KLASIFIKASI DATA MENGGUNAKAN JARINGAN SYARAF TIRUAN MODEL FRNN (*FULLY RECURRENT NEURAL NETWORK*)

Kornelis Letelay

Jurusan Ilmu Komputer, Fakultas Sains dan Teknik, Universitas Nusa Cendana

ABSTRACT

Artificial Neural Networks (ANN) can be used to solve specific problems such as prediction, classification, processing data, and robotics. Based on the exposure, this study tried to develop a system by applying ANN models Fully Recurrent Neural Network (FRNN). Fully Recurrent Neural Network structures have been presence of feedback that can make faster iteration thus making the update parameters and convergence speed become faster. The learning method used is Backpropagation Through Time. The system is implemented using the C# program. Input vectors used consisted of 7 variables.

The results showedt the developed system will rapidly converge and able to achieve the most optimal error value (minimum error) when using one hidden layer with 17 units of the number of neurons. The best accuracy can be obtained using the LR of 0.001, target of 0.1and momentum 0.95, with 30 the real data test, the system accuracy reaches 83.33%.

Keywords : *Artificial neural networks, fully recurrent neural network, data real non target*

PENDAHULUAN

Perkembangan *softcomputing* sebagai salah satu teknologi komputasi, telah banyak diaplikasikan dalam berbagai bidang. Salah satu metode *softcomputing* adalah jaringan syaraf tiruan (JST) yang memiliki kemampuan belajar terhadap informasi yang telah diterima dengan mensimulasikan proses pembelajaran seperti otak manusia. JST dapat digunakan untuk memecahkan permasalahan tertentu seperti prediksi, klasifikasi, pengolahan data, dan robotik.

Jaringan syaraf tiruan (*Artificial Neural Networks*) adalah salah satu cabang ilmu dari bidang kecerdasan buatan. Jaringan syaraf tiruan merupakan tiruan dari cara berpikir otak manusia, dapat berpikir dengan sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterima. Karakteristik dari *neural network* ditentukan

oleh beberapa hal, yaitu Arsitektur, *Learning algorithm* (algoritma untuk proses belajar), dan Fungsi aktifasi. Salah satu sifat menarik JST adalah kemampuannya untuk belajar (*learn*) dari lingkungannya dan dapat memperbaiki kinerjanya melalui pembelajaran (*learning*). JST belajar tentang lingkungannya melalui proses *iterative* penyesuaian yang diterapkan ke bobot selama proses belajar. Dalam penelitian ini digunakan metode JST *Fully Recurrent Neural Network* sebagai metode pembelajaran JST untuk melakukan pengujian terhadap data *real* non target. Data *real* non target adalah data nilai asli dalam bentuk data numerik yang dipakai sebagai *inputan* sistem yang tidak memiliki target *output*.

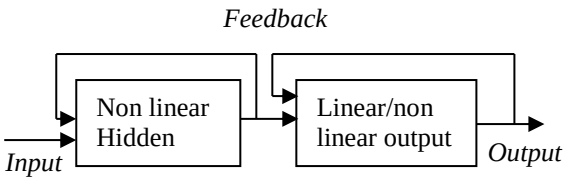
MATERI DAN METODE

Model FRNN

Model *Fully recurrent neural network* adalah Model pada jaringan syaraf tiruan yang memiliki masing-masing *context* unit pada *Hidden* layer dan *output* layer yang mengalami *feedback* pada dirinya sendiri (Samarasinghe, 2006).

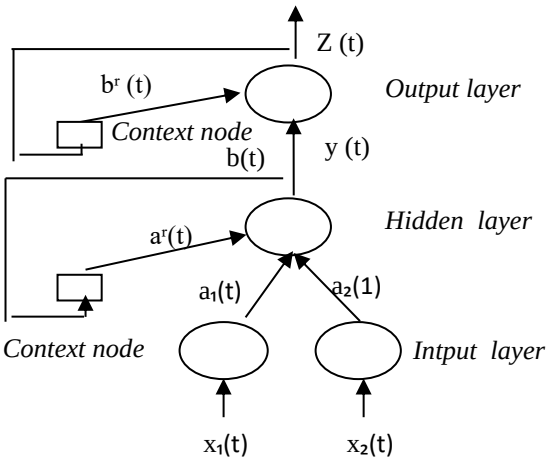
Fully Recurrent Neural Network juga mempunyai kompleksitas arsitektur sistem yang dinamis karena mempunyai *feedback* yang dapat mempercepat proses iterasi dan membuat kecepatan *update* parameter dan konvergensi menjadi lebih cepat (Samarasingghe, 2006). *Fully Recurrent Neural Network* dapat mengatasi masalah-masalah sederhana dengan memanfaatkan pengetahuan awal yang tersedia dengan lebih baik (Fransconi, 1994). Menurut William, (1989) model *Fully Recurrent Neural Network* juga dirancang untuk dapat mengatasi masalah-masalah khusus.

Keseluruhan sistem yang dibuat pada penelitian ini disesuaikan dengan konsep diagram *Fully recurrent neural network* pada Gambat 1.



Gambar 1. Konsep Diagram FRNN

Arsitektur *Fully Neural Network* hampir sama dengan arsitektur *Multilayer Feedforward* (MLP), namun ditambah dengan layer *context* untuk menampung hasil *output* dari *hidden* layer dan *output* layer seperti pada Gambar 2.



Gambar 2. Algoritma *Fully Recurrent NN*

Adapun tahapan dalam algoritma JST *Fully recurrent NN* adalah :

Sebuah jaringan *recurrent* sederhana, vektor masukannya juga sama disebarkan melalui lapisan berbobot, tetapi juga dikombinasikan dengan aktivasi keadaan sebelumnya melalui lapisan bobot tambahan *recurrent* "***r***" adalah *recurrent*.

Setiap unit akan menghitung aktivasinya seperti pada jaringan *feed forward* Gambar 2. Pada setiap layer akan memiliki indeks masing-masing. Variabel ***k*** untuk node *output*, ***j*** node tersembunyi, ***h*** untuk node *context*, dan ***i*** untuk node *input*. Sebuah jaringan umpan maju memiliki vektor masukan ***x***, yang disebarkan melalui lapisan berbobot ***a*** dan ***b***.

Proses algoritma *fully recurrent neural network* dapat dijelaskan sebagai berikut :

Training pola *input* saat time (t) dengan jaringan *recurrent* pada *context hidden* layer seperti persamaan (1.1) dan (1.2).

$$y_j\,(t) = f(net_j\,(t))$$

$$net_j(t) = \sum_i^n a_{ij}\; x_i(t) + \sum_h^m a^r\, y_h(t-1) + \theta_j$$

Training pola *output* saat time (t) dengan jaringan *recurrent* pada *context output* layer seperti persamaan (1.3) dan (1.4).

$$z_k(t) = f(net_k\,(t))$$

$$net_k(t) = \sum_j^n b_{jk}\; y(t) + \sum_h^m b^r\, z_h(t-1) + \theta_k$$

Back propagation untuk jaringan *recurrent* :

Setiap bobot dimodifikasi, dimana fungsi biaya (atau kesalahan) sehubungan dengan bobot dihitung dan kemudian disesuaikan. Fungsi biaya (*E*) yang paling sering digunakan adalah *sum square error* (SSE) pada persamaan (1.5).

$$SSE = \frac{1}{2}\sum_p^n \sum_k^m (T_{dk}(t) - \; Z_{pk}(t))^2$$

Menurut *gradien descent*, setiap perubahan bobot dalam jaringan harus sebanding dengan *gradien* negatif dari biaya sehubungan dengan bobot tertentu yang tertarik untuk dimodifikasi. ***α*** adalah sebuah *learning rate* seperti persamaan (1.6).

$$\Delta w = -\alpha \frac{\partial E}{\partial W}$$

Menggunakan aturan rantai dari differensiasi, bobot yang berkaitan dengan *error* pada *output* dapat dinyatakan dengan persamaan (1.7).

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial z}\frac{\partial z}{\partial net_k}\frac{\partial net_k}{\partial b} = \frac{\partial E}{\partial z}\frac{\partial z}{\partial net_k} = (z - T)z(1-z)$$

Sedangkan, *error* pada *hidden* layer dinyatakan dengan persamaan (1.8).

$$\frac{\partial E}{\partial a} = \delta b y_n(1-y_n) = \left[\sum_{i=1}^{k} \delta_{pk}\, b_{pk}\right] y_n\,(1-y_n)$$

Perubahan bobot untuk *output* dan perubahan bobot *input* dengan persamaan (1.9).

$$\Delta b_{jk} = \alpha \sum_p^n \delta_{pk}\, y_{pj}$$

Untuk perubahan bobot *input* dengan persamaan (1.10).

$$\Delta a_{ij} = \alpha \sum_p^n \delta_{pj}\, x_{pi}$$

Berdasarkan komponen waktu, perubahan bobot *recurrent* pada *hidden* dan *output* layer

dapat dinyatakan seperti persamaan (1.11) dan (1.12).

$$\Delta a^r = \alpha \sum_p^n \delta_{pj}(t)\pi(t)$$

$$\Delta b^r = \alpha \sum_p^m \delta_{pk}(t)\pi(t)$$

Perhitungan π *context* (untuk *time step* 1) pada *recurrent hidden* dengan persamaan (1.13). Sedangkan untuk *output* dengan persamaan (1.14).

$$\pi\,(t) = \frac{\partial y_j(t)}{\partial a^r(t)}$$

$$\pi\,(t) = f\left(net_j(t)\right)\left[y_h(t-1) + a^r(t)\frac{\partial y_j(t)}{\partial a^r(t)}\right]$$

$$= y_j(t)(1-y_j(t))\left[y_h(t-1) + a^r(t)\frac{\partial y_j(t)}{\partial a^r(t)}\right]$$

Perhitungan π *context* (untuk *time step* 1) pada *recurrent output* layer dengan persamaan (1.15) dan (1.16).

$$\pi(t) = \frac{\partial z_k(t)}{\partial b^r(t)}$$

$$\pi(t) = z_k(t)(1 - z_k(t))\left[z_h(t-1) + b^r(t)\frac{\partial z_k(t)}{\partial b^r(t)}\right]$$

Pelatihan akan berhenti apabila *error* < target *error*.

Adapun sistem yang dikembangkan diharapkan memiliki beberapa kemampuan antara lain :

Membaca data masukan

Melakukan *training* data

Melakukan *testing* data

Validasi Data

Analisis hasil

Menampilkan hasil

Untuk data *training* akan diproses dalam sistem sehingga kedua proses ini masing-masing diberikan data yang berbeda-beda yaitu data *training* sebanyak 70 data dan data *testing* sebanyak 30 data. Selama proses *training* dengan jumlah 70 data oleh JST FRNN didapat bobot akhir terbaik yang disimpan pada file bobot, untuk selanjutnya digunakan pada saat proses *testing*, dalam proses *testing* data di *load* dari data exel sebanyak 30 data. Proses *testing* dilakukan dengan mengambil bobot terbaik hasil *training* sebelumnya. Hasil dari proses *testing* berupa akurasi sistem terhadap pengenalan data *testing.* Contoh data yang diambil sebagai data *training* dan data *testing* adalah data hasil studi Bintara polri SPN Kupang sebanyak 100 data. Dengan data real sebanyak 30 data sebagai data uji, Tabel 1.

Rancangan arsitektur FRNN dari sistem yang sedang dikembangkan. FRNN terdiri dari beberapa lapisan, antara lain lapisan masukan (*input*), lapisan tersembunyi (*hidden*), dan lapisn keluaran (*output*). Pada lapisan masukan terdiri dari 7 *neuron*, dengan sejumlah unit neuron pada lapisan *hidden*, dan lapisan keluaran terdiri dari 2 *neuron*.

Tabel 1. Data Training da Testing

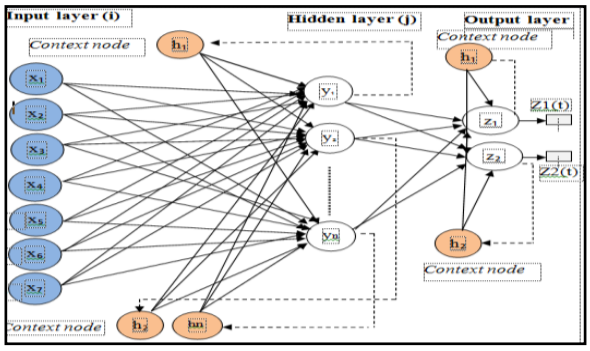
POP	KU	KUT	KKS	LATNIS	MENTAL	SAMJAS
90.55	90.89	90.9	86.97	73.8	72.5	80.13
89.89	90.98	90.99	87.7	76.63	72.9	81.05
89.73	86.78	89.79	85.88	76.61	72.7	78.7
88.84	90.23	88.74	83.3	74.76	72.8	80.93
85.72	86.69	86.66	86.99	74.28	72.8	87.75
85.25	90.67	90.57	87.73	74.76	72.9	80.8
84.97	90.89	87.97	90.9	74.45	72.8	86.88
84.81	86.85	84.49	85.86	75.09	72.9	79.5
84.63	83.38	86.36	87.25	75.09	72.7	81.95
83.93	90.69	84.7	88.94	75.41	72.7	85.93
83.85	81.96	83.52	88.66	74.78	72.7	82.13
83.85	89.65	87.7	85.96	76.44	72.8	74.3
83.79	82.42	80.57	87.64	75.93	72.7	85.38
83.75	87.09	84.09	86.24	73.99	72.8	72.25
83.75	81.52	82.87	85.92	76.19	72.5	88.2
83.69	83.47	84.67	81.87	76.69	72.9	86.68
82.86	82.57	82.84	86.93	73.79	72.8	83.5
82.69	84.92	85.85	82.97	76.34	72.5	73.3
82.64	83.79	86.63	81.93	76.82	72.1	78.05
82.61	78.35	82.64	83.99	76.16	72.8	86.63
82.49	83.98	86.66	82.35	75.89	72.7	75.13
82.4	87.47	89.07	90.79	76.63	72.7	86.63
82.35	79.88	80.87	84.78	78.08	72	86
82.12	84.89	84.59	86.34	75.29	72.8	86.5
90.55	90.89	90.9	86.97	73.8	72.5	80.13
89.89	90.98	90.99	87.7	76.63	72.9	81.05
89.73	86.78	89.79	85.88	76.61	72.7	78.7
88.84	90.23	88.74	83.3	74.76	72.8	80.93
85.72	86.69	86.66	86.99	74.28	72.8	87.75
85.25	90.67	90.57	87.73	74.76	72.9	80.8
84.97	90.89	87.97	90.9	74.45	72.8	86.88
84.81	86.85	84.49	85.86	75.09	72.9	79.5
84.63	83.38	86.36	87.25	75.09	72.7	81.95
83.93	90.69	84.7	88.94	75.41	72.7	85.93
83.85	81.96	83.52	88.66	74.78	72.7	82.13
83.85	89.65	87.7	85.96	76.44	72.8	74.3
83.79	82.42	80.57	87.64	75.93	72.7	85.38
83.75	87.09	84.09	86.24	73.99	72.8	72.25
83.75	81.52	82.87	85.92	76.19	72.5	88.2
83.69	83.47	84.67	81.87	76.69	72.9	86.68
82.86	82.57	82.84	86.93	73.79	72.8	83.5
82.69	84.92	85.85	82.97	76.34	72.5	73.3
82.64	83.79	86.63	81.93	76.82	72.1	78.05
82.61	78.35	82.64	83.99	76.16	72.8	86.63
82.49	83.98	86.66	82.35	75.89	72.7	75.13
82.4	87.47	89.07	90.79	76.63	72.7	86.63
82.35	79.88	80.87	84.78	78.08	72	86
82.12	84.89	84.59	86.34	75.29	72.8	86.5

Vektor *inputan* disebarkan melalui lapisan berbobot yang dilakukan secara random. inisialisasi bobot dan *threshold* dalam jaringan didistribusikan secara random dalam sebuah rentangan seperti persamaan 1, dimana *Fi* adalah jumlah *neuron input*.

$$\left(-\frac{2.4}{Fi}, +\frac{2.4}{Fi}\right)$$

Lapisan *hidden* beserta jumlah *neuron*-nya ditentukan secara *trial* dan *error*, dengan tujuan untuk mencapai nilai *error* yang paling optimal (*minimum error*), dengan jumlah *neuron* pada lapisan *hidden* akan sama dengan *neuron* pada *context unit*. Sedangkan untuk jumlah *neuron* pada lapisan *output* akan sama dengan *neuron* pada *context unit*.

Jumlah *neuron* pada *context* layer memiliki jumlah yang sama dengan *neuron* pada lapisan *hidden* dan *output* layer. Hal ini juga menyebabkan *context* layer disebut sebagai *copy context*, karena menduplikasi keluaran dari *hidden* layer dan *output* layer. *Neuron* pada lapisan *output* memiliki dua buah *neuron* yaitu Z1 dan Z2, serta memiliki dua *context unit*. Gambaran rancangan arsitektur FRNN, seperti pada Gambar 3. dilakukan dengan mengambil bobot terbaik hasil *training* sebelumnya. Hasil dari proses *testing* berupa akurasi sistem terhadap pengenalan terhadap data *testing*



Gambar 3. Rancangan arsitektur system RNN

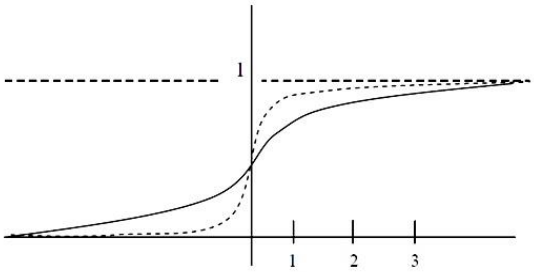
Pada diagram alir sistem, terdapat dua proses utama yaitu proses *training* dan proses *testing* yang masing-masing diberikan jumlah data yang berbeda-beda yaitu data *training* sebanyak 70 data dan data *testing* sebanyak 30 data. Sebelum melakukan pembelajaran terhadap data *training*, terlebih dahulu dilakukan seting nilai parameter

sebagai berikut *epoch, target error, learning rate*, jumlah *output*, jumlah *hidden layer*, *momentum*, yang selanjutnya dilakukan inisialisasi bobot jaringan, setelah melakukan inisialisasi bobot, proses selanjutnya adalah mengambil data *training*. Proses ini dilakukan secara umpan maju (*feedforward*) oleh jaringan dari sinyal masukan dipropagasi (dihitung maju) ke layer tersembunyi sampai layer keluaran dengan menngunakan fungsi aktivasi sigmoid biner. Fungsi ini digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode *back propagation*.

Fungsi sigmoid biner memiliki nilai pada *range* 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk JST yang membutuhkan nilai *output* yang terletak pada interval 0 sampai 1. Fungsi ini dirumuskan :

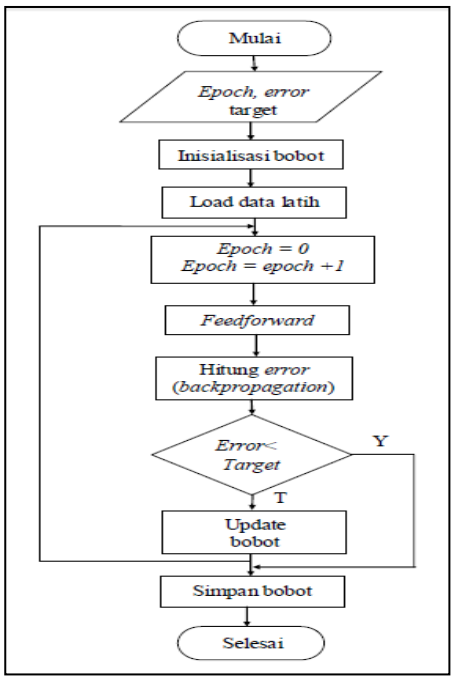
$$y = f(x) = \frac{1}{1+e^{-\sigma x}}$$

$$f'(x) = \sigma f(x)\,(1-f(x))$$



Gambar 4. Fungsi Aktivasi Sigmoid Biner ((σ = 1 dan σ = 3)

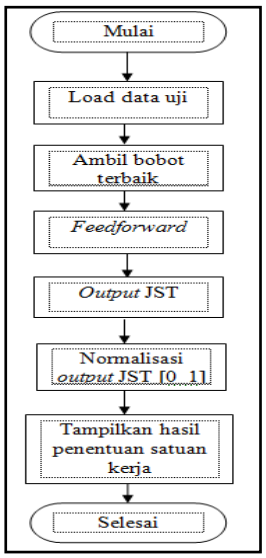
Fungsi ini digunakan untuk menghasilkan *output* JST. Proses selanjutnya dilakukan hitung *error* pada *neuron output* layer dan *neuron hidden* layer, dengan perhitungan mundur (*backward*), *error* yang diperoleh merupakan selisih antara *output* jaringan dengan data target yang terjadi pada *neuron output*, sehingga menghasilkan suatu bobot akhir. Proses peng-update-an bobot dilakukan dengan memodifikasi bobot untuk menurunkan kesalahan yang terjadi. Hal ini berlangsung pada perubahan bobot *hidden* layer, perubahan bobot *input* dan perubahan bobot *context unit*. Perubahan bobot ini berlangsung hingga selama *output* tidak sama dengan target. *Flowchart* untuk proses *training* ditunjukan pada Gambar 5.



Gambar 5. *Flowchart* untuk *training*

Pada proses *testing* bobot optimal hasil *training* diambil untuk dipergunakan pada proses *testing*, data *testing* diproses secara umpan maju (*feedforward*) oleh jaringan, sehingga menghasilkan *output* JST dan dinormalisasi dalam interval [0,1] sehingga

sama dengan rentang interval JST. *Flowchart* untuk proses *testing* dapat dilihat pada Gambar 6.



Gambar 6. *Flowchart* untuk *testing*

HASIL DAN PEMBAHASAN

Pengujian Sistem

Pada penelitian ini diperoleh hasil yaitu ini akan menjelaskan hasil pengujian pada penelitian dari sistem. Terdapat beberapa faktor yang dapat mempengaruhi kinerja sistem agar dapat menghasilkan keluaran yang baik, yaitu nilai *learning rate* = 0.0001, momentum=0.95, *epoch* maksimal, serta jumlah *hidden* layer=1 dan *neuron-neuron*-nya (10,12,13,14,15,16,17,18,19,20) Uji coba yang dilakukan dalam penelitian ini untuk mengetahui bagaimana pengaruh arsitektur jaringan serta setingan nilai parameter untuk mendapatkan hasil yang optimal. Terdapat beberapa percobaan yang dilakukan, dan hasil yang dijadikan sebagai acuan awal adalah dengan membandingkan target *error* dengan toleransi 0.1 dengan 0.01. Proses akan dihentikan apabila nilai fungsi biaya atau fungsi kinerja kurang dari atau sama dengan target *error*. Hasil *training* dengan toleransi 0.1 dapat dilihat seperti pada Tabel 1. setingan parameter yang digunakan pada pengujian pertama dapat dilihat seperti pada Tabel 2.

Tabel 2. Hasil *training* jumlah neuron dengan 1 hidden layer target 0.1

No	Neuron	Epoch	SSE	Durasi	Runtime (detik)
1	10	189277	0.009999939	21.05.801	2105801
2	12	170918	0.009999960	25.15.529	2515529
3	13	163122	0.009999963	27.35.425	2735425
4	14	165418	0.009999948	30.45.425	3045425
5	15	176950	0.009999968	36.07.302	3607302
6	16	117119	0.009999941	14.02.790	1402790
7	17	168253	0.009999906	40.12.525	4012525
8	18	186654	0.009999956	36.33.236	3633236
9	19	166231	0.009999952	45.07.079	4507079
10	20	182709	0.009999983	56.05.235	5605253

Tabel 3. Hasil *training* jumlah neuron dengan 1 hidden layer target 0.01

No	Neuron	Epoch	SSE	Durasi	Runtime (detik)
1	10	20137	0.09999531	00.59.066	59066
2	12	19713	0.09999914	01.24.034	124034
3	13	22622	0.09999487	02.20.097	220097
4	14	20058	0.09999932	03.38.015	338015
5	15	22483	0.09999866	04.43.493	443493
6	16	21681	0.09999451	04.00.083	400083
7	17	23614	0.09999370	04.52.089	452089
8	18	19949	0.09999971	04.26.077	426077
9	19	23953	0.09999610	04.37.595	437595
10	20	18272	0.09999978	04.11.721	411721

Pada pengujian yang dilakukan menunjukan bahwa dengan target 0.01 proses *training* berjalan lebih lama dibandingkan dengan target 0.1. Jumlah *neuron* pada *hidden* layer dengan nilai SSE terkecil diperoleh pada jumlah *neuron* 17, baik pada percobaan pertama atapun kedua. SSE mengambil nilai rata-rata kuadrat *error* yang terjadi antara output dengan target.

Pada pengujian dengan parameter *learning rate* (α) berpengaruh terhadap jumlah iterasi dalam pelatihan JST dapat diketahui dengan cara melatih JST dengan memvariasikan parameter *learning rate* (α). Beberapa nilai *learning rate* (α) yang sesuai digunakan antara 0.0001, 0.0002, 0.00015, dengan momentum 0.95 dan toleransi *error* 0.1, jumlah *neuron* 17, disajikan dalam Tabel 4 dan Tabel 5.

Tabel 4. Pengaruh antar LR dengan target 0.1

Learning rate (α)	Epoch	SSE	Durasi	Runtime (detik)
0.0001	23044	0.09999206	05.24.561	524561
0.0002	11822	0.09999527	01.57.028	11822
0.00015	20649	0.09999619	05.11.967	511967

Tabel 5. Pengaruh antara LR dengan target 0.01

Learning rate (α)	Epoch	SSE	Durasi	Runtime (detik)
0.0001	168253	0.009999906	40.12.525	4012525
0.0002	81866	0.009999932	20.57.933	2057933
0.00015	113651	0.009999937	28.22.333	2822333

Berdasarkan hasil pengujian dengan parameter *learning rate* (α) 0.0001 dan target 0.1 untuk jumlah *neuron* yang sama, maka jumlah *iterasi* yang terjadi semakin berkurang atau dengan kata lain kondisi konvergen semakin cepat terjadi. Namun apabila *learning rate* (α) dengan target 0.01, maka jumlah *iterasi* yang terajdi akan meningkat hal ini akan mengakibatkan kondisi konvergen semakin lama. Pengujian dengan parameter momentum, pemberian parameter momentum di dalam sistem JST berfungsi untuk mencegah sistem terjebak di dalam minimum lokal. Adapun nilai momentum yang coba diujikan adalah : 0.75, 0.85, dan 0.95 seperti pada Tabel 6 dan Tabel 7.

Tabel 6. Hasil pengujian momentum dengan target 0.1

LR	Momentum	Epoch	SSE	Durasi
0.0001	0.75	100040	0.09999951	24.15.455
	0.85	72987	0.09999875	17.47.025
	0.95	20406	0.09999459	04.56.663
0.0002	0.75	64832	0.09999932	16.07.646
	0.85	31391	0.09999987	07.48.118
	0.95	98460	0.09999918	20.22.150
0.00015	0.75	48174	0.09999972	06.46.289
	0.85	39173	0.09999942	08.30.355
	0.95	13963	0.09999561	03.16.798

Tabel 7. Hasil pengujian momentum dengan target 0.01

LR	Momentum	Epoch	SSE	Durasi
0.0001	0.75	741411	0.009999993	03:07:27:382
	0.85	499164	0.009999981	02:12:04:353
	0.95	168253	0.009999906	40:12:525
0.0002	0.75	511299	0.009999987	02:08:16:626
	0.85	250185	0.009999999	01:02:52:241
	0.95	85462	0.009999925	24:46:132
0.00015	0.75	583687	0.009999984	02:55:10:996
	0.85	342527	0.009999906	01:39:25:946
	0.95	106594	0.009999900	26:53:692

Berdasarkan hasil pengujian dengan momentum dengan target 0.1 dan target 0.01 menunjukan bahwa *epoch* dengan nilai SSE terkecil berada pada momentum dengan nilai 0.95 dengan nilai LR 0.0001. Pada pengujian momentum dengan target 0.01 memerlukan waktu yang lebih lama untuk mencapai konvergen dibandingkan dengan target 0.1. Berdasarkan hasil pengujian Tabel 5 dan

Tabel 6 dengan nilai momentum 0.95, dengan LR 0.0001 sistem dapat mempercepat proses konvergensi dengan nilai *error* yang sudah relatif kecil dari LR yang lain.

Pelatihan JST, peneliti menggunakan parameter sebagai berikut :

Bias = 1.

Learning rate (α) = 0.0001.

Momentum = 0.95.

Jumlah *hidden* layer = 1 lapis.

Jumlah *neuron* = 17.

Error = 0.1.

Proses *testing* dilakukan secara manual dengan memilah data sebanyak 30 data dari 100 data. Data yang digunakan untuk *testing* adalah data baru yang tidak diikutsertakan dalam *training*. Akurasi hasil *testing* sangat dipengaruhi oleh bobot hasil *training*, yang menunjukan kemampuan

Pada analisis sitem juga diketahui bahwa berdasarkan hasil percobaan yang dilakukan, maka nilai parameter yang digunakan untuk mendapatkan bobot optimal, *iterasi* dan SSE minimum untuk proses *testing* dalam jaringan dalam mengenali pola-pola yang sudah diberikan. Informasi hasil *testing* menggunakan JST FRNN dengan target jaringannya.

Hasil pengujian dengan JST model *fully recurrent neural network* menujukan bahwa sistem belum dapat mengenali semua pola data dengan baik terlihat pada 30 data *real* dengan tidak mengunakan target, hasil pengenalan pola data sebanyak 25 data, dan 5 data tidak dikenali, dengan tingkat akurasi mencapai 83,33%.

Tabel.8, menunjukan hasil uji data, sebanyak 30 data testing.

Nosis	Pop	Ku	Kut	Kks	Latnis	Mental	Samjas	*Output jst*	Hasil
001	90.55	90.89	90.9	86.97	73.8	72.5	80.13	0	Cocok
002	89.89	90.98	90.99	87.7	76.63	72.9	81.05	0	Cocok
003	89.73	86.78	89.79	85.88	76.61	72.7	78.7	0	Cocok
004	88.84	90.23	88.74	83.3	74.76	72.8	80.93	0	Cocok
005	85.72	86.69	86.66	86.99	74.28	72.8	87.75	0	Cocok
006	85.25	90.67	90.57	87.73	74.76	72.9	80.8	0	Cocok
007	84.97	90.89	87.97	90.9	74.45	72.8	86.88	0	Cocok
008	84.81	86.85	84.49	85.86	75.09	72.9	79.5	0	Cocok
009	84.63	83.38	86.36	87.25	75.09	72.7	81.95	1	Tidak cocok
010	83.93	90.69	84.7	88.94	75.41	72.7	85.93	0	Cocok
011	83.85	81.96	83.52	88.66	74.78	72.7	82.13	1	Tidak cocok
012	83.85	89.65	87.7	85.96	76.44	72.8	74.3	0	Cocok
013	83.79	82.42	80.57	87.64	75.93	72.7	85.38	0	Cocok
014	83.75	87.09	84.09	86.24	73.99	72.8	72.25	0	Cocok
015	83.75	81.52	82.87	85.92	76.19	72.5	88.2	0	Cocok
016	83.69	83.47	84.67	81.87	76.69	72.9	86.68	0	Cocok
017	82.86	82.57	82.84	86.93	73.79	72.8	83.5	1	Tidak cocok
018	82.69	84.92	85.85	82.97	76.34	72.5	73.3	0	Cocok
019	82.64	83.79	86.63	81.93	76.82	72.1	78.05	0	Cocok
020	82.61	78.35	82.64	83.99	76.16	72.8	86.63	0	Cocok
021	82.49	83.98	86.66	82.35	75.89	72.7	75.13	0	Cocok
022	82.4	87.47	89.07	90.79	76.63	72.7	86.63	0	Cocok
023	82.35	79.88	80.87	84.78	78.08	72	86	0	Cocok
024	82.12	84.89	84.59	86.34	75.29	72.8	86.5	0	Cocok
025	81.19	88.24	90.14	85.02	74.08	72.7	69.13	0	Cocok
026	81.03	79.46	80	82.85	77.44	72.7	87.37	0	Cocok
027	80.04	76.17	78.09	81.57	74.39	72.6	80.75	1	Tidak cocok
028	79.93	85.75	86.67	87.99	74.99	72.8	87.25	0	Cocok
029	79.84	78.79	78.98	81.84	74.08	72.9	82.75	1	Tidak cocok
030	79.81	88.99	86.95	83.99	76.59	72.8	78.75	0	Cocok

SIMPULAN

Berdasarkan hasil percobaan hasil pengujian pengaruh parameter-parameter JST terhadap *iterasi* dan SSE, maka untuk mendapatkan bobot yang paling optimal untuk proses *testing* dalam pelatihan JST dengan menggunakan 1 *hidden* layer dengan jumlah *neuron* 17 unit, LR sebesar 0.0001, momentum 0.95, dengan target *error* 0.1.Penerapan JST FRNN untuk pengenalan

data *real* sebanyak 30 data baru, data yang tidak dapat dikenali pada data ke- 9, 11, 17, 27 dan 29, sehingga akurasi yang dicapai sebesar 83,33%.

Perlu dilakukan pengujian dengan data yang lebih banyak lagi agar jaringan lebih banyak belajar dalam mengenal pola-pola baru. Perlu dikembangkan penelitian dengan masalah yang sama dengan menggunakan algoritma pembelajaran lain misalnya *Elman Recurrent Nueural Network, Levenberg Marquart Algorithm* (LMA), atau dengan pembelajaran *unsuvervised learning.*

DAFTAR PUSTAKA

Norvig, Russel., 2003, *Artificial Intelligence a modern approach*, USA, Prentice Hall

Samarasinghe, S., 2006, *Neural Networks for Applied Sciences and Engineering* : from fundamentals to complex pattern recognition, ISBN-13:978-0-8493-3375-0