PERBANDINGAN METODE JARINGAN SARAF TIRUAN *BACKPROPAGATION* DAN *LEARNING VECTOR QUANTIZATION* DALAM DETEKSI HAMA PENGEREK BATANG (Studi Kasus: Kabupaten Seram Bagian Barat Provinsi Maluku)

Safriyani Tomia[1], Z. A. Leleury[2], S. N. Aulele[3]
[1,2,3] Jurusan Matematika FMIPA Universitas Pattimura
Jln. Ir. M. Putuhena, Kampus Unpatti, Poka-Ambon, Indonesia, Kode Pos 97233
e-mail: : [2]zetharthur82@gmail.com

Abstrak

Jaringan saraf tiruan adalah pemrosesan sistem informasi pada karakteristik tertentu yang merupakan representatif buatan berdasarkan jaringan saraf manusia. Jaringan Saraf Tiruan (JST) telah banyak dikaji dalam berbagai bidang melalui pengenalan pola. Kemudian dengan memanfaatkan jaringan saraf tiruan LVQ dan Backpropagation dibuat sistem perbandingan deteksi dini hama pengerek batang padi dengan menggunakan Software MATLAB dengan melakukan pengujian. Dari hasil pengujian dengan menggunakan metode *backpropagation* diperoleh hasil akurasi 69,44% sedangkan untuk hasil pengujian metode LVQ diperoleh 80,56%. Dari hasil penelitian bahwa metode LVQ dianggap baik dalam mendeteksi hama pengerek batang padi.

Kata Kunci: *Backpropagation, hama pengerek batang padi, jaringan saraf tiruan, learning vector quantization, MATLAB*

COMPARISON OF ARTIFICIAL NEURAL NETWORK METHODS BACKPROPAGATION AND LEARNING VECTOR QUANTIZATION IN THE DETECTION OF STEM BORER (Case study: Western Seram, Maluku Province)

Abstract

The Artificial Neural Networks (ANN) is a process of informational system on certain traits which are artificial representatives of the human neural networks. Artificial Neural Networks has been studied in various fields through pattern recognition. Then with utilize Artificial neural networks Backpropagation and Learning Vector Quantization was made by system early detection compare gimlet Pest erects paddy by use of Software MATLAB by undertaking examination. From examination result by using backpropagation method obtained the accuration as 69,44% whereas for examination result method LVQ is 80,56%. Of that research result LVQ'S method is thought fit in detect Gimlet Pest Erects Paddy

Keywords: *Artificial neural networks, backpropagation,* Gimlet Pest erects Paddy*, internal diseases, learning vector quantization, MATLAB*

1. Pendahuluan

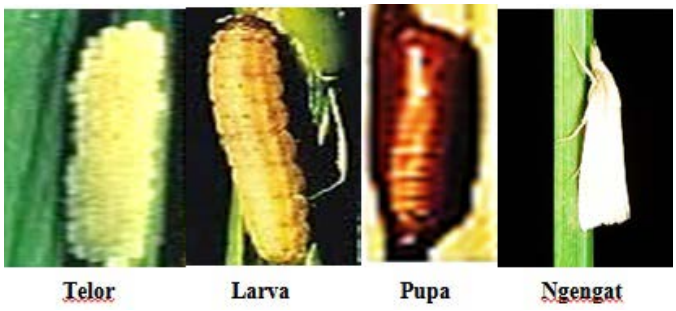
Pertanian Provinsi Maluku telah melakukan upaya untuk mengatasi masalah yang berkaitan dengan hama pengerek batang padi yang menyerang perkebunan-perkebunan yang ada di Maluku. Teknologi baru, seperti bioteknologi dan ilmu komputer dan kemajuan teknologi lainnya telah memungkinkan untuk mengembangkan bidang penelitian baru, termasuk di bidang rekayasa genetik, agrofisika, statistika pertanian dan pertanian presisi. Namun, sistematika ilmu pertanian bukan hanya bertumpu pada ilmu pertanian itu sendiri tetapi juga bertumpu pada ilmu pengetahuan yang lain di antaranya matematika, staistik dan teknik. Oleh sebab itu, ilmu pertanian sangatlah luas mencakup ilmu-ilmu yang lain.

Perkembangan teknologi pengenalan pola saat ini banyak menciptakan aplikasi-aplikasi baru sesuai dengan perkembangan zaman. Oleh sebab itu, penelitian-penelitian menggunakan aplikasi JST melalui pengenalan pola terjadinya sesuatu telah banyak dikaji, penggunaan jaringan saraf tiruan metode Learning Vector Quantization untuk mendiagnosa penyakit saluran pernapasan [1]. Selain itu JST dengan metode b*ackpropagation* dan *Learning Vector Quantization* untuk sistem diagnosa penyakit dalam [2]. Pada penelitian ini, peneliti akan menggunakan jaringan saraf tiruan metode *backpropagation* dan *Learning Vector Quantization* untuk memecahkan persoalan-persoalan terkait dengan faktor-faktor lain yang mempengaruhi hama pengerek batang yang sering menyerang tanaman-tanaman perkebunan, yang sebelumnya telah diselesaikan dalam [3] dengan menggunakan metode *backpropagation*.

2. Tinjauan Pustaka

2.1. Hama Pengerek Batang Padi

Pengerek batang adalah salah satu hama padi yang paling berbahaya dan merugikan. Serangan pengerek batang padi bisa terjadi semenjak di persemaian sampai masa pertumbuhan dan perkembangannya. Kadang-kadang lebih dari satu jenis pengerek batang menyerang tanaman padi dalam waktu yang tidak bersamaan, sehinggga sebagian petani merasa kesulitan dalam pengendaliannya. Pengerek batang dapat menyebabkan merosotnya hasil padi karena anakan yang rusak oleh sundep. Oleh sebab itu banyak padi yang kerap mengalami gagal panen.



Gambar 1. Hama Pengerek Batang

2.2. Faktor yang Mempengaruhi Hama Pengerek Batang Padi.

Faktor-faktor yang mempengaruhi hama pengerek batang padi diantaranya:

1) Curah hujan
2) Suhu udara
3) Kecepatan angin
4) Kelembapan udara
5) Intenstas cahaya

2.3. Jaringan Saraf Tiruan

Setiap pola-pola informasi *input* dan *output* yang diberikan ke dalam JST diproses dalam *neuron*. *Neuron-neuron* tersebut terkumpul di dalam lapisan-lapisan yang disebut *neuron layer*s. Lapisan-lapisan penyusun JST tersebut dapat dibagi menjadi 3, yaitu:

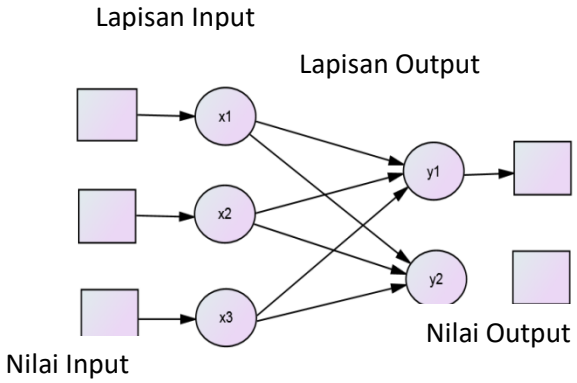
1) Lapisan *input*, unit-unit di dalam lapisan *input* disebut unit-unit *input*. Unit-unit *input* tersebut menerima pola data dari luar yang menggambarkan suatu permasalahan.
2) Lapisan tersembunyi, unit-unit di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Di mana *output*nya tidak dapat secara langsung diamati.
3) Lapisan *Output*, unit-unit di dalam lapisan *output* disebut unit-unit *output*. *Output* dari lapisan ini merupakan solusi JST terhadap suatu permasalahan.

2.3.1 Arsitektur Jaringan Saraf Tiruan

JST memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai aplikasi. Arsitektur JST tersebut, antara lain:

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

Jaringan dengan lapisan tunggal terdiri dari 1 lapisan *input* dan 1 lapisan *output*. Setiap *neuron* yang terdapat di dalam lapisan *input* selalu terhubung dengan setiap *neuron* yang terdapat pada lapisan *output*. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi.



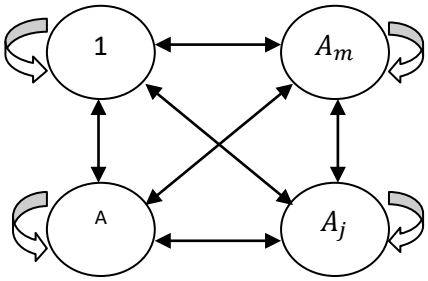
Gambar 2. Arsitektur Lapisan

2) Jaringan banyak lapisan (*Multilayer Net*)

Jaringan dengan lapisan jamak memiliki ciri khas tertentu yaitu memiliki 3 jenis lapisan yakni lapisan *input*, lapisan *output*, dan lapisan tersembunyi. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal.Namun, proses pelatihan sering membutuhkan waktu yang cenderung lama.

3) Jaringan lapisan kompetitif (*Competitive Layer*)

Pada jaringan ini sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Contoh algoritma yang menggunakan jaringan ini adalah LVQ yang ditunjukkan oleh Gambar 3:



Gambar 3. Arsitektur Lapisan Kompetitif

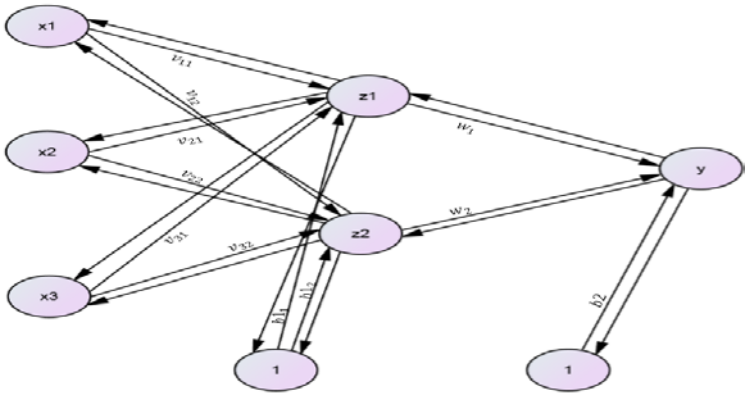
2.4. Jaringan *Backpropagation*

Model jaringan *backpropagation* merupakan suatu teknik pembelajaran atau pelatihan *supervised learning* yang paling banyak digunakan. Metode ini merupakan salah satu metode yang sangat baik dalam menangani masalah pengenalan pola-pola kompleks. Di dalam jaringan *backpropagation*, setiap unit yang berada di lapisan *input* berhubungan dengan setiap unit yang ada di lapisan tersembunyi. Setiap unit yang ada di lapisan tersembunyi terhubung dengan setiap unit yang ada di lapisan *output*. Jaringan ini terdiri dari banyak lapisan (*multilayer network*). Ketika jaringan ini diberikan pola masukan sebagai pola pelatihan, maka pola tersebut menuju unit-unit lapisan tersembunyi untuk selanjutnya diteruskan pada unit-unit di lapisan keluaran. Kemudian unit-unit lapisan keluaran akan memberikan respon sebagai keluaran JST. Saat hasil keluaran tidak sesuai dengan yang diharapkan, maka keluaran akan disebarkan mundur (*backward*) pada lapisan tersembunyi kemudian dari lapisan tersembunyi menuju lapisan masukan.

Tahap pelatihan ini merupakan langkah untuk melatih suatu JST, yaitu dengan cara melakukan perubahan bobot, sedangkan penyelesaian masalah akan dilakukan jika proses pelatihan tersebut telah selesai, tahap ini disebut tahap pengujian

2.4.1 Arsitektur *Backpropagation*

Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar 4, pada jaringan *backpropagation* terdiri dari tiga *neuron* pada lapisan *input*, yaitu x_1, x_2, x_3, 1 lapisan tersembunyi dengan 2 *neuron*, yaitu z_1, z_2, serta 1 unit pada lapisan *output*, yaitu y_1. Bobot yang menghubungkan x_1, x_2, x_3dengan *neuron* pertama pada lapisan tersembunyi, adalahv_1, v_2, v_3. Untuk $b1_1, b1_2$adalah bobot bias yang menuju ke *neuron* pertama dan kedua pada lapisan tersembunyi. Bobot yang menghubungkan z_1, z_2 dengan *neuron* pada lapisan *output*, adalah w_1, w_2. Bobot bias $b2$ menghubungkan lapisan tersembunyi dengan lapisan *output*.



Gambar 4. Arsitektur Jaringan *Backpropagation*

2.4.2 Pelatihan Standar *Backpropagation*

Pelatihan *backpropagation* meliputi 3 tahap. Tahap pertama adalah tahap maju. Pola masukan dihitung maju mulai dari lapisan masukan hingga lapisan keluaran menggunakan fungsi aktivasi yang ditentukan. Tahap kedua adalah Tahap mundur. Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, dimulai dari garis yang berhubungan langsung dengan unit – unit di layar keluaran. Tahap ketiga adalah modifikasi bobot untuk menurunkan kesalahan yang terjadi.

Tahap I. Propagasi maju

Selama propagasi maju, sinyal masukan (x_i) dipropagasikan ke lapisan tersembunyi menggunakan fungsi aktivasi yang ditentukan. Keluaran dari setiap unit lapisan tersembunyi (z_j) tersebut selanjutnya dipropagasikan maju lagi ke layar tersembunyi di atasnya menggunakan fungsiaktivasi yang ditentukan. Demikian seterusnya hingga menghasilkan keluaran jaringan (y_k).

Berikutnya, y_k dibandingkan dengan target yang harus dicapai (t_k). Selisih $t_k - y_k$ adalah kesalahan yang terjadi. Jika kesalahan ini lebih kecil dari batas toleransi yang ditentukan, maka iterasi dihentikan. Akan tetapi apabila kesalahan masih lebih besar dari batas toleransinya, maka bobot setiap baris dalam jaringan akan dimodifikasi untuk mengurangi kesalahan yang terjadi.

Tahap II. Propagasi mundur

Berdasarkan kesalahan $t_k - y_k$, dihitung faktor δ_k $(k = 1,2, \dots, m)$ yang dipakai untuk mendistribusikan kesalahan di unit y_k ke semua unit tersembunyi yang terhubung langsung dengan y_k. Faktor δ_k juga dipakai untuk mengubah bobot baris yang berhubungan langsung dengan unit keluaran.

Dengan cara yang sama, dihitung faktor δ_j di setiap unit di lapisan tersembunyi sebagai dasar perubahan bobot semua garis yang berasal dari unit tersembunyi di lapisan di bawahnya. Demikian seterusnya hingga semua faktor δ di unit tersembunyi yang berhubungan langsung dengan unit masukan dihitung.

Tahap III. Perubahan bobot

Setelah semua faktor δ dihitung, bobot semua garis dimodifikasi bersamaan. Perubahan bobot suatu garis didasarkan atas faktor δ *neuron* di lapisan atasnya. Sebagai contoh, perubahan bobot garis yang menuju ke lapisan keluaran didasarkan atas δ_k yang ada di unit keluaran.

Ketiga tahap tersebut diulang-ulang terus hingga kondisi penghentian dipenuhi. Umumnya kondisi penghentian yang sering dipakai adalah jumlah iterasi atau *kesalahan*. Iterasi akan dihentikan jika jumlah iterasi yang dilakukan sudah melebihi jumlah maksimum iterasi yang ditetapkan, atau jika kesalahan yang terjadi sudah lebih kecil dari batas toleransi yang diijinkan.

Algoritma pelatihan untuk jaringan dengan satu lapisan *tersembunyi* (dengan fungsi aktivasi sigmoid biner) adalah sebagai berikut:

Langkah 0: Inisialisasi semua bobot dengan bilangan acak kecil;

Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-8;

Langkah 2: Untuk setiap pasang data pelatihan, lakukan langkah 3-8;

Tahap I. Propagasi Maju

Langkah 3: Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi diatasnya;

Langkah 4: Hitung semua keluaran di unit tersembunyi z_j $(j = 1,2, \dots, p)$;

$$z\_net_j = v_{jo} + \sum_{i=1}^{n} x_i v_{ji}$$

$$z_j = f(z\_net_j) = \frac{1}{1 + e^{-z\_net_j}}$$

Langkah 5: Hitung semua keluaran jaringan di unit y_k $(k = 1,2, \dots, m)$;

$$y\_net_k = w_{ko} + \sum_{j=1}^{p} z_j w_{kj}$$

$$y_k = f(y\_net_k) = \frac{1}{1 + e^{-y\_net_k}}$$

Tahap II. Propagasi mundur

Langkah 6: Hitung faktor δ unit keluaran berdasarkan eror di setiap unit keluaran y_k $(k = 1,2, \dots, m)$;

$$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k) y_k (1 - y_k)$$

δ_k merupakan unit eror yang akan dipakai dalam perubahan bobot lapisan di bawahnya (langkah 7).

Hitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju percepatan α

$$\Delta w_{kj} = \alpha \delta_k z_j \quad ; k = 1,2, \dots, m \;\; ; j = 0,1,2, \dots, p;$$

Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan eror di setiap unit tersembunyi $z_j (j = 1,2, \dots, p)$

$$\delta\_net_j = \sum_{k=1}^{m} \delta_j w_{kj}$$

Faktor δ unit tersembunyi:

$$\delta_j = \delta\_net_j f'(z\_net_j) = \delta\_net_j z_j (1 - z_j).$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti untuk merubah bobot v_{ji})

$$\Delta v_{ji} = \alpha \delta_j x_i \qquad ; j = 1,2, \dots, p \;\; ; i = 0,1,2, \dots, n.$$

Tahap III. Perubahan bobot

Langkah 8: Hitung semua perubahan bobot;

Perubahan bobot garis yang menuju ke unit keluaran:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj}(k = 1,2, \dots, m \;\; ; j = 0,1,2, \dots, p);$$

Perubahan bobot garis yang menuju ke unit tersembunyi:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji}(j = 1,2, \dots, p \;\; ; i = 0,1,2, \dots, n).$$

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai untuk menentukan keluaran jaringan.

Apabila fungsi aktivasi yang dipakai bukan sigmoid biner, maka langkah 4 dan 5 harus diselesaikan. Demikian juga turunannya pada langkah 6 dan 7.

*Langkah 9***:** Uji kondisi berhenti (akhir iterasi).

2.5. Jaringan *Learning Vector Quantization* (LVQ)

Model jaringan *Learning Vector Quantization* merupakan salah satu jenis jaringan saraf tiruan berbasis *competitive learning* atau *winner take all* yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor *input*. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor *input*. Jika 2 vektor *input* mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor *input* tersebut ke dalam kelas yang sama. Arsitektur model jaringan LVQ dapat dilihat pada Gambar 5. [4])

2.5.1 Algoritma *Learning Vector Quantization*

Motivasi untuk algoritma jaringan LVQ adalah untuk mencari unit *output* yang terdekat dengan vektor *input*. Jika x dan w merupakan kelas yang sama, maka bobot dipindahkan terhadap vektor *input* baru, jika x dan w merupakan kelas-kelas yang berbeda, maka bobot dipindahkan dari *input* vektor.

Algoritma yang digunakan adalah sebagai berikut [4]:

x vektor training$(x_1, x_2, \dots, x_n)$;

T kategori/target yang benar untuk vektor training;

W_j vektor bobot untuk unit *output* ke-$j(w_{1j}, \dots, w_{ij}, \dots, w_{nj})$;

C_j kategori/kelas hasil komputasi oleh unit *output j;*

$\|x_i - W_j\|$ jarak *Euclidean* antara vektor *input* dengan unit *output;*

Langkah 0. Inisialisasi vektor referensi;

Inisialisasi laju pelatihan (*learning rate*) $\alpha(0)$;

Langkah 1. Bila kondisi STOP belum dipenuhi, kerjakan Langkah 2 – 6;

Langkah 2. Untuk setiap vektor *input training* kerjakan Langkah 3 – 4;

Langkah 3. Dapatkan j sedemikian hingga $\|x - w_{ij}\|$minimum;

Langkah 4. Update W_{ij} sebagai berikut:

Jika $C_j = T$, maka

$W_{ij}(baru) = W_{ij}(lama) + \alpha[x_i - W_{ij}(lama)]$;

Jika $C_j \neq T$, maka

$W_{ij}(baru) = W_{ij}(lama) - \alpha[x_i - W_{ij}(lama)$;

Langkah 5. Reduksi laju pelatihan

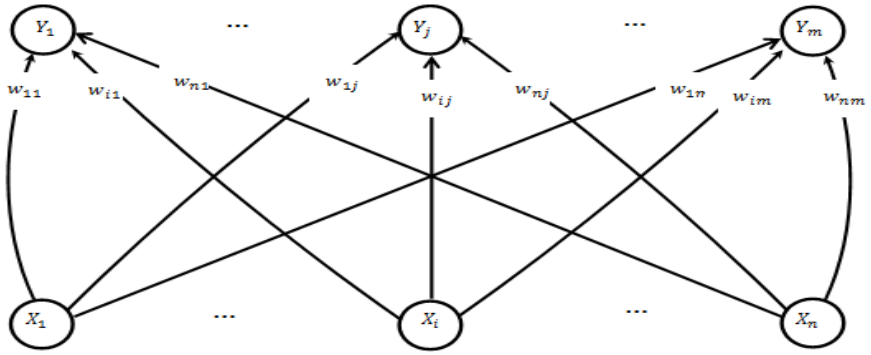
$\alpha = \alpha * Dec\ \alpha$;

Langkah 6. Tes kondisi STOP

Misal: 1. Dengan membatasi jumlah iterasi;

2. Setelah α mencapai nilai toleransi.

Metode paling sederhana dalam menginisialisasi vektor bobot (referensi) adalah dengan mengambil vektor *trainingm* pertama kali dan meggunakannya sebagai vektor bobot; vektor-vektor yang lain digunakan untuk *testing* (*Kohonen*, 1989).



Gambar 5. Jaringan *Learning Vector Quantization*

3. Hasil dan Pembahasan

3.1. Variabel Penelitian

Dalam penelitian ini variabel yang digunakan adalah curah hujan, suhu udara, kelembaban udara, kecepatan angin dan intensitas cahaya (lamanya penyinaran matahari) sebagai data *input*nya. Sedangkan hama pengerek batang sebagai target atau *output*nya.

Data curah hujan, suhu, kelembaban udara, kecepatan angin dan intensitas cahaya (lamanya penyinaran matahari) yang diperoleh dari Badan Meteorologi Klimatologi dan Geofisika (BMKG) provinsi Maluku, masing-masing data merupakan rekapan bulanan selama sebelas tahun terakhir dari tahun 2006-2016. Begitu pula untuk data potensi hama pengerek batang di peroleh dari Dinas Pertanian Provinsi Maluku. Data yang diperoleh adalah sebagai berikut:

Tabel 1. Data Curah Hujan Kabupaten Seram Bagian Barat 2006-2016 (mm)

Tahun	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH	CH
2006	147	209	188	107	147	946	158	32	105	11	78	91
2007	49	153	79	171	153	627	109	156	326	222	74	182
2008	153	137	221	229	227	553	650	1060	219	317	83	217
2009	239	234	135	213	97	209	193	91	17	73	118	91
2010	176	13	127	97	301	529	508	574	331	109	93	300
2011	127	125	268	80	671	487	299	126	281	235	128	154
2012	96	171	234	140	146	426	679	656	142	93	46	110
2013	323	144	62	153	285	206	528	378	180	128	69	97
2014	188	82	149	106	265	233	132	369	87	46	113	117
2015	107	244	120	100	156	674	129	3	1	17	111	29
2016	105	30	111	221	148	181	247	229	192	348	27	80

Sumber : BMKG provinsi maluku

Tabel 2. Data Suhu Udara Kabupaten Seram Bagian Barat 2006-2016 (o_c)

Tahun	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH	SH
2006	26,6	27,3	27,2	27,1	26,3	25,3	25,1	24,5	25,1	25,4	26,5	27,2
2007	27,7	26,9	27	26,5	26,2	26	25	24,7	25,2	26	26,7	26,7
2008	26,8	26,8	26,2	26,1	25,8	25,3	24,7	24,6	25,4	26,1	26,9	26,7
2009	26,9	26,8	26,6	26,5	26,5	25,5	24,9	25,1	25,6	26,3	27	27,4
2010	27	27,7	27,5	27,7	27,2	26,3	26,1	25,8	26,1	26,9	26,8	27,1
2011	26,8	26,7	26,5	27,1	25,9	25	25,3	25,1	25,6	26,4	27,4	27,5
2012	27,3	26,9	25	26,7	26,4	25,5	25,1	25	26	26,8	27,6	27,6
2013	27,4	27,3	28	27,1	26,8	26,5	25,3	26	26	27	27,4	27,4
2014	27,2	27,1	27,1	27,3	26,8	26,2	25,5	25,1	25,5	26,4	27,5	27,5
2015	27,4	26,9	26,9	27	26,6	25,8	25,1	24,7	25	25,8	27,6	28,2
2016	27,9	28	28	27,4	27,7	26,7	225,9	26,1	26,5	27	28,1	27,8

Sumber : BMKG provinsi maluku

Tabel 3. Data Kecepatan Angin Kabupaten Seram Bagian Barat 2006-2016 (*knots*)

Tahun	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
	KA	KA	KA	KA	KA	KA	KA	KA	KA	KA	KA	KA
2006	3	3	4	3	3	3	5	6	6	6	4	4
2007	4	4	3	3	4	1	2	2	2	2	2	1
2008	3	3	3	3	3	2	2	3	4	2	1	1
2009	1	1	1	1	1	1	2	2	2	2	1	1
2010	1	2	2	2	1	2	1	1	1	2	2	2
2011	1	2	1	1	3	6	6	7	6	7	5	4
2012	4	4	5	4	3	4	5	6	7	9	6	5
2013	7	7	8	7	6	7	4	5	5	5	3	3
2014	0	5	4	3	3	3	4	4	5	6	4	3
2015	5	4	4	3	3	3	4	4	6	7	5	5
2016	4	5	4	3	3	3	3	5	3	4	4	4

Sumber : BMKG provinsi maluku

Tabel 4. Data Kelembaban Udara Kabupaten Seram Bagian Barat 2006-2016 ($km/m^3, \%$)

Tahun	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
	KU	KU	KU	KU	KU	KU	KU	KU	KU	KU	KU	KU
2006	85	85	84	86	88	93	88	84	86	83	85	84
2007	85	83	83	87	88	82	87	89	88	89	86	87
2008	86	85	87	88	89	90	93	93	90	90	88	87
2009	88	87	87	88	89	89	90	87	87	86	86	83
2010	86	82	84	84	88	90	90	91	90	88	86	86
2011	86	85	87	87	91	60	88	86	89	87	85	85
2012	83	84	89	85	88	88	91	89	87	85	83	84
2013	84	84	82	86	88	88	91	88	87	84	83	84
2014	85	83	84	85	88	90	86	88	85	84	84	85
2015	84	84	81	86	86	89	86	84	83	86	84	81
2016	83	81	83	87	87	87	89	86	88	86	83	84

Sumber : BMKG provinsi maluku

Tabel 5. Data Intensitas Cahaya atau Penyinaran Matahari Kabupaten Seram Bagian Barat 2006-2016

Tahun	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
	PM	PM	PM	PM	PM	PM	PM	PM	PM	PM	PM	PM
2006	48	71	65	52	65	18	47	73	50	87	79	82
2007	68	49	60	47	56	51	34	27	49	62	57	53
2008	62	48	52	49	54	46	20	14	39	51	69	49
2009	53	58	67	64	52	36	36	62	75	7	77	81
2010	40	83	82	78	55	44	40	49	50	65	59	50
2011	53	52	48	62	30	21	38	45	34	68	77	50
2012	64	64	48	59	50	22	21	30	49	73	78	65
2013	46	65	70	53	57	48	20	29	~~41~~	0	64	62
2014	41	62	69	66	53	35	59	41	68	80	77	58
2015	48	62	66	53	66	46	60	68	89	83	76	81
2016	77	78	60	58	67	52	35	66	43	56	77	65

Sumber : BMKG provinsi maluku

Tabel 6 Data Luas Tambah Serangan Hama Pengerek Batang Kabupaten Seram Bagian Barat 2006-2016 (hektar)

	Luas Tambah Serangan Hama Setiap bulan di Seram Bagian Barat											
Tahun	Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
2006	0	0	0	0	0	0	0	0	0	0	0	0
2007	5	21	0	0	2	82	300	287	11	22	15	71
2008	71	0	0	0	25	27	91	2	0	0	46	0
2009	0	0	1	35	35	0	32	6	23	195	4	0
2010	2	0	165	198	196	0	16	235	299	98	0	47
2011	22,5	13	0	0	390,5	25,5	0,5	0,5	0	0	114,5	0
2012	8,5	0	80	0	5,8	0	0,5	13	4	0	0	4
2013	2	5,8	4	0	1,3	0,45	0,75	2	1,5	0	0,25	2
2014	5,75	2,5	0,5	8	0	0	31	16,5	5	0	0	2
2015	7,5	13	0,35	0	1	0	0,25	0	0	0	0	0,25
2016	14,5	29	0	0,3	2,3	9,4	7,2	4,1	1,3	0	0	5,3

Sumber : Dinas Pertanian Provinsi Maluku

Seluruh data yang telah terkumpul kemudian dipisahkan menjadi 2 bagian yaitu masukan dan keluaran. Data-data yang tergolong sebagai masukan secara berurut adalah sebagai berikut: Curah hujan sebagai x_1; Suhu udara sebagai x_2; Kecepatan angin sebagai x_3; Kelembapan udara sebagai x_4; Intensitas cahaya sebagai x_5; Sedangkan Luas Tambah Serangan Hama Pengerek Batang yang tergolong sebagai keluaran atau target yang diinginkan adalah sebagai variabel y.

Banyaknya data yang diperoleh dalam penelitian ini sebanyak 132, diantaranya 96 data digunakan sebagai data *training* dan 36 data digunakan untuk data *testing*. Dalam penelitian ini keluaran atau target yang diinginkan yaitu Deteksi Hama dengan pembagian pola menjadi 4 bagian yaitu :

Pola 0 untuk nilai $b \leq 0{,}5$ dengan ketentuan tidak ada hama atau bebas hama;

Pola 1 untuk nilai $0{,}5 < b < 1{,}5$ dengan ketetntuan ada hama berkisar 1-100;

Pola 2 untuk nilai $1{,}5 < b < 2{,}5$ dengan ketetntuan ada hama berkisar 101-200; dan

Pola 3 untuk nilai $b > 2{,}5$ dengan ketentuan ada hama berkisar 201-300.

3.2. Analisis Penelitian Menggunakan Metode *Backpropagation*

Pada metode *backpropagation*, arsitektur sebuah jaringan akan menentukan keberhasilan target yang akan dicapai karena tidak semua permasalahan dapat diselesaikan dengan arsitektur yang sama. Banyaknya lapisan tersembunyi ditentukan sendiri oleh pengguna sistem melalui cara percobaan konvergensi terbaik (*trial* dan *error*) sampai diperoleh hasil konvergensi pelatihan yang paling baik (jumlah *epoch* terkecil). Parameter sistem masukan untuk pembentukan pola yang dibentuk, yaitu:

Net Size: *Input Layer* : 5 *neuron*

Hidden Layer : 20 *neuron* 10 *neuron* 5 *neuron* dan 1 *neuron*

Output Layer : 1 *neuron*

Maksimum epoch / iterasi : 5000

Show Epoch : 100

Learning Rate : 0.85

Dari 96 data pelatihan dan 36 data uji, diperoleh analisis sebagai berikut. Untuk masing-masing variasi nilai α, jumlah iterasi maksimum sama yaitu jumlah iterasi (epoch) 5000.

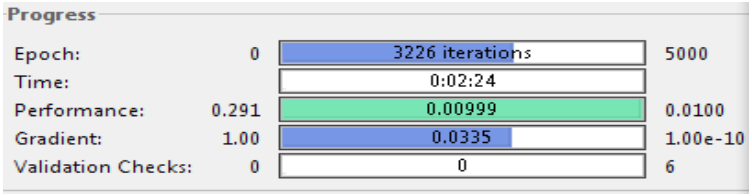
Berikut disajikan dalam Tabel 7 sebagai berikut:

Tabel 7. Hasil Analisa Data Metode Backpropagation

No	A	MSE	Epoch	Waktu	HasilPelatihan		HasilPengujian	
					Data yang dikenali	Tingkat keakuratan	Data yang dikenali	Tingkat keakuratan
1	0.1	0.0624	5000	02:40	66	68.75%	21	58.33%
2	0.3	0.0227	5000	02:52	75	78.13%	17	47.2%
3	0.4	0.0117	5000	03:31	73	76.04%	17	47.2%
4	0.5	0.0221	5000	03:31	70	72.92%	20	55.5%
5	0.6	0.00999	3692	02:35	95	99%	23	64%
6	0.7	0.0235	5000	04:20	56	58.33%	15	41.67%
7	0.8	0.0106	5000	03:40	53	55.21%	14	38.89%
8	***0.85***	***0.00999***	***3226***	***02:24***	***95***	***99%***	***25***	***69,44%***

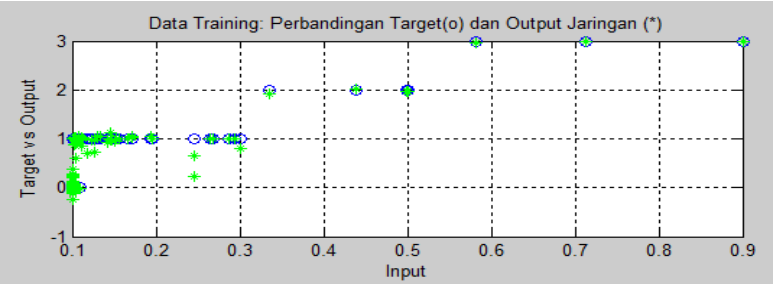
Berdasarkan Tabel 7, hasil yang didapat pada tahap training dan testing adalah arsitektur jaringan yang paling baik dalam proses deteksi dini hama pengerek batang adalah pada saat alpha 0,6 dengan jumlah iterasi, dan MSE yang sama. Namun yang membedakan ada pada data latih yang dikenali dan data non latih yang dikenali.

Tingkat keakuratan data yang dihasilkan dari proses training dan testing terbaik adalah pada saat α (*learning rate*) 0.85, dengan maksimum iterasi 3226, nilai MSE 0.00999 dan tingkat keakuratan data 99% dan 69,44% dari 96 data training dan 36 data testing. Untuk lebih jelasnya dapat dilihat pada hasil training (Gambar 6) dan hasil testing (Gambar 7).



Gambar 6. *Progress Neural Network Training*

Pada Gambar 7 akan ditunjukkan hasil analisa data (α) dari *learning rate* 0.85. Hasil perbandingan antara target (o) dan *output* jaringan (+) dapat diamati dengan cara memperhatikan penempatan posisi *output* jaringan (+). Jika *output* jaringan (+) menempati posisi yang sama dengan target (o) maka hasil analisa data tersebut dikatakan benar.

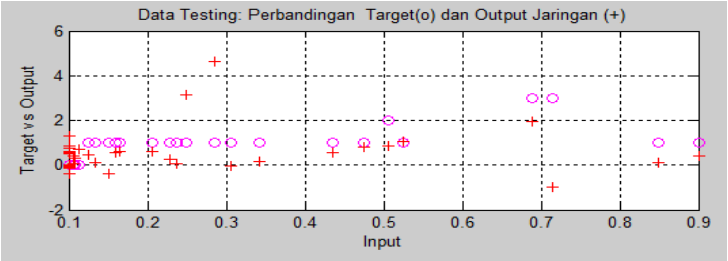


Gambar 7. *Grafik Hasil Training Metode Backpropagation*

Berikut ini disajikan tabel perbandingan antara target yang diharapkan dan target hasil pengujian data. Selisih antara target *input* dan *target output* yang lebih dari 0.5 dibaca sebagai hasil yang berbeda dari deteksi yang sebenarnya dan dianggap salah. Adapun hasil pengujian didapat sebagai berikut:

Tabel 8. Hasil Analisa deteksi tahap *Testing* jaringan Metode Backpropagation

No	Target *Input*	Target *Output*	Hasil Analisa Jaringan	Keterangan
1	3	3.1338	3	Benar
2	1	0.6184	1	Benar
3	1	0.6903	1	Benar
4	0	-0.0539	0	Benar
5	1	0.8468	1	Benar
6	3	1.2940	1	Salah
7	1	0.4133	0	Salah
8	0	10.662	1	Salah
9	1	0.0879	0	Salah
10	1	0.5118	1	Benar
11	0	0.8661	1	Salah
12	0	0.7330	1	Salah
13	1	0.0160	0	Salah
14	1	0.5453	1	Benar
15	2	1.9339	2	Benar
16	1	0.7982	1	Benar
17	0	0.4697	0	Benar
18	0	-0.1459	0	Benar
19	0	0.4142	0	Benar
20	0	-0.4031	0	Benar
21	0	0.0049	0	Benar
22	0	0.5751	1	Salah
23	0	0.7593	1	Salah
24	0	0.0405	0	Benar
25	0	-0.0566	0	Benar
26	0	0.2560	0	Benar
27	0	-0.0710	0	Benar
28	0	0.3347	0	Benar
29	1	0.5537	1	Benar
30	0	0.1726	0	Benar
31	1	.46451	5	Salah
32	1	0.6074	1	Benar
33	0	0.0890	0	Benar
34	1	0.1386	0	Salah
35	0	0.4900	0	Benar
36	1	0.6380	1	Benar



Gambar 8. Grafik Hasil Testing metode *Backpropagation*

Tabel 9. Hasil *testing* metode LVQ

No	Alpha	*Dec α*	Hasil		
			Data uji yang dikenali	Data Uji yang tidak dikenali	tingkat keakuratan
1	0.1	0,2	22	14	61,11%
		0,7	22	14	61,11%
		0,75	23	13	63,89%
2	0,3	0,2	21	15	58,33%
		0,3	22	14	61,11%
		0,6	23	13	63,88%
		0,85	24	12	66,67%
3	0,7	0,1	21	15	58,33%
		0,5	20	16	55,56%
		0,6	22	14	61,11%
		0,8	21	15	58,33%
4	0,8	0,1	21	15	58,33%
		0,5	22	14	61,11%
		0,7	23	13	63,88%
		0,75	24	12	66,67%
6	0,07	0,2	20	16	55,56%
		0,6	21	15	58,33%
		0,7	23	13	63,88%
		0,75	25	11	69,44%
7	**0,08**	0,2	21	15	58,33%
		0,5	23	13	63,88%
		0,7	26	11	72,22%
		0,75	**29**	**7**	**80,56%**

Dari Tabel 8 terdapat 11 data yang tidak sesuai dengan target *input* dimana tingkat keakuratannya 30.56% sedangan 25 data lainnya yang sesuai dengan target *input* dimana tingkat keakuratannya 69.44%. Hal ini dapat dilihat dari error yang didapat saat pelatihan dan pengujian hasilnya errornya cukup kecil yang menyebabkan pendeteksi dikatakan baik.

3.3. Analisis Penelitian Menggunakan metode Learning Vector Quantization

Pada metode *Learning Vector Quantization*, bobot awal menggunakan data yang diambil secara acak dari data yang ada. Kemudian bobot tersebut akan diubah (di-*update*) tergantung dari kelas vektor masukan sesuai dengan kelas yang dinyatakan sebagai *neuron* pemenang. Bobot awal diambil dari data yang sudah ada.

Setelah bobot awal ditentukan, proses *training* dan *testing* kemudian dilakukan dengan menggunakan pemrograman MATLAB dengan beberapa nilai α (*learning rate*) dan *Dec* α (penurunan *learning rate*) pada max *epoch* 10 iterasi, untuk mengetahui tingkat keakuratan terbaik dari metode LVQ.

Berdasarkan Tabel 9, dapat disimpulkan bahwa tingkat keakuratan yang didapatkan adalah 80,56% nilai α (*learning rate*) 0,08 dan *Dec* α (penurunan *learning rate*) 0,75 dinyatakan baik.

4. Kesimpulan

Berdasarkan hasil penelitian dapat disimpulkan bahwa:

1) Metode jaringan saraf tiruan *Backpropagation* dan *Learning Vector Quantization* dapat digunakan untuk deteksi dini hama pengerek batang.
2) Deteksi dini hama pengerek batang padi menggunakan Learning Vector Quantization memberikan hasil yang lebih baik daipada metode *Backpropagation*.
3) Terdapat 25 dari 36 (69,44%) data uji yang dikenali Metode Backpropagation dan 29 dari 36 (80,56%) data uji yang dikenali metode *Learning Vector Quantization*.
4) Metode *Learning Vector Quantization* pada jaringan saraf tiruan lebih baik dalam deteksi dini hama pengerek batang dengan tingkat keakuratannya adala 80,56% dibandingkan metode *Backpropagation.*

Daftar Pustaka

[1] V. Souisa, " Aplikasi Jaringan Saraf Tiruan untuk Mendiagnosa Penyakit Saluran Pernapasan dengan Metode Learning Vector Quantization.," 2016.

[2] J. Madiuw, " Sistem Diagnosa Penyakit Dalam dengan Menggunakan Jaringan Saraf Tiruan Metode Backpropagation dan Learning Vector Quantization," 2016.

[3] S. H. Hurasan, "Aplikasi Jaringan Saraf Tiruan Metode Backpropagation untuk Mendeteksi Hama Pengerek Batang dengan Mempertimbangkan Faktor Curah Hujan,Suhu,Kelembaban dan Kecepatan Angin.," 2016.

[4] L. Fausset, " Fundamentals of Neural Networks, Architectures, Algoritus, and Application,," 1994.