ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 05 Nomor 02, Desember Tahun 2017

STIKI

SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
Jl. Raya Tidar 100 Malang, 65146
Telp. (0341)560823, Fax (0341)562525

J-INTECH

Volume 05 Nomor 02, Desember Tahun 2017

ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 05 Nomor 02, Desember 2017

LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
STIKI
SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
Jl. Raya Tidar 100, Malang; Phone: 0341-560823; Fax: 0341-562525; http://www.stiki.ac.id; mail@stiki.ac.id

PENGANTAR REDAKSI

J-INTECH merupakan jurnal yang diterbitkan oleh Sekolah Tinggi Informatika dan Komputer Indonesia Malang guna mengakomodasi kebutuhan akan perkembangan Teknologi Informasi serta guna mensukseskan salah satu program DIKTI yang mewajibkan seluruh Perguruan Tinggi untuk menerbitkan dan mengunggah karya ilmiah mahasiswanya dalam bentuk terbitan maupun jurnal online.

Pada edisi ini, redaksi menampilkan beberapa karya ilmiah mahasiswa yang mewakili beberapa mahasiswa yang lain, yang dianggap cukup baik sebagai media pembelajaran bagi para lulusan selanjutnya.

Tentu saja diharapkan pada setiap penerbitan memiliki nilai lebih dari karya ilmiah yang dihasilkan sebelumya sehingga merupakan nilai tambah bagi para adik kelas maupun pihak-pihak yang ingin studi atau memanfaatkan karya tersebut selanjutnya.

Pada kesempatan ini kami juga mengundang pihak-pihak dari PTN/PTS lain sebagai kontributor karya ilmiah terhadap jurnal J-INTECH, sehingga Perkembangan IPTEK dapat dikuasai secara bersama- sama dan membawa manfaat bagi institusi masing-masing.

Akhir redaksi berharap semoga dengan terbitnya jurnal ini membawa manfaat bagi para mahasiswa, dosen pembimbing, pihak yang bekerja pada bidang Teknologi Informasi serta untuk perkembangan IPTEK di masa depan.

REDAKSI

ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 05 Nomor 02, Desember 2017

DAFTAR ISI

ISSN: 2303-1425 E-ISSN: 2580-720X

J-INTECH

Journal of Information and Technology

Volume 05 Nomor 02, Desember 2017

Aplikasi Pendeteksi Kelayakan Telur Menggunakan Metode *Backpropagation* dan *Thresholding*

Harman Tunggorono
Program Studi Teknik Informatika, Sekolah Tinggi Informatika & Komputer Indonesia (STIKI) Malang
Email: harmantunggorono@gmail.com

ABSTRAK

Perkembangan teknologi yang semakin maju membuat kebutuhan seseorang dalam hal kecepatan dalam penyelesaian masalah sangat dibutuhkan. Salah satunya yakni untuk mendeteksi kelayakan telur menggunakan pembiasan sinar pada telur (*candling*). Jaringan syaraf tiruan dapat diterapkan dalam berbagai bidang kebutuhan, salah satunya adalah untuk mengklasifikan suatu citra telur yang baik dan yang buruk. Algoritma *Thresholding* dan Algoritma *Backpropagation* dipilih karena algoritma tersebut memiliki beberapa unggulan. Penggunaan *threshold* yaitu untuk membuat input jaringan syaraf tiruan menjadi 2 dan selanjutnya dilakukan pelatihan menggunakan algoritma *Backropagation.* Dari hasil pengujian sistem hasil yang didapat sangat akurat dengan mengubah parameter *hidden node* dengan jumlah banyak atau *min error* dengan nilai yang sangat kecil.

Kata Kunci: *Pendeteksi Kelayakan Citra Telur, Jaringan Syaraf Tiruan Backpropagation, Thresholding.*

1. PENDAHULUAN

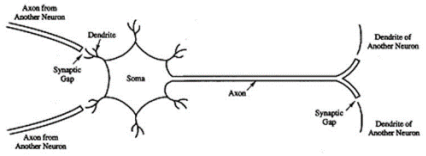
Telur adalah sumber protein hewani yang sangat baik bagi regenerasi sel yang terdapat dalam tubuh manusia. Terdapat banyak jenis telur ayam di dunia ini, dan salah satunya adalah telur ayam *leghorn* yang akan dipakai sebagai obyek dalam penelitian tugas akhir ini. Telur ayam leghorn/telur ayam dalam negeri adalah telur yang dihasilkan dari ayam leghorn atau ayam ternak. Pada umumnya telur tersebut dijual di toko-toko dengan harga yang relatif murah dan terjangkau. Ada perbedaan yang mencolok pada telur ayam leghorn dengan telur ayam yang lainnya yaitu dari segi ukuran relatif lebih besar.

Dalam memilih telur ayam leghorn untuk dikonsumsi tidak boleh sembarangan karena dari segi kelayakan akan sangat berpengaruh terhadap nilai gizi yang terkandung di dalamnya jika dikonsumsi bagi kesehatan manusia. Salah satu cara manual untuk mendeteksi telur apakah layak untuk dikonsumsi atau tidak, ialah dengan melihat pembiasan cahaya telur yang telah disinari atau biasa disebut sebagai *candling*. Jika dilakukan penyinaran dan cahaya tembus maka telur tersebut masih bagus dan layak untuk dikonsumsi, tetapi jika tidak tembus atau redup maka telur tersebut tidak layak untuk dikonsumsi. Pendeteksian tersebut seringkali kurang akurat dan berbeda-beda dalam penentuannya karena berbedanya persepsi pada setiap orang dalam menentukan kelayakan telur tersebut, dan yang bisa mendeteksi kelayakan secara *candling* hanya bisa dilakukan oleh seorang yang ahli atau pakar di bidang tersebut.

Untuk itu diusulkan sebuah usulan solusi terkait dengan permasalahan yang terdapat di latar belakang tersebut, yaitu dengan membuat sebuah sistem aplikasi otomatis yang dapat menentukan kelayakan sebuah telur ayam leghorn. Sistem tersebut dibangun menggunakan metode jaringan syaraf tiruan *backpropagation*. Cara kerja sistem tersebut yaitu mengklasifikasi telur yang baik dan buruk. Beberapa tahapan agar sistem dapat mengklasifikasi telur yang baik dan buruk yaitu dengan cara penentuan target citra telur baik (=1) dan buruk (=0). Langkah selanjutnya dilakukan pengambilan citra telur baik dan buruk, citra tersebut digunakan sebagi *input* jaringan dan jaringan tersebut akan dilatih menggunakan citra yang sudah diambil (baik dan buruk), pada tahap pelatihan ini dilakukan 3 langkah proses yakni umpan maju (*feed forward*), umpan mundur (*backpropagation of error*), dan pembaharuan bobot dan bias. Pada pengujian sistem, sistem tersebut hanya menjalankan langkah umpan maju (*feed forward*) dengan menggunakan jaringan yang sudah dilatih.

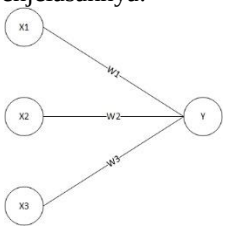
Jaringan syaraf manusia terdiri atas sel-sel yang disebut neuron. Ada tiga komponen utama neuron yang fungsinya dapat dianalogikan dengan yang terjadi pada jaringan syaraf tiruan, yaitu dendrit, soma, dan akson. Dendrit akan menerima sinyal-

sinyal dari neuron lain. Sinyal tersebut merupakan *input* listrik yang ditransmisikan melalui *synaptic gap* melalui proses kimia. Sedangkan, soma atau badan sel akan menjumlah sinyal-sinyal *input* yang masuk jika ada *input* yang masuk, sel akan aktif dan akan mentransmisikan sinyal ke sel lain melalui akson dan *synaptic gap* (Prof.Dr.Ir.Kuswara Setiawan, M.T, 2003, p. 2), berikut adalah penjelasannya :



Gambar 1. Susunan Neuron Biologis

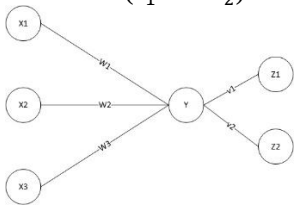
Jaringan syaraf tiruan merupakan sistem pengolahan informasi yang memiliki karakter seperti jaringan syaraf biologis, yaitu jaringan otak manusia. Pada jaringan syaraf tiruan, terdapat istilah neuron atau yang sering disebut sebagai unit, sel, atau node. Setiap neuron terhubung dengan neuron-neuron lain melalui *layer* (*layer* terdiri dari banyak neuron) dengan bobot tertentu. Bobot melambangkan informasi yang digunakan oleh jaringan untuk menyelesaikan persoalan. Pada jaringan syaraf biologis, bobot tersebut dapat dianalogikan sebagai aksi pada proses kimia yang terjadi pada *synaptic gap*. Sedangkan, setiap neuron mempunyai *internal state* atau yang disebut sebagai aktivasi. Aktivasi tersebut merupakan fungsi dari *input* yang diterima. Suaru neuron akan mengirim sinyal ke neuron-neuron lain, tetapi pada suatu saat hanya ada satu sinyal yang dapat dikeluarkan walaupun sinyal tersebut ditransmisikan pada beberapa neuron lain. Berikut adalah penjelasannya:



Gambar 2. Jaringan Syaraf Tiruan Layer Tunggal

Pada Gambar 1 terdapat neuron (y) menerima *input* dari neuron $(x_1, x_2, \text{dan } x_3)$. Aktivasi atau *output* sinyal adalah $(x_1, x_2, \text{dan } x_3)$. Bobot yang menghubungkan neuron $(x_1, x_2, \text{dan } x_3)$ ke neuron (y) adalah $(w_1, w_2, \text{dan } w_3)$. Maka, *input* jaringan y_in pada neuron y adalah jumlah sinyal berbobot dari masing-masing neuron $(x_1, x_2, \text{dan } x_3)$ dimana y_in = $w_1x_1 + w_2x_2 + w_3x_3$. Aktivasi y dari neuron (y) merupakan fungsi jaringan *input* yaitu y = f(y_in), misalnya dengan persamaan f(y) = $\frac{1}{1+\exp(-y\_in)}$ atau dengan fungsi aktivasi lainnya.

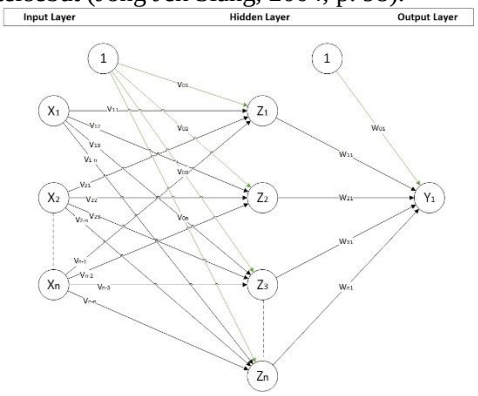
Jika neuron (y) terhubung lebih lanjut, misalnya ke neuron (z_1 dan z_2)



Gambar 3. Jaringan Syaraf Tiruan 3 Layer (input layer, hidden layer, output layer)

maka neuron (y) mengirim sinyal y ke neuron (z_1 dan z_2).Tetapi, nilai yang diterima z_1 dan z_2 berlainan, tergantung bobotnya. Misalnya bobot z_1 dan z_2 adalah v_1 dan v_2, maka aktivasi z_1 dan z_2 akan dihasilkan lewat fungsi aktivasi yang merupakan fungsi dari *input* neuron z_1 dan z_2 (Prof.Dr.Ir.Kuswara Setiawan, M.T, 2003, p. 2).

Jaringan syaraf pada metode *backpropagation* terdiri dari 3 lapis yaitu *input layer*, *hidden layer*, *output layer*. Pada *input layer* terdiri dari variabel masukan sel syaraf (*neuron*), lapisan tersebut digunakan untuk menampung *n* variabel yaitu X1 sampai Xn sedangkan pada *hidden layer* adalah sebagai variabel pemisah (Z1 sampai dengan Zn) dimana pada *hidden layer* tersebut untuk mengkategorikan sel syaraf (*neuron*) dan *output layer* (Y1 sampai dengan Yn) adalah hasil dari peng-kategorian tersebut (Jong Jek Siang, 2004, p. 98).



Gambar 4. Arsitektur Jaringan Backpropagation

Keterangan:

1. X= Input Layer
2. Z= Hidden Layer
3. Y= Output Layer
4. 1 = Bias
5. V= Bobot lapisan yang tersembunyi
6. V0= Bias lapisan tersembunyi
7. W=Bobot lapisan keluaran
8. W0= Bias lapiran keluaran

Pada intinya, pelatihan menggunakan metode *backpropagation* terdiri atas tiga langkah, yaitu:

1. Data masukan ke *input* jaringan (*feedforward*).
2. Perhitungan dan propagasi balik dari eror yang bersangkutan.
3. Pemaharuan (*adjustment*) bobot dan bias.

Menurut Prof. Dr. Ir. Kuswara Setiawan, M.T [Paradigma Sistem Cerdas, 2003, p. 20] pada umpan maju (*feedforward*), setiap unit *input* (Xi) akan menerima sinyal *input* dan akan menyebarkan sinyal tersebut pada tiap *hidden* unit (Zi). Setiap *hidden* unit kemudian menghitung aktivasinya dan mengirim sinyal (Zj) ke tiap unit *output*. Kemudian, setiap unit *output* (Yk) untuk menghasilkan respons terhadap *input* yang diberikan jaringan.

Saat proses pelatihan (*training*), setiap unit *output* membandingkan aktivasinya (Yk) dengan nilai target (*desired output*) untuk menentukan besarnya *error*. Berdasarkan *error* tersebut dihitung faktor δk. Faktor δk digunakan untuk mendistribusikan *error* dari *output* kembali ke *layer* sebelumnya. Dengan cara yang sama, faktor δj juga dihitung pada *hidden* unit Zj. Faktor δk digunakan untuk memperbaharui bobot antara *hidden layer* dan *input layer*.

Setelah semua faktor δ ditentukan, bobot untuk semua *layer* di *adjust* secara bersamaan. Pembaharuan bobot Wjk (dari *hidden* unit Zj ke unit *output* Yk) dilakukan berdasarkan faktor δk dan aktivasi zj dan *hidden* unit Zj. Sedangkan, pembaharuan bobot vij (dari *input* unit Xi ke *hidden* unit Zj) dilakukan berdasarkan faktor δj dan aktivasi Xi dari *input*. Berikut adalah notasi-notasi yang akan digunakan pada algoritma pelatihan:

1. X = Data *training* untuk *input*
2. T = Data *training* untuk *output*/ target/ *desired output*
3. α= *Learning rate* yaitu sebagai parameter yang mengontrol perubahan bobot selama pelatihan. Jika *learning rate* besar maka jaringan akan semakit cepat belajar tapi hasilnya tidak akurat. *Learning rate* biasanya dipilih antara 0 dan 1.
4. X_i = Unit *input* ke-i. Untuk unit *input*, sinyal yang masuk dan keluar dilambangkan dengan variabel yang sama yaitu Xi.
5. Z_i= *Hidden* unit ke-j. Sinyal *input* pada Zj dilambangkan dengan y_inj. Sinyal *output* (aktivasi) dan Zj dilambangkan dengan zj).
6. V_{0J} = Bias untuk *output* ke-k.
7. V_{ij}= Bobot antara unit *input* ke-i dan *hidden* unit ke-j.
8. Y_k = Unit *output* ke-k.Sinyal *input* ke Yk dilambangkan y_ink. Sinyal *output* (aktivasi) untuk Yk dilambangkan dengan yk.
9. W_{jk} = Bobot antara *hidden* unit ke-j dan unit *output* ke-k.
10. δ_k = Faktor koreksi *error* untuk bobot Wjk
11. δ_i = Faktor koreksi *error* untuk bobot Vij.

Menurut Prof. Dr. Ir. Kuswara Setiawan, M.T [Paradigma Sistem Cerdas, 2003, p. 21] Training session adalah pelatihan untuk pembelajaran sistem. Secara lebih detail langkah-langkah pelatihan pada metode *backpropagation* yaitu sebagai berikut:

Langkah 1

Inisialisasi bobot dan bias. Pada bobot dan bias tersebut dapat diberikan angka yang acak dan angka tersebut berkisar antara 0 dan 1 atau -1 (bias positif atau negatif).

Langkah 2

Jika stopping condition belum terpenuhi maka lakukan langkah 2 – langkah 9.

Umpan maju (feedforward)

Langkah 3

Pada langkah 3 ini, setiap unit *input* (Xi, i = 1, ... ,n) menerima sinyal *input* xi, dan menyebarkan sinyal tersebut pada seluruh unit atau *hidden* unit. Perlu diketahui bahwa *input* xi yang dipakai adalah data ini *input training* yang telah diskalakan. Pertama, *input* yang kemungkinan akan dipakai oleh sistem terlebih dahulu dicari nilai terendah dan nilai tertingginya. Selanjutnya, skala yang akan digunakan tergatung dari fungsi aktivasinya. Jika yang dipakai adalah fungsi sigmoid biner yang mempunyai nilai terendah = 0 dan nilai

tertinggi =1, maka nilai *input* terendah dianggap = 0 dan nilai tertinggi dianggap 1. Nilai diantaranya bervariasi antara 0 dan 1. Sedangkan yang dipakai adalah fungsi sigmoid bipolar maka range nilainya juga bervariasi mulai -1 sampai dengan 1.

Langkah 4

Setiap hidden unit (Zj, j = 1, ... ,p), akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot dan termasuk juga dengan biasnya,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} X_i V_{ij}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari hidden unit yang bersangkutan zj = f(z_inj) kemudian mengirim sinyal *output* tersebut ke seluruh unit pada unit *output*.

Langkah 5

Setiap unit *output* (Yk, k = 1, ... ,m) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot termasuk juga dengan biasnya,

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari unit *output* yang bersangkutan yk = f(y_ink), kemudian mengirim sinyal *output* ke seluruh unit pada unit *output*.

Propagasi error (backpropagation of error)

Langkah 6

Setiap unit *output* (Yk, k = 1, ... , m) menerima satu target pola/*pattern* (*desired output*) yang sesuai dengan *input* training pola untuk menghitung kesalahan (error) antara target dengan *output* yang dihasilkan jaringan

$\delta k = (t_k - t_k)\, f'(y\_in_k)$ → (target-hasil) x turunan

Sebagaimana *input training* data dan *output training* data telah diskalakan menurut fungsi aktivasi yang telah diskalakan menurut fungsi aktivasi yang telah dipakai. Faktor δk digunakan untuk menghitung koreksi error (Δwjk) yang akan dipakai untuk memperbaharui wjk , dimana Δwjk = αδkzj. Selain itu juga akan dilakukan penghitungan untuk mengkoreksi bias Δw0k yang nantinya akan dipakai memperbaharui Δw0k, dimana: Δwjk = αδk . Faktor dari δk nantinya akan dikirim ke *layer* yang berada pada langkah 7.

Langkah 7

Setiap hidden unit (Zj = 1, ... ,p) menjumlah *input* delta (Δ) → yang telah dikirim dari *layer* langkah 6 (sudah berbobot).

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}$$

Setelah dilakukan penghitungan, hasilnya akan dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi error j, dimana

$$\delta_j = \delta in_j\, f^1\,(z\_in_j)$$

Faktor dari δ_j digunakan untuk menghitung koreksi error (Δvij) yang nantinya akan digunakan untuk memperbaharui vij , dimana $\Delta v_{ij} = \alpha\delta_j x_i$. Selain itu juga akan dilakukan penghitungan untuk mengkoreksi bias Δ_{0j}, di mana $\Delta v_{ij} = \alpha\delta_j$.

Pembaharuan bobot (adjustment) dan bias

Langkah 8

Setiap unit *output* (Yk, k = 1, ... , m) akan memperbaharui bias dan bobotnya dari setiap *hidden* unit (j = 0, ... , p), $w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_{jk}$.

Demikian pula setiap hidden unit (Zj, j = 1, ... , p) juga akan memperbaharui bias dan bobot dari setiap unit *input* (i = 0, ... , n),

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

Langkah 9

Selanjutnya adalah memeriksa *stop condition*, jika stop condition telah terpenuhi maka pelatihan jaringan dapat dihentikan. Kesembilan langkah diatas adalah algoritma pelatihan untuk jaringan syaraf. Untuk menentukan stop condition dapat dilakukan dengan dua cara yaitu:

1. Dengan membatasi terasi yang akan dilakukan, misalnya jaringan akan dilatih sampai pada terasi ke-500. Yang dimaksud dengan satu iterasi adalah perulangan dari langkah 3 sampai dengan langkah 8.
2. Cara yang ke dua yaitu dengan melakukan pembatasan error / membatasi error. Pada metode *backpropagation* akan dipakai metode Mean Square Error untuk menghitung rata-rata error antara *output* yang dikehendaki pada

training data dengan training *output* yang dihasilkan oleh jaringan. Misal jika error telah mencapai nilai 0,01 (1%) maka pelatihan akan dihentikan. Besarnya sebuah nilai error tergantung dari kepresisian yang dibutuhkan oleh sistem.

Selain dengan kedua cara tersebut, ada sebuah pertimbangan atau cara lain untuk menghentikan pelatihan jika nilai error semakin besar karena belum mencapai kondisi yang diinginkan. Kejadian tersebut disebut juga sebagai over training. Kondisi error yang diperhatikan tidak hanya error dari training set tetapi juga test set. Jika salah satu dari training set error atau test set error bertambah besar dan pelatihan harus dihentikan.

Berikut ini adalah cara untuk memeriksa stopping condition dengan Mean Square Error:

1. Dengan bobot yang telah ada, dilakukan langkah umpan maju (langkah 3 sampai dengan langkah 5) di mana *input*-nya diambil dari *input* training set, jika yang ingin dihitung adalah training set error dan *input* test set jika yang ingin dihitung adalah test set error. Langkah tersebut dilakukan untuk semua data training/test yang ada.
2. Cara yang kedua yaitu mencari selisih antara target *output* (t_k) dengan *output* jaringan (y_k) dan diimplementasikan pada persamaan Mean Square Error. Jika terdapat m training data, maka:

   Mean Square Error =

   $E = 0{,}5 \times \{(t_{k1} - y_{k1})^2 + (t_{k2} - y_{k2})^2 + \ldots + (t_{km} - y_{km})^2\}$

Setelah pelatihan selesai dan kemudian jaringan diberikan *input* maka jaringan akan dapat menghasilkan *output* seperti yang diharapkan. Cara untuk mendapatkan *output* adalah dengan mengimplementasikan metode *backpropagation*, tetapi hanya pada bagian umpan majunya. Berikut ini adalah langkah-langkahnya:

Langkah 1

Inisiali bobot yang sesuai dengan bobot yang dihasilkan pada proses pelatihan di atas. Untuk setiap *input* akan dilakukan langkah 2 – 4.

Langkah 2

Untuk setiap *input* i = 1, ... , n

Skalakan menjadi bilangan dalam range fungsi aktivasi seperti yang dilakukan pada proses pelatihan di atas, yang dilambangkan dengan variabel x_i.

Langkah 3

Untuk j = 1, ... , p:

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_j = f(z\_in_j)$$

Langkah 4

Untuk k = 1, ... , m:

$$y\_in_j = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

$$y_k = f(y\_in_k)$$

Variabel y_k adalah *output* yang masih dalam skala menurut range fungsi aktivasi. Untuk mendapatkan nilai *output* yang sesungguhnya maka y_k harus dikembalikan seperti semula.

Pada pengujian (Testing Session) pada jaringan syaraf tiruan *backropagation*, langkah-langkah yang harus dilakukan adalah dengan *feedforward* dan *output* dari *feedforward* tersebut akan dibandingkan dengan *output* pada fase pelatihan (Prof.Dr.Ir.Kuswara Setiawan, M.T, 2003, p. 21). Berikut ini adalah langkah-langkah *testing session:*

- Setiap unit *input* (Xi, i = 1, ... ,n) menerima sinyal *input* xi, dan menyebarkan sinyal tersebut pada seluruh unit atau *hidden* unit. Perlu diketahui bahwa *input* xi yang dipakai adalah data ini *input training* yang telah diskalakan. Pertama, *input* yang kemungkinan akan dipakai oleh sistem terlebih dahulu dicari nilai terendah dan nilai tertingginya. Selanjutnya, skala yang akan digunakan tergatung dari fungsi aktivasinya. Jika yang dipakai adalah fungsi sigmoid biner yang mempunyai nilai terendah = 0 dan nilai tertinggi =1, maka nilai *input* terendah dianggap = 0 dan nilai tertinggi dianggap 1. Nilai diantaranya bervariasi antara 0 dan 1. Sedangkan yang dipakai adalah fungsi sigmoid bipolar maka range nilainya juga bervariasi mulai -1 sampai dengan 1.

  Langkah 4

  Setiap hidden unit (Zj, j = 1, ... ,p), akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot dan termasuk juga dengan biasnya,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} X_i V_{ij}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari hidden unit yang bersangkutan zj = f(z_inj) kemudian mengirim sinyal *output* tersebut ke seluruh unit pada unit *output*.

Langkah 5

Setiap unit *output* (Yk, k = 1, ... ,m) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot termasuk juga dengan biasnya,

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari unit *output* yang bersangkutan yk = f(y_ink), kemudian mengirim sinyal *output* ke seluruh unit pada unit *output*.

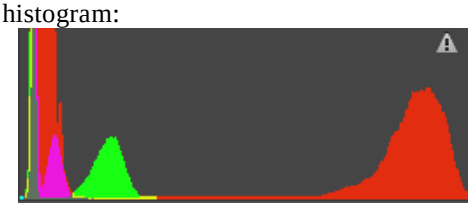
Langkah 6

Untuk menentukan hasil klasifikasi, maka *output feedforward* pada langkah pengujian akan dibandingkan dengan *output* dari pelatihan.

Menurut Fajar Astuti Hermawati, S.Kom, M.Kom [Pengolahan Citra Digital Konsep& Teori, 2003, p. 36] histogram adalah diagram yang menunjukan jumlah kemunculan nilai gray-level, dimana sumbu –x dari diagram menggambarkan nilai gray-level dan sumbu y mewakili jumlah kemunculan gray-level. Ada empat tipe dasar citra yang dapat digambarkan dengan sebuah histogram, yaitu:

- Citra gelap: histogram cenderung ke sebelah kiri
- Citra terang: histogram cenderung ke sebelah kanan
- Citra low contrast: histogram mengumpul di suatu tempat
- Citra high contrast: histogram merata di semua tempat

Berikut ini adalah contoh dari sebuah histogram:



Gambar 5. Histogram

Menurut Fajar Astuti Hermawati, S.Kom, M.Kom [Pengolahan Citra Digital Konsep& Teori, 2003, p. 187] *threshold* adalah sebuah teknik dalam pengolahan citra digital dengan memberikan sebuah nilai ambang (T) pada sebuah histogram. Pada histogram sebelah kiri mewakili citra *f(x,y)* , yang tersusun atas objek terang di atas *background* gelap. Piksel-piksel objek dan *background* dikelompokkan menjadi 2 mode yang dominan dengan pemberian nilai threshold.

Sembarang titik *(x,y)* yang memenuhi *f(x,y)>T* disebut titik objek, selain itu disebut titik *background*. Histogram sebelah kanan terbagi menjadi tiga mode, misalkan sebuah citra terdiri atas dua objek terang diatas *background* gelap. Thresholding multilevel digunakan untuk mengklasifikasikan suatu titik (*x,y)* sebagai bagian dari sebuah class objek. Titik *(x,y)* menjadi bagian dari suatu objek jika $T_1 < f(x,y) \leq T_2$, dan menjadi bagian dari objek yang lain jika *f(x,y)* > T_2, dan menjadi bagian dari *background* jika *f(x,y)* $\leq T_1$. Thresholding adalah pengesetan terhadap fungsi T= T[*x, y, p*(*x,y*), *f*(*x,y*)].

2. METODOLOGI PENELITIAN

A. Analisa Masalah

Masalah yang akan dibahas adalah bagaimana merancang sebuah aplikasi yang mampu untuk mendeteksi kelayakan telur, apakah telur tersebut layak untuk dikonsumsi atau tidak (baik atau buruk), karena dalam penentuan kelayakan sebuah telur, manusia tidak dapat secara langsung menentukan telur tersebut baik atau buruk dengan mata telanjang. Ada beberapa cara tradisional, salah satunya yaitu melalui penyinaran pada telur atau yang biasa disebut *candling*. Dalam penyinaran tersebut jika telur membiaskan cahaya maka telur tersebut baik namun jika telur tersebut disinari tidak tembus atau bias cahaya-nya redup maka telur tersebut dapat dikatakan telah buruk.

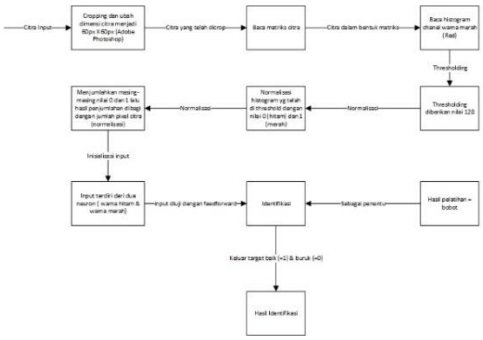
Dalam dunia industri khususnya yang bergerak di bidang produksi pangan seperti perusahaan (roti/kue) yang besar, kelayakan telur adalah alasan utama untuk menjamin sebuah kualitas produk pangan yang dihasilkan. Namun dalam menjamin hal tersebut tidak lah mungkin dilakukan *candling* satu persatu dari sekian banyak telur yang ada sebelum telur tersebut diolah.

B. Block Diagram

Blok Diagram Pelatihan Citra Telur

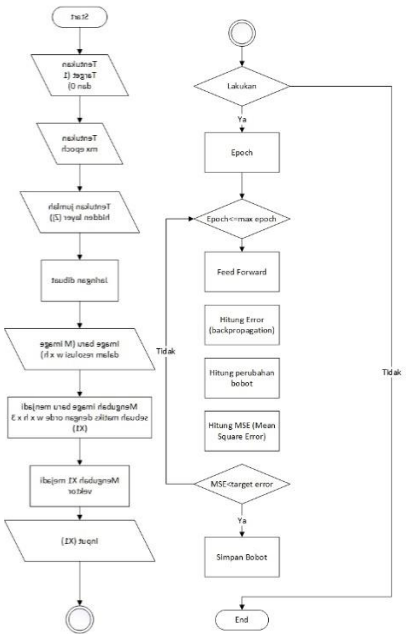
Gambar 6. Blok Diagram Pelatihan Citra Telur

Blok Diagram Pengujian Citra Telur



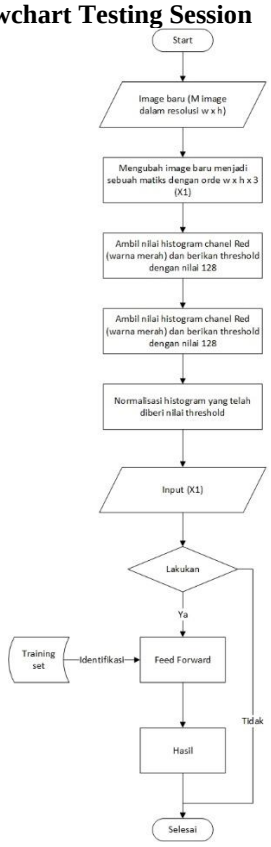
Gambar 7. Blok Diagram Pengujian Citra Telur

C. Flowchart Training Session



Gambar 8. Flowchart Training Session

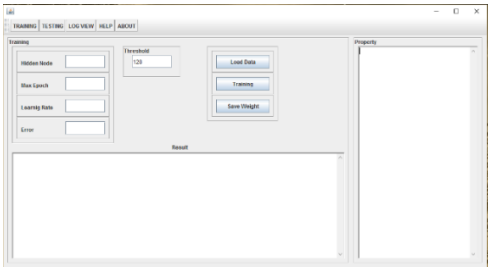
D. Flowchart Testing Session



Gambar 9. Flowchart Testing Session

E. Rancangan User Interface

Form Pelatihan



Gambar 10. Form Pelatihan

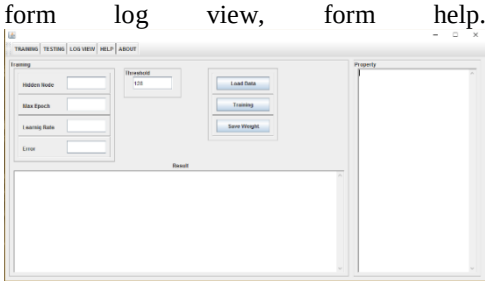
Form Pengujian

Gambar 11. Form Pengujian

3. HASIL DAN PEMBAHASAN

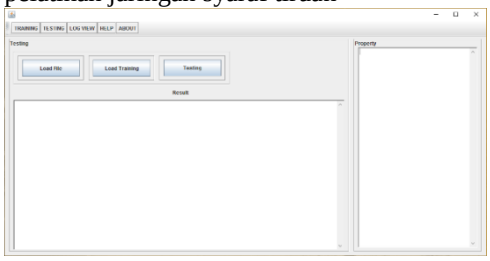
A. Implementasi

Pada Segemen program disusun sebagai berikut: Form training, form testing, form log view, form help.



Gambar 12. Form Training

Pada gambar 12 berfungsi untuk melakukan pelatihan jaringan syaraf tiruan

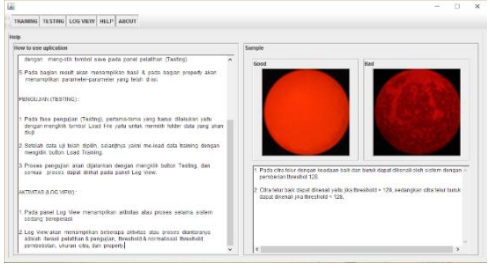


Gambar 13. Form Testing

Pada gambar 13 berfungsi untuk melakukan pengujian data menggunakan pembaharuan bobot dari hasil training.

Gambar 14. Form Log View

Pada gambar 14 menampilkan setiap proses yang dilakukan oleh sistem, mulai dari penghitungan bobot, thresholding, iterasi dan hasil dari pelatihan dan pengujian.



Gambar 15. Form Help

Pada gambar 15. berfungsi sebagai form untuk bantuan penggunaan aplikasi jika pengguna mengalami kesulitan dalam pengoperasian aplikasi.

B. Pembahasan

Pelatihan

Dalam pelatihan jaringan syaraf tiruan ini akan dilakukan pelatihan dengan menggunakan data training sebanyak 100 citra telur yang terdiri dari telur usia 1-7 hari (baik) dan telur usia 13-20 hari (buruk/tidak layak konsumsi). Berikut ini adalah tabel dari data telur:

Tabel 4.2 Hasil Uji Coba Pelatihan

Hidden Node	α	Min Error	Max Epoch	Hasil
10	0.1	0.1	1.000.000	MSE = 0.0902666019905 Iterasi = 20 Alfa = 0.099999998 Akurasi = 98 % (salah 2)
30	0.1	0.1	1.000.000	MSE = 0.0907985396052 Iterasi = 1050 Alfa = 0.099999895 Akurasi = 97 % (salah 3)
20	0.1	0.01	1.000.000	MSE = 0.00999999698 Iterasi = 48829 Alfa = 0.0999951171 Akurasi = 99 % (salah 1)
45	0.1	0.01	1.000.000	MSE = 0.00999999441 Iterasi = 426064 Alfa = 0.099957393602 Akurasi = 99 % (salah 1)
30	0.2	0.025	1.000.000	MSE = 0.002499343295149 Iterasi = 149959 Alfa = 0.1997000820001 Akurasi = 100% (salah 0)

Pengujian

Pada tahap pengujian akan dilakukan pengujian (*testing*) data telur usia 1-12 hari (140 citra telur baik) dan usia 13-20 hari (140 citra telur buruk). Data telur yang digunakan untuk pengujian berbeda dengan data telur pelatihan. Pengujian dilakukan dengan menggunakan data training dengan akurasi 100% benar dengan nilai parameter *hidden*

node: 30, *epoch*: 149.959, *learning rate*: 0.2, *min error*: 0.025.

Dari pengujian data telur dengan usia 1-12 (140 citra baik), dapat diambil kesimpulan bahwa sistem dari aplikasi yang dibuat mampu untuk mengklasifikasikan data telur baik sejumlah 140 dengan kegagalan klasifikasi 2 data telur. Keakurasian dari klasifikasi data telur baik yaitu:

$$\frac{138}{140}\ x\ 100\ \% = 98{,}58\ \%$$

Sedangkan pada klasifikasi data telur buruk masih belum sempurna, yakni kegagalan dalam mengklasifikasikan data telur usia 13-20 hari (140 citra buruk) sebanyak 107 dianggap sebagai data telur baik. Berikut ini adalah keakurasian dari sistem aplikasi untuk mengklasifikasi data telur buruk:

$$\frac{33}{140}\ x\ 100\ \% = 23{,}58\ \%$$

4. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian yang berupa pengembangan aplikasi untuk mengklasifikasikan telur yang baik dan buruk, dapat diambil sebuah kesimpulan yaitu:

1. Lamanya waktu untuk melakukan pelatihan sejumlah 100 data training dengan jumlah *hidden node* yang besar atau nilai *learning rate* yang sangat kecil, masih dapat dikatakan cukup lama dikarenakan iterasi yang cukup banyak untuk mencapai *error minimum*.
2. Pada fase pelatihan keakurasian sistem dalam mengklasifikasikan data telur dapat sangat lama dikarenakan kemiripan data telur antara yang baik dan buruk, sehingga sistem perlu melakukan iterasi sebanyak-banyaknya untuk mencapai *error minimum*.
3. Pada fase pengujian keakurasian sistem dalam mengklasifikasi data telur usia 1-12 hari (140 citra telur baik) dapat dikatakan akurat (98,58%), namun pada beberapa pengujian data telur usia 13-20 hari (140 citra telur buruk) masih belum dapat dikatakan akurat (25,58%) karena kegagalan sistem aplikasi untuk mengklasifikasikan data telur buruk yang hampir mirip dengan data telur baik.

B. Saran

Dari penelitian ini masih terdapat banyak kemungkinan untuk pengembangan aplikasi dan meningkatkan kemampuan serta menutupi kekurangan yang telah dipaparkan di kesimpulan. Beberapa hal yang diusulkan untuk penelitian dan pengembangan aplikasi klasifikasi menggunakan jaringan syaraf tiruan *backpropagation*, yaitu:

1. Penambahan *multilayer hidden* agar memiliki keakurasian yang tinggi dalam klasifikasi.
2. Penambahan *image processing*, untuk mendapatkan ekstrasi ciri citra yang lebih *detail*.
3. Dikembangkannya aplikasi yang lebih komplek dengan penambahan conveyor khusus untuk telur dan penggunaan sensor untuk menggantikan kamera.

5. DAFTAR PUSTAKA

[1] Hartati, G Sri. B Herry Suharto, M Soesilo Wijono. (2006). Pemrogaman Gui Swing Java dengan Netbeans 5. Yogyakarta. Andi Offset.

[2] Hermawati, Fajar Astuti. (2006). Pengolahan Citra Digital Konsep & Teori. Surabaya. Andi Offset.

[3] Juliasmi, Tito., Kusuma, Kartina Diah., Nugroho, Erwin Setyo. (2012). Aplikasi Pengenalan Karakter Manusia Melalui Bentuk Bagian Wajah Menggunakan Metode Backpropagation. Jurnal Teknik Informatika Jurusan Komputer Politeknik Caltex Riau.

[4] Nurhayati., Iskandarianto, Fitri Adi. (2010). Penerapan Metode Back Propagation Neural Network pada Pendeteksian Kelainan Otak Ischemic Cerebral Infraction dengan Bahasa Pemrogaman Delphi. Jurnal Fisika dan Aplikasinya. Institut Tegnologi Sepuluh November (ITS) Surabaya. Vol. 6, No. 1.

[5] Setiawan, Kuswara. (2003). Paradigma Sistem Cerdas. Surabaya. Bayumedia Publishing.

[6] Setiawan, Sari Indah Anatta. (2011). Penerapan Jaringan Saraf Tiruan Metode Backpropagation Menggunakan VB 6. Jurnal Ultimatics Universitas Multimedia Nusantara Tangerang, Vol. III, No. 2.

[7] Siang, Jong Jek. (2004). Jaringan Syaraf Tiruan & Pemrogamannya Menggunakan Matlab. Yogyakarta. Andi Offset.

[8] T Sutojo, Bowo N, Erna Z A, Setia Astuti, Yuniarsih Rahayu, Edy Mulyanto. (2009). Teori dan Aplikasi Aljabar Linier & Matriks. Yogyakarta. Andi Offset.