

BAB II KAJIAN PUSTAKA

A. JARINGAN SYARAF TIRUAN

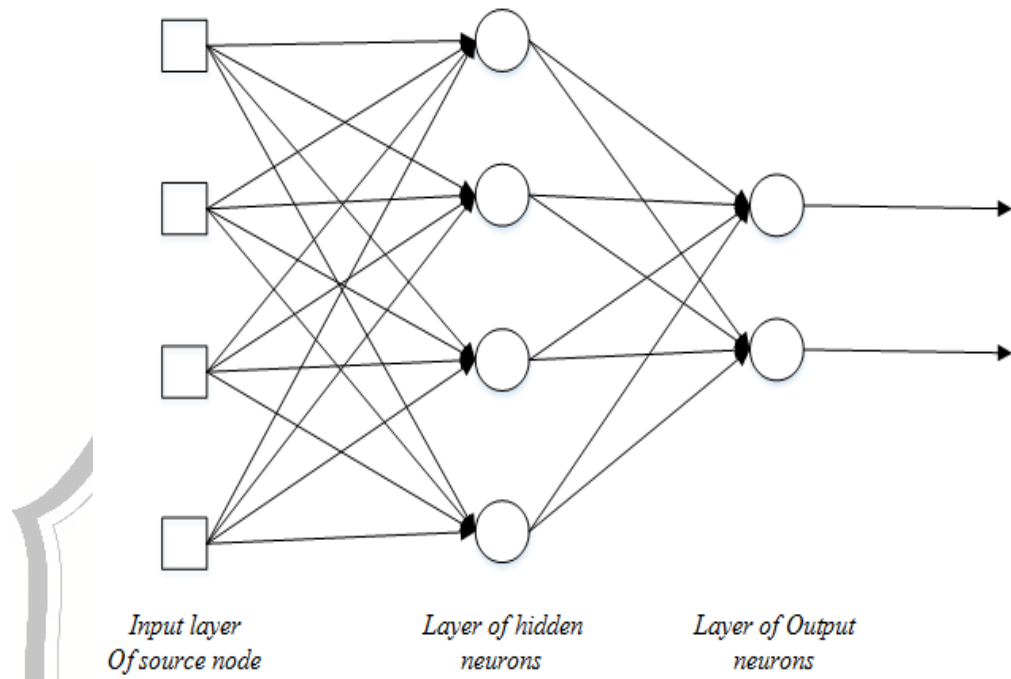
1. Definisi Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST), yaitu metode *learning* yang bisa digunakan untuk permasalahan yang bernilai diskrit, real, maupun vektor. JST adalah proses tersebar paralel (*parallel distributed processor*) yang sangat besar yang memiliki kecenderungan untuk menyimpan pengetahuan yang bersifat pengalaman dan membuatnya siap untuk digunakan. JST menyerupai otak manusia dalam dua hal, yaitu : Pengetahuan diperoleh jaringan melalui proses belajar. Kekuatan hubungan antar sel syaraf (neuron) yang dikenal sebagai bobot-bobot sinaptik digunakan untuk menyimpan pengetahuan (Suyanto, 2014).

2. Arsitektur Jaringan Syaraf Tiruan

Pola dimana neuron-neuron pada JST disusun berhubungan erat dengan algoritma belajar yang digunakan untuk melatih jaringan. Berikut arsitektur jaringan syaraf tiruan (*multi-layer*). Kelas ke dua dari *feedforward neural network* adalah jaringan dengan satu atau lebih lapis tersembunyi (*hidden layer*), dengan *computation nodes* yang berhubungan disebut *hidden neuron* atau *hidden units*. Gambar 1 mengilustrasikan *multilayer feedforward neural network* untuk kasus satu *hidden layer*. Untuk meningkatkan jaringan pada Gambar 1 tersebut, biasanya disebut sebagai

jaringan 4-4-2, dalam arti bahwa jaringan tersebut mempunyai 4 *node* masukan, 4 *hidden neurons* dan 2 *output neurons* (Suyanto, 2014).



Gambar 1. *Fully connected feedforward network* dengan satu *hidden layer* dan satu *output layer* (jaringan 4-4-2)

B. ALGORITMA PEMBELAJARAN *BACKPROPAGATION*

Menurut Kusumadewi (2004) *backpropagation* merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot–bobot yang terhubung dengan neuron–neuron yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobot– bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan

maju, neuron–neuron diaktifkan dengan menggunakan fungsi aktivasi yang dapat dideferensikan seperti *sigmoid* pada persamaan 1 :

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

dengan : $f'(x) = \sigma f(x)[1 - f(x)]$

atau *tansig* seperti pada persamaan 2 :

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

atau $y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \dots\dots\dots(2)$

dengan $f'(x) = [1 + f(x)][1 - f(x)]$

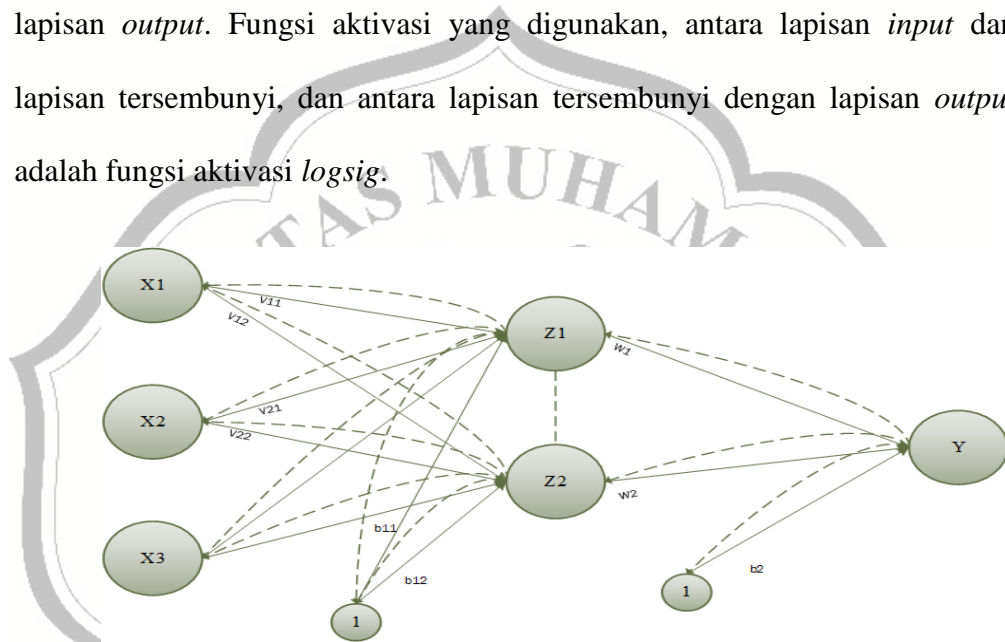
atau *purelin* seperti pada persamaan 3 :

$$y = f(x) = x$$

$f'(x) = 1 \dots\dots\dots(3)$

Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar 2, jaringan terdiri atas 3 unit (*neuron*) pada lapisan *input*, yaitu x_1 , x_2 , dan x_3 ; 1 lapisan *output*, yaitu y . Bobot yang meghubungkan x_1 , x_2 , dan x_3 dengan neuron pertama pada lapisan tersembunyi, adalah v_{11} , v_{21} , v_{31} (v_{ij} : bobot yang menghubungkan neuron *input* ke-i ke neuron ke-j pada lapisan tersembunyi). Perlu diingat bahwa, untuk pemakaian *toolbox* nnet pada Matlab, bobot v_{ij} memiliki pengertian yang sebaliknya (v_{ij} : bobot yang menghubungkan neuron ke-j pada suatu lapisan neuron ke-i pada lapisan sesudahnya). Misal: v_{12} adalah bobot yang menghubungkan neuron ke-2 pada lapisan *input*, ke neuron ke-1

pada lapisan tersembunyi. Kembali ke Gambar 2, b_{1_1} dan b_{1_2} adalah bobot bias yang menuju ke neuron pertama dan kedua pada lapisan tersembunyi. Bobot yang menghubungkan z_1 dan z_2 dengan neuron pada lapisan *output*, adalah w_1 dan w_2 . Bobot bias b_2 menghubungkan lapisan tersembunyi dengan lapisan *output*. Fungsi aktivasi yang digunakan, antara lapisan *input* dan lapisan tersembunyi, dan antara lapisan tersembunyi dengan lapisan *output* adalah fungsi aktivasi *logsig*.



Gambar 2. Arsitektur Jaringan *Backpropagation*

Berikut adalah algoritma *backpropagation* :

- Inisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil).
- Tetapkan: Maksimum Epoch, Target *Error*, dan *Learning Rate* (α).
- Inisialisasi: Epoch = 0, MSE = 1.
- Berikut langkah – langkah selama (Epoch < Maksimum Epoch) dan (MSE > Target *Error*):
 - a. Epoch = Epoch + 1
 - b. Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, dikerjakan *layer feedforward* :

Feedforward seperti pada persamaan 4-7 :

1. Tiap–tiap unit *input* (X_i , $i=1,2,3,\dots,n$) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).

2. Tiap–tiap unit tersembunyi (Z_j , $j=1,2,3,\dots,p$) menjumlahkan sinyal – sinyal *input* terbobot :

$$z_{in_j} = v_{0j} + \sum_{i=1}^n x_i v_{ij} \dots\dots\dots (4)$$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*nya :

$$z_j = f(z_{in_j}) \dots\dots\dots (5)$$

Dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit–unit *output*).

3. Tiap–tiap unit *output* (Y_k , $k=1,2,3,\dots,m$) menjumlahkan sinyal – sinyal *input* terbobot.

$$y_{in_k} = w_{0k} + \sum_{i=1}^p z_i w_{jk} \dots\dots\dots (6)$$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*nya :

$$y_k = f(y_{in_k}) \dots\dots\dots (7)$$

Dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit – unit *output*).

Backpropagation seperti pada persamaan 8-22 :

4. Tiap – tiap unit output ($Y_k, k=1,2,3,\dots,m$) menerima target pola yang berhubungan dengan pola *input* pembelajaran, menghitung informasi *error*nya :

$$\delta z_k = (t_k - t_y) f'(y_{in_k}) \dots\dots\dots(8)$$

$$\varphi z_{jk} = \delta_k z_j \dots\dots\dots(9)$$

$$\beta z_k = \delta_k \dots\dots\dots(10)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}):

$$\Delta w_{jk} = \alpha \varphi z_{jk} \dots\dots\dots(11)$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai awal $b z_k$):

$$\Delta b z_k = \alpha \beta z_k \dots\dots\dots(12)$$

Langkah (d) ini juga dilakukan sebanyak jumlah lapisan tersembunyi, yaitu menghitung informasi *error* dari suatu lapisan tersembunyi ke lapisan tersembunyi sebelumnya.

5. Tiap – tiap unit tersembunyi ($Z_j, j=1,2,3,\dots,p$) menjumlahkan delta *input*nya (dari unit – unit yang berada pada lapisan di atasnya):

$$\delta_{-in_j} = \sum_{k=1}^m \delta z_k w_{jk} \dots\dots\dots(13)$$

Kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi *error*:

$$\delta 1_j = \delta_{in} f'(z_{in_j}) \dots\dots\dots(14)$$

$$\varphi 1_{ij} = \delta 1_j x_j \dots\dots\dots(15)$$

$$\beta 1_j = \delta 1_j \dots\dots\dots(16)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}):

$$\Delta v_{ij} = \alpha \varphi 1_{ij} \dots\dots\dots(17)$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai b_{1j}):

$$\Delta b_{1j} = \alpha \beta 1_j \dots\dots\dots(18)$$

6. Tiap – tiap unit *output* ($Y_k, k=1,2,3,\dots,p$) memperbaiki bias dan bobotnya ($j=0,1,2,\dots,p$):

$$w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_{jk} \dots\dots\dots(19)$$

$$b_{2k}(\text{baru}) = b_{2k}(\text{lama}) + \Delta b_{2k} \dots\dots\dots(20)$$

Tiap – tiap unit tersembunyi ($Z_j, j=1,2,3,\dots,p$) memperbaiki bias dan bobotnya ($i=0,1,2,3,\dots,n$):

$$v_{ij}(\text{baru}) = v_{ij}(\text{lama}) + \Delta v_{ij} \dots\dots\dots(21)$$

$$b_{1j}(\text{baru}) = b_{1j}(\text{lama}) + \Delta b_{1j} \dots\dots\dots(22)$$

C. ALGORITMA PELATIHAN

Dalam JST terdapat algoritma pelatihan yang bisa digunakan, diantaranya adalah *Fletcher-Reeves Update* (*traincgf*), *Polak-Ribiere* (*traincgp*), *Powell-Beale Restarts* (*traincgb*), *Scaled Conjugate Gradient* (*traincsg*), *Gradien*

Descent (traingd), *Gradient Descent* dengan *Adaptive Learning Rate (traingda)*, *Gradient Descent* dengan *momentum (traingdm)*, *Gradient Descent* dengan *Momentum* dan *Adaptive Learning Rate (traingdx)*, *Resilent Backpropagation (trainrp)*, *BFGS (trainbfg)*, *One Step Secant (trainoss)*, *Levenberg-Marquardt (trainlm)* yang tergolong dalam metode *Backpropagation* (Kusumadewi, 2004).

D. HIDDEN LAYER

Menurut Heaton (2008) menentukan jumlah neuron di lapisan tersembunyi adalah bagian yang penting dalam menentukan asitektur *neural*, walaupun dalam lapisan ini tidak secara langsung beinteraksi dengan lingkungan eksternal akan tetapi, lapisan ini memiliki pengaruh besar pada hasil akhir. Pada penelitian ini menentukan jumlah neuron yang digunakan dalam lapisan tersembunyi berdasarkan :

1. Jumlah neuron tersembunyi sebaiknya berada di antara neuron lapisan *input* dan neuron lapisan *output*.
2. Jumlah neuron yang tersembunyi sebaiknya $2/3$ ukuran lapisan *input* ditambah neuron lapisan keluaran.
3. Jumlah neuron yang tersembunyi sebaiknya kurang dari dua kali neuron dari lapisan *input*.

E. PARAMETER JARINGAN

Beberapa parameter algoritma seperti yang dijelaskan oleh Kusumadewi (2004) adalah :

a. Maksimum epoh

Maksimum epoh adalah jumlah epoh maksimum yang boleh dilakukan selama proses pelatihan. Iterasi akan dentikan apabila nilai epoh melebihi maksimum epoh.

Intruksi: `net.trainParam.epochs = MaxEpoh`

Nilai *default* untuk maksimum epoh adalah 10.

b. Kinerja tujuan

Kinerja tujuan adalah target nilai fungsi kinerja. Iterasi akan dihentikan apabila nilai fungsi kinerja kurang dari atau sama dengan kinerja tujuan.

Intruksi: `net.trainParam.goal = TargetError`

Nilai *default* untuk kinerja tujuan adalah 0.

c. Learning rate

Learning rate adalah laju pembelajaran.

Intruksi: `net.trainParam.lr = LearningRate`

Nilai *default* untuk *learning rate* adalah 0.01

d. Rasio untuk menaikkan *learning Rate*

Rasio ini berguna sebagai faktor penggali untuk menaikkan *learning rate* apabila *learning rate* yang ada terlalu rendah untuk mencapai kekonvergenan.

Intruksi: `net.trainParam.lr_inc = IncLearningRate`

Nilai *default* untuk rasio kenaikan *learning rate* adalah 1.05.

e. Rasio untuk menurunkan *learning Rate*

Rasio ini berguna sebagai faktor pengali untuk menurunkan *learning rate* apabila *learning rate* yang ada terlalu tinggi dan menuju ke ketidakstabilan

Intruksi: `net.trainParam.lr_decc = DecLearningRate`

Nilai *default* untuk rasio penurunan *learning rate* adalah 0.7.

f. Maksimum kegagalan

Maksimum kegagalan diperlukan apabila pada algoritma disertai dengan validitas (optional). Maksimum kegagalan adalah ketidakvalitan terbesar yang diperbolehkan. Apabila gradien pada iterasi ke-*k* lebih besar daripada gradien iterasi ke-*(k-1)*, maka kegagalannya akan bertambah 1. Iterasi akan dihentikan apabila jumlah kegagalan lebih dari maksimum kegagalan.

Instruksi: `net.trainParam.max_fail = MaxFail`

Nilai *default* untuk maksimum kegagalan adalah 5.

g. Maksimum kenaikan kinerja

Maksimum kenaikan kinerja adalah nilai maksimum kenaikan *error* yang diijinkan, antara *error* saat ini dan *error* sebelumnya.

Intruksi: `net.trainParam.max_perf_inc = MaxPerflnc`

Nilai *default* untuk maksimum kenaikan kinerja adalah 1.04.

h. Gradien minimum

Gradien minimum adalah akar dari jumlah kuadrat semua gradien (bobot *input*, bobot lapisan, bobot bias) terkecil yang diperbolehkan. Iterasi akan

dihentikan apabila nilai akar jumlah kuadrat semua gradien ini kurang dari gradien minimum.

Intruksi: `net.trainParam.min_grad = MinGradien`

Nilai *default* untuk gradien minimum adalah 10^{-10} .

- i. Jumlah epoch yang akan ditunjukkan kemajuannya

Menunjukkan berapa jumlah epoch berselang yang akan ditunjukkan kemajuannya.

Intruksi: `net.trainParam.show = EpochShow`

Nilai *default* untuk jumlah epoch yang akan ditunjukkan adalah 25.

- j. Waktu maksimum untuk pelatihan

Menunjukkan waktu maksimum yang diijinkan untuk melakukan pelatihan.

Iterasi akan dihentikan apabila waktu pelatihan melebihi waktu maksimum.

Intruksi : `net.trainParam.time = MaxTime`

Nilai *default* untuk waktu maksimum adalah tak terbatas (inf).

F. MATLAB

Menurut Tjolleng (2017) MATLAB merupakan perangkat lunak yang digunakan untuk pemrograman, analisis, serta komputasi teknis dan matematis berbasis matriks. MATLAB adalah singkatan dari *Matrix Laboratory* karena mampu menyelesaikan masalah perhitungan dalam bentuk matriks. MATLAB versi pertama dirilis pada tahun 1970 oleh Cleve Moler. Pada awalnya, MATLAB didesain untuk menyelesaikan masalah-masalah persamaan aljabar

linear. Seiring berjalannya waktu, program ini terus mengalami perkembangan dari segi fungsi dan performa komputasi.

Bahasa pemrograman yang kini dikembangkan oleh MathWorks Inc. Menggabungkan proses pemrograman, komputasi, dan visualisasi melalui lingkungan kerja yang mudah digunakan. MATLAB juga memiliki keunggulan umum lainnya, seperti analisis dan eksplorasi data, pengembangan algoritma, permodelan dan simulasi, visualisasi plot dalam bentuk 2D dan 3D, hingga pengembangan aplikasi antar muka grafis. Dalam ruang lingkup perguruan tinggi, MATLAB digunakan sebagai alat pembelajaran pemrograman matematika, teknik, dan sains pada level pengenalan dan lanjutan, sedangkan dalam dunia industri, MATLAB dipilih sebagai alat penelitian, pengembangan dan analisis produk industri.

MATLAB dapat dioperasikan pada sistem operasi Windows, Linux, maupun macOS. Selain itu, MATLAB juga bisa dihubungkan dengan aplikasi atau bahasa pemrograman eksternal lainnya, seperti C, Java, .NET dan Microsoft Excel. Dalam MATLAB tersedia pula kotak kakas (*toolbox*) yang dapat digunakan untuk aplikasi-aplikasi khusus, seperti pengolahan sinyal, sistem kontrol, logika *fuzzy*, jaringan saraf tiruan, optimasi, pengolahan citra digital, bioinformatika, simulasi dan berbagai teknologi lainnya.

G. SPSS

SPSS singkatan dari *Statistical Package For Social Science* yaitu merupakan paket statistika untuk ilmu – ilmu sosial, akan tetapi SPSS banyak

juga digunakan untuk bidang – bidang lain yang memang membutuhkan statistika. Sejak dikeluarkannya SPSS dengan versi under DOS sampai sekarang dengan versi under Windows, sudah dikembangkan SPSS sampai generasi atau release 17 yang paling baru dengan penambahan fasilitas yang makin lengkap seperti grafik pengendali untuk *Quality control* dan penambahan fasilitas untuk link S-Plus yaitu Package Statistika terbaru yang sangat cocok untuk tujuan ilmiah dan pengembangannya, tidak hanya pengolahan data semata. SPSS berfungsi untuk membantu memproses data–data statistik secara cepat dan tepat, serta menghasilkan berbagai *output* yang dikehendaki oleh para pengambil keputusan. Statistik dapat diartikan sebagai suatu kegiatan yang bertujuan untuk mengumpulkan data, meringkas atau menyajikan data kemudian menganalisis data dengan menggunakan metode tertentu, dan menginterpretasikan hasil dari analisa tersebut (Taniredja & Mustafidah, 2011).

H. ANAVA

Anava atau disebut juga Anova (*Analysis of Variance*) merupakan perluasan dari uji rata – rata k sampel, dengan $k > 2$.

➤ $H_0: \mu_1 = \mu_2 = \dots = \mu_k$ vs H_1 : minimal 2 mean tidak sama.

➤ $F_{hit} = \frac{MST}{MSE} \sim F_{k-1, N-k}$

Dimana $MST = \text{Mean Square of Treatment}$

$MSE = \text{Mean Square of Error}$

Pada ANAVA jika H_0 ditolak maka masih harus ditentukan mean–mean mana saja yang berbeda. Untuk itu perlu dilakukan MCA (*Multiple Comparison Analysis*) atau Analisis Perbandingan Ganda (Taniredja & Mustafidah, 2011).

I. PENELITIAN TERKAIT

Beberapa algoritma pelatihan telah diuji pada penelitian Mustafidah & Suwarsito (2015a) melakukan pengujian terhadap 12 algoritma pelatihan yang terdapat dalam jaringan *backpropagation*. Pengujian menggunakan parameter jaringan seperti target *error* = 0,001 (10^{-3}), maksimum epoh = 10000, *learning rate* (*lr*) = 0,01, dengan 5 neuron *input* dan satu neuron *output*. Berdasarkan hasil tes dari 12 algoritma pelatihan, algoritma *trainlm* memiliki *error* terkecil dengan level $\alpha = 5\%$ dan memberikan *error* 0,0001986858. Dengan demikian algoritma pelatihan *trainlm* adalah algoritma pelatihan dalam *backpropagation* yang dapat diaplikasikan untuk menyelesaikan permasalahan.

Menurut penelitian Mustafidah & Suwarsito (2015b) telah dilakukan pengujian terhadap 12 algoritma pelatihan. Simpulan dari penelitian menyatakan bahwa algoritma pelatihan *trainlm* merupakan algoritma pelatihan yang paling optimal pada tingkat laju pembelajaran (*learning rate*/*lr*) = 0,05. Hal ini dibuktikan berdasarkan uji ANOVA dengan tingkat $\alpha = 5\%$, hasil uji menunjukkan tingkat signifikansi sebesar 0,00. Rata-rata *error* yang dihasilkan oleh algoritma *trainlm* ini adalah sebesar 0,0002196 dari target *error* 0,001.

Pada penelitian yang dilakukan Mustafidah, et al. (2014) menyatakan dalam penelitiannya bahwa 12 algoritma pelatihan dalam jaringan syaraf telah

diuji, yaitu *traingd*, *traingdm*, *traingdx*, *trainrp*, *traingda*, *traingcf*, *traingcp*, *traingcb*, *traingscg*, *trainbfg*, *trainoss*, dan *trainlm* dalam kesesuaian mengenali pola data. Parameter jaringan yang digunakan yaitu maksimum epoch = 1000, *learning rate* = 0,05, dan *target error* = 0,001. Hasil pengujian diperoleh bahwa pada interval kepercayaan 95% algoritma *trainlm* adalah algoritma yang paling tepat dalam mengenali pola data dengan tingkat kelayakan rata-rata 87,5%.

