Computer Based Information System Journal

ISSN (Print): 2337-8794 | E- ISSN : 2621-5292
web jurnal : http://ejournal.upbatam.ac.id/index.php/cbis

PROPAGASI BALIK MENENTUKAN PREDIKSI PRODUKSI USAHA SONGKET SILUNGKANG KOTA SAWAHLUNTO

Rima Liana Gema, Devia Kartika
Universitas Putra Indonesia YPTK Padang, Indonesia

INFORMASI ARTIKEL

Diterima Redaksi: Tanggal
Diterbitkan *Online*: Tanggal

KATA KUNCI

Production, Songket, Back Propagation

KORESPONDENSI

E-mail: rimalianagema@gmail.com
devia.kartika11@gmail.com

A B S T R A C T

Artificial Neural Networks is a computational paradigm in which the way it works mimics the biological nerve cell system based on the characteristics of the function of the human brain. One method used in Artificial Neural Networks is a backpropagation algorithm that is widely used, especially in dealing with the problem of identification, prediction, recognition of complex patterns because this method is able to predict what will happen in the future based on patterns that existed in the past. Songket is one of the works of skilled hands of the original Silungkang craftsmen, Sawahlunto City, West Sumatra who have varied and unique patterns and motifs. Sawahlunto City Government, West Sumatra prioritizes the development of Silungkang songket craft business, which is a regional specialty, to enter the export market. At the initial stage, the regional government's priority is to increase the production of crafters by facilitating the development of micro, small and medium enterprises (MSMEs), especially those engaged in songket, to continue to be developed by improving quality and creativity. The city of Sawahlunto can help several parties such as the government, micro, small and medium enterprises in making good handling and decision making efforts to increase the production of Songket Silungkang MSMEs in Sawahlunto City.

I. 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.

II. Kajian Literatur

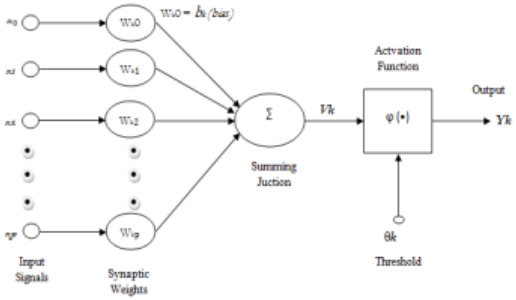
2.1 Jaringan Syaraf Tiruan

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 [1].

Jaringan syaraf tiruan merupakan generalisasi model matematis yang disusun dengan asumsi yang sama seperti jaringan syaraf [2] :

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 1.



Gambar 1. Model Matematis Jaringan Syaraf Tiruan

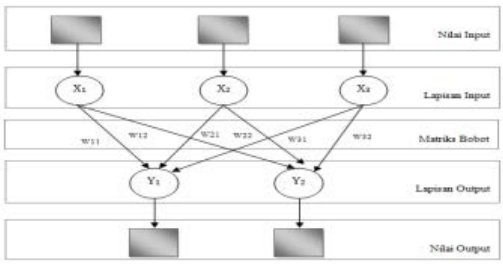
2.1.2 Arsitektur Jaringan

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* [3]. Lapisan-lapisan penyusun jaringan syaraf tiruan dapat dibagi menjadi tiga yaitu [4] :

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.
2. Lapisan tersembunyi
   Node-node di dalam lapisan tersembunyi disebut unit-unit tersembunyi. *Output* dari lapisan ini tidak secara langsung dapat diamati.
3. 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.

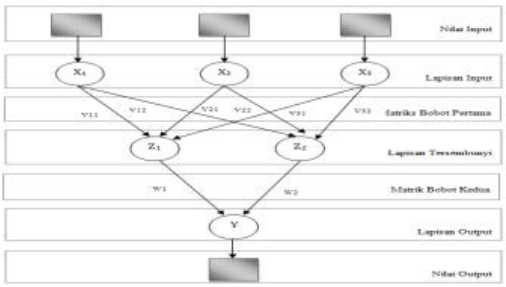
Ada beberapa arsitektur jaringan syaraf tiruan, antara lain [5]:

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*.



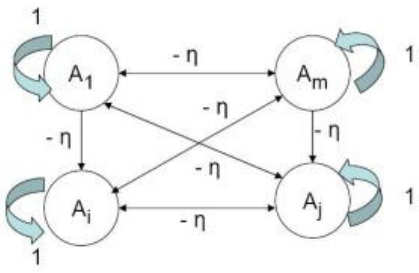
Gambar 2. Jaringan Syaraf Tiruan Dengan Lapisan Tunggal

2. 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 3. Jaringan Syaraf Tiruan Dengan Lapisan Banyak

3. 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 4. Jaringan Syaraf Dengan Lapisan Kompetitif Dengan Bobot –η

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 [6]. Fungsi aktivasi atau fungsi transfer merupakan fungsi yang menggambarkan hubungan antara aktivasi internal (*summation function*) yang mungkin berbentuk linear atau *non*-linear [4].

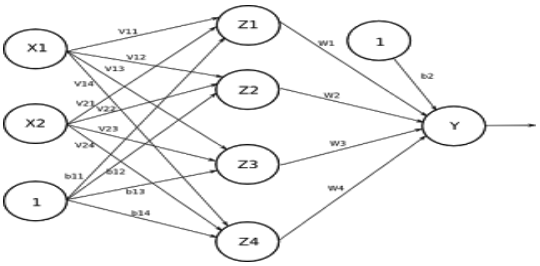
2.1.4 Algoritma Pembelajaran

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 [3].

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 [7]. 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.

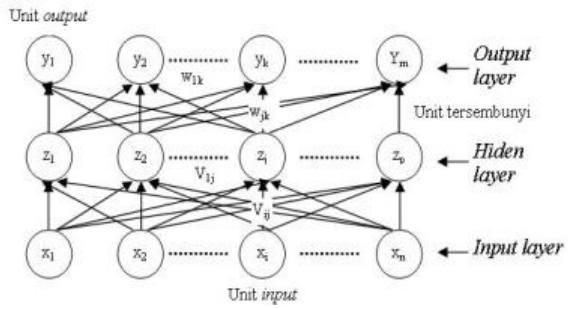


Gambar 6. JST *Backpropagation* Dengan Satu Lapisan Tersembunyi

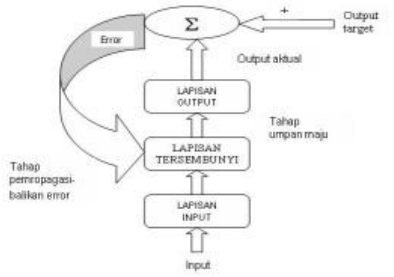
2.2.2 Arsitekur Metode Backpropagation

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* [4]. 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* x_1 sampai unit *input* x_n.
2. Lapisan tersembunyi (minimal satu). Lapisan tersembunyi terdiri dari unit-unit tersembunyi mulai dari unit tersembunyi z_1 sampai z_p.
3. Lapisan *output* (satu buah). Lapisan *output* terdiri dari unit-unit *output* mulai dari unit *output* y_1 sampai unit *output* y_m.



Gambar 7. Arsitektur *Backpropagation*



Gambar 8. Alur Kerja Jaringan *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. [4].

Penggunaan propagasi balik terdiri dari dua tahap yaitu :

a. Tahap belajar atau pelatihan, di mana pada tahap ini pada propagasi balik diberikan sejumlah data pelatihan dan target.
b. 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 [2]. Untuk mentransformasikan seluruh data *real* tersebut, digunakan fungsi sebagai berikut :

$$x^1 = \frac{0.8(x-a)}{b-a} + 0.1 \quad (1)$$

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 [7]:

1. *Initialization*
   Memberikan nilai awal terhadap nilai-nilai yang diperlukan oleh *neural network* seperti *weight, threshold.*
2. *Activation*
   Nilai-nilai yang diberikan pada tahap *Initilization* akan digunakan pada tahap *Activation.*
   Dengan melakukan perhitungan :
   a. Menentukan *aktual output* pada *hidden layer*
   b. Menghitung *aktual output* pada *output layer*
3. *Weight Training*
   Pada tahap *weight training* dilakukan dua kegiatan yaitu :
   a. Menghitung *error gradient* pada *output layer*
   b. Menghitung *error gradient* pada *hidden layer*
4. *Iteration*
   Pada tahap ini dilakukan proses pengulangan sampai mendapat *error* yang minimal.

Berikut ini adalah algoritma pelatihan untuk *backpropagation* dengan sebuah lapisan tersembunyi [4] :

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).
2. *While* kondisi berhenti tidak terpenuhi *do* langkah ke-2 sampai langkah ke-9.
3. Untuk setiap pasangan pola pelatihan, lakukan langkah ke-4 sampai langkah ke-9
   Tahapan Umpan Maju
4. Untuk setiap pasangan pola pelatihan, lakukan langkah ke-4 sampai langkah ke-9
   Tahapan Umpan Maju
5. Setiap unit *input* x_i (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); x_i
6. Pada setiap unit di lapisan tersembunyi z_j (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 x_i *:*
   $Z_j = f\ (Vo_j + \sum X_I\ V_{ij})_{i=1}^{n}$
   kemudian dikirim ke semua unit di lapisan atasnya.
7. Setiap unit di lapisan *output* y_k (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 Z_j bagi lapisan ini :
   $Y_k = f\ (W_{ok} + \sum Z_j\ W_{jk})_{i=1}^{p}$

Tahap Pemprogasibalikan *Error*

8. 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* (δ) dihitung δ_k dikirim ke lapisan dibawahnya dan digunakan untuk menghitung besar koreks bobot dan *bias* (Δw_{jk} dan Δw_{ok}) antara lapisan tersembunyi dengan lapisan *output* :

$\delta_k = (t_k - y_k)\ f'(W_{ok} + \sum Z_j\ W_{jk})_{j=1}^{p}$

$\Delta W_{jk} = \alpha\ \delta_k\ Z_j$

$\Delta W_{0k} = \alpha\ \delta_k$

Tahap Peng-update-an Bobot Bias

9. 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 (δ_j). δ_j kemudian digunakan untuk menghitung besar koreksi bobot dan *bias* (ΔV_{ij} dan ΔV_{oj}) antara lapisan *input* dan lapisan tersembunyi.

$\delta j = (\sum \delta k\ Wjk)_{k=1}^{m}\ f'(V_{oj} + \sum X_i\ V_{ij})_{i=1}^{n}$

$\Delta V_{ij} = \alpha\ \delta_j\ X_i$

$\Delta V_{0j} = \alpha\ \delta_j$

Tahap Perubahan bobot dan *bias*

10. 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 :

$W_{jk}\ (baru) = W_{jk}\ (lama) + \Delta Vij$

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

$V_{ij}\ (baru) = V_{ij}\ (lama) + \Delta Vij$ (6)

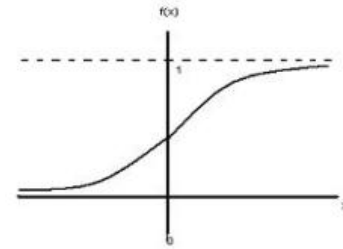
11. Tes kondisi berhenti.
12. Setelah proses pelatihan selesai, nilai-nilai ternormalisasi *output* jaringan (hasil peramalan Jaringan Syaraf Tiruan) harus dikembalikan (denormalisasi) ke nilai aslinya dengan persamaan sebagai berikut:

$$x = \frac{(x^1 - 0.1)(b-a)}{0.8} + a$$

2.2.4 Fungsi Aktivasi Pada Algoritma Pelatihan Backpropagation

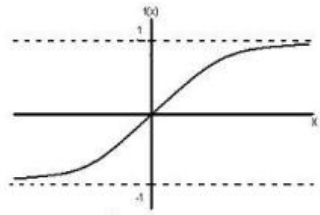
Beberapa fungsi aktivasi yang digunakan di dalam metode *backpropagation* seperti fungsi *sigmoid biner*, *sigmoid bipolar*, dan *tangen hiperbolik*. Karakteristik yang harus dimiliki fungsi aktivasi tersebut adalah kontinu, diferensial dan tidak menurun secara monoton.

1. Fungsi *Sigmoid Biner*
   Fungsi ini umum digunakan. *Range*-nya adalah (0,1) dan didefinisikan sebagai berikut
   $f_1(x) = \frac{1}{1+e^{-x}}$ (4)
   dengan turunan $f_1'(x) = f_1(x)\ (1 - f_1(x))$ (4)



Gambar 9. Fungsi *Sigmoid Biner* Dengan *Range* (0,1)

2. Fungsi *Sigmoid Bipolar*
   Fungsi *Sigmoid Bipolar* memiliki *Range*-nya adalah (-1,1)
   $f_2(x) = 2\ f_1(x) - 1$ (5)
   dengan turunan $f_2'(x) = \frac{1}{2}(1 + f_2(x))\ (1 - f_2(x))$ (6)



Gambar 10. Fungsi *Sigmoid Bipolar* Dengan *Range* (-1,1)

3. Fungsi *Tangen hiperbolik*
   Fungsi *tangen hiperbolik* didefinisikan sebagai berikut :

   $$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (10)$$

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

III. Metodologi

3.1 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 11.

Kerangka kerja dalam melakukan penelitian ini adalah sebagai berikut :

Gambar 11. Kerangka Kerja Penelitian

Pembahasan ini berisi penjelasan tentang kerangka kerja penelitian berdasarkan gambar 11.

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.

2. 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.

3. 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.

4. Merancang Metode JST
   Pada tahap ini, yang perlu dilakukan yaitu mengolah data input serta arsitektur sistem. Adapun tahap-tahapnya adalah sebagai berikut :
   a. 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).
   b. Pembagian data dilakukan dengan membagi data penelitian menjadi data pelatihan dan data pengujian.
   c. Perancangan arsitektur jaringan yang optimum.

d. Memilih dan menggunakan arsitektur jaringan yang optimum.
e. Pemilihan jaringan optimum dan penggunaannya untuk peramalan.

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

6. 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.

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

IV. Pembahasan

a. 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 1. Produksi Songket

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

b. 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 2 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.

a. $x_1 = (0.8 (136 – 130) / (194 - 130)) + 0.1 = 0.1750$
b. $x_2 = (0.8 (136 – 130) / (194 - 130)) + 0.1 = 0.1750$
c. $x_3 = (0.8 (149 – 130) / (194 - 130)) + 0.1 = 0.3375$

d. $x_4 = (0.8 (147 – 130) / (194 - 130)) + 0.1 = 0.3125$

e. $x_5 = (0.8 (194 – 130) / (194 - 130)) + 0.1 = 0.9000$

Tabel 2 merupakan hasil transformasi data pada tabel 2 yang akan dipakai sebagai data pelatihan *Backpropagation.*

Tabel 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 3.

Tabel 3. Hasil Penyusunan Pola Data

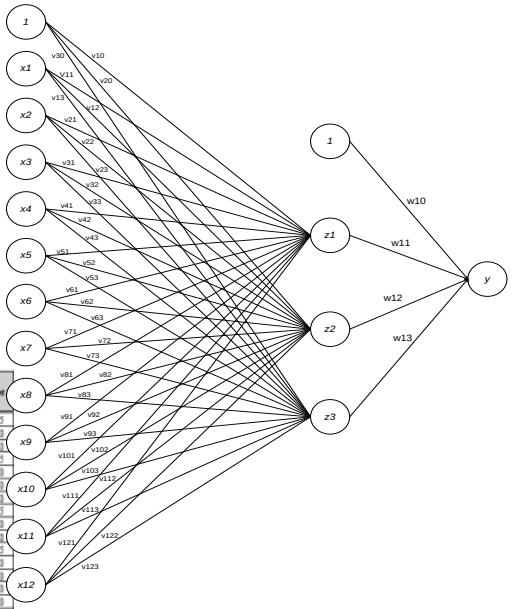
Pola	Data Input												Target
	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	
Pola-1	0.1750	0.1750	0.3375	0.3125	0.9000	0.2250	0.1000	0.6750	0.4875	0.2750	0.7000	0.4375	0.5625
Pola-2	0.1750	0.3375	0.3125	0.9000	0.2250	0.1000	0.6750	0.4875	0.2750	0.7000	0.4375	0.5625	0.5250
Pola-3	0.3375	0.3125	0.9000	0.2250	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-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

c. 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 :

a. Lapisan masukan (*input*) dengan 12 simpul (j_1, j_2,j_3,j_4,j_5,j_n).
b. Lapisan tersembunyi (*hidden*) dengan jumlah simpul ditentukan oleh pengguna (z_1,z_2,z_3).
c. Lapisan keluaran (*output*) dengan 1 simpul (y)

Hasil perancangan Jaringan Syaraf Tiruan Prediksi Produksi Usaha Songket Silungkang dilihat pada gambar 12.



Gambar 12. Arsitektur Jaringan Syaraf Tiruan

Pada gambar 12, 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 :

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

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

d. 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.

e. 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 4 dan data pengujian terdiri dari pola 22 sampai pola 28 dapat dilihat pada tabel 4. Pola data yang digunakan untuk pelatihan jaringan syaraf tiruan dapat dilihat pada Tabel 4.

Tabel 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-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 :

$x_1 = 136$ $x_2 = 136$ $x_3 = 149$ $x_4 = 147$ $x_5 = 194$
$x_6 = 140$ $x_7 = 130$ $x_8 = 176$ $x_9 = 161$ $x_{10} = 144$
$x_{11} = 178$ $x_{12} = 157$

Data tersebut ditransformasikan menjadi :

$j_1 = 0.1750$ $j_2 = 0.1750$ $j_3 = 0.3375$ $j_4 = 0.3125$
$j_5 = 0.9000$ $j_6 = 0.2250$ $j_7 = 0.1000$ $j_8 = 0.6750$
$j_9 = 0.4875$ $j_{10} = 0.2750$ $j_{11} = 0.7000$ $j_2 = 0.4375$

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.

a. Inisiasi bobot (*v*) dan *bias* (*i*) secara acak dari *input* ke lapisan tersembunyi (*hidden* layer) dapat dilihat pada tabel 5.

Tabel 5 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

b. Inisiasi bobot (*w*) dan bias (*l*) secara acak dari lapisan tersembunyi ke *output* dapat dilihat pada tabel 6.

Tabel 6. 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 :

$$z\_net_j = \sum_{i=1}^{2} v_{ji} j_i$$

z_net_1 = v_{10} + (j1*v_{11}) + (j_2*v_{21}) + (j_3*v_{31}) + (j_4*v_{41}) + (j_5*v_{51}) + (j_6*v_{61}) + (j_7*v_{71}) + (j_8*v_{81}) + (j_9*v_{91}) + (j_{10}*v_{101}) + (j_{11}*v_{111}) + (j_{12}*v_{121})

z_net_1=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_net_2 = v_{20} + (j1*v_{12}) + (j_2*v_{22}) + (j_3*v_{32}) + (j_4*v_{42}) + (j_5*v_{52}) + (j_6*v_{62}) + (j_7*v_{72}) + (j_8*v_{82}) + (j_9*v_{92}) + (j_{10}*v_{102})+ (j_{11}*v_{112}) + (j_{12}*v_{122})

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_net_3 = v_{20} + (j1*v_{13}) + (j_2*v_{23}) + (j_3*v_{33}) + (j_4*v_{43}) + (j_5*v_{53}) + (j_6*v_{63}) + (j_7*v_{73}) + (j_8*v_{83}) + (j_9*v_{93}) + (j_{10}*v_{103}) + (j_{11}*v_{113}) + (j_{12}*v_{123})

z_net_3 =5.4724+(0.7564*0.1750)+(1.8556*0.1750)+(2.2755*0.3375)+ (2.0668*0.3125)+(.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

z_1 = sigmoid [-4.4843] = $\frac{1}{1+e^{-z\_net_j}}$ = 0.0112

z_2 = sigmoid [0.4829] = $\frac{1}{1+e^{-z\_net_j}}$ = 0.6184

z_3 = sigmoid [-0.2413] = $\frac{1}{1+e^{-z\_net_j}}$ = 0.4400

Langkah 3.

Menghitung keluaran unit y_k dengan menggunakan persamaan :

$$y\_net_k = w_{k0} \sum_{j=1}^{3} z_j w_{kj}$$

= 0.3443+ (0.0112*0.6924) + (0.6184*0.0503) + (0.4400*-0.5947) = 0.1215 = sigmoid [0.1215] = $\frac{1}{1+e^{-z\_net_j}}$ = 0.5303

Langkah 4.

Menghitung faktor δ di unit keluaran y_k dengan menggunakan persamaan :

$\delta k = (tk - yk)\ f`(y\_net_k) = (tk - yk)\ yk\ (1-yk)$

$\delta k = (0.5625 - 0.5303) * 0.5303\ (1 - 0.5303)\ \delta k = 0.0080$

Hitung suku perubahan bobot W_{jk} (yang akan digunakan untuk merubah bobot W_{jk}) dengan laju pelatihan α = 0.1 dengan persamaan :

$\Delta Wkj = \alpha\ \delta k\ zj$

$\Delta W_{10} = \alpha\ \delta k\ zj$

$= 0.1 * 0.0080 * 1$

$= 0.00080$

$\Delta W_{11} = \alpha\ \delta k\ z_1$

$= 0.1 * 0.0080 * 0.0112$

$= 0.00001$

$\Delta W_{12} = \alpha\ \delta k\ z_2$

$= 0.1 * 0.0080 * 0.6184$

$= 0.00050$

$\Delta W_{13} = \alpha\ \delta k\ z_3$

$= 0.1 * 0.0080 * 0.4400$

$= 0.00035$

Langkah 5.

Hitung penjumlahan kesalahan dari unit tersembunyi dengan persamaan :

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

$\delta\_net_1 = \delta_k w_{kj}$

$\delta\_net_1 = 0.0080 * 0.6924$

$= 0.0055$

$\delta\_net_2 = \delta_k w_{kj}$

$\delta\_net_2 = 0.0080 * 0.0503$

$= 0.0004$

$\delta\_net_3 = \delta_k w_{kj}$

$\delta\_net_3 = 0.0080 * -0.5947$

$= -0.0048$

Kemudian hitung faktor kesalahan δ di unit tersembunyi dengan persamaan:

$$\delta_j = \delta_{net_j} f'\left(z_{net_j}\right) = \delta_{net_j} z_j\ (1 - z_j)$$

$\delta_1 = 0.0055 * 0.0112 * (1 - 0.0112)$

$\delta_1 = -0.0001$

$\delta_2 = 0.0004 * 0.6184 * (1 - 0.6184)$

$\delta_2 = 0.0001$

$\delta_3 = -0.0048 * 0.4400 * (1 - 0.4400)$

$\delta_3 = -0.0012$

Hitung suku perubahan bobot ke unit tersembunyi dengan persamaan:

$\Delta vji = \alpha\ \delta_j\ j_i$ di mana α = 0.1

$\Delta v_{11} = 0.1 * (0.0001) * 0.1750 = 0.0000$

$\Delta v_{21} = 0.1 * (0.0001) * 0.1750 = 0.0000$

$\Delta v_{31} = 0.1 * (0.0001) * 0.3375 = 0.0000$

$\Delta v_{41} = 0.1 * (0.0001) * 0.3125 = 0.0000$

$\Delta v_{51} = 0.1 * (0.0001) * 0.9000 = 0.0000$

$\Delta v_{61} = 0.1 * (0.0001) * 0.2250 = 0.0000$

$\Delta v_{71} = 0.1 * (0.0001) * 0.1000 = 0.0000$

$\Delta v_{81} = 0.1 * (0.0001) * 0.6750 = 0.0000$

$\Delta v_{91} = 0.1 * (0.0001) * 0.4875 = 0.0000$

$\Delta v_{101} = 0.1 * (0.0001) * 0.2750 = 0.0000$

$\Delta v_{111} = 0.1 * (0.0001) * 0.7000 = 0.0000$

$\Delta v_{121} = 0.1 * (0.0001) * 0.4375 = 0.0000$

Δv_{12} = 0.1 * (0.0001) * 0.1750 = 0.0000
Δv_{22} = 0.1 * (0.0001) * 0.1750 = 0.0000
Δv_{32} = 0.1 * (0.0001) * 0.3375 = 0.0000
Δv_{42} = 0.1 * (0.0001) * 0.3125 = 0.0000
Δv_{52} = 0.1 * (0.0001) * 0.9000 = 0.0000
Δv_{62} = 0.1 * (0.0001) * 0.2250 = 0.0000
Δv_{72} = 0.1 * (0.0001) * 0.1000 = 0.0000
Δv_{82} = 0.1 * (0.0001) * 0.6750 = 0.0000
Δv_{92} = 0.1 * (0.0001) * 0.4875 = 0.0000
Δv_{102} = 0.1 * (0.0001) * 0.2750 = 0.0000
Δv_{112} = 0.1 * (0.0001) * 0.7000 = 0.0000
Δv_{122} = 0.1 * (0.0001) * 0.4375 = 0.0000
Δv_{13} = 0.1 * (-0.0002) * 0.1750 = 0.0000
Δv_{23} = 0.1 * (-0.0002) * 0.1750 = 0.0000
Δv_{33} = 0.1 * (-0.0002) * 0.3375 = 0.0000
Δv_{43} = 0.1 * (-0.0002) * 0.3125 = -0.0000
Δv_{53} = 0.1 * (-0.0002) * 0.9000 = -0.0001
Δv_{63} = 0.1 * (-0.0002) * 0.2250 = 0.0000
Δv_{73} = 0.1 * (-0.0002) * 0.1000 = 0.0000
Δv_{83} = 0.1 * (-0.0002) * 0.6750 = -0.0001
Δv_{93} = 0.1 * (-0.0002) * 0.4875 = -0.0001
Δv_{103} = 0.1 * (-0.0002) * 0.2750= 0.0000
Δv_{113} = 0.1 * (-0.0002) * 0.7000 = -0.0001
Δv_{123} = 0.1 * (-0.0002) * 0.4375= -0.0001

Sehingga didapatkan suku perubahan bobot ke unit tersembunyi yang dapat dilihat pada tabel 7 di bawah ini.

Tabel 7. Suku Perubahan Bobot Unit Tersembunyi (Δv)

Δv	*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 :

W_{kj} *(baru)* = w_{kj} *(lama)* + Δw_{kj}

W_1 *(baru)* = w_1*(lama)* + Δw_1

= 0.6924+ 0.00001 = 0.6924

W_2 *(baru)* = w_2*(lama)* + Δw_2

= 0.0503 + 0.00050 = 0.0508

W_3 *(baru)* = w_3*(lama)* + Δw_3

= -0.5947 + 0.00035 = -0.5943

W_o *(bias baru)* = w_0*(lama)* + Δw_0

= 0.3443 + 0.00080 = 0.3451

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

$V_{[oj]}$:$\Delta V_{[o,j]} = \alpha * \delta_j$

$\Delta V_{[o,1]}$= 0.1 * 0.0001 = -0.00001

$\Delta V_{[o,2]}$= 0.1 * 0.0001 = 0.00001

$\Delta V_{[o,3]}$= 0.1 * -0.0012= -0.00012

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

$V_{[o,j]}(baru) = V_{[o,j]}(lama) + \Delta V_{[o,j]}$

$V_{[o,1]}$= -3.3578 + 0.00001 = -3.3578

$V_{[o,2]}$= -3.1172 + 0.00001 = -3.1172

$V_{[o,3]}$= 5.4724 + (-0.00012) = 5.4723

Hitung perubahan bobot garis menuju ke unit tersembunyi dengan persamaan :

V_{kj} *(baru)* = v_{kj}*(lama)* + Δv_{kj}

V_{11} *(baru)* = v_{11}*(lama)* + Δv_{11}

= 3.4759 + 0.0000 = 3.4759

V_{21} (baru) = v_{21}(lama) + Δv_{21}

= -0.1082 + 0.0000 = -0.1082

V_{31} (baru) = v_{31}(lama) + Δv_{31}

= -0.3362 + 0.0000 = -0.3362

V_{41} (baru) = v_{41}(lama) + Δv_{41}

= -0.4270 + 0.0000 = -0.4270

V_{51} (baru) = v_{51}(lama) + Δv_{51}

= 3.2572 + 0.0000 = 3.2572

V_{61} (baru) = v_{61}(lama) + Δv_{61}

= -0.9708 + 0.0000 = -0.9708

V_{71} (baru) = v_{71}(lama) + Δv_{71}

= -0.9239 + 0.0000 = -0.9239

V_{81} (baru) = v_{81}(lama) + Δv_{81}

= -1.5149 + 0.0000 = -1.5149

V_{91} (baru) = v_{91}(lama) + Δv_{91}

= -3.7180 + 0.0000 = -3.7180

V_{101} (baru) = v_{101}(lama) + Δv_{101}

= 1.0686 + 0.0000 = 1.0686

V_{111} (baru) = v_{91}(lama) + Δv_{91}

= -4.9907 + 0.0000 = -4.9907

V_{121} (baru) = v_{101}(lama) + Δv_{101}

= 4.4459 + 0.0000 = 4.4459

V_{12} (baru) = v_{12}(lama) + Δv_{12}

= -1.9049 + 0.0000 = -1.9049

V_{22} (baru) = v_{22}(lama) + Δv_{22}

= 2.7724 + 0.0000 = 2.7724

V_{32} (baru) = v_{32}(lama) + Δv_{32}

= -3.4114 + 0.0000 = -3.4114

V_{42} (baru) = v_{42}(lama) + Δv_{42}

= 0.8178 + 0.0000 = 0.8178

V_{52} (baru) = v_{52}(lama) + Δv_{52}

= 1.6877 + 0.0000 = 1.6877

V_{62} (baru) = v_{62}(lama) + Δv_{62}

= 4.1138 + 0.0000 = 4.1138

V_{72} (baru) = v_{72}(lama) + Δv_{72}

= 3.7187 + 0.0000 = 3.7187

V_{82} (baru) = v_{82}(lama) + Δv_{82}

= 2.9584 + 0.0000 = 2.9584

V_{92} (baru) = v_{92}(lama) + Δv_{92}

= -2.8079 + 0.0000 = -2.8079

V_{102} (baru) = v_{102}(lama) + Δv_{102}

= -2.1521 + 0.0000 = -2.1521

V_{112} (baru) = v_{91}(lama) + Δv_{91}

= 2.3313 + 0.0000 = 2.3313

V_{122} (baru) = v_{101}(lama) + Δv_{101}

= -0.3212 + 0.0000 = -0.3212

V_{13} (baru) = v_{13}(lama) + Δv_{13}

= 0.7564 + 0.0000 = 0.7564

V_{23} (baru) = v_{23}(lama) + Δv_{23}

= 1.8556 + 0.0000 = 1.8556

V_{33} (baru) = v_{33}(lama) + Δv_{33}

= 2.2755 + 0.0000 = 2.2755

V_{43} (baru) = v_{43}(lama) + Δv_{43}

= 2.0668 + 0.0000 = 2.0668

V_{53} (baru) = v_{53}(lama) + Δv_{53}

= -2.2919 + (-0.0001) = -2.2920

V_{63} (baru) = v_{63}(lama) + Δv_{63}

= 3.9354 + 0.0000 = 3.9354

V_{73} (baru) = v_{73}(lama) + Δv_{73}

= -4.1733 + 0.0000 = -4.1733

V_{83} (baru) = v_{83}(lama) + Δv_{83}

= -4.6267 + (-0.0001) = -4.6268

V_{93} (baru) = v_{93}(lama) + Δv_{93}

= -2.8439 + (-0.0001) = -2.8440

V_{103} (baru) = v_{103}(lama) + Δv_{103}

= -2.8431 + 0.0000 = -2.8431

V_{113} (baru) = v_{91}(lama) + Δv_{91}

= -0.5183 + (-0.0001) = -0.5184

V_{123} (baru) = v_{101}(lama) + Δv_{101}

= -0.7679 + (-0.0001) = -0.7680

Setelah selesai, akan didapatkan tabel nilai bobot baru pada dari *input layer* ke *hidden layer* seperti pada tabel 8

Tabel 8. 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 :

$$x = \frac{(0.1215 - 0.1)(194 - 130)}{0.8} + 130$$

= 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).

f. 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 10-3-1 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.

g. Pola Pelatihan Dengan Model Arsitektur 12-9-1

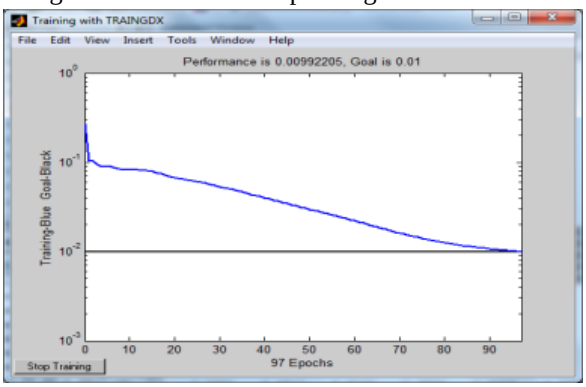
Berdasarkan hasil penelitian yang telah dilakukan oleh peneliti dengan judul *Algoritma Propagasi Balik Dalam Pencarian Pola Training Terbaik Untuk Menentukan Prediksi Produksi Usaha Songket Silungkang Dengan Menggunakan Matlab.* Model arsitektur 12 – 9 – 1 terdiri atas satu lapisan input yang memiliki 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. Adapun tahapan pengolahan data pelatihan pola 12-9-1 yang dilakukan adalah 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 13.

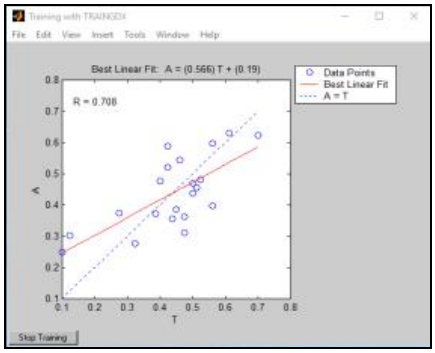


Gambar 13 Antarmuka Grafik Pelatihan Model JST 12-9-1

Hasil pelatihan mencapai goal pada *epochs* ke-97 dan *error* 0.00992205. Untuk melihat analisis regresi antara respon jaringan dan target yang ditentukan pada pola pelatihan yang disediakan oleh postreg dapat dilihat pada gambar 14, dan hasil yang lebih rinci untuk mengetahui *output* dan *error* dapat dilihat pada tabel 9. Adapun perintah matlab untuk menampilkannya adalah sebagai berikut:

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

>>[m1,y1,r1] = postreg (y,t)



Gambar 14. Output Grafis Pada Postreg

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

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	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

h. Pengujian Dengan Model Arsitektur 12-9-1

Setelah dilakukan pelatihan pola, maka langkah selanjutnya adalah melakukan pengujian terhadap data uji di mana data tersebut adalah pola 22 sampai pola 28 yang ada pada Tabel 10. Setelah semua tahap-tahap yang ada pada sub bab 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 0.4750 0.4250]

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

2. 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 10.

Tabel 10. 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

Tabel 11. Hasil Pengujian Dengan Arsitektur Jaringan Backpropagation Model 12-9-1

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	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

i. Hasil Pengujian Akurasi Prediksi

Setelah proses pengujian selesai, data *output* yang dihasilkan jaringan masih dalam bentuk normalisasi, sehingga harus dilakukan proses denormalisasi data dengan tujuan mengkonversikan kembali *output* 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 :

a. $x_1 = ((0.4497 – 0.1000) * (194-130)) / 0.8 + 130 = 158$
b. $x_1 = ((0.4465 – 0.1000) * (194-130)) / 0.8 + 130 = 158$
c. $x_1 = ((0.3723 – 0.1000) * (194-130)) / 0.8 + 130 = 152$
d. $x_1 = ((0.4736 – 0.1000) * (194-130)) / 0.8 + 130 = 160$
e. $x_1 = ((0.4037 – 0.1000) * (194-130)) / 0.8 + 130 = 154$
f. $x_1 = ((0.4313 – 0.1000) * (194-130)) / 0.8 + 130 = 157$
g. $x_1 = ((0.5185 – 0.1000) * (194-130)) / 0.8 + 130 = 163$

Berdasarkan hasil pengujian dengan pola arsitektur jaringan 12-9-1 pada pola 22 sampai pola 28 dapat dilihat bahwa persentasi keakuratan prediksi produksi usaha songket silungkang mencapai 96,38%. Diharapkan dengan hasil prediksi yang didapatkan ini dapat membantu pihak UKM serta pihak yang berkepentingan dalam mengambil keputusan yang terbaik dalam hal yang berhubungan dengan produksi Songket Silungkang Kota Sawahlunto.

V. Kesimpulan

Berdasarkan penelitian tentang Propagasi Balik Menentukan Prediksi Produksi Usaha Songket Silungkang Kota Sawahlunto yang telah dilakukan, adapun kesimpulan yang dapat diambil adalah 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.

Ucapan Terima Kasih

Terima Kasih kepada Dipa Direktorat Riset dan Pengabdian Masyarakat. Direktorat Jendral Penguat Riset dan Pengembangan. Kementerian Riset, Teknologi dan Pendidikan Tinggi.

Daftar Pustaka

[1] P. Inggit, *Implementasi Jaringan Syaraf Tiruan Algoritma Backpropagation Untuk Memprediksi Curah Hujan*. Yogyakarta: STMIK Amikom.

[2] J. J. Siang, *Jaringan Syaraf Tiruan & Pemrogramannya*. Yogyakarta: Andi Offset, 2009.

[3] E. M. and V. suhartono Sutojo, T., *Kecerdasan Buatan*. Yogyakarta: Andi Offset, 2011.

[4] D. Puspitaningrum, *Pengantar Jaringan Syaraf Tiruan*. Yogyakarta: Andi Offset, 2006.

[5] S. Kusumadewi, *Artificial Intelligence Teknik dan Aplikasinya*. Yogyakarta: Graha Ilmu, 2003.

[6] Lanny W Pandjaitan, *Dasar-dasar Komputasi Cerdas*. Yogyakarta: Andi Offset, 2007.

[7] B. Anwar, "Penerapan algoritma jaringan syaraf tiruan backpropagation dalam memprediksi tingkat suku bunga bank," *J. SAINTIKOM*, 2011.