|  |  |
| --- | --- |
| |  | | --- | | Kode/Nama Rumpun Ilmu\*\* : 458 / Teknik Informatika  Bidang Fokus : Teknologi Informasi dan Komunikasi |       **LAPORAN**  **PENELITIAN DOSEN UPI “YPTK” PADANG**  logo_upi          **PROPAGASI BALIK MENENTUKAN PREDIKSI PRODUKSI USAHA SONGKET SILUNGKANG KOTA SAWAHLUNTO**  **TIM PENGUSUL**  **RIMA LIANA GEMA, S.KOM, M.KOM / 1013098901 (Ketua )**  **DEVIA KARTIKA, S.KOM, M.KOM / 1021129101 (Anggota 1)**      **UNIVERSITAS PUTRA INDONESIA “YPTK” PADANG**  **NOVEMBER 2018** |



**RINGKASAN**

Jaringan Syaraf Tiruan adalah paradigma komputasi yang mana cara kerjanya meniru sistim sel syaraf biologi berdasarkan karakteristik fungsi otak manusia. Salah satu metode yang digunakan dalam Jaringan Syaraf Tiruan adalah algoritma *backpropagation* yang banyak digunakan terutama dalam menangani masalah identifikasi, prediksi, pengenalan pola-pola kompleks karena metode ini mampu meramalkan apa yang akan terjadi di masa yang akan datang berdasarkan pola yang ada pada masa lalu. Songket merupakan salah satu hasil karya tangan-tangan terampil pengrajin asli Silungkang, Kota Sawahlunto, Sumatera Barat yang memiliki corak dan motif yang bervariasi dan unik. Pemerintah Kota Sawahlunto, Sumatra Barat memprioritaskan pengembangan usaha kerajinan songket Silungkang, yang menjadi kekhasan daerah, guna masuk pasar ekspor. Pada tahap awal, prioritas pemda adalah meningkatkan produksi perajin dengan memfasilitasi pembinaan kepada pelaku usaha mikro kecil dan menengah (UMKM) terutama yang bergerak di bidang kerajinan songket, untuk terus dikembangkan dengan meningkatkan kualitas dan kreativitas. Dengan menerapkan metode algoritma *backpropagation* dalam memprediksi produksi Songket Silungkang Kota Sawahlunto dapat membantu beberapa pihak seperti pemerintah, pelaku usaha mikro kecil dan menengah dalam melakukan upaya penanganan dan pengambilan keputusan yang baik terhadap peningkatan produksi UMKM Songket Silungkang Kota Sawahlunto.

**Kata Kunci : *Produksi, Songket, Propagasi balik***

**DAFTAR ISI**

**Halaman**

**HALAMAN PENGESAHAN i**

**RINGKASAN UMUM ii**

**DAFTAR ISI iii**

**BAB I. PENDAHULUAN 1**

1.1 Latar Belakang 1

1.2 Rumusan Masalah 2

1.3 Hipotesis 2

**BAB II. TINJAUAN PUSTAKA 3**

2.1 Jaringan Syaraf Tiruan 3

2.1.1 Definisi Jaringan Syaraf Tiruan 3

2.1.2 Arsitektur Jaringan 3

2.1.3 Fungsi Aktivasi 5

2.2 Metode Backpropagation 6

2.2.1 Definisi Metode Backpropagation 6

2.2.2 Arsitektur Metode Backpropagation 7

2.2.3 Algoritma Backpropagation 7

2.2.3 Algoritma Pelatihan Backpropagation 8

2.2.4 Fungsi Aktivasi Pada Pelatihan Backpropagation 11

**BAB III. TUJUAN DAN MANFAAT PENELITIAN 13**

3.1 Tujuan Penelitian 13

3.2 Manfaat Penelitian 13

**BAB IV. METODE PENELITIAN 14**

4.1 Pendahuluan 14

4.2 Kerangka Kerja Penelitian 14

**BAB V. HASIL DAN LUARAN YANG DICAPAI 18**

5.1 Analisa Data 18

5.2 Transformasi Data Real Menjadi Data Pelatihan 18

5.3 Perancangan Arsitektur Jaringan Syaraf Tiruan 22

5.4 Postprocessing / Denormalisasi 23

5.5 Pelatihan Jarinagn Syaraf Tiruan 24

5.6 Analisa Hasil Pelatihan Jaringan Syaraf Tiruan 69

**BAB VI. IMPLEMENTASI DAN PENGUJIAN 70**

6.1 Implementasi dan Pengujian Sistem 20

6.2 Hasil Pengenalan Data Training 70

6.2.1 Model Arsitektur 12-3-1 70

6.2.2 Model Arsitektur 12-5-1 76

6.2.3 Model Arsitektur 12-7-1 80

6.2.4 Model Arsitektur 12-8-1 85

6.2.5 Model Arsitektur 12-9-1 89

6.3 Perbandingan Model Data Training 93

6.4 Pengujian Data Dengan Backpropagation Model 12-9-1 93

6.5 Hasil Pengujian Akurasi Prediksi 96

**BAB VII. KESIMPULAN DAN SARAN 97**

7.1 Kesimpulan 97

7.2 Saran 98

**DAFTAR PUSTAKA 99**

**DAFTAR TABEL**

Tabel 5.1 Produksi Songket Silungkang 18

Tabel 5.2 Hasil Transformasi Data Produksi Songket 19

Tabel 5.3 Hasil Penyusunan Pola Data 20

Tabel 5.4 Pola Data Pelatihan 25

Tabel 5.5 Pola Data Pengujian 26

Tabel 5.6 Nilai Bobot Dari Input ke Hidden Layer 59

Tabel 5.7 Nilai Bobot Dari Hidden ke Output Layer 60

Tabel 5.8 Suku Perubahan Bobot Unit Tersembunyi 64

Tabel 5.9 Bobot Baru Dari Input Layer ke Hidden Layer 68

Tabel 6.1 Hasil dan Error Data Input Pelatihan dengan Pola 12-3-1 75

Tabel 6.2 Hasil dan Error Data Input Pelatihan dengan Pola 12-5-1 79

Tabel 6.3 Hasil dan Error Data Input Pelatihan dengan Pola 12-7-1 84

Tabel 6.4 Hasil dan Error Data Input Pelatihan dengan Pola 12-8-1 88

Tabel 6.5 Hasil dan Error Data Input Pelatihan dengan Pola 12-9-1 92

Tabel 6.6 Perbandingan Epoch dan MSE Dari Lima Pola Uji 93

Tabel 6.7 Hasil Pengujian Dengan arsitektur Jaringan Model 12-9-1 95

**DAFTAR GAMBAR**

Gambar 2.1 Model Matematis Jaringan Syaraf Tiruan 3

Gambar 2.2 Jaringan Syaraf Tiruan Dengan Lapisan Tunggal 4

Gambar 2.3 Jaringan Syaraf Tiruan Dengan Lapisan Banyak 5

Gambar 2.4 Jaringan Syaraf Tiruan Dengan Lapisan Kompetitif 5

Gambar 2.5 JST *Backpropagation* Dengan Satu Lapisan Tersembunyi 6

Gambar 2.6 Arsitektur Backpropagation 7

Gambar 2.7 Alur Kerja Jaringan Backpropagation 8

Gambar 2.8 Fungsi Sigmoid Biner 12

Gambar 2.9 Fungsi Sigmoid Bipolar 12

Gambar 3.1 Kerangka Kerja Penelitian 15

Gambar 5.1 Arsitektur Jaringan Syaraf Tiruan 22

Gambar 6.1 Antarmuka Grafik Pelatihan Model JST 12-3-1 74

Gambar 6.2 Antarmuka Grafik Pelatihan Model JST 12-5-1 78

Gambar 6.3 Antarmuka Grafik Pelatihan Model JST 12-7-1 83

Gambar 6.4 Antarmuka Grafik Pelatihan Model JST 12-8-1 87

Gambar 6.5 Antarmuka Grafik Pelatihan Model JST 12-9-1 91

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Jaringan Syaraf Tiruan adalah paradigma komputasi yang mana cara kerjanya meniru sistim sel syaraf biologi berdasarkan karakteristik fungsi otak manusia. Salah satu metode yang digunakan dalam Jaringan Syaraf Tiruan adalah algoritma *backpropagation* yang banyak digunakan terutama dalam menangani masalah identifikasi, prediksi, pengenalan pola-pola kompleks karena metode ini mampu meramalkan apa yang akan terjadi di masa yang akan datang berdasarkan pola yang ada pada masa lalu.

Songket salah satu hasil karya tangan-tangan terampil pengrajin asli Silungkang, Kota Sawahlunto, Sumatera Barat yang memiliki corak dan motif yang bervariasi dan unik. Mengambil unsur-unsur dari alam yang dikemas dalam motif berbentuk bungo lobak, anggur, pucuak rabuang, bintang, kapalo samek, dan masih banyak unsur lainnya. Kombinasi warna yang cantik dan elegan di setiap helai kain songket menambah keanggunan bagi setiap pemiliknya. Songket sangat cocok untuk semua perempuan Indonesia yang ingin selalu tampil cantik dan menawan di setiap kesempatan, formal dan semi formal.

Pemerintah Kota Sawahlunto, Sumatra Barat memprioritaskan pengembangan usaha kerajinan songket Silungkang, yang menjadi kekhasan daerah, guna masuk pasar ekspor. Potensi ekspor kerajinan songket Silungkang sangat besar, mengingat kerajinan tersebut cukup diminati di mancanegara, terutama Malaysia. Pada tahap awal, prioritas pemda adalah meningkatkan produksi perajin dengan memfasilitasi pembinaan kepada pelaku usaha mikro kecil dan menengah (UMKM) terutama yang bergerak di bidang kerajinan songket, untuk terus dikembangkan dengan meningkatkan kualitas dan kreativitas.

Songket Silungkang memiliki keunikan tersendiri.

Dengan menerapkan metode algoritma *backpropagation* dalam memprediksi produksi Songket Silungkang Kota Sawahlunto dapat membantu beberapa pihak seperti pemerintah, pelaku usaha mikro kecil dan menengah dalam melakukan upaya penanganan dan pengambilan keputusan yang baik terhadap peningkatan produksi UMKM Songket Silungkang Kota Sawahlunto. Untuk mewujudkan hal tersebut, maka perlu dilakukan penelitian dengan judul: **“PROPAGASI BALIK MENENTUKAN PREDIKSI PRODUKSI USAHA SONGKET SILUNGKANG KOTA SAWAHLUNTO”.**

* 1. **Rumusan Masalah**

Berdasarkan analisa, pengamatan dari latar belakang telah diuraikan, maka dapat dirumuskan beberapa masalah sebagai berikut :

1. Bagaimana mengolah data-data perkembangan produksi UMKM Songket Silungkang dalam periode tertentu untuk mendapatkan nilai-nilai yang dibutuhkan dalam metode *backpropagation?*
2. Bagaimana mendapatkan prediksi yang akurat atas produksi UMKM Songket Silungkang?
   1. **Hipotesis**

Berdasarkan perumusan masalah di atas, penulis membuat suatu hipotesis atau dugaan sementara yaitu sebagai berikut :

1. Diharapkan dengan menerapkan metode Propagasi balik, data-data perkembangan produksi UMKM Songket Silungkang dalam periode tertentu bisa diolah dengan tepat untuk mendapatkan nilai-nilai yang dibutuhkan.
2. Diharapkan dengan menerapkan metode Propagasi balik dalam mengolah data perkembangan produksi UMKM Songket Silungkang dalam periode tertentu diharapkan dapat memprediksi produksi usaha Songket Silungkang Kota Sawahlunto dengan akurat, sehingga bisa membantu pihak-pihak yang terkait dalam mengambil keputusan yang baik untuk kemajuan perkembangan UMKM Songket Silungkang Kota Sawahlunto.

**BAB II**

**TINJAUAN PUSTAKA**

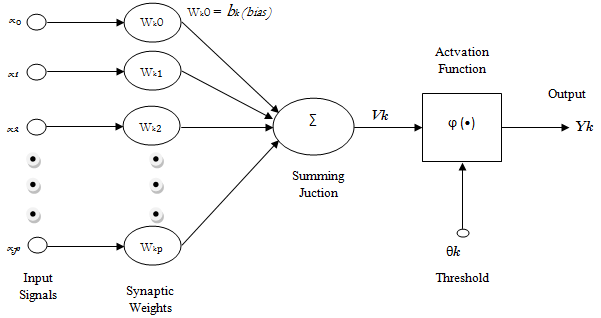
# 2.1 Jaringan Syaraf Tiruan

**2.1.1 Definisi Jaringan Syaraf Tiruan**

Jaringan syaraf tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba mensimulasikan proses pembelajaran pada otak manusia tersebut, istilah buatan digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran (Inggit Prahesti, 2013).

Jaringan syaraf tiruan merupakan generalisasi model matematis yang disusun dengan asumsi yang sama seperti jaringan syaraf biologis (Jong Jek Siang, 2009) :

1. Pemrosesan informasi terjadi pada banyak elemen sederhana (*neuron*).
2. Sinyal dikirimkan di antara *neuron*-*neuron* melalui penghubung-penghubung.
3. Penghubung antar *neuron* memiliki bobot yang akan memperkuat atau memperlemah sinyal.
4. Untuk menentukan *output*, setiap *neuron* menggunakan fungsi aktivasi (fungsi aktivasi yang digunakan biasanya fungsi yang *non*-linier, bukan fungsi linier) yang dikenakan pada jumlahan *input* yang diterima. Secara matematis, proses ini dijelaskan pada gambar 2.1.



**Gambar 2.1 Model Matematis Jaringan Syaraf Tiruan**

**2.1.2 Arsitektur Jaringan**

Baik tidaknya suatu model JST salah satunya ditentukan oleh hubungan antar *neuron* atau yang biasa disebut sebagai arsitektur jaringan. *Neuron*-*neuron* tersebut terkumpul dalam lapisan-lapisan yang disebut *neuron layer* (Sutojo dkk, 2011). Lapisan-lapisan penyusun jaringan syaraf tiruan dapat dibagi menjadi tiga yaitu (Diyah Puspitaningrum, 2009) :

1. Lapisan *Input*

*Node-node* di dalam lapisan *input* disebut unit-unit *input*. Unit-unit *input* menerima *input* dari dunia luar yang merupakan penggambaran dari suatu masalah.

1. Lapisan tersembunyi

*Node-node* di dalam lapisan tersembunyi disebut unit-unit tersembunyi. *Output* dari lapisan ini tidak secara langsung dapat diamati.

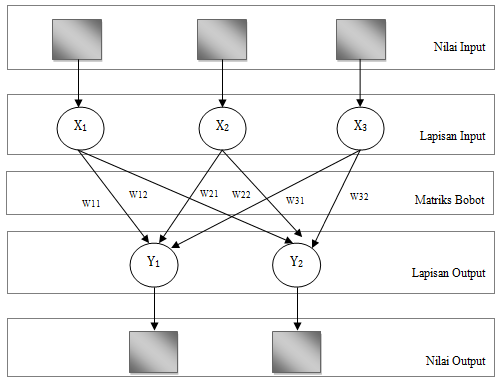
1. Lapisan *Output*

*Node-node* pada lapisan *output* disebut unit-unit *output*. Keluaran atau *output* dari lapisan ini merupakan *output* jaringan syaraf tiruan terhadap suatu permasalahan.

Ada beberapa arsitektur jaringan syaraf tiruan, antara lain (Sri Kusumadewi, 2003):

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

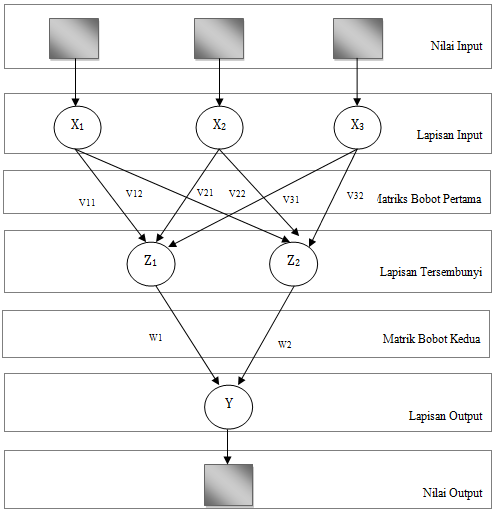
Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi. Contoh JST yang menggunakan jaringan lapisan tunggal adalah *ADALINE, Hopfield, Perceptron*.



**Gambar 2.2 Jaringan Syaraf Tiruan Dengan Lapisan Tunggal**

1. Jaringan Lapisan Banyak (*Multi Layer Net*)

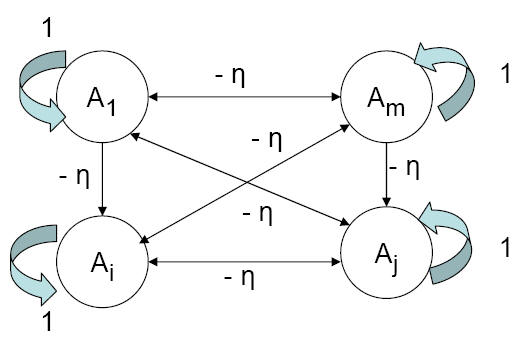
Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan yang terletak di antara lapisan *input* dan lapisan *output* (memiliki satu atau lebih lapisan tersembunyi). Umumya, ada lapisan bobot-bobot yang terletak antara dua lapisan yang bersebelahan. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada dengan lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Contoh JST yang menggunakan jaringan lapisan banyak adalah *MADALINE, backpropagation, neocognitron.*



**Gambar 2.3 Jaringan Syaraf Tiruan Dengan Lapisan Banyak**

1. Jaringan Dengan Lapisan Kompetitif (*Competitive Layer Net*)

Jaringan ini memiliki bobot yang telah ditentukan dan tidak memiliki proses pelatihan. Jaringan ini digunakan untuk mengetahui *neuron* pemenang dari sejumlah *neuron* yang ada. Akibatnya, pada jaringan ini sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Nilai bobot setiap *neuron* untuk dirinya sendiri adalah 1, sedangkan untuk *neuron* lainnya bernilai random negatif. Contoh JST yang menggunakan jaringan dengan lapisan kompetitif adalah *LVQ*.



**Gambar 2.4 Jaringan Syaraf Dengan Lapisan Kompetitif Dengan Bobot –Ƞ**

**2.1.3 Fungsi Aktivasi**

Faktor paling menentukan keaktifan suatu *neuron* adalah fungsi transfer yang biasa dikenal sebagai fungsi aktifasi, yang akan mengaktifkan *neuron*. Fungsi aktifasi menentukan bagaimana suatu *neuron* menanggapi sinyal-sinyal masukan, sehingga terjadi aktifitas satu *neuron*. Jika aktifitas *neuron* kuat, maka *neuron* akan menghasilkan sinyal keluaran yang dapat dihubungkan ke *neuron* lain (Lanny W. Pandjaitan, 2007). Fungsi aktivasi atau fungsi transfer merupakan fungsi yang menggambarkan hubungan antara aktivasi internal (*summation function*) yang mungkin berbentuk linear atau *non*-linear (Diyah Puspitaningrum, 2009).

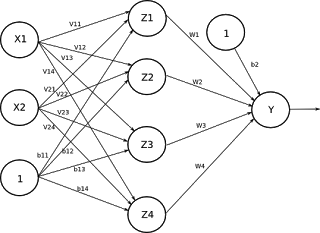
**2.1.4 Algoritma Pembelajaran**

Pelatihan jaringan syaraf tiruan dibagi menjadi dua, yaitu pelatihan dengan *supervise* (pembimbing) dan pelatihan tanpa supervisi. Pada proses pelatihan, suatu *input* dimasukan ke jaringan, kemudian jaringan akan memproses dan mengeluarkan suatu keluaran. Keluaran yang dihasilkan oleh jaringan akan dibandingkan dengan target, jika keluaran jaringan tidak sama dengan target, maka perlu dilakukan modifikasi bobot. Tujuan dari pelatihan ini adalah memodifikasi bobot hingga diperoleh bobot yang bisa membuat keluaran jaringan sama dengan target yang diinginkan (Sutojo dkk, 2011).

# 2.2 Metode Backpropagation

**2.2.1 Definisi Metode Backpropagation**

*Backpropagation* adalah sebuah metode matematik untuk pelatihan *multilaye*r jaringan syaraf tiruan. Jaringan *Backpropagation* merupakan salah satu algoritma yang sering digunakan dalam menyelesaikan masalah-masalah yang rumit. Algoritma ini memiliki dasar matematis yang kuat dan dilatih dengan menggunakan metode belajar terbimbing (Badrul Anwar, 2011). Propagasi balik melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respons yang benar terhadap pola masukan yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan (Andra Riztyan dkk, 2009).



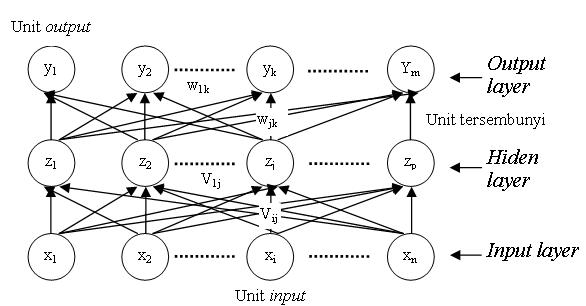
**Gambar 2.5 JST *Backpropagation* Dengan Satu Lapisan Tersembunyi**

**2.2.2 Arsitekur Metode Backpropagation**

Di dalam jaringan *backpropagation*, setiap unit yang berada di lapisan terhubung dengan setiap unit yang ada di lapisan tersembunyi. Setiap unit yang ada di lapisan tersembunyi terhubung dengan setiap unit yang ada di lapisan *output* (Diyah Puspitaningrum, 2009).

Jaringan syaraf tiruan *backpropagation* terdiri dari banyak lapisan (*multilayer neural networks*) :

1. Lapisan *input* (satu buah). Lapisan *input* terdiri dari *neuron*-*neuron* atau unit-unit *input*, mulai dari unit *input* *x1* sampai unit *input* *xn*.
2. Lapisan tersembunyi (minimal satu). Lapisan tersembunyi terdiri dari unit-unit tersembunyi mulai dari unit tersembunyi *z1* sampai *zp*.
3. Lapisan *output* (satu buah). Lapisan *output* terdiri dari unit-unit *output* mulai dari unit *output* *y1*sampai unit *output* *ym*.

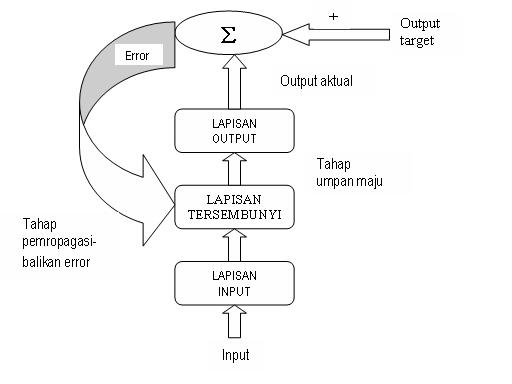


**Gambar 2.6 Arsitektur *Backpropagation***

**2.2.3 Algoritma Backpropagation**

Cara kerja *backpropagation*  adalah sebagai berikut : mula-mula jaringan diinisialisasikan dengan bobot yang diset dengan bilangan acak. Lalu contoh-contoh pelatihan dimasukan ke dalam jaringan. Contoh pelatihan terdiri dari pasangan vektor *input* dan vektor target. Keluaran dari jaringan berupa sebuah vektor *output* aktual. Selanjutnya vektor *output* aktual jaringan dibandingkan dengan vektor *output* target untuk mengetahui apakah *output* jaringan sudah sesuai dengan harapan (*output* aktual sudah sama dengan *output* target).

*Error* yang timbul akibat perbedaan antara *output* aktual dengan *output* target tersebut kemudian dihitung dan digunakan untuk mengubah bobot-bobot yang relevan dengan jalan mempropagasikan kembali *error*.setiap perubahan bobot yang terjadi diharapkan dapat mengurangi besar *error*. *Epoch* (siklus setiap pola pelatihan) seperti ini dilakukan pada semua set pelatihan sampai unjuk kerja jaringan mencapai tingkat yang diinginkan atau sampai kondisi berhenti terpenuhi. (Diyah Puspitaningrum, 2009).



**Gambar 2.7 Alur Kerja Jaringan *Backpropagation***

Penggunaan propagasi balik terdiri dari dua tahap (Andra Riztyan dkk, 2009) yaitu :

1. Tahap belajar atau pelatihan, di mana pada tahap ini pada propagasi balik diberikan sejumlah data pelatihan dan target.
2. Tahap pengujian atau penggunaan dilakukan setelah propagasi balik selesai belajar.

**2.2.4 Algoritma Pelatihan Backpropagation**

Sebelum memasukan data yang akan dilatih dan diuji, terlebih dahulu lakukan transformasi data. Transformasi data merupakan tahap di mana data *real* akan diubah menjadi data yang dibutuhkan dalam pelatihan Jaringan Syaraf Tiruan. Data yang diperoleh harus ditransformasikan terlebih dahulu dengan melakukan penskalaan terhadap *input* dan target hingga data-data *input*  dan target tersebut masuk dalam *range* tertentu, dengan begitu proses *training*  pada Jaringan Syaraf Tiruan akan lebih efisien dan efektif. Tujuan utama transformasi adalah agar terjadi sinkronisasi data, di samping itu juga untuk memudahkan dalam proses komputasi. Menyajikan data mentah secara langsung pada Jaringan Syaraf Tiruan akan membuat *neuron* mengalami saturasi dan gagal melakukan *training*. Data bisa ditransformasikan ke interval yang lebih kecil, misal pada interval [0,1], tapi akan lebih baik jika ditransformasikan ke interval yang lebih kecil, misal pada interval [0.1, 0.9], ini mengingat fungsi *sigmoid* merupakan fungsi *asimtotik* yang nilainya tidak pernah mencapai 0 ataupun 1 (Jong Jek Siang, 2009). Untuk mentransformasikan seluruh data *real* tersebut, digunakan fungsi sebagai berikut :

(11)

Di mana:

*a* = data minimum *b* = data maksimum

*x* = nilai asli dari data *x*1= nilai transformasi dari data

Algoritma pelatihan pada *backpropagation* sebagai berikut (Badrul Anwar, 2011) :

1. *Initialization*

Memberikan nilai awal terhadap nilai-nilai yang diperlukan oleh *neural network* seperti *weight, threshold.*

1. *Activation*

Nilai-nilai yang diberikan pada tahap *Initilization* akan digunakan pada tahap *Activation.*

Dengan melakukan perhitungan :

1. Menentukan *aktual output*  pada *hidden layer*
2. Menghitung *aktual output* pada *output layer*
3. *Weight Training*

Pada tahap *weight training* dilakukan dua kegiatan yaitu :

1. Menghitung *error gradient*  pada *output layer*
2. Menghitung *error gradient* pada *hidden layer*
3. *Iteration*

Pada tahap ini dilakukan proses pengulangan sampai mendapat *error* yang minimal.

Berikut ini adalah algoritma pelatihan untuk *backpropagation* dengan sebuah lapisan tersembunyi (Diyah Puspitaningrum, 2009) :

1. Inisialisasi bobot-bobot

Tentukan angka pembelajaran (α). Tentukan pula nilai toleransi eror atau nilai ambang (bila menggunakan nilai ambang sebagai kondisi berhenti); atau set maksimal *epoch* (bila menggunakan banyaknya *epoch* sebagai kondisi terhenti).

1. *While*  kondisi berhenti tidak terpenuhi *do* langkah ke-2 sampai langkah ke-9.
2. Untuk setiap pasangan pola pelatihan, lakukan langkah ke-4 sampai langkah ke-9

Tahapan Umpan Maju

1. Setiap unit *input* *xi*(dari unit ke-1 sampai unit ke-n pada lapisan *input*) mengirimkan sinyal *input* ke semua unit yang ada di lapisan atasnya (ke lapisan tersembunyi); *xi* (12)
2. Pada setiap unit di lapisan tersembunyi *zj* (dari unit ke-1 sampai unit ke-n ke-*p; i=1,……,n; j=l,....p*) sinyal *output* lapisan tersembunyinya dihitung dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal *input* berbobot *xi :*

*Zj =*  (13)

kemudian dikirim ke semua unit di lapisan atasnya.

1. Setiap unit di lapisan *output* *yk* (dari unit ke-1 sampai unit ke-m; i=1,….n; k=1,……m) dihitung sinyal *output*-nya dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal *input* berbobot *Zj* bagi lapisan ini :

*Yk =*  (14)

Tahap Pemprogasibalikan *Error*

1. Setiap unit *output* *Yk* (dari unit ke-1sampai unit ke-m ; j=1,……,p; k=1,……..m) menerima pola target *tk* lalu informasi kesalahan lapisan *output* ( ) dihitung k dikirim ke lapisan dibawahnya dan digunakan untuk menghitung besar koreks bobot dan *bias* (∆wjk dan ∆wok) antara lapisan tersembunyi dengan lapisan *output* :

k = (tk - yk) f’ (15.)

*jk = k Zj* (15.)

*0k = k* (15.) *Tahap Peng-update-an Bobot Bias*

1. Pada setiap unit di lapisan tersembunyi (dari unit ke-1 sampai unit ke-*p*; *i*=1,….,*n*; *j*=1…..*p*; *k*=1….m) dilakukan perhitungan informasi kesalahan lapisan tersembunyi (*j*). *j* kemudian digunakan untuk menghitung besar koreksi bobot dan *bias* (∆Vij dan ∆Voj) antara lapisan *input* dan lapisan tersembunyi.

= *f’* (16.)

*ij = j Xi* (16.)

*0j = j* (16.)

Tahap Perubahan bobot dan *bias*

1. Pada setiap unit *output* *Yk* (dari unit ke-1 sampai unit ke-m) dilakukan perubahan *bias* dan bobot (*j*=0,….,*p*; *k*=1,….,*m*) sehingga *bias* dan bobot yang baru menjadi :

*Wjk (baru) = Wjk (lama) +* (17.)

dari unit ke-1 sampai unit ke-*p* di lapisan tersembunyi juga dilakukan perubahan pada *bias* dan bobotnya (*i*=0,….,*n*; *j*=1,….,*p*):

*Vij (baru)*= *Vij (lama) +* (17.)

1. Tes kondisi berhenti.
2. Setelah proses pelatihan selesai, nilai-nilai ternormalisasi *output* jaringan (hasil peramalan Jaringan Syaraf Tiruan) harus dikembalikan (denormalisasi) ke nilai aslinya dengan persamaan sebagai berikut:

(18)

**2.2.4 Fungsi Aktivasi Pada Algoritma Pelatihan Backpropagation**

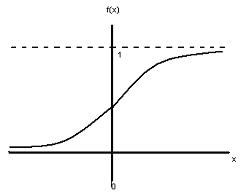
Beberapa fungsi aktivasi yang digunakan di dalam metode *backpropagation*  seperti fungsi *sigmoid biner, sigmoid bipolar,* dan *tangen hiperbolik.* Karakteristik yang harus dimiliki fungsi aktivasi tersebut adalah kontinu, diferensial dan tidak menurun secara monoton.

1. Fungsi *Sigmoid Biner*

Fungsi ini umum digunakan. *Range-*nya adalah (0,1) dan didefinisikan sebagai berikut

*f1* (x) = (22.)

dengan turunan *f1’* (x) = *f1* (x) (1- f1 (x)) (22.)



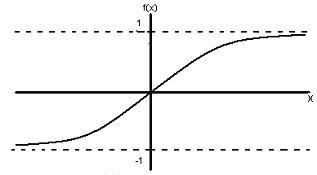
**Gambar 2.8 Fungsi *Sigmoid Biner*  Dengan *Range* (0,1)**

1. Fungsi *Sigmoid Bipolar*

Fungsi *Sigmoid Bipolar* memiliki *Range*-nya adalah (-1,1)

*f2*(x) = 2 *f1* (x) – 1 (23.)

dengan turunan *f2*‘(x) = (1 + *f2* (x)) (1 - *f2* (x)) (23.)



**Gambar 2.9 Fungsi *Sigmoid Bipolar* Dengan *Range* (-1,1)**

1. Fungsi *Tangen hiperbolik*

Fungsi *tangen hiperbolik* didefinisikan sebagai berikut :

tanh (x) = = (24.)

tanh’(x)= (1 + tanh(x)) (1-tanh(x)) (24.b)

**BAB III**

**TUJUAN DAN MANFAAT PENELITIAN**

**3.1 Tujuan Penelitian**

Tujuan dari penelitian ini adalah sebagai berikut:

1. Menganalisa data-data perkembangan produksi UMKM Songket Silungkang dalam periode tertentu di Kota Sawahlunto.
2. Mengetahui variabel-variabel apa saja yang dibutuhkan dalam memprediksi produksi UMKM Songket Silungkang Kota Sawahlunto.
3. Merancang dan membangun Jaringan Syaraf Tiruan dalam memprediksi produksi UMKM Songket Silungkang Kota Sawahlunto.
4. Menguji metode *backpropagation* dalam memprediksi produksi UMKM Songket Silungkang Kota Sawahlunto dengan menggunakan *software* MATLAB 6.1.

**4.1 Manfaat Penelitian**

Manfaat dari penelitian ini adalah sebagai berikut:

1. Penelitian tentang penerapan metode algoritma *backpropagation* dalam memprediksi produksi Songket Silungkang Kota Sawahlunto dapat membantu pihak pemerintah dalam pengambilan keputusan yang baik terhadap peningkatan produksi UMKM Songket Silungkang Kota Sawahlunto.
2. Dengan menerapkan metode algoritma *backpropagation* dalam memprediksi produksi Songket Silungkang Kota Sawahlunto dapat membantu pihak pelaku usaha mikro kecil dan menengah dalam melakukan upaya penanganan untuk meningkatkan produksi songket silungkang pada UMKM Songket Silungkang Kota Sawahlunto.
3. Memperluas ilmu pengetahuan tentang metode backpropagation dalam pengeloaan data yang berkaitan dengan prediksi produksi Songket Silungkang.

**BAB IV**

**METODE PENELITIAN**

**4.1 Pendahuluan**

Metodologi penelitian ini merupakan langkah-langkah kerja yang harus dilakukan agar penyusunan tesis menjadi lebih mudah, berurutan dan juga dapat digunakan sebagai pedoman untuk peneliti dalam melaksanakan penelitian. Metodologi yang digunakan dalam penyusunan tesis ini adalah pengumpulan data. Data-data yang dikumpulkan kemudian dilakukan identifikasi masalah dan kebutuhan, serta cara kerja dan ruang lingkup sistem yang kan dibuat.

Penelitian merupakan rangkaian kegiatan ilmiah dalam rangka pemecahan suatu permasalahan. Fungsi penelitian adalah mencarikan penjelasan dan jawaban terhadap permasalahan serta memberikan alternatif bagi kemungkinan yang dapat digunakan untuk pemecahan masalah.

**4.2 Kerangka Kerja Penelitian**

Kerangka kerja diperlukan dalam acuan langkah-langkah untuk mengerjakan suatu penelitian secara terstruktur dengan membuat sebuah tahapan metodologi penelitian sehingga hasil yang dicapai menjadi lebih maksimal. Kerangkan kerja pada penelitian ini dapat dilihat pada gambar 3.1.

Kerangka kerja dalam melakukan penelitian ini adalah sebagai berikut :

Memulai Penelitian

Identifikasi Masalah

Batasan Masalah

Perumusan Masalah

Menganalisa Metodologi

Paper Pendukung

Teori Pendukung

Menganalisa Data

Pengumpulan Data

Pengelompokan Data

Pengelolaan Data

Merancang Metode JST

Merancang Arsitektur JST

Analisa Hasil JST

Pelatihan JST

Install Matlab 6.1

Mengimplementasi Software

Mengelola Hasil JST dengan Aplikasi

Matlab 6.1

Mensintesa Hasil

Validasi Hasil

Menarik Kesimpulan

Pengujian Hasil

**Gambar 3.1 Kerangka Kerja Penelitian**

Pembahasan ini berisi penjelasan tentang kerangka kerja penelitian berdasarkan gambar 3.1.

1. Memulai Penelitian

Pada tahap ini dilakukan identifikasi masalah yang bertujuan untuk mengidentifikasikan masalah yang akan diteliti,batasan masalah bertujuan untuk mendapatkan hasil penelitian yang baik dan perumusan masalah bertujuan untuk menjelaskan garis besar permasalahan yang dihadapi dalam penelitian.

1. Menganalisa Metodologi

Pada tahap ini dilakukan analisa terhadap metodologi yang digunakan meliputi bahan pendukung dan teori pendukung dan hal-hal lain diperlukan dalam menyelesaikan penelitian.

1. Menganalisa Data

Pada tahap ini dilakukan studi pustaka yang bertujuan untuk mengetahui metode apa yang akan digunakan untuk menyelesaikan permasalahan yang akan diteliti yang nantinya menjadi referensi kuat bagi peneliti dalam menerapkan suatu metode yang digunakan.

1. Merancang Metode JST

Pada tahap ini, yang perlu dilakukan yaitu mengolah data input serta arsitektur sistem. Adapun tahap-tahapnya adalah sebagai berikut :

1. Transformasi data dilakukan agar terjadi kestabilan data yang dicapai dan juga menyesuaikan nilai data dengan *range* fungsi aktivasi yang digunakan dalam jaringan. Data ditransformasikan ke *interval* (0,1).
2. Pembagian data dilakukan dengan membagi data penelitian menjadi data pelatihan dan data pengujian.
3. Perancangan arsitektur jaringan yang optimum.
4. Memilih dan menggunakan arsitektur jaringan yang optimum.
5. Pemilihan jaringan optimum dan penggunaannya untuk peramalan.
6. Mengimplementasikan software

Tahap ini merupakan proses implementasi metode JST yang dibuat dengan menggunakan bahasa pemrograman *Matlab* 6.1 dan algoritma yang digunakan algoritma *backpropagation*.

1. Mensintesa Hasil

Tahap ini akan dilakukan pengevaluasian metode JST terhadap hasil yang didapatkan melalui pencarian secara manual dengan hasil yang didapatkan dengan mengunakan *software Matlab* 6.1. Hal ini dilakukan unuk menguji keakuratan hasil tersebut dan melihat apakah ada kesalahan-kesalahan yang harus diperbaiki untuk mendapatkan prediksi yang tepat terhadap produksi usaha songket silungkang di Kota Sawahlunto.

1. Kesimpulan

Pada akhir pembahasan dilakukan proses penarikan kesimpulan yang bertujuan untuk membandingkan hasil yang diperoleh dari tahap implementasi sistem yang dibuat secara manual.

**BAB V**

**HASIL DAN LUARAN YANG DICAPAI**

* 1. **Analisa Data**

Analisa data merupakan sebuah cara untuk mengolah data menjadi informasi agar karakteristik data tersebut mudah dipahami dan bermanfaat untuk solusi permasalahan, terutama hal yang berkaitan dengan penelitian. Analisis data bisa juga diartikan sebagai kegiatan yang dilakukan untuk merubah data hasil dari penelitian menjadi informasi yang nantinya dapat dipergunakan untuk mengambil kesimpulan.

Analisis data merupakan bagian yang amat penting, sebab dengan analisislah suatu data dapat diberi makna yang berguna untuk masalah penelitian. Data yang telah dikumpulkan oleh peneliti tidak akan ada gunanya apabila tidak dianalisis terlebih dahulu. Data yang akan digunakan dalam proses prediksi ini adalah jumlah produksi Songket Silungkang di UKM Arena Songket INJ dari januari 2015 sampai dengan april 2018. Data ini akan digunakan untuk mengetahui prediksi produksi songket pada ukm tersebut.

**Tabel 5.1: Produksi Songket Silungkang**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 | Data 8 | Data 9 | Data 10 |
| 136 | 136 | 149 | 147 | 194 | 140 | 130 | 176 | 161 | 144 |
| Data 11 | Data 12 | Data 13 | Data 14 | Data 15 | Data 16 | Data 17 | Data 18 | Data 19 | Data 20 |
| 178 | 157 | 167 | 164 | 160 | 157 | 178 | 130 | 132 | 167 |
| Data 21 | Data 22 | Data 23 | Data 24 | Data 25 | Data 26 | Data 27 | Data 28 | Data 29 | Data 30 |
| 158 | 144 | 163 | 148 | 162 | 156 | 154 | 171 | 159 | 153 |
| Data 31 | Data 32 | Data 33 | Data 34 | Data 35 | Data 36 | Data 37 | Data 38 | Data 39 | Data 40 |
| 160 | 156 | 162 | 158 | 144 | 168 | 160 | 160 | 156 | 160 |

* 1. **Transformasi Data *Real* Menjadi Data Pelatihan**

Langkah awal dalam melakukan transformasi adalah menentukan variabel *(x)*, kemudian menentukan nilai maksimum dan nilai minimum pada data. Berdasarkan data pada Tabel 4.1 didapatkan data tertinggi dan terendah adalah sebagai berikut :

* Nilai data maksimum = 194
* Nilai data minimum = 130

Dengan transformasi ini maka data terkecil akan menjadi 0.1 dan data terbesar menjadi 0.9. Berikut akan ditampilkan proses transformasi beberapa data secara manual.

1. x1 = (0.8 (136 – 130) / (194 - 130)) + 0.1 = 0.1750
2. x2 = (0.8 (136 – 130) / (194 - 130)) + 0.1 = 0.1750
3. x3 = (0.8 (149 – 130) / (194 - 130)) + 0.1 = 0.3375
4. x4 = (0.8 (147 – 130) / (194 - 130)) + 0.1 = 0.3125
5. x5 = (0.8 (194 – 130) / (194 - 130)) + 0.1 = 0.9000

Tabel 5.2 merupakan hasil transformasi data pada tabel 4.1 yang akan dipakai sebagai data pelatihan *Backpropagation.*

**Tabel 5.2 Hasil Transformasi Data Produksi Songket**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 | Data 8 | Data 9 | Data 10 |
| 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 |
| Data 11 | Data 12 | Data 13 | Data 14 | Data 15 | Data 16 | Data 17 | Data 18 | Data 19 | Data 20 |
| 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 |
| Data 21 | Data 22 | Data 23 | Data 24 | Data 25 | Data 26 | Data 27 | Data 28 | Data 29 | Data 30 |
| 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 |
| Data 31 | Data 32 | Data 33 | Data 34 | Data 35 | Data 36 | Data 37 | Data 38 | Data 39 | Data 40 |
| 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 | 0.4750 |

Selanjutnya hasil transformasi ini akan disusun membentuk 28 pola data. Data yang digunakan untuk pelatihan adalah data ke-1 sampai dengan data ke-12 dan target adalah data ke-13. Adapun 28 pola data tersebut dapat dilihat dalam Tabel 5.3.

**Tabel 5.3 Hasil Penyusunan Pola Data**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pola** | **Data Input** | | | | | | | | | | | | **Target** |
| **J1** | **J2** | **J3** | **J4** | **J5** | **J6** | **J7** | **J8** | **J9** | **J10** | **J11** | **J12** |
| **Pola-1** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 |
| **Pola-2** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 |
| **Pola-3** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 |
| **Pola-4** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 |
| **Pola-5** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 |
| **Pola-6** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 |
| **Pola-7** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 |
| **Pola-8** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 |
| **Pola-9** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 |
| **Pola-10** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 |
| **Pola-11** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 |
| **Pola-12** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 |
| **Pola-13** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 |
| **Pola-14** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 |
| **Pola-15** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 |
| **Pola-16** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 |
| **Pola-17** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 |
| **Pola-18** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 |
| **Pola-19** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 |
| **Pola-20** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 |
| **Pola-21** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 |
| **Pola** | **Data Input** | | | | | | | | | | | | **Target** |
| **J1** | **J2** | **J3** | **J4** | **J5** | **J6** | **J7** | **J8** | **J9** | **J10** | **J11** | **J12** |
| **Pola-22** | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 |
| **Pola-23** | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 |
| **Pola-24** | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 |
| **Pola-25** | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 |
| **Pola-26** | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 |
| **Pola-27** | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 |
| **Pola-28** | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 | 0.4750 |

* 1. **Perancangan Arsitektur Jaringan Syaraf Tiruan**

Proses selanjutnya adalah penentuan jumlah dari lapisan masukan (*input*), lapisan tersembunyi (*hidden layers*) dan lapisan keluaran (*output layers*). Pada penelitian ini arsitektur Jaringan Syaraf Tiruan yang digunakan adalah Jaringan Syaraf Tiruan dengan banyak lapisan (*multilayer net*) dengan algoritma *Backpropagation* dengan menggunakan fungsi aktivasi *sigmoid*, yang terdiri dari :

1. Lapisan masukan (*input*) dengan 12 simpul (*j1, j2,j3,j4,j5,jn*).
2. Lapisan tersembunyi (*hidden*) dengan jumlah simpul ditentukan oleh pengguna (*z1,z2,z3*).
3. Lapisan keluaran (*output*) dengan 1 simpul (*y*)

Hasil perancangan Jaringan Syaraf Tiruan Prediksi Produksi Usaha Songket Silungkang dilihat pada gambar 5.1.



**Gambar 5.1. Arsitektur Jaringan Syaraf Tiruan**

Pada gambar 5.1, penulis merancang Jaringan Syaraf Tiruan dengan algoritma *backpropagation* dengan fungsi aktivasi *sigmoid* untuk menetukan keluaran satu *neuron.*  Ditetapkan 12 buah variabel *input* *j1, j2,….j10* dan 3 buah *neuron* pada *hidden layer z1,….z3,* serta 1 buah *output layer* *y.*

Tahap selanjutnya yaitu menentukan nilai bobot dan *bias* untuk masing–masing *neuron* pada Jaringan Syaraf Tiruan. Bobot dan *bias* yang berperan penting dalam pelatihan Jaringan Syaraf Tiruan. Hal tersebut dikarenakan Jaringan Syaraf Tiruan belajar dengan cara meng-*update* bobot. Bobot yang tersimpan pada proses pelatihan nantinya akan digunakan pada proses pengujian. Pada awal pelatihan akan dilakukan penginisialisasikan nilai bobot jaringan termasuk juga nilai bobot *bias* untuk *hidden* dan *output layer*. Pada analisa ini, nilai bobot dan bias ditentukan secara acak. Inisialisai bobot dan bias secara acak dilakukan dengan menggunakan *syntax* matlab yaitu dengan :

1. Bobot pada layar *input* ke *hidden* : *net.IW {1,1}*
2. Bobot pada layar *hidden* ke *output* : *net.LW {2,1}*
3. Bobot bias pada layar *hidden* : *net.b{1}*
4. Bobot bias pada layar *output* : *net.b{2}*

Jika ada penambahan *hidden layer*, maka bobot ke *hidden layer* tambahan tersebut juga harus diinisialisasikan yaitu dengan menambah angka parameter *x* pada *net.LW{x,1}* dan bobot bias dengan *net.b{x}.*

**5.4 Postprocessing / Denormalisasi**

Setelah proses pelatihan selesai, nilai-nilai ternormalisasi *output* jaringan harus didenormalisasikan ke nilai aslinya untuk mendapatkan nilai *output* pada *range* yang sebenarnya. Setelah diperoleh nilai *output* yang sebenarnya maka hasil peramalan tersebut dapat dibandingkan dengan data *real* sehingga dapat diperoleh kesimpulan apakah jaringan syaraf tiruan telah benar dan akurat dalam meramalkan produksi usaha Songket Silungkang. Setelah tahap perancangan jaringan syaraf tiruan dan proses denormalisasi telah selesai, maka tahap selanjutnya yaitu proses pelatihan jaringan syaraf tiruan.

* 1. **Pelatihan Jaringan Syaraf Tiruan**

Pada proses pelatihan pada jaringan syaraf tiruan akan menggunakan beberapa pola data dengan parameter yang telah dirancang. Pelatihan menggunakan 21 pola dari 28 pola data. Data pelatihan tersebut terdiri dari pola 1 sampai pola 21 yang dapat dilihat pada tabel 5.4, dan data pengujian terdiri dari pola 22 sampai pola 28 dapat dilihat pada tabel 5.5.

Pola data yang digunakan untuk pelatihan jaringan syaraf tiruan dapat dilihat pada Tabel 5.4.

**Tabel 5.4 Pola Data Pelatihan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pola** | **Data Input** | | | | | | | | | | | | **Target** |
| **J1** | **J2** | **J3** | **J4** | **J5** | **J6** | **J7** | **J8** | **J9** | **J10** | **J11** | **J12** |
| **Pola-1** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 |
| **Pola-2** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 |
| **Pola-3** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 |
| **Pola-4** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 |
| **Pola-5** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 |
| **Pola-6** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 |
| **Pola-7** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 |
| **Pola-8** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 |
| **Pola-9** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 |
| **Pola-10** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 |
| **Pola-11** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 |
| **Pola-12** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 |
| **Pola-13** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 |
| **Pola-14** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 |
| **Pola-15** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 |
| **Pola-16** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 |
| **Pola-17** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 |
| **Pola-18** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 |
| **Pola-19** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 |
| **Pola-20** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 |
| **Pola-21** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 |

Pola data yang digunakan untuk pengujian jaringan syaraf tiruan dapat dilihat pada Tabel 5.5.

**Tabel 5.5 Pola Data Pengujian**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pola** | **Data Input** | | | | | | | | | | | | **Target** |
| **J1** | **J2** | **J3** | **J4** | **J5** | **J6** | **J7** | **J8** | **J9** | **J10** | **J11** | **J12** |
| **Pola-22** | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 |
| **Pola-23** | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 |
| **Pola-24** | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 |
| **Pola-25** | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 |
| **Pola-26** | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 |
| **Pola-27** | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 |
| **Pola-28** | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 | 0.4750 |

Perancangan data pelatihan dan pengujian memiliki 12 buah variabel *input* yaitu *x1, x2, x3, x4, x5, x6, x7, x8 ,x9, x10, x11* dan *x12* di mana nilainya adalah sebagai berikut :

*x1* = 136 *x2*= 136 *x3 =* 149 *x4 =* 147 *x5 =* 194 *x6*= 140

*x7* = 130 *x8*= 176 *x9 =* 161 *x10 =* 144 *x11 =* 178 *x12* = 157

Data tersebut ditransformasikan menjadi :

*j1* = 0.1750 *j2*= 0.1750 *j3 =* 0.3375 *j4 =* 0.3125 *j5 =* 0.9000 *j6 =* 0.2250

*j7* = 0.1000 *j8*= 0.6750 *j9 =* 0.4875 *j10 =* 0.2750 *j11 =* 0.7000 *j12 =* 0.4375

Target = 0.5625 Learning Rate (*α*) = 0.1

Arsitektur jaringan yang akan dibentuk adalah 12-3-1, di mana jumlah unit pada lapisan *input* adalah dua belas variabel, jumlah unit pada lapisan tersembunyi (*hidden layer*) adalah tiga dan jumlah unit pada lapisan *output*  adalah satu.

*Langkah 1.*

Inisialisasi bobot dan bias, nilai bobot dan bias ditentukan secara acak dengan menggunakan fungsi matlab.

1. Inisiasi bobot (*v*) dan *bias* (*i*) secara acak dari *input* ke lapisan tersembunyi (*hidden* layer) dapat dilihat pada tabel 5.6.

**Tabel 5.6 Nilai Bobot Dari Input ke Hidden Layer**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***z1*** | ***z2*** | ***z3*** |
| ***x1*** | 3.4759 | -1.9049 | 0.7564 |
| ***x2*** | -0.1082 | 2.7724 | 1.8556 |
| ***x3*** | -0.3362 | -3.4114 | 2.2755 |
| ***x4*** | -0.4270 | 0.8178 | 2.0668 |
| ***x5*** | 3.2572 | 1.6877 | -2.2919 |
| ***x6*** | -0.9708 | 4.1138 | 3.9354 |
| ***x7*** | -0.9239 | 3.7187 | -4.1733 |
| ***x8*** | -1.5149 | 2.9584 | -4.6267 |
| ***x9*** | -3.7180 | -2.8079 | -2.8439 |
| ***x10*** | 1.0686 | -2.1521 | -2.8431 |
| ***x11*** | -4.9907 | 2.3313 | -0.5183 |
| ***x12*** | 4.4459 | -0.3212 | -0.7679 |
| ***I*** | -3.3578 | -3.1172 | 5.4724 |

1. Inisiasi bobot (*w*) dan bias (*l*) secara acak dari lapisan tersembunyi ke *output* dapat dilihat pada tabel 5.7.

**Tabel 5.7 Nilai Bobot Dari Hidden Layer ke Output Layer**

|  |  |
| --- | --- |
|  | ***l*** |
| ***Z1*** | 0.6924 |
| ***Z2*** | 0.0503 |
| ***Z3*** | -0.5947 |
| ***l*** | 0.3443 |

*Langkah 2.*

Menghitung keluaran dari *hidden layer* (z) dengan menggunakan persamaan :

*z\_net1*= v10 + (j1\*v11) + (j2\*v21) + (j3\*v31) + (j4\*v41) + (j5\*v51) + (j6\*v61) + (j7\*v71) + (j8\*v81) + (j9\*v91) + (j10\*v101) + (j11\*v111) + (j12\*v121)

*z\_net1*= -3.3578+( 3.4759\*0.1750)+( -0.1082\*0.1750)+( -0.3362\*0.3375)+

(-0.4270\* 0.3125)+( 3.2572\*0.9000)+(-0.9708\*0.2250)+

(-0.9239\*0.1000)+(-1.5149\* 0.6750)+(-3.718\*0.4875)+

(1.0686\*0.2750)+(-4.9907\*0.7000)+(4.4459\*0.4375)

= -4.4843

*z\_net2*= v20 + (j1\*v12) + (j2\*v22) + (j3\*v32) + (j4\*v42) + (j5\*v52) + (j6\*v62) + (j7\*v72) + (j8\*v82) + (j9\*v92) + (j10\*v102)+ (j11\*v112) + (j12\*v122)

*z\_net*2 = -3.1172+( -1.9049\*0.1750)+( 2.7724\*0.1750)+( -3.4114\*0.3375)+

(0.8178\* 0.3125)+( 1.6878\*0.9000)+(4.1138\*0.2250)+

(3.7187\*0.1000)+(2.9584\* 0.6750)+(-2.8079\*0.4875)+

(-2.1521\*0.2750)+(2.3313\*0.7000)+(-0.3212\*0.4375)

= 0.4829

*z\_net3*= v20 + (j1\*v13) + (j2\*v23) + (j3\*v33) + (j4\*v43) + (j5\*v53) + (j6\*v63) + (j7\*v73) + (j8\*v83) + (j9\*v93) + (j10\*v103) + (j11\*v113) + (j12\*v123)

*z\_net*3 = 5.4724+( 0.7564\*0.1750)+( 1.8556\*0.1750)+( 2.2755\*0.3375)+

(2.0668\* 0.3125)+( -2.2919\*0.9000)+(3.9354\*0.2250)+

(-4.1733\*0.1000)+(-4.6267\* 0.6750)+(-2.8439\*0.4875)+

(-2.8431\*0.2750)+(-0.5183\*0.7000)+(-0.7679\*0.4375)

= -0.2413

z1 = sigmoid [-4.4843] = = 0.0112

z2 = sigmoid [0.4829] = = 0.6184

z3 = sigmoid [-0.2413] = = 0.4400

*Langkah 3***.**

Menghitung keluaran unit *yk* dengan menggunakan persamaan :

= 0.3443+ (0.0112\*0.6924) + (0.6184\*0.0503) + (0.4400\*-0.5947)

= 0.1215

= sigmoid [0.1215] = = 0.5303

*Langkah 4.*

Menghitung faktor δ di unit keluaran yk dengan menggunakan persamaan :

δk = (tk – yk) f` (y\_netk) = (tk – yk) yk (1-yk)

δk = (0.5625- 0.5303) \* 0.5303 (1 - 0.5303)

δk = 0.0080

Hitung suku perubahan bobot *Wjk* (yang akan digunakan untuk merubah bobot *Wjk*) dengan laju pelatihan α = 0.1 dengan persamaan :

∆Wkj = α δk zj

∆W10 = α δk zj

= 0.1 \* 0.0080\* 1

= 0.00080

∆W11 = α δk z1

= 0.1 \* 0.0080\* 0.0112

= 0.00001

∆W12 = α δk z2

= 0.1 \* 0.0080\* 0.6184

= 0.00050

∆W13 = α δk z3

= 0.1 \* 0.0080\* 0.4400

= 0.00035

*Langkah 5.*

Hitung penjumlahan kesalahan dari unit tersembunyi dengan persamaan :

=

= 0.0080 \* 0.6924

= 0.0055

=

= 0.0080 \* 0.0503

= 0.0004

=

= 0.0080 \* -0.5947

= -0.0048

Kemudian hitung faktor kesalahan di unit tersembunyi dengan persamaan:

= 0.0055 \* 0.0112 \* (1 - 0.0112)

= -0.0001

= 0.0004 \* 0.6184 \* (1 - 0.6184)

= 0.0001

= -0.0048 \* 0.4400 \* (1 - 0.4400)

= -0.0012

Hitung suku perubahan bobot ke unit tersembunyi dengan persamaan:

∆vji = α δj ji di mana α = 0.1

0.1 \* (0.0001) \* 0.1750 = 0.0000

0.1 \* (0.0001) \* 0.1750 = 0.0000

0.1 \* (0.0001) \* 0.3375 = 0.0000

0.1 \* (0.0001) \* 0.3125 = 0.0000

0.1 \* (0.0001) \* 0.9000 = 0.0000

0.1 \* (0.0001) \* 0.2250 = 0.0000

0.1 \* (0.0001) \* 0.1000 = 0.0000

0.1 \* (0.0001) \* 0.6750 = 0.0000

0.1 \* (0.0001) \* 0.4875 = 0.0000

0.1 \* (0.0001) \* 0.2750 = 0.0000

0.1 \* (0.0001) \* 0.7000 = 0.0000

0.1 \* (0.0001) \* 0.4375 = 0.0000

0.1 \* (0.0001) \* 0.1750 = 0.0000

0.1 \* (0.0001) \* 0.1750 = 0.0000

0.1 \* (0.0001) \* 0.3375 = 0.0000

0.1 \* (0.0001) \* 0.3125 = 0.0000

0.1 \* (0.0001) \* 0.9000 = 0.0000

0.1 \* (0.0001) \* 0.2250 = 0.0000

0.1 \* (0.0001) \* 0.1000 = 0.0000

0.1 \* (0.0001) \* 0.6750 = 0.0000

0.1 \* (0.0001) \* 0.4875 = 0.0000

0.1 \* (0.0001) \* 0.2750 = 0.0000

0.1 \* (0.0001) \* 0.7000 = 0.0000

0.1 \* (0.0001) \* 0.4375 = 0.0000

0.1 \* (-0.0002) \* 0.1750 = 0.0000

0.1 \* (-0.0002) \* 0.1750 = 0.0000

0.1 \* (-0.0002) \* 0.3375 = 0.0000

0.1 \* (-0.0002) \* 0.3125 = -0.0000

0.1 \* (-0.0002) \* 0.9000 = -0.0001

0.1 \* (-0.0002) \* 0.2250 = 0.0000

0.1 \* (-0.0002) \* 0.1000 = 0.0000

0.1 \* (-0.0002) \* 0.6750 = -0.0001

0.1 \* (-0.0002) \* 0.4875 = -0.0001

0.1 \* (-0.0002) \* 0.2750 = 0.0000

0.1 \* (-0.0002) \* 0.7000 = -0.0001

0.1 \* (-0.0002) \* 0.4375 = -0.0001

Sehingga didapatkan suku perubahan bobot ke unit tersembunyi yang dapat dilihat pada tabel 5.8 di bawah ini.

**Tabel 5.8 Suku Perubahan Bobot Unit Tersembunyi (**

|  |  |  |  |
| --- | --- | --- | --- |
|  | *z1* | *z2* | *z3* |
| *x1* | 0.0000 | 0.0000 | 0.0000 |
| *x2* | 0.0000 | 0.0000 | 0.0000 |
| *x3* | 0.0000 | 0.0000 | 0.0000 |
| *x4* | 0.0000 | 0.0000 | 0.0000 |
| *x5* | 0.0000 | 0.0000 | -0.0001 |
| *x6* | 0.0000 | 0.0000 | 0.0000 |
| *x7* | 0.0000 | 0.0000 | 0.0000 |
| *x8* | 0.0000 | 0.0000 | -0.0001 |
| *x9* | 0.0000 | 0.0000 | -0.0001 |
| *x10* | 0.0000 | 0.0000 | 0.0000 |
| *x11* | 0.0000 | 0.0000 | -0.0001 |
| *x12* | 0.0000 | 0.0000 | -0.0001 |

Langkah berikutnya hitung perubahan bobot garis yang menuju ke unit keluaran dengan persamaan :

*Wkj (baru) = wkj (lama) +*

*W1 (baru) = w1(lama) +*

= 0.6924+ 0.00001 = 0.6924

*W2 (baru) = w2(lama) +*

= 0.0503 + 0.00050 = 0.0508

*W3 (baru) = w3(lama) +*

= -0.5947 + 0.00035 = -0.5943

*Wo (bias baru) = w0(lama) +*

= 0.3443 + 0.00080 = 0.3451

Hitung koreksi nilai *bias* pada *neuron hidden* yang nantinya akan digunakan untuk memperbarui nilai dengan persamaan :

*V[oj]*:

= 0.1 \* 0.0001 = -0.00001

= 0.1 \* 0.0001 = 0.00001

= 0.1 \* -0.0012= -0.00012

Hitung nilai *bias* baru pada *neuron hidden* dengan persamaan :

= -3.3578 + 0.00001 = -3.3578

= -3.1172 + 0.00001 = -3.1172

= 5.4724 + (-0.00012) = 5.4723

Hitung perubahan bobot garis menuju ke unit tersembunyi dengan persamaan :

*Vkj (baru) = vkj (lama) +*

*V11 (baru) = v11(lama) +*

= 3.4759 + 0.0000 = 3.4759

*V21 (baru) = v21(lama) +*

= -0.1082 + 0.0000 = -0.1082

*V31 (baru) = v31(lama) +*

= -0.3362 + 0.0000 = -0.3362

*V41 (baru) = v41(lama) +*

= -0.4270 + 0.0000 = -0.4270

*V51 (baru) = v51(lama) +*

= 3.2572 + 0.0000 = 3.2572

*V61 (baru) = v61(lama) +*

= -0.9708 + 0.0000 = -0.9708

*V71 (baru) = v71(lama) +*

= -0.9239 + 0.0000 = -0.9239

*V81 (baru) = v81(lama) +*

= -1.5149 + 0.0000 = -1.5149

*V91 (baru) = v91(lama) +*

= -3.7180 + 0.0000 = -3.7180

*V101 (baru) = v101(lama) +*

= 1.0686 + 0.0000 = 1.0686

*V111 (baru) = v91(lama) +*

= -4.9907 + 0.0000 = -4.9907

*V121 (baru) = v101(lama) +*

= 4.4459 + 0.0000 = 4.4459

*V12 (baru) = v12(lama) +*

= -1.9049 + 0.0000 = -1.9049

*V22 (baru) = v22(lama) +*

= 2.7724 + 0.0000 = 2.7724

*V32 (baru) = v32(lama) +*

= -3.4114 + 0.0000 = -3.4114

*V42 (baru) = v42(lama) +*

= 0.8178 + 0.0000 = 0.8178

*V52 (baru) = v52(lama) +*

= 1.6877 + 0.0000 = 1.6877

*V62 (baru) = v62(lama) +*

= 4.1138 + 0.0000 = 4.1138

*V72 (baru) = v72(lama) +*

= 3.7187 + 0.0000 = 3.7187

*V82 (baru) = v82(lama) +*

= 2.9584 + 0.0000 = 2.9584

*V92 (baru) = v92(lama) +*

= -2.8079 + 0.0000 = -2.8079

*V102 (baru)* = *v102(lama) +*

= -2.1521 + 0.0000 = -2.1521

*V112 (baru)* = *v91(lama) +*

= 2.3313 + 0.0000 = 2.3313

*V122 (baru)* = *v101(lama) +*

= -0.3212 + 0.0000 = -0.3212

*V13 (baru) = v13(lama) +*

= 0.7564 + 0.0000 = 0.7564

*V23 (baru)* = *v23(lama) +*

= 1.8556 + 0.0000 = 1.8556

*V33 (baru) = v33(lama) +*

= 2.2755 + 0.0000 = 2.2755

*V43 (baru) = v43(lama) +*

= 2.0668 + 0.0000 = 2.0668

*V53 (baru) = v53(lama) +*

= -2.2919 + (-0.0001) = -2.2920

*V63 (baru) = v63(lama) +*

= 3.9354 + 0.0000 = 3.9354

*V73 (baru) = v73(lama) +*

= -4.1733 + 0.0000 = -4.1733

*V83 (baru) = v83(lama) +*

= -4.6267 + (-0.0001) = -4.6268

*V93 (baru) = v93(lama) +*

= -2.8439 + (-0.0001) = -2.8440

*V103 (baru) = v103(lama) +*

= -2.8431 + 0.0000 = -2.8431

*V113 (baru) = v91(lama) +*

= -0.5183 + (-0.0001) = -0.5184

*V123 (baru) = v101(lama) +*

= -0.7679 + (-0.0001) = -0.7680

Setelah selesai, akan didapatkan tabel nilai bobot baru pada dari *input layer* ke *hidden layer* seperti pada tabel 5.9.

**Tabel 5.9 Bobot Baru dari Input Layer ke Hidden Layer**

|  |  |  |  |
| --- | --- | --- | --- |
|  | *z1* | *z2* | *z3* |
| *x1* | 3.4759 | -1.9049 | 0.7564 |
| *x2* | -0.1082 | 2.7724 | 1.8556 |
| *x3* | -0.3362 | -3.4114 | 2.2755 |
| *x4* | -0.4270 | 0.8178 | 2.0668 |
| *x5* | 3.2572 | 1.6877 | -2.2920 |
| *x6* | -0.9708 | 4.1138 | 3.9354 |
| *x7* | -0.9239 | 3.7187 | -4.1733 |
| *x8* | -1.5149 | 2.9584 | -4.6268 |
| *x9* | -3.7180 | -2.8079 | -2.8440 |
| *x10* | 1.0686 | -2.1521 | -2.8431 |
| *x11* | -4.9907 | 2.3313 | -0.5184 |
| *x12* | 4.4459 | -0.3212 | -0.7680 |

Setelah proses pelatihan maupun proses pengujian selesai, data keluaran yang dihasilkan jaringan masih dalam bentuk normalisasi, sehingga perlu dilakukan proses denormalisasi data dengan tujuan mengkonversikan kembali hasil keluaran menjadi data *real* yang telah diprediksi. Proses denormalisasi nantinya akan dilakukan dengan menggunakan rumus denormalisasi. Misal hasil *output* dari jaringan adalah 0.1215 dengan *a* = 194 dan *b =* 130 maka untuk mendenormalisasikannya dapat dilakukan dengan :

= 132

Hasil denormalisasi dari 0.1215 adalah 132. Hasil tersebut masih memiliki nilai *error* yang besar, karena *target* yang sebenarnya adalah 136. Hal tersebut dikarenakan nilai *output* 0.1215 belum dilakukan proses pembelajaran sampai batas toleransi *error* (0.01).

**5.6 Analisa Hasil Pelatihan Jaringan Syaraf Tiruan**

Hasil perhitungan di atas merupakan hasil analisa yang dilakukan untuk menggambarkan bagaimana proses pelatihan yang dilakukan Jaringan Syaraf Tiruan *Backpropagation* secara manual terhadap produksi songket silungkang dalam menggali pola arsitektur yang sudah dirancang.

Berdasarkan pelatihan manual iterasi P = 1 dengan pola arsitektur jaringan 12-3-1 pada pola 1 dapat bahwa nilai yang didapatkan masih jauh dari nilai *output* yang diharapkan. Hal ini dikarenakan penganalisaan baru dilakukan pada satu proses iterasi saja. Sehingga masih perlu dilakukan pelatihan dengan pola yang lain hingga mencapai persentasi yang lebih mendekati nilai yang diharapkan.

Untuk mendapatkan hasil yang lebih akurat dalam penelitian ini penulis akan menggunakan *software Matlab* 6.1 dengan pola yang akan dilatih dan diuji dengan variasi yang lebih banyak lagi untuk mendapatkan kesimpulan yang benar.

Pelatihan dilakukan dengan algoritma *backpropagation* dengan *weight*-*elimination* yang bertujuan untuk mengenali pola-pola dari masukan pada data latih untuk dilatih pada jaringan yang akan menghasilkan keluaran untuk dibandingkan dengan data target. Hasil akhir dari pelatihan berupa bobot-bobot optimal akan diterapkan pada prediksi produk songket silungkang dan *output* akan dibandingkan dengan data yang sebelumnya untuk diketahui akurasinya.

**BAB VI**

**IMPLEMENTASI DAN PENGUJIAN SISTEM**

**6.1 Implementasi dan Pengujian Sistem**

Pada tahap ini menjelaskan bagaimana menguji data sampel dengan menggunakan *software* *matlab* untuk memprediksi produksi Songket Silungkang. *Matlab* merupakan *software* tepat sebagai alat komputasi yang melibatkan penggunaan *matriks* dan *vector*. Matlab menyediakan fungsi-fungsi khusus untuk menyelesaikan model jaringan syaraf tiruan. *Matlab* juga mendukung berbagai algoritma untuk jaringan syaraf tiruan di antaranya adalah algoritma *backpropagation*.

Hasil percobaan menunjukan bahwa *backpropagation* yang sudah dilatih dengan baik akan memberikan keluaran yang sesuai jika diberi masukan yang serupa (tidak harus sama) dengan pola yang dipakai dalam latihan. Sifat generalisasi ini membuat pelatihan lebih efesien karena tidak perlu dilakukan pada semua data.

**6.2 Hasil Pengenalan Data Training**

Untuk menguji akurasi pelatihan jaringan syaraf tiruan digunakan enam model arsitektur jaringan syaraf tiruan yaitu 12-3-1, 12-5-1, 12-7-1, 12-8-1, 12-9-1. Berikut ini tampilan hasil pelatihan dan pengujian jaringan syaraf tiruan dari model arsitektur tersebut.

**6.2.1 Model Arsitektur 12 – 3 – 1**

Model Arsitektur 12 – 3 – 1, bearti terdiri dari satu lapisan input yang memiliki 12 unit *neuron* yang terhubung langsung dengan lapisan tersembunyi yang memiliki 1 unit *neuron* tersembunyi. Kemudian, 1 unit *neuron* tersembunyi terhubung langsung dengan lapisan *output*  yang memiliki 1 unit neuron. Untuk mengolah data pelatihan 10-1-1 dengan menggunakan matlab melalui tahapan-tahapan sebagai berikut :

1. Membuat data *input.*

*>> p=[0.1750 0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500;*

*0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750;*

*0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125;*

*0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250;*

*0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000;*

*0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250;*

*0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000;*

*0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125;*

*0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625;*

*0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875;*

*0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750;*

*0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250]*

1. Membuat data *output.*

*>> t=[0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000]*

1. Membuat jaringan syaraf tiruan metode *backpropagation.*

*>> net=newff(minmax(p),[3,1],{'logsig','purelin'},'traingdx')*

Perintah di atas digunakan untuk jaringan syaraf tiruan dengan model arsitektur 12 – 3 – 1.

1. Melihat bobot dari *input* ke *hidden layer*.

*>>net.iw{1,1}*

1. Melihat bobot *hidden layer* ke *output layer.*

*>>Net.LW{2,1}*

1. Melihat bias *input* ke *hidden layer*

*>> net.b{1}*

1. Melihat bias *output layer*

*>>net.b{2}*

1. Melihat hasil jaringan syaraf tiruan pada iterasi 1.

*>>[y,Pf,Af,e,Perf]=sim(net,pn,[],[],tn)*

1. Dikarenakan jaringan syaraf tiruan belum mampu mengenali target yang diinginkan. Oleh karna itu jaringan syaraf tersebut harus diberikan pelatihan, untuk melakukan pelatihan maka ditentukan terlebih dahulu parameter - parameter yang diperlukan dalam proses pelatihan adalah sebagai berikut :

*>>net.trainParam.epochs=5000;*

Parameter ini digunakan untuk menentukan *epoch* maksimum dalam pelatihan.

*>>net.trainParam.goal=0,01;*

Parameter ini digunakan untuk menentukan batasan *MSE* agar iterasi diberhentikan. Iterasi akan diberhentikan jika *MSE* < batas yang ditentukan dalam *net. trainParam.epochs.*

*>>net.trainParam.lr = 0 ,1;*

Parameter ini digunakan untuk laju pemahaman (*α =* *learning rate*). *Default* = 0.01, semakin besar nilai α, maka semakin cepat proses pelatihan, tetapi jika nilai α terlau besar, maka algoritma terjadi tidak stabil dan akan mencapai titik minimum.

*>>net.trainParam.show=100;*

Parameter ini digunakan untuk menampilkan frekuensi perubahan *MSE* (*default* setiap 100 epoch)

*>> net = train(net,p,t);*

TRAINGDX, Epoch 0/5000, MSE 0.291638/0.01, Gradient 1.45864/1e-006

TRAINGDX, Epoch 25/5000, MSE 0.0172327/0.01, Gradient 0.05298/1e-006

TRAINGDX, Epoch 50/5000, MSE 0.0143517/0.01, Gradient 0.0127608/1e-006

TRAINGDX, Epoch 75/5000, MSE 0.0136676/0.01, Gradient 0.00397255/1e-006

TRAINGDX, Epoch 100/5000, MSE 0.0132021/0.01, Gradient 0.00244149/1e-006

TRAINGDX, Epoch 125/5000, MSE 0.0127037/0.01, Gradient 0.0255689/1e-006

TRAINGDX, Epoch 150/5000, MSE 0.0125949/0.01, Gradient 0.00972842/1e-006

TRAINGDX, Epoch 175/5000, MSE 0.0125203/0.01, Gradient 0.00431887/1e-006

TRAINGDX, Epoch 200/5000, MSE 0.0123365/0.01, Gradient 0.00238437/1e-006

TRAINGDX, Epoch 225/5000, MSE 0.0117253/0.01, Gradient 0.00453711/1e-006

TRAINGDX, Epoch 250/5000, MSE 0.0116404/0.01, Gradient 0.0147632/1e-006

TRAINGDX, Epoch 275/5000, MSE 0.0115709/0.01, Gradient 0.00722021/1e-006

TRAINGDX, Epoch 300/5000, MSE 0.0114474/0.01, Gradient 0.00245831/1e-006

TRAINGDX, Epoch 325/5000, MSE 0.0110578/0.01, Gradient 0.00227645/1e-006

TRAINGDX, Epoch 350/5000, MSE 0.0106662/0.01, Gradient 0.020146/1e-006

TRAINGDX, Epoch 375/5000, MSE 0.0105741/0.01, Gradient 0.00278169/1e-006

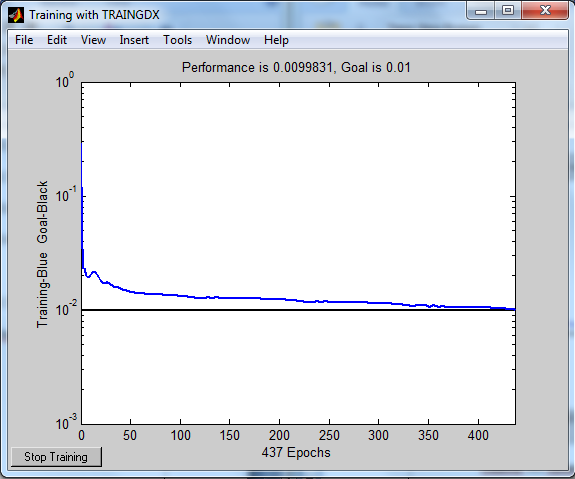
TRAINGDX, Epoch 400/5000, MSE 0.010495/0.01, Gradient 0.00373408/1e-006

TRAINGDX, Epoch 425/5000, MSE 0.0102519/0.01, Gradient 0.00235882/1e-006

TRAINGDX, Epoch 437/5000, MSE 0.0099831/0.01, Gradient 0.00222762/1e-006

TRAINGDX, Performance goal met.

Dengan perintah di atas maka akan menghasilkan keluaran pada gambar 6.1.



**Gambar 6.1 Antarmuka Grafik Pelatihan Model JST 12-3-1**

Hasil pelatihan mencapai goal pada *epochs* atau iterasi ke-437 dan *error* 0.0099837. Untuk hasil yang lebih rinci dan mengetahui *output* dan *error* dapat dilihat pada tabel 6.1. Dengan menggunakan perintah pada *matlab >>[y,Pf,Af,e,Perf]=sim(net,pn,[],[],tn)*

**Tabel 6.1 Hasil dan Error Data Input Pelatihan dengan Pola 12-3-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** | ***X4*** | ***X5*** | ***X6*** | ***X7*** | ***X8*** | ***X9*** | ***X10*** | ***X11*** | ***X12*** | ***Target*** | ***JST*** | |
| ***Act*** | ***Error*** |
| ***Pola-1*** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5301 | 0.0324 |
| ***Pola-2*** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4917 | 0.0333 |
| ***Pola-3*** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.3575 | 0.1175 |
| ***Pola-4*** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.4463 | -0.0088 |
| ***Pola-5*** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.6744 | 0.0256 |
| ***Pola-6*** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.3499 | -0.2499 |
| ***Pola-7*** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.3205 | -0.1955 |
| ***Pola-8*** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4791 | 0.0834 |
| ***Pola-9*** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.4469 | 0.0031 |
| ***Pola-10*** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.2617 | 0.0133 |
| ***Pola-11*** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.5181 | -0.0056 |
| ***Pola-12*** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.4528 | -0.1278 |
| ***Pola-13*** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4700 | 0.0300 |
| ***Pola-14*** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4489 | -0.0239 |
| ***Pola-15*** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.3545 | 0.0455 |
| ***Pola-16*** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4629 | 0.1496 |
| ***Pola-17*** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.6269 | -0.1644 |
| ***Pola-18*** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.3298 | 0.0577 |
| ***Pola-19*** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4146 | 0.0604 |
| ***Pola-20*** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.3790 | 0.0460 |
| ***Pola-21*** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4182 | 0.0818 |

**6.2.2 Model Arsitektur 12 – 5 – 1**

Model Arsitektur 12 – 5 – 1, bearti terdiri dari satu lapisan input yang memiliki 12 unit *neuron* yang terhubung langsung dengan lapisan tersembunyi yang memiliki 5 unit *neuron* tersembunyi. Kemudian, 5 unit *neuron* tersembunyi terhubung langsung dengan lapisan *output*  yang memiliki 1 unit neuron. Untuk mengolah data pelatihan 10-5-1 dengan menggunakan *matlab* melalui tahapan-tahapan sebagai berikut :

*>> p=[0.1750 0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500;*

*0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750;*

*0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125;*

*0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250;*

*0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000;*

*0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250;*

*0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000;*

*0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125;*

*0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625;*

*0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875;*

*0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750;*

*0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250]*

*>> t=[0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000]*

*>> net = newff(minmax(p),[5,1],{'logsig','purelin'},'traingdx');*

*>> net.iw{1,1}*

*>> net.LW{2,1}*

*>> net.b{1}*

*>> net.b{2}*

*>> [y,Pf,Af,e,perf]=sim(net,p,[],[],t)*

*>> net.trainParam.epochs=5000;*

*>> net.trainParam.goal=0.01;*

*>> net.trainParam.lr=0.1;*

*>> net=train(net,p,t);*

TRAINGDX, Epoch 0/5000, MSE 2.01887/0.01, Gradient 4.60758/1e-006

TRAINGDX, Epoch 25/5000, MSE 0.0337521/0.01, Gradient 0.0843096/1e-006

TRAINGDX, Epoch 50/5000, MSE 0.0203898/0.01, Gradient 0.03059/1e-006

TRAINGDX, Epoch 75/5000, MSE 0.0138992/0.01, Gradient 0.00446848/1e-006

TRAINGDX, Epoch 100/5000, MSE 0.0126263/0.01, Gradient 0.00414104/1e-006

TRAINGDX, Epoch 125/5000, MSE 0.0125001/0.01, Gradient 0.0850001/1e-006

TRAINGDX, Epoch 150/5000, MSE 0.01171/0.01, Gradient 0.0179072/1e-006

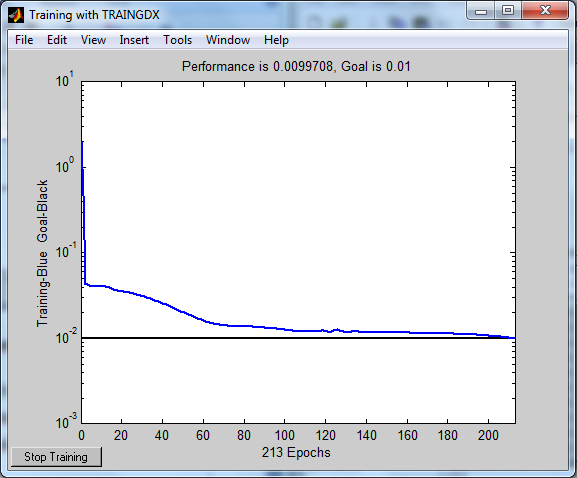
TRAINGDX, Epoch 175/5000, MSE 0.0114466/0.01, Gradient 0.003927/1e-006

TRAINGDX, Epoch 200/5000, MSE 0.0107424/0.01, Gradient 0.00384044/1e-006

TRAINGDX, Epoch 213/5000, MSE 0.0099708/0.01, Gradient 0.00479839/1e-006

TRAINGDX, Performance goal met.

Dengan perintah di atas maka akan menghasilkan keluaran pada gambar 6.2.



**Gambar 6.2 Antarmuka Grafik Pelatihan Model JST 12-5-1**

Hasil pelatihan mencapai goal pada *epochs* atau iterasi ke-213 dan *error* 0.0099708. Untuk hasil yang lebih rinci dan mengetahui *output* dan *error* dapat dilihat pada tabel 4.2. Dengan menggunakan perintah pada matlab *>>[y,Pf,Af,e,Perf]=sim(net,pn,[],[],tn)*

**Tabel 6.2 Hasil dan Error Data Input Pelatihan dengan Pola 12-5-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** | ***X4*** | ***X5*** | ***X6*** | ***X7*** | ***X8*** | ***X9*** | ***X10*** | ***X11*** | ***X12*** | ***Target*** | ***JST*** | |
| ***Act*** | ***Error*** |
| ***Pola-1*** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.4355 | 0.1270 |
| ***Pola-2*** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.5009 | 0.0241 |
| ***Pola-3*** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.3600 | 0.1150 |
| ***Pola-4*** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.4632 | -0.0257 |
| ***Pola-5*** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.5209 | 0.1791 |
| ***Pola-6*** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.3029 | -0.2029 |
| ***Pola-7*** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.1732 | -0.0482 |
| ***Pola-8*** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.5464 | 0.0161 |
| ***Pola-9*** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.3742 | 0.0758 |
| ***Pola-10*** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5034 | -0.2284 |
| ***Pola-11*** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.4265 | 0.0860 |
| ***Pola-12*** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.3703 | -0.0453 |
| ***Pola-13*** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4556 | 0.0444 |
| ***Pola-14*** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4391 | -0.0141 |
| ***Pola-15*** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.4978 | -0.0978 |
| ***Pola-16*** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.5823 | 0.0302 |
| ***Pola-17*** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.4682 | -0.0057 |
| ***Pola-18*** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.3863 | 0.0012 |
| ***Pola-19*** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.3809 | 0.0941 |
| ***Pola-20*** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5424 | -0.1174 |
| ***Pola-21*** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4841 | 0.0159 |

**6.2.3 Model Arsitektur 12 – 7 – 1**

Model Arsitektur 12 – 7 – 1, bearti terdiri dari satu lapisan input yang memiliki 12 unit *neuron* yang terhubung langsung dengan lapisan tersembunyi yang memiliki 7 unit *neuron* tersembunyi. Kemudian, 7 unit *neuron* tersembunyi terhubung langsung dengan lapisan *output*  yang memiliki 1 unit neuron. Untuk mengolah data pelatihan 10-7-1 dengan menggunakan *matlab* melalui tahapan-tahapan sebagai berikut :

*>> p=[0.1750 0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500;*

*0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750;*

*0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125;*

*0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250;*

*0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000;*

*0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250;*

*0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000;*

*0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125;*

*0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625;*

*0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875;*

*0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750;*

*0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250]*

*>> t=[0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000]*

*>> net = newff(minmax(p),[7,1],{'logsig','purelin'},'traingdx');*

*>> net.iw{1,1}*

*>> net.LW{2,1}*

*>> net.b{1}*

*>> net.b{2}*

*>> [y,Pf,Af,e,perf]=sim(net,p,[],[],t)*

*>> net.trainParam.epochs=5000;*

*>> net.trainParam.goal=0.01;*

*>> net.trainParam.lr=0.1;*

*>> net=train(net,p,t);*

TRAINGDX, Epoch 0/5000, MSE 0.339794/0.01, Gradient 1.19896/1e-006

TRAINGDX, Epoch 25/5000, MSE 0.139316/0.01, Gradient 0.179699/1e-006

TRAINGDX, Epoch 50/5000, MSE 0.0490026/0.01, Gradient 0.0676458/1e-006

TRAINGDX, Epoch 75/5000, MSE 0.0153044/0.01, Gradient 0.0102961/1e-006

TRAINGDX, Epoch 100/5000, MSE 0.01306/0.01, Gradient 0.084188/1e-006

TRAINGDX, Epoch 125/5000, MSE 0.0125901/0.01, Gradient 0.0307127/1e-006

TRAINGDX, Epoch 150/5000, MSE 0.0124621/0.01, Gradient 0.00354474/1e-006

TRAINGDX, Epoch 175/5000, MSE 0.0123278/0.01, Gradient 0.00314814/1e-006

TRAINGDX, Epoch 200/5000, MSE 0.012004/0.01, Gradient 0.00226695/1e-006

TRAINGDX, Epoch 225/5000, MSE 0.0119135/0.01, Gradient 0.0401035/1e-006

TRAINGDX, Epoch 250/5000, MSE 0.0118075/0.01, Gradient 0.0200255/1e-006

TRAINGDX, Epoch 275/5000, MSE 0.0117448/0.01, Gradient 0.00267277/1e-006

TRAINGDX, Epoch 300/5000, MSE 0.0116563/0.01, Gradient 0.00213139/1e-006

TRAINGDX, Epoch 325/5000, MSE 0.0113933/0.01, Gradient 0.00443801/1e-006

TRAINGDX, Epoch 350/5000, MSE 0.0117178/0.01, Gradient 0.070481/1e-006

TRAINGDX, Epoch 375/5000, MSE 0.0113612/0.01, Gradient 0.0157047/1e-006

TRAINGDX, Epoch 400/5000, MSE 0.0113284/0.01, Gradient 0.0060971/1e-006

TRAINGDX, Epoch 425/5000, MSE 0.0112836/0.01, Gradient 0.0016651/1e-006

TRAINGDX, Epoch 450/5000, MSE 0.0111487/0.01, Gradient 0.00159863/1e-006

TRAINGDX, Epoch 475/5000, MSE 0.0111602/0.01, Gradient 0.0394692/1e-006

TRAINGDX, Epoch 500/5000, MSE 0.011063/0.01, Gradient 0.0202031/1e-006

TRAINGDX, Epoch 525/5000, MSE 0.0110103/0.01, Gradient 0.00244311/1e-006

TRAINGDX, Epoch 550/5000, MSE 0.0109535/0.01, Gradient 0.00160824/1e-006

TRAINGDX, Epoch 575/5000, MSE 0.0109015/0.01, Gradient 0.0417804/1e-006

TRAINGDX, Epoch 600/5000, MSE 0.0107845/0.01, Gradient 0.0161751/1e-006

TRAINGDX, Epoch 625/5000, MSE 0.0107597/0.01, Gradient 0.0103667/1e-006

TRAINGDX, Epoch 650/5000, MSE 0.0107254/0.01, Gradient 0.00171742/1e-006

TRAINGDX, Epoch 675/5000, MSE 0.010646/0.01, Gradient 0.00150432/1e-006

TRAINGDX, Epoch 700/5000, MSE 0.0106255/0.01, Gradient 0.0424591/1e-006

TRAINGDX, Epoch 725/5000, MSE 0.0105029/0.01, Gradient 0.0172112/1e-006

TRAINGDX, Epoch 750/5000, MSE 0.0104656/0.01, Gradient 0.0063834/1e-006

TRAINGDX, Epoch 775/5000, MSE 0.0104193/0.01, Gradient 0.00195662/1e-006

TRAINGDX, Epoch 800/5000, MSE 0.0102761/0.01, Gradient 0.00151709/1e-006

TRAINGDX, Epoch 825/5000, MSE 0.0108/0.01, Gradient 0.0886342/1e-006

TRAINGDX, Epoch 850/5000, MSE 0.0101933/0.01, Gradient 0.00496641/1e-006

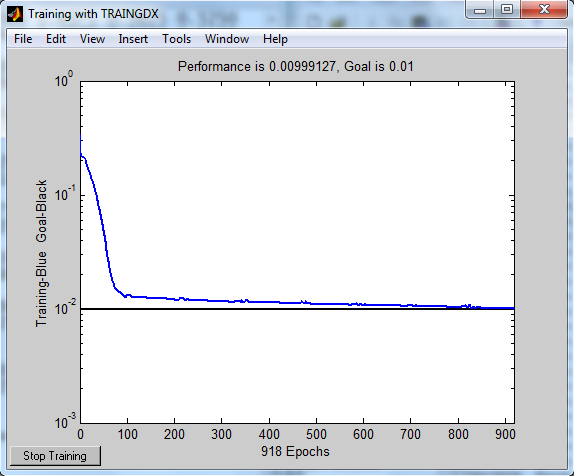
TRAINGDX, Epoch 875/5000, MSE 0.0101685/0.01, Gradient 0.00315204/1e-006

TRAINGDX, Epoch 900/5000, MSE 0.0101084/0.01, Gradient 0.00171382/1e-006

TRAINGDX, Epoch 918/5000, MSE 0.00999127/0.01, Gradient 0.00178408/1e-006

TRAINGDX, Performance goal met.

Dengan perintah di atas maka akan menghasilkan keluaran pada gambar 6.3.



**Gambar 6.3 Antarmuka Grafik Pelatihan Model JST 12-7-1**

Hasil pelatihan mencapai goal pada *epochs* atau iterasi ke-918 dan *error* 0.00999127. Untuk hasil yang lebih rinci dan mengetahui *output* dan *error* dapat dilihat pada tabel 6.3. Dengan menggunakan perintah pada matlab *>>[y,Pf,Af,e,Perf]=sim(net,pn,[],[],tn)*

**Tabel 6.3 Hasil dan Error Data Input Pelatihan dengan Pola 12-7-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** | ***X4*** | ***X5*** | ***X6*** | ***X7*** | ***X8*** | ***X9*** | ***X10*** | ***X11*** | ***X12*** | ***Target*** | ***JST*** | |
| ***Act*** | ***Error*** |
| ***Pola-1*** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5223 | 0.0402 |
| ***Pola-2*** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.5372 | -0.0122 |
| ***Pola-3*** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4008 | 0.0742 |
| ***Pola-4*** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.4636 | -0.0261 |
| ***Pola-5*** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.5879 | 0.1121 |
| ***Pola-6*** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.2574 | -0.1574 |
| ***Pola-7*** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.2967 | -0.1717 |
| ***Pola-8*** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4603 | 0.1022 |
| ***Pola-9*** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.4906 | -0.0406 |
| ***Pola-10*** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.3444 | -0.0694 |
| ***Pola-11*** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.5108 | 0.0017 |
| ***Pola-12*** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.4399 | -0.1149 |
| ***Pola-13*** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.5427 | -0.0427 |
| ***Pola-14*** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4664 | -0.0414 |
| ***Pola-15*** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.4115 | -0.0115 |
| ***Pola-16*** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4544 | 0.1581 |
| ***Pola-17*** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.6364 | -0.1739 |
| ***Pola-18*** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.3196 | 0.0679 |
| ***Pola-19*** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4067 | 0.0683 |
| ***Pola-20*** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.3800 | 0.0450 |
| ***Pola-21*** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.3137 | 0.1863 |

**4.2.4 Model Arsitektur 12 – 8 – 1**

Model Arsitektur 12 – 8 – 1, bearti terdiri dari satu lapisan input yang memiliki 12 unit *neuron* yang terhubung langsung dengan lapisan tersembunyi yang memiliki 8 unit *neuron* tersembunyi. Kemudian, 8 unit *neuron* tersembunyi terhubung langsung dengan lapisan *output*  yang memiliki 1 unit neuron. Untuk mengolah data pelatihan 10-8-1 dengan menggunakan *matlab* melalui tahapan-tahapan sebagai berikut :

*>> p=[0.1750 0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500;*

*0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750;*

*0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125;*

*0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250;*

*0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000;*

*0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250;*

*0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000;*

*0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125;*

*0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625;*

*0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875;*

*0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750;*

*0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250]*

*>> t=[0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000]*

*>> net = newff(minmax(p),[8,1],{'logsig','purelin'},'traingdx');*

*>> net.iw{1,1}*

*>> net.LW{2,1}*

*>> net.b{1}*

*>> net.b{2}*

*>> [y,Pf,Af,e,perf]=sim(net,p,[],[],t)*

*>> net.trainParam.epochs=5000;*

*>> net.trainParam.goal=0.01;*

*>> net.trainParam.lr=0.1;*

*>> net=train(net,p,t);*

TRAINGDX, Epoch 0/5000, MSE 0.357281/0.01, Gradient 1.78036/1e-006

TRAINGDX, Epoch 25/5000, MSE 0.0559169/0.01, Gradient 0.185671/1e-006

TRAINGDX, Epoch 50/5000, MSE 0.0255835/0.01, Gradient 0.0503442/1e-006

TRAINGDX, Epoch 75/5000, MSE 0.0160438/0.01, Gradient 0.0144371/1e-006

TRAINGDX, Epoch 100/5000, MSE 0.0108082/0.01, Gradient 0.0115982/1e-006

TRAINGDX, Epoch 125/5000, MSE 0.0105906/0.01, Gradient 0.00366786/1e-006

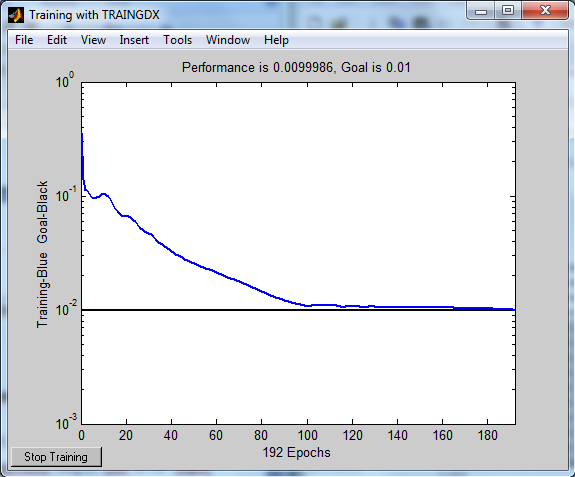
TRAINGDX, Epoch 150/5000, MSE 0.0105183/0.01, Gradient 0.00972229/1e-006

TRAINGDX, Epoch 175/5000, MSE 0.0103161/0.01, Gradient 0.00337138/1e-006

TRAINGDX, Epoch 192/5000, MSE 0.0099986/0.01, Gradient 0.0032102/1e-006

TRAINGDX, Performance goal met.

Dengan perintah di atas maka akan menghasilkan keluaran pada gambar 6.4.



**Gambar 6.4 Antarmuka Grafik Pelatihan Model JST 12-8-1**

Hasil pelatihan mencapai goal pada *epochs* atau iterasi ke-192 dan *error* 0.0099986 Untuk hasil yang lebih rinci dan mengetahui *output* dan *error* dapat dilihat pada tabel 6.4. Dengan menggunakan perintah pada matlab *>>[y,Pf,Af,e,Perf]=sim(net,pn,[],[],tn)*

**Tabel 6.4 Hasil dan Error Data Input Pelatihan dengan Pola 12-8-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** | ***X4*** | ***X5*** | ***X6*** | ***X7*** | ***X8*** | ***X9*** | ***X10*** | ***X11*** | ***X12*** | ***Target*** | ***JST*** | |
| ***Act*** | ***Error*** |
| ***Pola-1*** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.3964 | 0.1661 |
| ***Pola-2*** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4976 | 0.0274 |
| ***Pola-3*** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.3877 | 0.0873 |
| ***Pola-4*** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.4928 | -0.0553 |
| ***Pola-5*** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.6603 | 0.0397 |
| ***Pola-6*** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.3027 | -0.2027 |
| ***Pola-7*** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.2795 | -0.1545 |
| ***Pola-8*** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.5681 | -0.0056 |
| ***Pola-9*** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.3141 | 0.1359 |
| ***Pola-10*** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.4195 | -0.1445 |
| ***Pola-11*** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.4605 | 0.0520 |
| ***Pola-12*** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.3363 | -0.0113 |
| ***Pola-13*** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.3865 | 0.1135 |
| ***Pola-14*** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.3583 | 0.0667 |
| ***Pola-15*** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.4438 | -0.0438 |
| ***Pola-16*** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.5022 | 0.1103 |
| ***Pola-17*** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.5123 | -0.0498 |
| ***Pola-18*** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.3633 | 0.0242 |
| ***Pola-19*** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.5202 | -0.0452 |
| ***Pola-20*** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5792 | -0.1542 |
| ***Pola-21*** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4560 | 0.0440 |

**62.5 Model Arsitektur 12 – 9 – 1**

Model Arsitektur 12 – 9 – 1, bearti terdiri dari satu lapisan input yang memiliki 12 unit *neuron* yang terhubung langsung dengan lapisan tersembunyi yang memiliki 9 unit *neuron* tersembunyi. Kemudian, 9 unit *neuron* tersembunyi terhubung langsung dengan lapisan *output*  yang memiliki 1 unit neuron. Untuk mengolah data pelatihan 10-9-1 dengan menggunakan *matlab* melalui tahapan-tahapan sebagai berikut :

*>> p=[0.1750 0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500;*

*0.1750 0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750;*

*0.3375 0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125;*

*0.3125 0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250;*

*0.9000 0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000;*

*0.2250 0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250;*

*0.1000 0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000;*

*0.6750 0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125;*

*0.4875 0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625;*

*0.2750 0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875;*

*0.7000 0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750;*

*0.4375 0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250]*

*>> t=[0.5625 0.5250 0.4750 0.4375 0.7000 0.1000 0.1250 0.5625 0.4500 0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000]*

*>> net = newff(minmax(p),[9,1],{'logsig','purelin'},'traingdx');*

*>> net.iw{1,1}*

*>> net.LW{2,1}*

*>> net.b{1}*

*>> net.b{2}*

*>> [y,Pf,Af,e,perf]=sim(net,p,[],[],t)*

*>> net.trainParam.epochs=5000;*

*>> net.trainParam.goal=0.01;*

*>> net.trainParam.lr=0.1;*

*>> net=train(net,p,t);*

TRAINGDX, Epoch 0/5000, MSE 0.272252/0.01, Gradient 1.64713/1e-006

TRAINGDX, Epoch 25/5000, MSE 0.0614728/0.01, Gradient 0.170975/1e-006

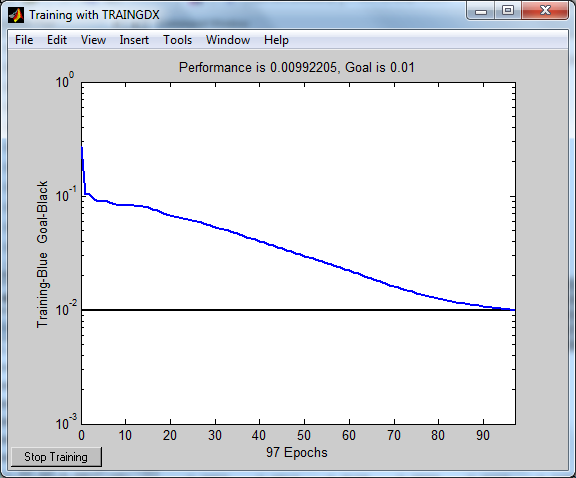
TRAINGDX, Epoch 50/5000, MSE 0.0296766/0.01, Gradient 0.0483948/1e-006

TRAINGDX, Epoch 75/5000, MSE 0.0140348/0.01, Gradient 0.0157855/1e-006

TRAINGDX, Epoch 97/5000, MSE 0.00992205/0.01, Gradient 0.00565601/1e-006

TRAINGDX, Performance goal met.

Dengan perintah di atas maka akan menghasilkan keluaran pada gambar 6.5.



**Gambar 6.5 Antarmuka Grafik Pelatihan Model JST 12-9-1**

Hasil pelatihan mencapai goal pada *epochs* atau iterasi ke-97 dan *error* 0.00992205 Untuk hasil yang lebih rinci dan mengetahui *output* dan *error* dapat dilihat pada tabel 4.5. Dengan menggunakan perintah pada matlab *>>[y,Pf,Af,e,Perf]=sim(net,pn,[],[],tn)*

**Tabel 6.5 Hasil dan Error Data Input Pelatihan dengan Pola 12-9-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** | ***X4*** | ***X5*** | ***X6*** | ***X7*** | ***X8*** | ***X9*** | ***X10*** | ***X11*** | ***X12*** | ***Target*** | ***JST*** | |
| ***Act*** | ***Error*** |
| ***Pola-1*** | 0.1750 | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.3972 | 0.1653 |
| ***Pola-2*** | 0.1750 | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4812 | 0.0438 |
| ***Pola-3*** | 0.3375 | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.3126 | 0.1624 |
| ***Pola-4*** | 0.3125 | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.3559 | 0.0816 |
| ***Pola-5*** | 0.9000 | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.6229 | 0.0771 |
| ***Pola-6*** | 0.2250 | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.2483 | -0.1483 |
| ***Pola-7*** | 0.1000 | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.3017 | -0.1767 |
| ***Pola-8*** | 0.6750 | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.5973 | -0.0348 |
| ***Pola-9*** | 0.4875 | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.3862 | 0.0638 |
| ***Pola-10*** | 0.2750 | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.3752 | -0.1002 |
| ***Pola-11*** | 0.7000 | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.4554 | 0.0571 |
| ***Pola-12*** | 0.4375 | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.2777 | 0.0473 |
| ***Pola-13*** | 0.5625 | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4692 | 0.0308 |
| ***Pola-14*** | 0.5250 | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.5200 | -0.0950 |
| ***Pola-15*** | 0.4750 | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.4765 | -0.0765 |
| ***Pola-16*** | 0.4375 | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.6295 | -0.0170 |
| ***Pola-17*** | 0.7000 | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.5433 | -0.0808 |
| ***Pola-18*** | 0.1000 | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.3718 | 0.0157 |
| ***Pola-19*** | 0.1250 | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.3625 | 0.1125 |
| ***Pola-20*** | 0.5625 | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5884 | -0.1634 |
| ***Pola-21*** | 0.4500 | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4378 | 0.0622 |

**6.3 Perbandingan Model Data Training**

Setelah dilakukan proses pelatihan pada beberapa model yang digunakan pada sistem Jaringan Syaraf Tiruan algoritma *backpropagation* dengan menggunakan *Matlab 6.1*, maka didapatkan data perbandingan dari masing-masing model yang dapat dilihat pada tabel 4.6 berikut ini:

**Tabel 6.6 Perbandingan *Epoch* dan MSE Dari Lima Pola yang Diuji**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Perbandingan | 12-3-1 | 12-5-1 | 12-7-1 | 12-8-1 | 12-9-1 |
| Epoch Pelatihan | 437 | 213 | 918 | 192 | 97 |
| MSE Pengujian | 0.0099831 | 0.0099708 | 0.0099913 | 0.0099986 | 0.0099221 |
| Ranking MSE Pengujian | 3 | 4 | 2 | 1 | 5 |
| Akurasi (100-MSE) | 99.990017 | 99.990029 | 99.990009 | 99.990001 | 99.990078 |
| Ranking Akurasi (100-MSE) | 3 | 2 | 4 | 5 | 1 |

Berdasarkan hasil perbandingan beberapa model yang digunakan pada proses pelatihan sistem Jaringan Syaraf Tiruan algoritma *backpropagation* dengan menggunakan *Matlab 6.1* dapat disimpulkan bahwa model terbaik adalah model 12-9-1 dengan proses perulangan (*epoch*) pada saat pelatihan dengan nilai *epoch = 97*  dan pencapaian *MSE* pada saat pengujian dengan *MSE* = 0.0099221.

**6.4 Pengujian Data Dengan Backpropagation Model 12-9-1**

Setelah didapatkan model terbaik yang digunakan dalam pengujian ini yaitu model arsitektur jaringan 12-9-1, maka langkah selanjutnya yaitu melakukan pengujian terhadap data uji di mana data tersebut adalah pola 22 sampai pola 28 yang ada pada Tabel 5.5.

Tahap yang harus dilakukan dalam pengujian data adalah sama dengan tahap-tahap yang ada pada sub sebelumnya yaitu pengujian dengan model arsitektur jaringan 12–9-1. Setelah semua tahap-tahap yang ada pada sub bab tersebut dilaksanakan, maka selanjutnya adalah mengikuti tahap yang ada di bawah ini :

1. Membuat data *input*

>> p2=[0.2750 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125; 0.5125 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625; 0.3250 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875; 0.5000 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750; 0.4250 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250; 0.4000 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000; 0.6125 0.4625 0.3875 0.4750 0.4250 0.5000 0.4500;

0.4625 0.3875 0.4750 0.4250 0.5000 0.4500 0.2750; 0.3875 0.4750 0.4250 0.5000 0.4500 0.2750 0.5750; 0.4750 0.4250 0.5000 0.4500 0.2750 0.5750 0.4750; 0.4250 0.5000 0.4500 0.2750 0.5750 0.4750 0.4750; 0.5000 0.4500 0.2750 0.5750 0.4750 0.4750 0.4250]

2. Membuat data target atau *ouput*

>> t2=[0.4500 0.2750 0.5750 0.4750 0.4750 0.4250 0.4750]

1. Melihat hasil jaringan syaraf tiruan

*>>[y,Pf,Af,e,Perf]=sim(net,p2,[],[],t2)*

Untuk melihat hasil yang lebih rinci dan mengetahui *output* dan *error* dapat pada data pengujian dilihat pada tabel 6.7.

**Tabel 6.7 Hasil Pengujian dengan Arsitektur Jaringan Backpropagation Model 12-9-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** | ***X4*** | ***X5*** | ***X6*** | ***X7*** | ***X8*** | ***X9*** | ***X10*** | ***X11*** | ***X12*** | ***Target*** | ***JST*** | |
| ***Act*** | ***Error*** |
| ***Pola-22*** | 0.2750 | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0,4497 | 0,0003 |
| ***Pola-23*** | 0.5125 | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0,4465 | -0,1715 |
| ***Pola-24*** | 0.3250 | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0,3723 | 0,2027 |
| ***Pola-25*** | 0.5000 | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0,4736 | 0,0014 |
| ***Pola-26*** | 0.4250 | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0,4037 | 0,0713 |
| ***Pola-27*** | 0.4000 | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 | 0,4313 | -0,0063 |
| ***Pola-28*** | 0.6125 | 0.4625 | 0.3875 | 0.4750 | 0.4250 | 0.5000 | 0.4500 | 0.2750 | 0.5750 | 0.4750 | 0.4750 | 0.4250 | 0.4750 | 0,5185 | -0,0435 |

**6.5 Hasil Pengujian Akurasi Prediksi**

Setelah proses pengujian selesai, data *output* yang dihasilkan jaringan masih dalam bentuk normalisasi, sehingga perlu dilakukan proses denormalisasi data dengan tujuan mengkonversikan kembali hasil keluaran menjadi data *real* yang telah diprediksi. Hasil *output* dari jaringan adalah 0.4497, 0.4465, 0.3723, 0,4736, 0.4037, 0,4313 dan 0.5185 dengan *a* = 194 dan *b =* 130 maka untuk mendenormalisasikannya dapat dilakukan dengan :

1. x1 = ((0.4497 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 158
2. x2 = ((0.4465 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 144
3. x3 = ((0.3723 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 168
4. x4 = ((0.4736 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 160
5. x5 = ((0.4037 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 160
6. x6 = ((0.4313 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 156
7. x7 = ((0.5185 – 0.1000) \* (194 - 130)) / 0.8 + 130 = 160

Berdasarkan pengujian dengan pola arsitektur jaringan 12-9-1 pada pola 22 sampai pola 28 dapat dilihat bahwa persentasi keakuratan mencapai 96,38%. Berdasarkan nilai akurasi yang didapatkan maka dapat disimpulkan bahwasannya nilai prediksi yang didapatkan sudah mendekati dengan nilai yang diharapkan.

**BAB VII**

**KESIMPULAN DAN SARAN**

* 1. **Kesimpulan**

Berdasarkan penelitian dilakukan pada Usaha Songket Silungkang Kota Sawahlunto maka penulis mengambil kesimpulan sebagai berikut:

1. Penelitian yang dilakukan untuk memprediksi produksi Usaha Songket Silungkang Kota Sawahlunto dengan algoritma pembelajaran *Backpropagation* ini menggunakan *Software* Matlab dalam proses pelatihan dan pengujiannya. Data ditransformasi menjadi 28 pola data, dimana pola ke-1 sampai dengan pola ke-21 dijadikan sebagai data training atau pelatihan dan pola ke -22 sampai dengan pola ke-28 dijadikan data uji.
2. Model arsitektur jaringan yang digunakan dalam pelatihan dan pengujian data adalah model 12-9-1 yang terdiri atas satu lapisan input yang memiliki 12 unit *neuron* yang terhubung langsung dengan lapisan tersembunyi yang memiliki 9 unit *neuron* tersembunyi dan 9 unit *neuron* tersembunyi terhubung langsung dengan lapisan *output*  yang memiliki 1 unit neuron
3. Hasil pengujian dengan menggunakan model arsitektur jaringan 12-9-1 didapatkan persentasi keakuratan prediksi produksi usaha songket silungkang mencapai 96,38%
4. Keakurasian dan ketepatan dalam jaringan syaraf tiruan tergantung pada jumlah data yang akan diuji dan pola arsitektur yang dipakai dalam pengujian. Semakin banyak data dan pola yang diuji maka tingkat keakurasian dan ketepatannya akan semakin tinggi pula.
5. Dalam menggunakan metode *backpropagation* pola arsitektur yang dipakai sangat mempengaruhi dalam proses penentuan hasil. Setiap hasil yang diperoleh oleh suatu model arsitektur memungkinkan berbeda dengan hasil yang didapatkan dengan model arsitektur yang lain.

**7.2 Saran-Saran**

Dari hasil penelitian ini maka penulis dapat menyampaikan saran-saran sebagai berikut:

1. Jaringan perlu dilatih dengan jumlah data yang lebih berbervariasi agar hasil prediksi yang akan dihasilkan semakin akurat dengan nilai persentasi keakuratan yang semakin tinggi.
2. Komposisi pembagian data yang berbeda perlu dicoba untuk memperoleh hasil yang lebih optimal.
3. Jumlah lapisan tersembunyi lebih dari satu dan parameter yang digunakan di-*set* lebih tepat lagi agar jaringan dapat mengenali semua pola data input sehingga mampu memprediksi dengan lebih akurat.

# DAFTAR PUSTAKA

[1] Ali M. Al Salihi, Alaa M. Al Lami dan Ali J. Mohammed (2013). *“Prediction of Monthly Rainfall for Selected Meteorogical Stations in Iraq Using Back Propagation Algorithms (Journal of Environmental Science and Technology)”.* Iraq : Al Mustansiriyah University.

[2] Badrul Anwar (2011). **“***Penerapan Jaringan Syaraf Tiruan Backpropagation Dalam Memprediksi Tingkat suku Bunga (Jurnal Saintikom)”.* Medan : STMIK Triguna Dharma.

[3] Ch. Jyosthna Devi, dkk (2012). **“***ANN Approach for Weather Prediction Using Back Propagation (Journal of Engineering Trends and Technology)”.* India : Department of Computer Science and Engineering KLCE Vaddeswaram.

[4] Inggit Prahesti (2013). **“***Implementasi Jaringan Syaraf Tiruan Algoritma Backpropagation Untuk Memprediksi Curah Hujan di Yogyakarta (Naskah Publikasi)***.”** Yogyakarta : STMIK Amikom.

[5] Jayanta Kumar, dkk (2010). **“***Use of Artificial Neural Network in Pattern Recognition (Journal of Software Engineering and Its Applications)***.”** India : Computer Science and Engineering Department Heritage Institute of Technology.

[6] Kusumadewi, Sri (2004). **“***Membangun Jaringan Syaraf Tiruan Menggunakan Matlab dan Excel Link.***”** Yogyakarta : Graha Ilmu.

[7] Maharani Dessy Wuryandari (2012). **“***Jurnal Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation dan Learning Vector Quantization Pada Pengenalan Wajah (Jurnal Komputer dan Informatika)***.”** Bandung : Universitas Komputer Indonesia.

[8] Pandjaitan, Lanny W. (2007). *“Dasar-dasar Komputasi Cerdas***.”** Yogyakarta: Andi Offset.

[9] Puspitaningrum, Diyah (2006). **“***Pengantar Jaringan Syaraf Tiruan***.”**Yogyakarta: Andi Offset.

[10] Sutojo, T., Edy Mulyanto dan Vincent suhartono (2011). **“***Kecerdasan Buatan.”*Yogyakarta: Andi

Offset.