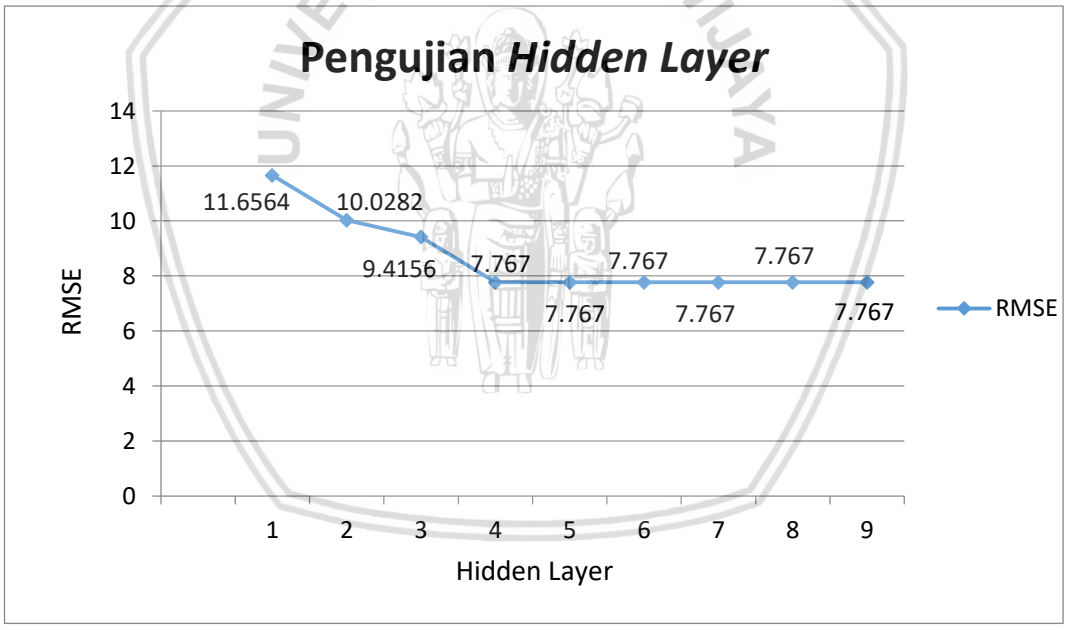
Tabel 6.2 Pengujian Perbedaan Jumlah *Hidden Layer*

Hidden Layer	***Data Latih = 135 & Data Uji = 13, learning rate = 0.8, neuron = 7, epoch = 10000***					***Rata-rata RMSE***
	RMSE					
	Percobaan ke 1	**Percobaan ke 2**	**Percobaan ke 3**	**Percobaan ke 4**	**Percobaan ke 5**	
1	8.671	14.625	14.315	11.056	9.615	11.6564
2	9.137	9.164	10.018	10.198	11.624	10.0282
3	9.137	9.164	10.018	10.198	8.561	9.4156
4	7.767	7.767	7.767	7.767	7.767	7.767
5	7.767	7.767	7.767	7.767	7.767	7.767
6	7.767	7.767	7.767	7.767	7.767	7.767

Hidden Layer	*Data Latih = 135 & Data Uji = 13, learning rate = 0.8, neuron = 7, epoch = 10000*					*Rata-rata RMSE*
	RMSE					
	Percobaan ke 1	Percobaan ke 2	Percobaan ke 3	Percobaan ke 4	Percobaan ke 5	
7	7.767	7.767	7.767	7.767	7.767	7.767
8	7.767	7.767	7.767	7.767	7.767	7.767
9	7.767	7.767	7.767	7.767	7.767	7.767
10	7.767	7.767	7.767	7.767	7.767	7.767

Dari hasil pengujian pada Tabel 6.2, didapatkan nilai rata-rata RMSE terbaik yaitu 7.767 pada *input hidden layer* berjumlah 4, 5, 6, 7, 8, 9, dan 10. Namun, pada pengujian ini nilai *input hidden layer* yang diambil berjumlah 3 dikarenakan pada *input hidden layer* 4 sampai dengan 10 menghasilkan hasil prediksi yang tidak ada perubahan dan cenderung seragam. Berikut salah satu pengujian sistem menggunakan *hidden layer* berjumlah 4 dapat dilihat pada tabel 6.3.

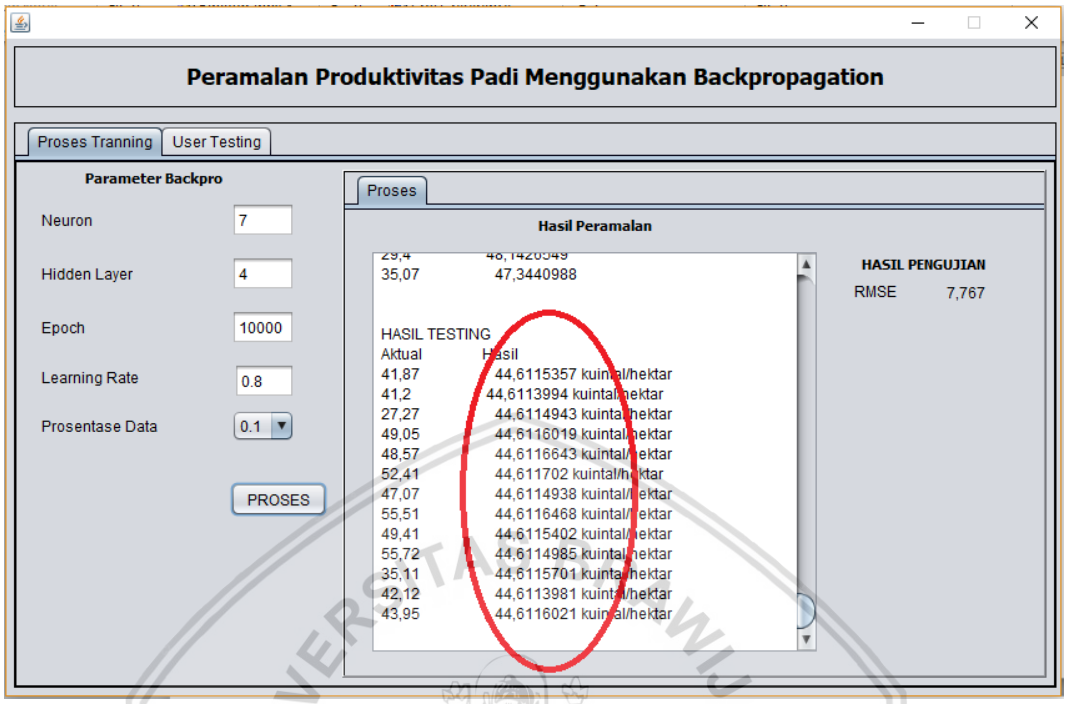
Dari hasil rata-rata RMSE disetiap pengujian didapatkan grafik pada Gambar 6.2:



Gambar 6.2 Grafik Pengujian Perbedaan *Hidden Layer*

Berdasarkan Gambar 6.2 nilai *hidden layer* dengan jumlah 4, 5, 6, 7, 8, 9, dan 10 memiliki nilai rata – rata RMSE yang sama yang mengakibatkan tidak ada perubahan yang signifikan terhadap hasil predikasi yang didapatkan. Kecilnya nilai rata – rata RMSE pada *hidden layer* 4 sama dengan 10 dikarenakan hasil prediksi yang cenderung sama lebih banyak mendekati nilai target, sehingga nilai rata – rata RMSE yang didapatkan kecil. Sedangkan hasil predikasi yang di dapatkan pada hidden layer 1, 2, dan 3 berbanding terbalik dengan hasil predikasi yang didapatkan *hidden layer* dengan jumlah 4, 5, 6, 7, 8, 9, dan 10. Hasil prediksi yang didapatkan cenderung berbeda. Namun, pada *hidden layer* 3 memiliki nilai rata – rata RMSE terkecil dibandingkan dengan *hidden layer*

1 dan 2 dengan nilai rata – rata RMSE sebesar 9.4156, sehingga nilai *hidden layer* yang digunakan untuk proses pengujian berikutnya menggunakan *hidden layer* 3.



Gambar 6.3 Pengaruh *Hidden Layer*

6.1.3 Pengujian dan Analisis Perbedaan *Hidden Neuron*

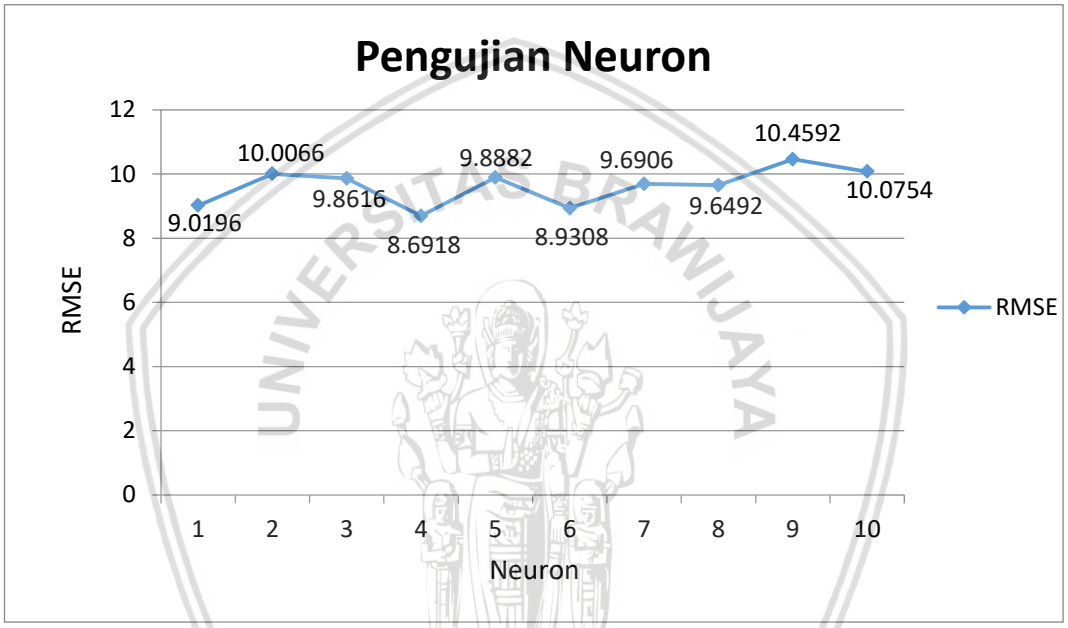
Pengujian perbedaan *hidden neuron* dilakukan dengan menggunakan *epoch* 10000, *learning rate*, dan jumlah *hidden layer* terbaik yang telah dilakukan pada sub bab sebelumnya. Pengujian dilakukan dengan variasi jumlah *input hidden neuron* yaitu, 1, 2, 3, 4, 5, 6, 7, 8, 9, dan 10 dengan jumlah *hidden layer* 3 dilakukan percobaan sebanyak 5 kali yang selanjutnya akan dihitung rata-rata dari nilai RMSE.

Tabel 6.3 Pengujian Perbedaan *Hidden Neuron*

Neuron	***Data Latih = 135 & Data Uji = 13, learning rate = 0.8, hidden layer = 3, epoch = 10000***					***Rata-rata RMSE***
	RMSE					
	Percobaan ke 1	**Percobaan ke 2**	**Percobaan ke 3**	**Percobaan ke 4**	**Percobaan ke 5**	
1	9.076	9.183	8.665	8.558	9.616	9.0196
2	10.298	10.331	9.082	10.359	9.963	10.0066
3	8.808	10.945	10.829	8.918	9.808	9.8616
4	6.228	8.773	9.398	9.881	9.179	8.6918
5	7.968	10.656	9.745	11.343	9.729	9.8882
6	9.69	9.851	7.479	10.543	7.091	8.9308
7	7.633	9.527	9.175	11.241	10.877	9.6906
8	10.377	10.004	8.803	9.63	9.432	9.6492
9	13.277	10.251	8.437	10.242	10.089	10.4592
10	8.472	9.625	11.577	11.118	9.585	10.0754

Dari hasil pengujian pada Tabel 6.3, didapatkan nilai rata-rata RMSE terbaik yaitu 8.6918 yang didapat dari jumlah *hidden neuron* sebanyak 4. Grafik hasil rata-rata RMSE disetiap pengujian *hidden neuron* dapat pada Gambar 6.3:

Berdasarkan Gambar 6.4 *hidden neuron* yang berjumlah 4 menunjukkan rata-rata RMSE terkecil. Dimana pada *hidden neuron* 4 didapat rata-rata RMSE sebesar 8.6918. Dari 5 kali percobaan *hidden neuron* 4 menunjukkan bahwa nilai RMSE yang diperoleh lebih baik dari hidden *neuron* yang lain dan stabil. Sedangkan pada *hidden neuron* lain menunjukkan hasil RMSE yang cenderung tidak stabil dan naik-turun pada *range* tertentu, namun tidak lebih kecil dari *hidden neuron* 4. Ini membuktikan bahwa tidak selalu pada *hidden neuron* yang banyak mampu mendapatkan nilai RMSE terbaik, begitu juga sebaliknya. Hal ini



Gambar 6.4 Grafik Pengujian Perbedaan *Hidden Neuron*

dikarenakan semakin banyaknya *hidden neuron* pada *hidden layer*, maka bobot yang akan inisialisasi akan semakin banyak sehingga perhitungan terhadap masukkan dan keluaran pada masing – masing *neuron* akan semakin banyak yang membuat hasil yang tidak optimal. Dan jika *hidden neuron* pada *hidden layer* terlalu sedikit, maka proses untuk mengenali input pada sistem menjadi tidak optimal.

6.1.4 Pengujian *K – Fold Cross Validation*

Pengujian yang keempat adalah pengujian *K-Fold Cross Validation*. Pengujian *K-fold* dilakukan untuk melihat kestabilan pengujian data dengan mengkelompokkan data berdasarkan beberapa K. Selain itu, pengujian ini bertujuan untuk melihat nilai RMSE terhadap perubahan data latih dan data uji. Pada pengujian ini menggunakan *epoch* 10000, *learning rate*, *hidden layer*, *neuron* pada *hidden layer*, dan 119/118 data latih dan 29/30 data uji yang telah dibagi berdasarkan K tertentu terhadap keseluruhan data. Hasil pengujian *K-fold* dapat dilihat pada Tabel 6.4.

Tabel 6.4 Pengujian *K – Fold Cross Validation*

Learning Rate = 0.8, Hidden Layer = 3, Neuron = 4, Epoch = 10000			
Nomor	**Data Latih**	**Data Uji**	**Hasil RMSE**
Percobaan 1	fold 1 - fold 2 - fold 3 - fold 4	fold 5	11.712
Percobaan 2	fold 2 - fold 3 - fold 4 - fold 5	fold 1	5.826
Percobaan 3	fold 3 - fold 4 - fold 5 - fold 1	fold 2	6.924
Percobaan 4	fold 4 - fold 5 - fold 1 - fold 2	fold 3	8.237
Percobaan 5	fold 5 - fold 1 - fold 2 - fold 3	fold 4	8.364
Rata - rata RMSE			8.2126

Berdasarkan Tabel 6.4 hasil rata – rata RMSE pada seluruh percobaan adalah 8.2126. Hasil RMSE terbesar didapatkan pada percobaan 1 dengan hasil RMSE 11.712 dengan data latih *fold* 1 – *fold* 2 – *fold* 3 – *fold* 4 dan data uji *fold* 5. Sedangkan untuk hasil RMSE terkecil didapatkan pada percobaan 2 dengan hasil RMSE 5.826 dengan data latih *fold* 2 – *fold* 3 – *fold* 4 – *fold* 5 dan data uji *fold* 1.

BAB 7 PENUTUP

7.1 Kesimpulan

Beberapa kesimpulan yang dapat ditarik dari penelitian ini sebagai berikut:

1. Optimalnya nilai *learning rate* yang digunakan sangat berpengaruh terhadap hasil prediksi produktivitas padi. Terlalu besarnya nilai *learning rate* yang digunakan dikhawatirkan akan membuat sistem divergen dan membuat *error* mencapai *local minima*. Pada penelitian ini nilai *learning rate* yang optimal yaitu 0.8 yang memiliki nilai rata – rata RMSE terkecil dibanding dengan nilai *learning rate* 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, dan 1.
2. *Hidden layer* sangat berpengaruh dalam proses *learning backpropagation*. Pada penelitian ini variasi *hidden layer* yang cocok untuk proses prediksi produktivitas padi berjumlah 3. Semakin besarnya *hidden layer* yang digunakan maka nilai prediksi yang dihasilkan akan semakin buruk hampir tidak mempengaruhi hasil prediksi. Sebaliknya, jika *hidden layer* yang digunakan terlalu kecil maka akan mendapatkan hasil prediksi yang kurang optimal.
3. Pengaruh *hidden neuron* terhadap proses prediksi produktivitas padi sangat signifikan. Terlalu banyaknya *hidden neuron* yang digunakan membuat proses *learning backpropagation* lama karena *variable* yang semakin banyak dan belum tentu menghasilkan hasil yang optimal. Pada penelitian ini variasi *hidden neuron* optimal terhadap proses prediksi produktivitas padi berjumlah 4 *neuron*.
4. Pengaruh perubahan data latih dan data uji terhadap hasil peramalan sangat signifikan. Dengan menggunakan *k-fold cross validation* didapatkan hasil RMSE terbesar dan terkecil pada percobaan 1 dan percobaan 2 dengan nilai RMSE 11.712 dan 5.826 dengan rata – rata RMSE seluruh percobaan 8.2126.

7.2 Saran

Berberapa saran yang dikemukakan untuk penelitian selanjutnya guna membangun sistem lebih baik lagi yaitu:

1. Pada penelitian selanjutnya diharapkan menggunakan data bulanan ataupun harian dengan menambahkan parameter lain yang mempengaruhi hasil produktivitas padi.
2. Pada penelitian selanjutnya dapat dilakukan modifikasi dengan menambahkan metode lain pada prediksi produktivitas padi ini, agar proses *learning* tidak memerlukan *epoch* yang besar serta menambahkan kondisi berhenti selain *epoch* untuk mencapai konvergensi.

DAFTAR PUSTAKA

Amrullah, Sopandie, D., Sugianta, & Junaedi, A. (2014). Peningkatan Produktivitas Tanaman Padi (Oryza sativa L.) melalui Pemberian Nano Silika. *PANGAN, 23*, 17-32.

Andrian, Y., & Wayahdi, M. R. (2014). Analisis Algoritma Inisialisasi Nguyen-Widrow Pada Proses Prediksi Curah Hujan Kota Medan Menggunakan Metode Backpropagtion Neural Network. *Seminar Nasional Informatika*.

Anwar, B. (2011). Penerapan Algoritma Jaringan Syaraf Tiruan Backpropagation Dalam Memprediksi Tingkat Suku Bunga Bank. *SAINTIKOM, 2*.

Asnawi, R. (2014). Peningkatan Produktivitas dan Pendapatan Petani Melalui Penerapan Model Pengelolaan Tanaman Terpadu Padi Sawah di Kabupaten Pesawaran, Lampung. *Jurnal Penelitian Pertanian Terapan, 14*, 44-52.

Brian, T. (2016). Analisis Learning Rates Pada Algoritma Backpropagation Untuk Klasifikasi Penyakit Diabetes. *Jurnal Ilmiah Edutic, 3*.

Chamidah, N., Wiharto, & Salamah, U. (2012). Pengaruh Normalisasi Data pada Jaringan Syaraf Tiruan Backpropagation Gradient Descent Adaptive Gain (BPGDAG) untuk Klasifikasi. *ITSMART, 1*.

Cox, R., Priyambada, R., Winardi, W., Jamzuri, M., Sutyanto, D., Budiono, A., et al. (2017). *INDONESIA-INVESTMENTS*. Dipetik January 5, 2018, dari https://www.indonesia-investments.com/id/bisnis/komoditas/beras/item183?

Dewi, C., & Muslikh, M. (2013). Perbandingan Akurasi Backpropagation Neural Network dan ANFIS Untuk Memprediksi Cuaca. *Journal of Scientific Modeling & Computation, 1*.

Fitri, Setyawati, O., & S, D. R. (2013). Aplikasi Jaringan Syaraf Tiruan Untuk Penentuan Status Gizi Balita Dan Rekomendasi Menu Makanan Yang Dibutuhkan. *Jurnal EECCIS, 7*.

Hartati, S. (2013). Pengaruh Pengolahan terhadap Kandungan Poliphenol dan Antosianin Beras Wulung yang Berpotensi sebagai Makanan Diet Penderita Diabetes Mellitus. *Jurnal Pangan dan Gizi, 04*.

Haryanti, S. (2010). Jumlah dan Distribusi Stomata pada Daun Beberapa Spesies Tanaman Dikotil dan Monokotil. *Buletin Anatomi dan Fisiologi, 18*.

Herawati, S. (2013). Peramalan Harga Saham Menggunakan Integrasi Empirical Mode Decomposition dan Jaringan Syaraf Tiruan. *Jurnal Ilmiah Mikrotek, 1*, 23-28.

Herjanto, E. (2008). *Manajemen Operasi* (Edisi ketiga ed.). Jakarta: Grasindo.

Hidayati, D., Aldrian, E., Sucahyono, D., Abdurrahim, A. Y., Surtiari, G. A., & Yogaswara, H. (2017). *Upaya Peningkatan Pengetahuan Dan Adaptasi Petani Dan Nelayan Melalui Radio* (Vol. 16). Bogor: ResearchGate.

Kholis, I. (2015). Analisis Variasi Parameter Backpropagation Aritificial Neural Terhadap Pengenalan Pola Data Iris. *Jurnal Teknik dan Ilmu Komputer, 4*.

Lesnussa, Y. A., Latuconsina, S., & Persulessy, E. R. (2015). Aplikasi Jaringan Saraf Tiruan Backpropagation untuk Memprediksi Prestasi Siswa SMA (Studi kasus: Prediksi Prestasi Siswa SMAN 4 Ambon). *Jurnal Matematika Integratif, 11*.

Matondang, Z. A. (2013). Jaringan Syaraf Tiruan Dengan Algoritma Backpropagation Untuk Penentuan Kelulusan Sidang Skripsi. *Pelita Informatika Budi Darma, IV*.

Nguyen, D. H., & Widrow, B. (1990). Neural Networks for Self-Learning Control Systems. *IEEE*.

Pakaja, F., Naba, A., & Purwanto. (2012). Peramalan Penjualan Mobil Menggunakan Jaringan Syaraf Tiruan dan Certainty Factor. *Jurnal EECCIS, 6*.

Purwaningsih, N. (2016). Penerapan Multilayer Perceptron Untuk Klasifikasi Jenis Kulit Sapi Ternak. *TEKNOIF, 4*.

Rachman, A. S., Cholissodin, I., & Fauzi, M. A. (2018). Peramalan Produksi Gula Menggunakan Metode Jaringan Syaraf Tiruan Backpropagation Pada PG Candi Baru Sidoarjo. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2*, 1683-1689.

Rahmat, Setiawan, R., & Purnomo, M. H. (2006). Perbandingan Algoritma Levenberg-Marquardt Dengan Metoda Backpropagation Pada Proses Learning Jaringan Syaraf Tiruan Untuk Pengenalan Pola Sinyal Elektrokardiograf. *Seminar Nasional Aplikasi Teknologi Informasi*.

Ramadhiat, A., Tritoasmoro, I. I., & Wijayanto, I. (2016). Analisis Penggunaan Algoritma Genetika Untuk Meningkatkan Perfomansi Dari Klasifikasi Genre Musik Berbasis Jaringan Syaraf Tiruan Back-Propagation. *e-Proceeding of Engineering, 3*, 1527-1535.

Riaddy, D. (2015). *KOMPAS.com*. Dipetik Desember 1, 2017, dari http://nasional.kompas.com/read/2015/09/02/095100026/Ini.5.Negara.Penghasil.Beras.Terbesar.di.Dunia?page=all

Riska, S. Y., Cahyani, L., & Rosadi, M. I. (2015). Klasifikasi Jenis Tanaman Mangga Gadung dan Mangga Madu Berdasarkan Tulang Daun. *Jurnal Buana Informatika, 6*, 41-50.

Ruminta. (2016). Analisis penurunan produksi tanaman padi akibat perubahan iklim di Kabupaten Bandung Jawa Barat. *Jurnal Kultivasi, 15*.

Santoso, A. B. (2015). Pengaruh Luas Lahan dan Pupuk Bersubsidi Terhadap Produksi Padi Nasional. *Jurnal Ilmu Pertanian Indonesia (JIPI), 20*.

Setiawan, E. (2009). Kajian Hubungan Unsur Iklim Terhadap Produktivitas Cabe Jamu (Piper Retrofractum Vahl) Di Kabupaten Sumenep. *AGROVIGOR, 2*.

Subagya, E. H., Iswadi, Poerwaningsih, R., Drajat, D., Siagian, S. H., Hartini, M., et al. (2015). *Badan Pusat Statistik*. Dipetik Desember 1, 2017, dari https://www.bps.go.id/dynamictable/2015/09/09/865/produksi-padi-menurut-provinsi-ton-1993-2015.html

Suhartanto, R. S., Dewi, C., & Muflikhah, L. (2017). Implementasi Jaringan Syaraf Tiruan Backpropagation Untuk Mendiagnosis Penyakit Kulit Pada Anak. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1*, 555-562.

Wahed, M. (2015). Pengaruh Luas Lahan, Produksi, Ketahanan Pangan dan Harga Gabah Terhadap Kesejahteraan Petani Padi di Kabupaten Pasuruan. *JESP, 7*.

Wiranti, R. D. (2013). *Metode Pembelajaran dan Fungsi Aktivasi JST*. Dipetik Desember 1, 2017, dari http://r_d_w-fst10.web.unair.ac.id/artikel_detail-76032-Umum-Metode%20Pembelajaran%20dan%20Fungsi%20Aktivasi%20JST.html

Yana, Y. (2017). *Manfaat.co.id*. Dipetik May 1, 2018, dari https://manfaat.co.id/manfaat-padi

Yuliawan, T., & Handoko. (2012). Pengaruh Kenaikan Suhu Terhadap Produktivitas Tanaman Padi Sawah Irigasi Dan Tadah Hujan Di Indonesia Menggunakan Model Simulasi Pertanian Sheirary Rice Berbasiskan Sistem Informasi Geografis (SIG). *JSPUI*.

Zamani, A. M., Amaliah, B., & Munif, A. (2012). Implementasi Algoritma Genetika pada Struktur Backpropagation Neural Network untuk Klasifikasi Kanker Payudara. *JURNAL TEKNIK ITS, 1*.