METODE *SECOND POSITION ASYMMETRIC WINDOWING* DAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION* UNTUK PENGENALAN FONEM DALAM BAHASA INDONESIA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oleh :
ALDO SEPTYAN PRAHERDA
0310960005-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2008

METODE *SECOND POSITION ASYMMETRIC WINDOWING* DAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION* UNTUK PENGENALAN FONEM DALAM BAHASA INDONESIA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oleh :
ALDO SEPTYAN PRAHERDA
0310960005-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2008


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

METODE *SECOND POSITION ASYMMETRIC WINDOWING* DAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION* UNTUK PENGENALAN FONEM DALAM BAHASA INDONESIA

Oleh:
ALDO SEPTYAN PRAHERDA
0310960005-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 7 Januari 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Marji, MT	**Bayu Rahayudi, ST, MT**
NIP. 131 993 386	**NIP. 132 318 424**

Mengetahui,
a.n. Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya
Sekretaris,

Dra. Ani Budi Astuti, MSi
NIP. 131 993 385


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Aldo Septyan Praherda
NIM : 0310960005
Jurusan : Matematika
Penulis skripsi berjudul : Metode *Second Position Asymmetric Windowing* dan Jaringan Syaraf Tiruan *Backpropagation* untuk Pengenalan Fonem dalam Bahasa Indonesia.

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 7 Januari 2008
Yang menyatakan,

(Aldo Septyan Praherda)
NIM. 0310960005

METODE *SECOND POSITION ASYMMETRIC WINDOWING* DAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION* UNTUK PENGENALAN FONEM DALAM BAHASA INDONESIA

ABSTRAK

Text-to-speech dalam Bahasa Indonesia dapat dibangun dengan beberapa metode diantaranya adalah metode *phoneme synthesis*. Salah satu kelemahan yang ada pada metode *phoneme synthesis* adalah belum bisa mengatasi masalah perbedaan pengucapan pada suatu huruf. Kelemahan pada metode *phoneme synthesis* perlu diselesaikan sebab dalam Bahasa Indonesia vokal ‘e’ dan ‘o’ mempunyai lebih dari satu cara pengucapan. Dalam penelitian ini diimplementasikan metode *Second Position Asymmetric Windowing* (SPAW) dan jaringan syaraf tiruan *backpropagation* untuk mengatasi masalah perbedaan pengucapan pada suatu huruf. Data yang digunakan untuk proses pembelajaran adalah kata yang mengandung huruf 'e' atau 'o' sebanyak 500 kata. Sedangkan data untuk proses pengujian adalah 50 kata yang mengandung huruf 'e' atau 'o' serta belum pernah dilakukan pembelajaran. Pada proses pembelajaran dilakukan uji coba jumlah unit pada *hidden layer*, besar *learning rate*, dan jumlah *max epoch* untuk menghasilkan model yang terbaik. Tolok ukur keberhasilan pengujian model dilakukan dengan menghitung nilai *mean squared error* (MSE) dan jumlah kata yang berhasil dikenali fonemnya. Dari penelitian ini diketahui bahwa metode jaringan syaraf tiruan *backpropagation* dengan metode *Second Position Asymmetric Windowing* mampu mengenali huruf yang mempunyai lebih dari satu cara pengucapan dengan nilai keakuratan sebesar 97.80% (MSE : 0.0054) untuk kata-kata yang pernah dilakukan pembelajaran dan menghasilkan nilai keakuratan rata-rata sebesar 63.6% untuk kata-kata yang belum pernah dilakukan pembelajaran. Pembelajaran dilakukan menggunakan metode *Second Position Asymmetric Windowing* dengan model 1-5 dan metode jaringan syaraf tiruan *backpropagation* dengan jumlah unit pada *input layer* sebanyak 189 unit, jumlah unit pada *hidden layer* sebanyak 80 unit, jumlah unit pada *output layer* sebanyak 30 unit, nilai *learning rate* sebesar 0.005 dan *max epoch* sebesar 3000.

SECOND POSITION ASYMMETRIC WINDOWING METHOD AND BACKPROPAGATION NEURAL NETWORKS FOR PHONEME RECOGNITION IN INDONESIAN LANGUAGE

ABSTRACT

Text-to-speech in Indonesian language can be build by using several methods, one of them is phoneme synthesis. Yet, this methode still has some weakness, that is still not able to handle dual phoneme cases (i.e., one letter is mapped to two phonemes). This weakness of phoneme synthetis method needs to be solved, since in Indonesian language, in this case is the vocal 'e' and 'o' have more than one pronounciation. This research will implement Second Position Asymmetric WIndowing (SPAW) method and Backpropagation Neural Networks method, in handling the dual phoneme cases. Data used as training data in this research is some words which contains letter 'e' or 'o'. The amount of this data itself is 500 words. Meanwhile, data used as testing data is in the amount of 50 words which cointains letter 'e' or 'o' and not used as training data. In learning process, we do the testing of number of units in hidden layer, learning rate value, and number of max epoch to obtain the best model. The success of model testing is measured by calculating mean square error (MSE) and number of words which its phoneme succesfully recognized. From this research, it can be conclude that Second Position Asymmetric Windowing method using Backpropagation Neural Networks, is able to recognize the letter that has dual phoneme cases, with accuration rate at 97.80% (MSE : 0.0054) for the words that has passed the learning process. Meanwhile, for the words that is not passed the learning process, the accuration rate is at 63.6%. The best net for the dual phoneme case used a 1-5 SPAW and one hidden layer with 189 units in input layer, 80 units in hidden layer, 30 units in output layer, 0.005 of learning rate and 3000 max epoch.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, skripsi yang berjudul "Metode *Second Position Asymmetric Windowing* dan Jaringan Syaraf Tiruan *Backpropagation* untuk Pengenalan Fonem dalam Bahasa Indonesia" ini dapat diselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Marji, MT selaku Dosen Pembimbing Utama dan Bayu Rahayudi, ST, MT selaku Dosen Pembimbing Kedua. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Wayan Firdaus Mahmudy, SSi, MT selaku Penasihat Akademik dan Ketua Program Studi Ilmu Komputer, Universitas Brawijaya.
3. Dr. Agus Suryanto, MSc selaku ketua Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
5. Segenap staf dan karyawan di Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.
6. Ayah dan Ibu serta adikku Windy dan Merry. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
7. Teman-teman sekontrakan di Bamboong. Terima kasih atas segala bantuan dan dukungannya.
8. Sahabat-sahabat labdevilz dan arek ilkom `03. Terima kasih atas semangat dan persahabatannya.

9. Santi, Rosihan, Apriliyah dan semua orang yang telah memberikan bantuan, semangat dan do'a yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan skripsi ini bermanfaat bagi pembaca sekalian. Dengan tidak lupa kodratnya sebagai manusia, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 31 Desember 2007

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Salah satu contoh perkembangan bidang multimedia adalah munculnya sub bidang ilmu yang memungkinkan komputer untuk memproduksi suara manusia berdasarkan suara manusia yang telah disimpan terlebih dahulu dan kemudian dikenal dengan istilah *stored speech*. Dalam perkembangannya sub bidang ilmu *stored speech* mampu mengenali *input* yang berupa teks menjadi suara manusia dan kemudian dikenal dengan istilah *text-to-speech* (Tjiputra, 2003).

Ada beberapa cara untuk membuat sebuah sistem *text-to-speech* diantaranya adalah dengan menyimpan suara pengucapan semua kata pada sebuah *database*, yang nantinya akan dirangkai sesuai dengan kata yang dicari. Metode ini dikenal dengan sebutan *dictionary method. Dictionary method* mempunyai keunggulan yaitu keakuratan yang tinggi, tetapi memerlukan *database* yang besar untuk menyimpan semua data suara pengucapan kata, selain itu waktu akses yang diperlukan akan semakin lama bila ukuran *database* semakin besar (Hendessi, 2004).

Metode lain yang dapat digunakan untuk membangun sebuah sistem *text-to-speech* adalah metode jaringan syaraf tiruan. Metode jaringan syaraf tiruan memerlukan data berupa struktur pengucapan beberapa kata yang diperlukan dalam proses pembelajaran. Struktur pengucapan suatu kata dipengaruhi oleh fonem, yaitu aspek bunyi bahasa yang membedakan bentuk dan makna kata. Salah satu contoh sistem *text-to-speech* dalam Bahasa Inggris yang berbasiskan jaringan syaraf tiruan *backpropagation* adalah NETtalk, yang dikembangkan oleh Sejnowski dan Rosenberg. Sistem NETtalk dapat mengubah teks normal menjadi bentuk fonemnya dengan tingkat keakuratan maksimum mencapai 98%. Teks fonem ini kemudian dijadikan masukan pada sistem *text-to-speech* DECtalk untuk menghasilkan suara manusia (Sejnowski, 1986).

Dalam Bahasa Indonesia sebuah sistem t*ext-to-speech* dapat dibangun dengan menggunakan beberapa metode yang salah satunya adalah metode *phoneme synthesis*. Metode *phoneme synthesis* bekerja dengan cara menyimpan *file* suara yang mewakili suatu suku kata tertentu dan kemudian *file* suara tersebut disambungkan sesuai

dengan aturan pemenggalan suku kata. Metode *phoneme synthesis* membutuhkan pengetahuan tentang tata cara pemenggalan suku kata dalam suatu bahasa (Tjiputra, 2003).

Salah satu kelemahan yang ada pada metode *phoneme synthesis* adalah belum bisa mengatasi masalah perbedaan pengucapan pada suatu huruf. Kelemahan pada metode *phoneme synthesis* perlu diselesaikan sebab dalam Bahasa Indonesia vokal 'e' dan 'o' mempunyai perbedaan pengucapan. Vokal 'e' memiliki tiga variasi bentuk pengucapan misalnya 'e' pada kata 'kare' dan 'sore', 'e' pada kata 'perak' dan 'remeh' dan variasi pengucapan 'e' pada kata 'tipe' dan 'bandeng'. Sedangkan vokal 'o' memiliki dua variasi misalnya 'o' pada kata 'toko' dan 'soto' dan variasi pengucapan 'o' pada kata 'obat', 'bodoh' dan 'olah'.

Dalam Bahasa Inggris juga terdapat huruf yang dapat menghasilkan suara pengucapan yang berbeda pada suatu kata. Misalnya huruf 'a' pada kata 'CAME' mempunyai perbedaan pengucapan dengan huruf 'a' pada kata 'CAMERA'. Masalah perbedaan pengucapan pada suatu huruf belum bisa diatasi dengan baik oleh sistem NETtalk. Untuk mengatasi masalah perbedaan pengucapan pada suatu huruf, Arciniegas dan Embrechts mengusulkan metode *windowing* yang disebut dengan *Second Position Asymmetric Windowing* (SPAW) untuk menggantikan metode *windowing* pada NETtalk (Arciniegas, 2000).

Berdasarkan latar belakang tersebut diperlukan suatu metode untuk mengenali perbedaan pengucapan (fonem) pada suatu huruf. Dalam skripsi ini metode yang dipilih adalah metode jaringan syaraf tiruan *backpropagation* dengan *Second Position Asymmetric Windowing*. Metode jaringan syaraf tiruan *backpropagation* digunakan untuk proses pengenalan fonem sedangkan metode *Second Position Asymmetric Windowing* digunakan dalam proses *windowing* untuk mengatasi masalah huruf yang mempunyai lebih dari satu cara pengucapan. Oleh karena itu penulis mengambil judul **Metode *Second Position Asymmetric Windowing* dan Jaringan Syaraf Tiruan *Backpropagation* untuk Pengenalan Fonem dalam Bahasa Indonesia**.

1.2. Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah bagaimana merancang jaringan syaraf tiruan *backpropagation* dengan *Second Position Asymmetric Windowing* agar dapat mengenali perbedaan pengucapan pada vokal ‘e’ dan ‘o’ dalam Bahasa Indonesia.

1.3. Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Pengenalan fonem ini diterapkan dalam Bahasa Indonesia pada vokal ’e’ dan ’o’.
2. Tidak dilakukan proses analisis semantik terhadap suatu kata.
3. Tidak dilakukan perbandingan penerapan algoritma lainnya.
4. Jumlah kata yang dijadikan data latih hanya sebanyak 500 kata.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Membangun sebuah sistem pengenalan fonem yang dapat membaca suatu teks sesuai dengan tata bahasa baku Bahasa Indonesia.
2. Menerapkan metode jaringan syaraf tiruan *backpropagation* dan metode *Second Position Asymmetric Windowing* dalam pengenalan fonem.

1.5. Manfaat Penelitian

Manfaat yang dapat diambil dari penelitian ini adalah mengetahui kegunaan metode jaringan syaraf tiruan *backpropagation* dan metode *Second Position Asymmetric Windowing* dalam mengatasi perbedaan pengucapan vokal ‘e’ dan ‘o’ pada Bahasa Indonesia. Sistem pengenalan fonem ini juga bisa dimanfaatkan pada sistem *text-to-speech* dalam Bahasa Indonesia.


UNIVERSITAS BRAWIJAYA

BAB II
TINJAUAN PUSTAKA

2.1. Beberapa Istilah Dalam Tata Bahasa Baku Bahasa Indonesia

Berikut adalah beberapa istilah dalam tata bahasa baku Bahasa Indonesia (Alwi, 2000) :

1. Fonem, merupakan satuan bahasa terkecil berupa bunyi atau aspek bunyi bahasa yang membedakan bentuk dan makna kata. Dalam ilmu bahasa, fonem ditulis di antara dua garis miring: /…/. Jadi dalam Bahasa Indonesia /p/ dan /b/ adalah dua fonem karena kedua bunyi itu membedakan bentuk dan arti. Contoh:
   Pola - /pola/ Bola - /bola/
   Parang - /parang/ Barang - /barang/
2. Vokal, merupakan puncak dari suku kata. Dalam Bahasa Indonesia dikenal enam vokal yaitu :
   a. /i/.
   b. /u/.
   c. /a/.
   d. /o/, mempunyai dua variasi pengucapan yaitu [o] contohnya pada ‘toko’ dan [ɔ] contohnya pada kata ‘rokok’.
   e. /e/, mempunyai dua variasi pengucapan yaitu [e] contohnya pada kata ‘besok’ dan [ɛ] contohnya pada kata ‘tokek’.
   f. /ë/, contohnya pada kata ‘emas’ dan ‘bandeng’.

2.2. Jaringan Syaraf Tiruan

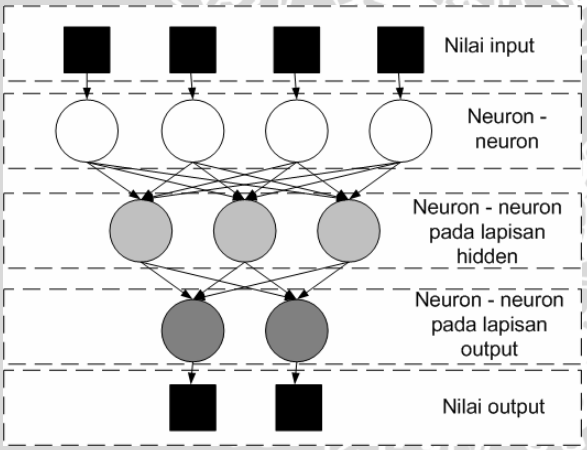
Menurut Kusumadewi (2003), jaringan syaraf tiruan ialah suatu sistem pengolah informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologis tubuh manusia. Jaringan syaraf tiruan telah dikembangkan dengan menggunakan model matematis untuk meniru cara kerja jaringan syaraf biologis, dengan berdasarkan asumsi :

a. Pengolah informasi terdiri dari elemen-elemen sederhana yang disebut *neuron* atau unit.
b. Sinyal dilewatkan dari satu *neuron* ke *neuron* yang lain melalui hubungan koneksi.
c. Tiap hubungan koneksi mempunyai nilai bobot tersendiri.

d. Tiap *neuron* mempergunakan fungsi aktivasi terhadap masukan yang diterimanya untuk menentukan sinyal keluarannya.

Informasi (disebut dengan : *input*) akan dikirim ke *neuron* dengan bobot kedatangan tertentu. *Input* akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui fungsi aktivasi setiap *neuron*.

Pada jaringan syaraf tiruan, *neuron-neuron* dikumpulkan dalam lapisan-lapisan (*layer*). Suatu jaringan syaraf minimum tersusun atas *input layer* dan *output layer*. Dalam beberapa tipe jaringan diantara *input layer* dan *output layer* terdapat *hidden layer*. Hal ini berarti bahwa semua neuron pada *input layer* akan berhubungan ke semua *neuron* dalam *hidden layer* yang selanjutnya setiap *neuron* dalam *hidden layer* nantinya akan dihubungkan ke semua *neuron* di *output layer* (Siang, 2005). Contoh model jaringan syaraf dengan 3 lapisan dapat dilihat pada Gambar 2.1.



Gambar 2.1 Jaringan syaraf dengan 3 lapisan

2.2.1. Proses Pembelajaran

Proses pembelajaran adalah proses untuk menentukan bobot penghubung antar *neuron*. Nilai bobot bertambah jika informasi yang diberikan oleh *neuron* yang bersangkutan tersampaikan, sebaliknya

jika informasi tidak disampaikan oleh *neuron* ke *neuron* yang lain, maka nilai bobot yang menghubungkan keduanya dikurangi.

Pada saat pembelajaran dilakukan pada *input* yang berbeda, nilai bobot diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Nilai yang cukup seimbang ini mengindikasikan bahwa tiap-tiap *input* telah berhubungan dengan *output* yang diharapkan.

Berdasarkan metode pembelajarannya jaringan syaraf tiruan dibagi menjadi (Siang, 2005) :

1. *Supervised* (pembelajaran yang terawasi)
   Pada sistem pembelajaran *supervised* terdapat pasangan data (masukan dan target keluaran) yang dipakai untuk melatih jaringan hingga memperoleh bobot yang diinginkan.
2. *Unsupervised* (pembelajaran yang tidak terawasi)
   Sistem pembelajaran *unsupervised* kebalikan dari sistem pembelajaran *supervised*, tidak ada pasangan data (masukkan dan target keluaran) yang dipakai untuk melatih, tetapi perubahan bobot jaringan berdasarkan parameter tertentu dan jaringan dimodifikasi menurut parameter tersebut.

Di dalam proses pembelajaran, *learning rate* adalah sebuah konstanta yang nilainya diantara 0 sampai 1 yang berguna untuk mempercepat proses pembelajaran. Sedangkan *max epoch* adalah batas maksimum perulangan untuk melakukan proses pembelajaran.

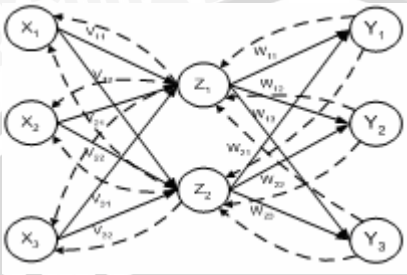
2.2.2. Algoritma *Backpropagation*

Algoritma *backpropagation* merupakan algoritma pembelajaran yang terawasi. Algoritma *backpropagation* menggunakan nilai *error* pada *output layer* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan nilai *error*, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, *neuron-neuron* diaktifkan dengan menggunakan fungsi aktivasi *sigmoid*. Fungsi *sigmoid* yang sering dipakai yaitu :

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2.1)$$

Keterangan :
e : bilangan natural yaitu 2.71828.
x : hasil penjumlahan dari sinyal-sinyal *input* terbobot.

Arsitektur jaringan *backpropagation* seperti terlihat pada gambar 2.2.



Gambar 2.2 Arsitektur jaringan *backpropagation.*

Algoritma *backpropagation* (Kusumadewi, 2003) :

1. Inisialisasi bobot dengan nilai *random* yang cukup kecil.
2. Kerjakan langkah–langkah berikut selama kondisi berhenti belum terpenuhi :

   Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan :

   feedforward :

   a. Tiap-tiap unit *input* (X_i = 1,2,3,..,n) menerima sinyal x_i dan meneruskan *sinyal* tersebut ke semua unit pada lapisan di atasnya (*hidden layer*). Dimana n adalah jumlah unit *input.*

   b. Tiap-tiap unit tersembunyi (Z_j, j= 1,2,3,…,p) menjumlahkan sinyal – sinyal *input* terbobot (z_in_j) :

   $$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (2.2)$$

   Keterangan :

   v_{0j} : nilai bobot dari bias ke unit tersebunyi j.

   v_{ij} : nilai bobot dari input i ke unit tersebunyi j.

   p : jumlah unit tersembunyi.

   Gunakan fungsi aktivasi untuk menghitung sinyal *output* (z_j) :

   $$z_j = f(z\_in_j) \quad (2.3)$$

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

   c. Tiap-tiap unit *output* (Y_k, k = 1,2,3,..,m) menjumlahkan sinyal-sinyal *input* terbobot (y_in_k) :

   $$y\_in_k = W_{0k} + \sum_{i=1}^{p} z_i w_{jk} \quad (2.4)$$

   Keterangan :

W_{0k} : nilai bobot dari bias ke unit *output.*

W_{jk} : nilai bobot dari unit tersembunyi j ke unit *output* k.

m : jumlah unit *output.*

Gunakan fungsi aktivasi untuk menghitung sinyal *output* (y_k) :

$$y_k = f(y\_in_k) \quad (2.5)$$

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

Backpropagation :

d. Tiap-tiap unit *output* (Y_k, k=1,2,3,…,m) menerima target pola yang berhubungan dengan pola *input* pembelajaran, hitung informasi *error*-nya (δ_k) :

$$\delta_k = (t_k - y_k)f'(y\_in_k) \quad (2.6)$$

Keterangan :

t_k : target pola pada *output* k.

$f'(y\_in_k)$: turunan dari fungsi aktivasi.

Kemudian hitung koreksi bobot dari unit-unit tersembunyi j ke unit-unit *output* k (Δw_{jk}) :

$$\Delta w_{jk} = \alpha\, \delta_k\, z_j \quad (2.7)$$

Keterangan :

α : nilai *learning rate.*

Hitung juga koreksi bias ke unit-unit *output* k (Δw_{0k}) :

$$\Delta w_{0k} = \alpha\, \delta_k \quad (2.8)$$

Kirimkan informasi *error* (δ_k) ke unit-unit yang ada di lapisan bawahnya.

e. Tiap-tiap unit tersebunyi (Z_j, j = 1,2,3,…,p) menjumlahkan delta *input*-nya (δ_in_j) dari unit-unit yang berada pada lapisan di atasnya :

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.9)$$

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

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (2.10)$$

Kemudian hitung koreksi bobot dari unit-unit *input* i ke unit-unit tersembunyi j (Δv_{ij}) :

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (2.11)$$

Hitung juga koreksi bias ke unit-unit tersembunyi j (Δv_{0j}) :

$$\Delta v_{0j} = \alpha \delta_j \quad (2.12)$$

f. Tiap-tiap unit *output* (Y_k, k = 1,2,3,…,m) memperbaiki bias dan bobotnya (j=0,1,2,…,p) :

$W_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$ (2.13)

Tiap–tiap unit tersebunyi (z_j, j = 1,2,3,…,p) memperbaiki bias dan bobotnya (i=0,1,2,…,n) :

$V_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$ (2.14)

3. Tes kondisi berhenti, kondisi berhenti terpenuhi bila proses *feedforward* dan *backpropagation* telah diulang sebanyak *max epoch* atau *error* pada unit *output* telah memenuhi batas yang ditentukan.

2.2.3. Algoritma *Backpropagation* Dengan Metode *Stochastic Diagonal Levenberg-Marquardt*

Algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* diperkenalkan oleh LeCun (1998). Algoritma tersebut hampir sama dengan algoritma *backpropagation* pada umumnya namun berbeda dalam proses memperbarui bobot-bobotnya. Algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* kira-kira tiga kali lebih cepat konvergen (sampai pada nilai *error* yang konstan) daripada algoritma *backpropagation* biasa. Pada proses pembelajaran, tiap-tiap bobot diperbarui dengan menggunakan *learning rate* yang berbeda-beda yang diperoleh dari perhitungan nilai *hessian* ($\frac{\partial^2 E}{\partial w_k^2}$).

Fungsi aktivasi yang dipakai dalam algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* adalah :

$$f(x) = 1.7159 \tanh(\frac{2}{3}x) \quad (2.15)$$

yang mempunyai turunan :

$$f'(x) = \frac{1.7159(\frac{2}{3})}{(\cosh(\frac{2}{3}x))^2} \quad (2.16)$$

Fungsi aktivasi pada persamaan 2.15 digunakan sebab fungsi tersebut dapat menghasilkan nilai yang bervariasi (sehingga mempunyai rata-rata mendekati nol) dibandingkan dengan fungsi aktivasi pada persamaan 2.1 yang selalu menghasilkan nilai positif.

Langkah-langkah untuk mendapatkan nilai *hessian* adalah sebagai berikut *:*

1. Pada *output layer*, untuk tiap-tiap unit *output* (Y_k, k=1,2,3,…,m) hitung nilai turunan kedua fungsi *error* (persamaan 2.29) terhadap sinyal *output* terbobot ($\frac{\partial^2 E}{\partial y_k^2}$) :

$$\frac{\partial^2 E}{\partial y_k^2} = (f'(y\_in_k))^2 \quad (2.17)$$

Kemudian nilai turunan kedua fungsi *error* terhadap sinyal *output* terbobot digunakan untuk menghitung nilai *hessian* pada bobot dari *hidden layer* ke *output layer* ($\frac{\partial^2 E}{\partial w_{jk}^2}$) :

$$\frac{\partial^2 E}{\partial w_{jk}^2} = \frac{\partial^2 E}{\partial y_k^2} (z_j)^2 \quad (2.18)$$

Hitung turunan kedua fungsi *error* terhadap unit *hidden* ($\frac{\partial^2 E}{\partial x_j^2}$), untuk digunakan pada *hidden layer* :

$$\frac{\partial^2 E}{\partial x_j^2} = \sum_k \frac{\partial^2 E}{\partial y_k^2} W_{jk}^2 \quad (2.19)$$

2. Pada *hidden layer*, untuk tiap-tiap unit *hidden* (Z_j, j = 1,2,3,…,p) hitung turunan kedua fungsi *error* terhadap sinyal *hidden* terbobot ($\frac{\partial^2 E}{\partial y_j^2}$) dengan menggunakan nilai turunan kedua fungsi *error* terhadap unit *hidden* ($\frac{\partial^2 E}{\partial x_j^2}$) dari *output layer* :

$$\frac{\partial^2 E}{\partial y_j^2} = \frac{\partial^2 E}{\partial x_j^2} (f'(z\_in_j))^2 \quad (2.20)$$

Kemudian nilai turunan kedua fungsi *error* terhadap sinyal *hidden* terbobot digunakan untuk menghitung nilai *hessian* pada bobot dari *input layer* ke *hidden layer* ($\frac{\partial^2 E}{\partial w_{ij}^2}$) :

$$\frac{\partial^2 E}{\partial w_{ij}^2} = \frac{\partial^2 E}{\partial y_j^2} (x_i)^2 \quad (2.21)$$

Algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* mempunyai langkah-langkah sebagai berikut *:*

1. Inisialisasi tiap-tiap bobot dengan nilai acak yang cukup kecil (mendekati nol) yaitu berkisar diantara -0.05 sampai 0.05.
2. Untuk tiap-tiap elemen lakukan proses normalisasi input dengan persamaan :

$$x' = \frac{0.8(x - minvalue)}{maxvalue - minvalue} + 0.1 \qquad (2.22)$$

   Dalam sistem pengenalan fonem ini nilai *input* tertinggi (*maxvalue*) adalah 1 dan nilai terendah (*minvalue*) adalah 0.
3. Ambil beberapa contoh pasangan elemen secara acak dan lakukan proses a sampai d yaitu:
   a. Proses *feedforward* seperti pada algoritma *backpropagation* biasa namun menggunakan fungsi aktivasi pada persamaan 2.15, untuk menghitung sinyal *output* (z_j) (dengan menggunakan persamaan 2.2) pada *hidden layer* dan untuk menghitung sinyal *output* (y_k) (dengan menggunakan persamaan 2.4) pada *output layer* yang akan digunakan untuk menghitung nilai *hessian*.
   b. Proses mendapatkan nilai *hessian* untuk masing-masing bobot, dengan menggunakan persamaan 2.17 sampai 2.21.
   c. Perbarui nilai *hessian* total dengan persamaan 2.23 untuk nilai *hessian* total pada bobot *input* ke *hidden* dan persamaan 2.24 untuk nilai *hessian* total pada bobot *hidden* ke *output.* Nilai γ digunakan agar nilai *hessian* total tidak terlalu besar. Nilai γ yang dipakai dalam sistem pengenalan fonem ini adalah 0.2.

$$\left\langle \frac{\partial^2 E}{\partial w_{ij}^2} \right\rangle = (1-\gamma) \left\langle \frac{\partial^2 E}{\partial w_{ij}^2} \right\rangle_{\text{lama}} + \gamma \frac{\partial^2 E}{\partial w_{ij}^2} \qquad (2.23)$$

$$\left\langle \frac{\partial^2 E}{\partial w_{jk}^2} \right\rangle = (1-\gamma) \left\langle \frac{\partial^2 E}{\partial w_{jk}^2} \right\rangle_{\text{lama}} + \gamma \frac{\partial^2 E}{\partial w_{jk}^2} \qquad (2.24)$$

   d. Bagi nilai *hessian* total pada persamaan 2.23 dan persamaan 2.24 dengan jumlah elemen yang diambil secara acak tadi.
4. Proses pembelajaran, untuk tiap-tiap elemen lakukan proses 5 sampai 7.
5. Proses *feedforward* seperti pada algoritma *backpropagation* biasa namun menggunakan fungsi aktivasi pada persamaan 2.15.

6. Proses menghitung nilai *error* pada tiap *layer*, pada *output layer* menggunakan persamaan 2.6 sedangkan pada *hidden layer* menggunakan persamaan 2.10.
7. Perbarui nilai bobot dengan menggunakan *learning rate* (η) yang berbeda untuk tiap-tiap bobot melalui persamaan 2.25 untuk bobot dari *input* ke *hidden layer* dan persamaan 2.26 untuk bobot dari *hidden* ke *output layer*. Nilai μ digunakan untuk menjaga agar nilai *learning rate* tidak terlalu besar apabila nilai *hessian* terlalu kecil. Nilai μ yang dipakai dalam sistem pengenalan fonem ini adalah 0.1. Sedangkan α adalah nilai *learning rate*.

$$\eta_{ij} = \frac{\alpha}{\left(\frac{\partial^2 E}{\partial w_{ij}^2}\right) + \mu} \quad (2.25)$$

$$\eta_{jk} = \frac{\alpha}{\left(\frac{\partial^2 E}{\partial w_{jk}^2}\right) + \mu} \quad (2.26)$$

Sehingga nilai bobot dari *input* ke *hidden layer* yang baru dapat dihitung dengan :

$$W_{ij} = W_{ij} \text{ lama} + \eta_{ij}\ \delta_j\ x_i \quad (2.27)$$

Dan nilai bobot dari *hidden* ke *output layer* yang baru dapat dihitung dengan :

$$W_{jk} = W_{jk} \text{ lama} + \eta_{jk}\ \delta_k\ z_j \quad (2.28)$$

8. Apabila tiap-tiap elemen telah melalui proses pembelajaran maka dilakukan proses penghitungan ulang nilai *hessian* melalui proses 3. Lakukan proses 3 sampai 8 hingga mencapai nilai *mean squared error* atau MSE (persamaan 2.29) yang dibutuhkan atau sampai *max epoch* tercapai.

$$\text{MSE} = \frac{1}{p}\sum_{p=1}\frac{1}{2}(\text{target} - y)^2 \quad (2.29)$$

Dengan p adalah jumlah elemen. Untuk mempercepat proses pembelajaran, dalam tiap *epoch* dilakukan proses pengacakan urutan elemen. Sehingga tiap elemen akan mendapat giliran pembelajaran yang berbeda-beda.

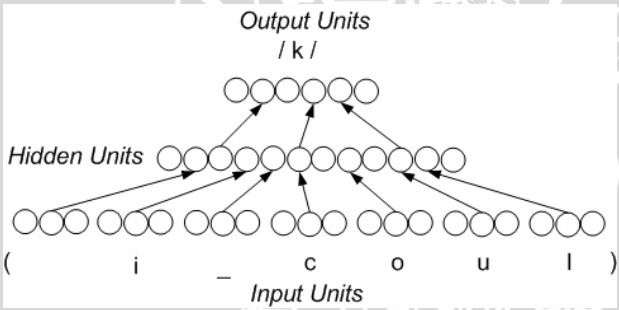
Menurut O'Neill (2006), nilai *learning rate* awal pada algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* sebaiknya dimulai dengan nilai 0.001. Kemudian setiap

memproses 120.000 elemen nilai *learning rate* berkurang dengan cara mengalikannya dengan nilai 0.7941833.

2.3. Pengenalan Fonem Menggunakan Jaringan Syaraf Tiruan

Pengenalan fonem dengan menggunakan metode jaringan syaraf tiruan memerlukan data berupa struktur pengucapan beberapa kata yang nantinya diperlukan dalam proses pembelajaran pengucapan. Salah satu contoh sistem pengenalan fonem yang berbasiskan jaringan syaraf tiruan adalah NETtalk. Struktur jaringan NETtalk adalah sebagai berikut, pada *input layer* terdapat tujuh grup unit *neuron*, yang pada setiap grup terdiri atas 29 unit *neuron*, satu grup pada *hidden layer* yang terdiri atas 80 unit *neuron* dan satu grup pada *output layer* yang terdiri atas 26 unit *neuron*.

Seperti terlihat pada Gambar 2.3, pada *input layer*, tiap-tiap grup merepresentasikan satu karakter. Sehingga masukan pada *input layer* adalah berupa *string* yang terdiri atas tujuh huruf. Sedangkan target pada *output layer* adalah berupa fonem yang diinginkan berdasarkan atas huruf yang ada di tengah atau huruf keempat beserta dengan atribut pengucapan seperti cara artikulasi dan daerah artikulasinya. Representasi karakter seperti ini disebut juga dengan istilah *window.*



Gambar 2.3 Arstitektur jaringan pada NETtalk (Sejnowski, 1986)

Teks masukan akan bergerak melewati *window* huruf demi huruf dan pada tiap langkah dihitung bobotnya untuk menentukan fonem yang sesuai dengan huruf pada *window* keempat. Contoh proses *windowing* dapat dilihat pada Tabel 2.1. Unit-unit pada *hidden layer*

hanya digunakan untuk menyelesaikan masalah pemetaan dari karakter ke fonem.

Tiap-tiap huruf, tanda baca (.) dan (-) serta tanda pemisah antar kata (_) pada *input layer* direpresentasikan dengan 29 bit yang unik (Tabel 2.2). Sehingga jumlah semua unit pada *input layer* adalah 203 unit (29 x 7). Pada *output layer*, target *output* direpresentasikan dengan 26 bit yang didalamnya sudah termasuk daerah artikulasi, cara artikulasi, fonem, intonasi dan batas suku kata (Sejnowski, 1986).

Tabel 2.1 Contoh proses *windowing* (Bakiri, 1997)

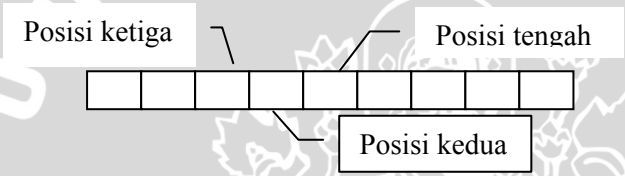
***Window* (x)**	**Target pengucapan (y)**
_ _ _ l o l l	l >
_ _ l o l l y	a 1
_ l o l l y p	l <
l o l l y p o	- >
o l l y p o p	i 0
l l y p o p _	p >
l y p o p _ _	a 2
y p o p _ _ _	p <

Tabel 2.2 Contoh representasi abjad dalam bit

Huruf	Bit
a	10000000000000000000000000000
b	01000000000000000000000000000
…	…
z	00000000000000000000000001000
-	00000000000000000000000000100
_	00000000000000000000000000010
.	00000000000000000000000000001

2.4. Metode *Second Position Asymmetric Windowing* (SPAW)

Model *window* yang dipakai pada NETtalk belum bisa mengatasi ketidakkonsistenan pada huruf yang mempunyai lebih dari satu cara pengucapan. Untuk mengatasi masalah ketidakkonsistenan ini, Arciniegas dan Embrechts mengusulkan model *window* yang disebut dengan *Second Position Asymmetric Windowing* (SPAW). Pada SPAW, huruf yang akan dipetakan ke fonemnya tidak berada pada *window* yang di tengah (*center window*) melainkan berada di tempat kedua dari *center window* (Gambar 2.4). Notasi yang dipakai pada SPAW adalah "N-M SPAW", dimana N adalah jumlah ruang (*space*) yang ada sebelum *center window* dan M adalah jumlah ruang yang ada setelah *center window*.



Gambar 2.4 Posisi dari sebuah *window*

Beberapa model SPAW yang dapat mengatasi ketidakkonsistenan dalam pengucapan suatu huruf diantarnya adalah 2-6 SPAW, 1-5 SPAW dan 3-5 SPAW (Arciniegas, 2000).

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan akan dibahas penggunaan metode serta langkah-langkah yang dilakukan dalam pembuatan perangkat lunak pengenalan fonem dalam skripsi ini.

Tahapan pembuatan perangkat lunak pengenalan fonem ini adalah sebagai berikut:

a. Mempelajari metode yang digunakan dari jurnal yang pernah ada, yang telah disinggung pada bab 1.
b. Menganalisa dan merancang perangkat lunak dengan menggunakan penggabungan metode pada penelitian sebelumnya.
c. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
d. Uji coba perangkat lunak.

3.1. Deskripsi Umum Sistem

Sistem pengenalan fonem ini akan dimulai dengan proses pembelajaran untuk mengenali vokal ‘e’ dan ‘o’ yang mempunyai lebih dari satu cara pengucapan. Proses pembelajaran vokal ‘e’ dan ‘o’ dilakukan dengan menggunakan metode *Second Position Asymmetric Windowing* (SPAW) dengan model 1-5. Untuk mendapatkan hasil yang optimal dari metode SPAW diperlukan algoritma pembelajaran jaringan syaraf tiruan *backpropagation* yang efisien. Dalam sistem pengenalan fonem ini algoritma *backpropagation* yang dipakai adalah dengan metode *stochastic diagonal Levenberg-Marquardt.*

Pada tahap pembelajaran diperlukan data pembelajaran yang berupa kata-kata yang mengandung huruf ‘e’ atau huruf ‘o’ beserta cara pengucapannya. Cara pengucapan yang dimaksudkan adalah susunan fonem dalam kata tersebut. Proses pembelajaran diperlukan untuk mengatasi masalah pengucapan huruf ‘e’ dan ‘o’ dalam Bahasa Indonesia. *Flowchart* dari proses pembelajaran terdapat pada Gambar 3.1.

Gambar 3.1 *Flowchart* proses pembelajaran.

Sebelum proses pembelajaran dimulai, dilakukan proses penentuan parameter-parameter pembelajaran oleh *user*. Parameter-parameter tersebut diantaranya adalah jumlah *neuron* pada *hidden layer*, nilai *learning rate* dan besar *max epoch.* Proses berikutnya adalah proses pembelajaran berdasarkan atas data yang ada pada

tabel kata dan parameter-parameter pembelajaran yang telah ditentukan sebelumnya.

Proses pembelajaran dimulai dengan pengambilan data tiap kata dari tabel kata. Untuk setiap kata yang ada pada tabel kata, kata bergerak melewati *window* huruf demi huruf dan pada tiap langkah, *string* dalam *window* tersebut dikonversikan menjadi 189 (27 x 7) bit. Masing-masing huruf dalam *window* direpresentasikan sebagai 27 bit, sedangkan banyak *window* yang dipakai adalah sebanyak tujuh. Pada setiap langkah dipilih target fonem yang sesuai dengan karakter pada posisi *window* kedua dari kiri. Posisi *window* kedua dari kiri merupakan posisi *center window* berdasarkan model 1-5 SPAW. Setiap target fonem kemudian dikonversikan ke dalam 30 bit (karena pada sistem pengenalan fonem ini hanya terdapat 30 fonem).

Deretan bit dari tiap *window* dan deretan bit dari target fonem pada posisi *center window*, kemudian dijadikan *input* pada proses pembelajaran dengan menggunakan algoritma *backpropagation*. Proses pembelajaran dilakukan sampai mendapat nilai *error* yang diinginkan atau telah mencapai *epoch* maksimum. Setelah proses pembelajaran selesai, kemudian dilakukan proses penyimpanan nilai bobot ke dalam tabel bobot pada *database.*

3.2. Perancangan Proses

Proses yang ada dalam sistem pengenalan fonem ini dibagi menjadi dua proses yaitu proses pembelajaran dan proses pengenalan fonem. Proses pembelajaran diperlukan untuk mengenali pola pengucapan vokal ‘e’ dan ‘o’. Sedangkan proses pengenalan fonem bertugas mengenali fonem pada suatu kata sesuai dengan bobot yang diperoleh dari proses pembelajaran.

3.2.1. Perancangan Proses Pembelajaran

Sebelum proses pembelajaran dimulai, dilakukan proses penentuan parameter-parameter pembelajaran oleh *user*. Parameter-parameter tersebut diantaranya adalah jumlah *neuron* pada *hidden layer*, nilai *learning rate* dan besar *max epoch.* Proses pembelajaran memerlukan data pembelajaran yaitu kata-kata yang mengandung huruf ’e’ atau ’o’ beserta susunan fonemnya. Data pembelajaran ini nantinya disimpan pada tabel kata. Tabel kata tersebut diisi sebelum

proses pembelajaran dimulai. Dalam sistem pengenalan fonem ini jumlah kata yang ada pada tabel kata adalah sebanyak 500 kata (Lampiran 1).

Pada saat proses pembelajaran berlangsung, setiap kata beserta fonemnya dalam tabel kata dipilih secara berurutan untuk dijadikan sebagai *input* pembelajaran. Kata yang terpilih kemudian melalui proses *windowing* dengan model *window* 1-5 SPAW, sehingga sebuah kata yang mempunyai huruf sejumlah *n* dikonversikan menjadi sejumlah *n* seri *window*. Sedangkan fonem dari kata tersebut digunakan sebagai target pembelajaran untuk tiap seri *window*-nya. Pada tiap seri *window,* fonem yang dipilih menjadi terget adalah fonem yang bersesuaian dengan huruf pada *window* kedua (Tabel 3.1).

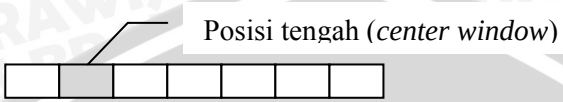
Untuk setiap seri *window,* tiap abjad dikonversikan ke dalam bentuk deretan bit dengan menggunakan data pada tabel abjad. Sedangkan setiap target fonem dikonversikan ke dalam bentuk deretan bit dengan menggunakan data pada tabel fonem. Deretan bit dalam tiap seri *window* kemudian dijadikan sebagai *input* pada proses pembelajaran dengan jaringan syaraf tiruan *backpropagation* sedangkan targetnya adalah deretan bit dari target fonem.

Proses pembelajaran dengan jaringan syaraf tiruan *backpropagation* berhenti apabila telah mencapai *max epoch* atau nilai *mean squared error* telah memenuhi. Nilai bobot yang diperoleh dari proses pembelajaran dengan jaringan syaraf tiruan *backpropagation* kemudian disimpan dalam sebuah tabel bobot. Berikut ini akan dijelaskan lebih terperinci masing-masing proses dalam proses pembelajaran yang meliputi :

1. Proses *windowing* dengan model 1-5 SPAW

Lebar *window* dengan model 1-5 SPAW adalah 7 *window* (Gambar 3.2). Sebuah kata direpresentasikan menjadi beberapa seri *window* tergantung pada jumlah huruf yang ada pada kata tersebut. Huruf pertama suatu kata ditempatkan pada *window* kedua dari kiri, baru diikuti dengan huruf kedua dan seterusnya. Pada seri berikutnya huruf pertama kata tersebut ditempatkan pada *window* pertama dari kiri, demikian proses penempatan huruf berlangsung sampai mencapai huruf terakhir kata tersebut. Pada tiap seri *window* dipilih fonem yang bersesuaian dengan huruf pada *window* kedua dari kiri.

Contoh proses *windowing* dengan model 1-5 SPAW ditunjukkan pada Tabel 3.1.



Gambar 3.2 Model *window* 1-5 SPAW

Tabel 3.1 Contoh proses *windowing* dengan model 1-5 SPAW

Seri *window* (x)	Target fonem (y)
_ b e l a j a	/b/
b e l a j a r	/ê/
e l a j a r _	/l/
l a j a r _ _	/a/
a j a r _ _ _	/j/
j a r _ _ _ _	/a/
a r _ _ _ _ _	/r/

2. Proses konversi *string* pada seri *window* menjadi deretan bit

Setelah melalui proses *windowing* dengan model 1-5 SPAW, setiap karakter pada tiap *window* dikonversikan menjadi bit sepanjang 27 digit. Panjang bit sepanjang 27 digit didasarkan atas jumlah abjad yang ada yaitu sebanyak 26 ditambah dengan tanda (_) untuk merepresentasikan karakter kosong. Proses konversi dilakukan dengan bantuan tabel abjad (Tabel 3.2) yang di dalamnya berisi abjad beserta representasi deretan bitnya. Jumlah total digit bit hasil konversi pada model 1-5 SPAW adalah sebanyak 189 (27 x 7) digit untuk tiap seri *window*. Deretan bit untuk tiap seri *window* kemudian dijadikan sebagai *input* pada proses pembelajaran dengan jaringan syaraf tiruan *backpropagation.*

Tabel 3.2 Tabel abjad

abjad	Deretan bit
a	10000000000000000000000000
b	01000000000000000000000000
c	00100000000000000000000000
…	…
z	00000000000000000000000010
-	00000000000000000000000001

3. Proses konversi fonem menjadi deretan bit

Setiap fonem yang menjadi target pada tiap seri *window* dari proses *windowing* dengan model 1-5 SPAW dikonversikan menjadi bit sepanjang 30 digit. Panjang bit sepanjang 30 digit didasarkan atas jumlah fonem yang ada pada sistem pengenalan fonem ini yaitu sebanyak 30 fonem. Proses konversi dilakukan dengan bantuan tabel fonem (Tabel 3.3) yang didalamnya berisi simbol fonem beserta representasi deretan bitnya. Deretan bit hasil konversi kemudian dijadikan sebagai target pada proses pembelajaran dengan jaringan syaraf tiruan *backpropagation.*

Tabel 3.3 Tabel fonem

fonem	contoh	Deretan bit
a		100000000000000000000000000000
b		010000000000000000000000000000
c		001000000000000000000000000000
d		000100000000000000000000000000
e	sate	000010000000000000000000000000
ê	tipe,bandeng	000001000000000000000000000000
ë	perak,remeh	000000100000000000000000000000
f		000000010000000000000000000000
…		…

fonem	contoh	Deretan bit
n		00000000000000010000000000000000
o	toko,soto	00000000000000001000000000000000
ö	tobat,tokoh	00000000000000000100000000000000
p		00000000000000000010000000000000
…		…
z		00000000000000000000000000000010
-		00000000000000000000000000000001

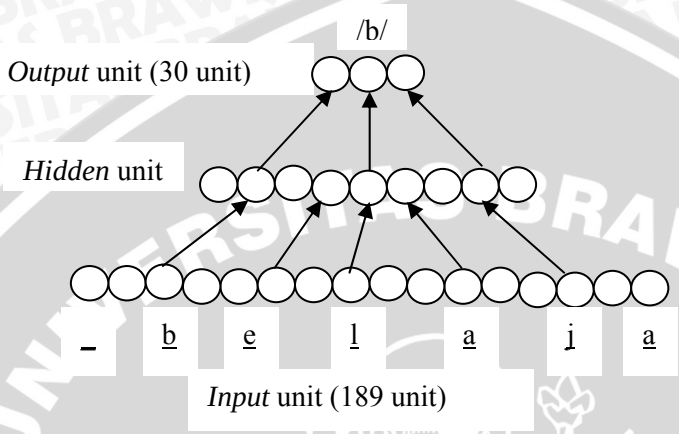
4. Proses pembelajaran fonem menggunakan jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt*

Proses pembelajaran fonem dimulai dengan proses memasukkan data yang berupa deretan bit yang merepresentasikan tiap seri *window* dan deretan bit yang merepresentasikan target fonem dari tiap seri *window* tersebut. Tiap deretan bit kemudian dikelompokkan sesuai dengan asal katanya, yang nantinya disebut sebagai satu *pattern*.

Parameter-parameter untuk proses pembelajaran dengan jaringan syaraf tiruan *backpropagation* (yang telah ditentukan oleh *user* sebelumnya) adalah jumlah *neuron* pada *hidden layer*, nilai *learning rate* dan besar *max epoch*. Standar nilai *learning rate* yang dipakai dalam sistem pengenalan fonem ini adalah 0.001. Sedangkan standar besar *max epoch* yang dipakai adalah sebanyak 2000 *epoch*. Nilai *learning rate* akan berkurang tiap melakukan pembelajaran sebanyak 120.000 seri *window*, dengan cara mengalikannya dengan nilai 0.7941833. Proses pengurangan nilai *learning rate* perlu dilakukan untuk mencegah agar kondisi jaringan tidak divergen, karena nilai *learning rate* yang terlalu besar cenderung mempercepat perubahan pada bobot. Pengurangan nilai *learning rate* berhenti apabila telah mencapai nilai 0.00005. Arsitektur jaringan yang dipakai pada proses pembelajaran dapat dilihat pada Gambar 3.3.

Untuk mempercepat proses pembelajaran maka setiap *input* dinormalisasi terlebih dahulu dengan menggunakan persamaan 2.22. Selain itu tiap-tiap bobot pada tiap *layer* diinisialisasi menggunakan

nilai acak yang cukup kecil (mendekati nol) yaitu berkisar diantara -0.05 sampai 0.05.



Gambar 3.3 Arsitektur jaringan pada proses pembelajaran

Setelah proses normalisasi dan inisialisasi bobot selesai, kemudian dilajutkan dengan proses pembelajaran dengan algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt.*

Algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* dimulai dengan proses perhitungan nilai *hessian* total. Untuk mendapatkan nilai *hessian* total mula-mula dipilih *pattern* secara acak sejumlah setengah dari jumlah total *pattern* yang ada. Dalam satu *pattern* terdiri atas beberapa pasangan deretan bit *input* dan *target.*

Untuk tiap-tiap *pattern* yang terpilih, dilakukan proses *feedforward* terhadap deretan bit *input* dengan menggunakan fungsi aktivasi pada persamaan 2.15. Setelah proses *feedforward,* kemudian dilanjutkan dengan proses perhitungan nilai *hessian* dengan menggunakan persamaan 2.17 sampai 2.21 untuk tiap-tiap *input*. Setelah perhitungan nilai *hessian,* dilanjutkan dengan proses memperbarui nilai *hessian* total menggunakan persamaan 2.23 dan 2.24.

Nilai *hessian* total yang didapatkan setelah proses *feedforward* dan proses perhitungan nilai *hessian* pada tiap-tiap *pattern* yang

terpilih, kemudian dibagi dengan jumlah *pattern* yang terpilih tersebut. *Pseudo code* untuk menghitung nilai *hessian* total dapat dilihat pada Gambar 3.4.

```
Acak urutan m_vOrder
for sample=1 to JumKata/2
  urutankata = m_vOrder[sample]
  //tiap seri window
  untuk tiap seriwindow dalam kata[urutankata]
    FeedForward(seriwindow)
    Hitung Hessian(seriwindow)
    Perbarui nilai Hessian Total
Bagi Nilai Hessian Total dengan JumKata/2
```

Gambar 3.4 *Pseudo code* untuk menghitung nilai *hessian* total

Setelah didapatkan nilai *hessian* total, kemudian dilanjutkan dengan proses pembelajaran untuk tiap-tiap *pattern.* Untuk tiap-tiap *pattern* mula-mula dilakukan proses *feedforward. Input* dari proses *feedforward* adalah deretan bit dari sebuah seri *window* pada sebuah *pattern.* Persamaan yang dipakai dalam proses *feedforward* adalah persamaan 2.2 sampai 2.5 dan fungsi aktivasi yang dipakai adalah persamaan 2.15.

Setelah proses *feedforward* selesai, kemudian dilanjutkan dengan proses perhitungan nilai *error.* Proses perhitungan nilai *error* antara *output* dengan *hidden layer* dilakukan dengan cara menghitung selisih antara nilai hasil aktivasi dengan nilai dari target fonem pada sebuah seri *window.* Persamaan yang dipakai pada proses perhitungan nilai *error* antara *output layer* dengan *hidden layer* adalah persamaan 2.6. Sedangkan proses perhitungan nilai *error* antara *hidden layer* dengan *input layer* dilakukan dengan menggunakan persamaan 2.9 dan 2.10.

Setelah proses perhitungan nilai *error* selesai, kemudian dilanjutkan dengan proses pembaharuan nilai bobot. Nilai bobot antara *output* dengan *hidden layer* diperbarui dengan menggunakan persamaan 2.27. Sedangkan nilai bobot antara *hidden* dengan *input layer* diperbarui dengan menggunakan persamaan 2.28.

Setelah semua *pattern* telah melalui proses pembelajaran, kemudian dilanjutkan dengan proses perhitungan nilai MSE Total

(persamaan 2.29) dan proses perhitungan nilai *average error* untuk semua *pattern. Pseudo code* untuk menghitung nilai MSE Total dan nilai *average error* untuk semua *pattern* dapat dilihat pada Gambar 3.5.

```
m_dMSE = 0
m_dAvgError = 0
for urutan = 1 to JumKata
  //tiap seri window
  untuk tiap seriwindow dalam kata[urutan]
    FeedForward(seriwindow)
    For i=1 to NumOutput
        //error pada tiap output
        selisih = fabs(seriwindow.output[i]-y[i]);
        m_dMSE = m_dMSE + (selisih*selisih)
       m_dAvgError =  m_dAvgError + selisih
m_dMSE = m_dMSE / (jumlahseriwindow*2);
m_dAvgError = m_dAvgError / jumlahseriwindow
```

Gambar 3.5 *Pseudo code* untuk menghitung nilai *error*

Setelah proses perhitungan nilai MSE total selesai, kemudian dilakukan proses pengacakan urutan *pattern*. Proses pengacakan urutan *pattern* bertujuan untuk mendapatkan urutan pembelajaran kata yang berbeda-beda pada tiap *epoch*.

Proses pembelajaran *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* dilakukan sampai nilai MSE pada proses *backpropagation* memenuhi target yang telah ditentukan sebelumnya atau telah mencapai *max epoch*. Standar target nilai MSE yang dipakai dalam sistem pengenalan fonem ini adalah 0.01.

3.2.2. Proses pengenalan fonem

Proses pengenalan fonem pada sebuah kata dimulai dengan proses *windowing* dengan model 1-5 SPAW. Setiap seri *window* kemudian dikonversikan kedalam bentuk deretan bit sesuai dengan tabel abjad. Deretan bit hasil konversi tiap seri *window*, kemudian dikenali menggunakan jaringan syaraf tiruan *backpropagation* lewat proses *feedforward* (persamaan 2.2 sampai 2.5). Bobot yang dipakai pada proses *feedforward* adalah bobot yang diperoleh setelah proses pembelajaran. Dari proses *feedforward* didapatkan deretan bit yang

kemudian dikonversikan ke dalam sebuah fonem lewat tabel fonem. Hasil dari proses pengenalan fonem adalah kata dalam bentuk fonemnya.

3.3. Contoh Proses Pembelajaran

Pada contoh permasalahan proses pembelajaran data yang dipakai hanya sebanyak 2 kata (Tabel 3.4). Dengan jumlah unit pada *hidden layer* sebanyak 2 unit, nilai *learning rate* yang dipakai adalah 0.001 dan nilai target *error* yang ingin dicapai adalah 0.01.

Tabel 3.4 Contoh kata

Kata	fonem
absensi	absënsi
adegan	adêgan

Sebelum proses pembelajaran, dilakukan proses inisialisasi yang meliputi proses konversi tiap seri *window* dan proses inisialisasi bobot awal. Berikut adalah langkah-langkah pada proses inisialisasi :

1. Setiap kata diubah menjadi beberapa seri *window* sesuai dengan jumlah huruf pada kata tersebut (Tabel 3.5).

Tabel 3.5 Tabel seri *window*

Seri *window*	*Input*							Target
1	-	a	b	s	e	n	s	a
2	a	b	s	e	n	s	i	b
3	b	s	e	n	s	i	-	s
4	s	e	n	s	i	-	-	ë
5	e	n	s	i	-	-	-	n
6	n	s	i	-	-	-	-	s
7	s	i	-	-	-	-	-	i
1	-	a	d	e	g	a	n	a
2	a	d	e	g	a	n	-	d
3	d	e	g	a	n	-	-	ê

Seri *window*	*Input*							Target
4	e	g	a	n	-	-	-	g
5	g	a	n	-	-	-	-	a
6	a	n	-	-	-	-	-	n

2. Tiap-tiap seri *window* dari Tabel 3.5 dikonversikan menjadi deretan bit dengan bantuan tabel abjad (Tabel 3.6).

Tabel 3.6 Deretan bit dari tiap seri *window*

#	*Input*	Target
1	00000000000000000000000000100100000000000000 00000000000000010000000000000000000000000000 00000000000000000100000000000001000000000000 00000000000000000000000000100000000000000000 00000000000000010000000000	100000 000000 000000 000000 000000
2	10000000000000000000000000000010000000000000 00000000000000000000000000000000100000000000 00010000000000000000000000000000000000000100 00000000000000000000000000000010000000000000 00000010000000000000000000	100000 000000 000000 000000 000000
3	01000000000000000000000000000000000000000000 00010000000000000100000000000000000000000000 00000000000010000000000000000000000000000000 00100000000000000000100000000000000000000000 00000000000000000000000100	000000 000000 000000 000100 000000
4	00000000000000000010000000000000010000000000 00000000000000000000000000010000000000000000 00000000000000000100000000000000000010000000 00000000000000000000000000000000000000010000 00000000000000000000000100	000000 100000 000000 000000 000000
5	00001000000000000000000000000000000000000010 00000000000000000000000000000000100000000000 00000001000000000000000000000000000000000000 00000000001000000000000000000000000000010000 00000000000000000000000100	000000 000000 000100 000000 000000

#	*Input*	Target
6	000000000000010000000000000000000000000000 000100000000000000001000000000000000000000 000000000000000000000001000000000000000000 000000000010000000000000000000000000010000 00000000000000000000000100	000000 000000 000000 000100 000000
7	000000000000000001000000000000000001000000 000000000000000000000000000000000000001000 000000000000000000000001000000000000000000 000000000010000000000000000000000000010000 00000000000000000000000100	000000 000010 000000 000000 000000
1	000000000000000000000000001001000000000000000 000000000000000010000000000000000000000000 000100000000000000000000000000000001000000000 000000000000010000000000000000000000000000 00000000001000000000000	100000 000000 000000 000000 000000
2	100000000000000000000000000000100000000000 000000000000000001000000000000000000000000 000001000000000000000000000010000000000000 000000000000000000000000010000000000000000 00000000000000000000000100	000100 000000 000000 000000 000000
3	000100000000000000000000000000010000000000 000000000000000000010000000000000000000001 000000000000000000000000000000000000000100 000000000000000000000000000000000000010000 00000000000000000000000100	000001 000000 000000 000000 000000
4	000010000000000000000000000000000100000000 000000000000010000000000000000000000000000 000000000001000000000000000000000000000000 000000000100000000000000000000000000010000 00000000000000000000000100	000000 001000 000000 000000 000000
5	000000100000000000000000000001000000000000 000000000000000000000000001000000000000000 000000000000000000000000100000000000000000 000000000100000000000000000000000000010000 00000000000000000000000100	100000 000000 000000 000000 000000

#	*Input*	Target
6	10000000000000000000000000000000000000000010	000000
	00000000000000000000000000000000000000001000	000000
	00000000000000000000000010000000000000000000	000100
	00000000001000000000000000000000000010000	000000
	000000000000000000000000100	000000

3. Setiap bit dari Tabel 3.6 dinormalisasi menggunakan persamaan 2.16 sehingga nilai 0 berubah menjadi 0.1 sedangkan nilai 1 berubah menjadi 0.8 (Tabel 3.7). Dalam Tabel 3.7 yang dijadikan contoh hanya seri pertama *window* dari kata 'absensi'.

Tabel 3.7 Hasil normalisasi

#	*Input*
1	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.9 0.1 0.1 0.9 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.9 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.8 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.8 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.9 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	0.9 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
	Target
	10000000000000000000000000000

4. Inisialisasi bobot awal dari *input layer* ke *hidden layer* (Tabel 3.8). Dalam Tabel 3.8 bobot yang diperlihatkan hanya 10 bobot pertama karena terlalu banyaknya data yaitu sebanyak 189 bobot.

Tabel 3.8 Nilai bobot awal dari *input layer* ke *hidden layer* (v_{ij})

	Z_1	Z_2
X_1	-0.0493	-0.0491
X_2	0.0441	-0.0060
X_3	0.0102	0.0008
X_4	0.0286	0.0106
X_5	0.0077	0.0259
X_6	-0.0358	-0.0099
X_7	-0.0278	-0.0150
X_8	-0.0117	0.0343
X_9	-0.0496	0.0442
X_{10}	-0.0082	0.0123

5. Inisialisasi bobot awal dari *hidden layer* ke *output layer* (Tabel 3.9). Dalam Tabel 3.9 bobot yang diperlihatkan hanya 10 bobot pertama karena terlalu banyaknya data yaitu sebanyak 60 bobot.

Tabel 3.9 Nilai bobot awal dari *hidden layer* ke *output layer* (w_{jk})

	Z_1	Z_2
Y_1	0.0212	0.0499
Y_2	0.0276	0.0142
Y_3	0.0121	0.0446
Y_4	0.0188	-0.0210
Y_5	0.0067	-0.0210
Y_6	0.0048	0.0212
Y_7	-0.0254	0.0047
Y_8	-0.0007	0.0452
Y_9	0.0170	0.0265
Y_{10}	0.0123	-0.0054

Proses pembelajaran dimulai dengan proses perhitungan nilai *hessian* total (proses a) dan kemudian dilanjutkan dengan proses pembelajaran untuk tiap-tiap *window* (proses b sampai h).

a. Proses perhitungan nilai *hessian* total dimulai dengan proses pengambilan beberapa kata secara acak untuk mendapatkan nilai

hessian total awal. Dalam contoh berikut yang dijadikan contoh sebagai kata yang terpilih adalah kata 'absensi' dan yang dijadikan contoh perhitungan nilai *hessian* hanya seri pertama *window* dari kata tersebut. Nilai *hessian* total kemudian dibagi dengan jumlah dari kata yang terpilih secara acak (dalam contoh berikut jumlah kata yang terpilih hanya satu).

Berikut adalah contoh proses perhitungan nilai *hessian* total :

1. Untuk tiap-tiap seri *window* dalam satu kata yang terpilih, lakukan perhitungan nilai *hessian* :

   1.1 Hitung nilai *output* dari unit-unit di *hidden layer*

   $z\_in_j = \sum_{i=1}^{n} x_i v_{ji}$

   $z\_in_1 = 0.0597$

   $z\_in_2 = 0.0436$

   $z_j = f(z\_in_j)$

   $z_j = 1.7159 \tanh(\frac{2}{3} x)$

   $z_1 = 0.0682$

   $z_2 = 0.0499$

   1.2 Hitung nilai *output* dari unit-unit di *output layer.*

   Dalam Tabel 3.10 hanya ditampilkan 10 nilai *output* pertama saja dari 30 nilai *output*.

   $y\_in_k = \sum_{j=1}^{m} z_j w_{kj}$

   $y_k = f(y\_in_k) = 1.7159 \tanh(\frac{2}{3} y\_in_k)$

Tabel 3.10 Nilai *output* dari unit-unit di *output layer*

K	Y_in	Y
1	0.0039	0.0045
2	0.0026	0.0030
3	0.0031	0.0035
4	0.0002	0.0003
5	-0.0006	-0.0007
6	0.0014	0.0016
7	-0.0015	-0.0017
8	0.0022	0.0025

K	Y_in	Y
9	0.0025	0.0028
10	0.0006	0.0007

1.3 Hitung nilai turunan kedua fungsi *error* terhadap sinyal *output* terbobot $\left(\frac{\partial^2 E}{\partial y_k^2}\right)$.

$$\frac{\partial^2 E}{\partial y_k^2} = \{f'(y\_in_k)\}^2.$$

Dalam Tabel 3.11 hanya ditampilkan sepuluh nilai turunan kedua fungsi *error* terhadap sinyal *output* terbobot pertama saja dari 30 nilai.

Tabel 3.11 Nilai turunan kedua fungsi *error* terhadap sinyal *output* terbobot

K	$\frac{\partial^2 E}{\partial y_k^2}$
1	1.3086
2	1.3086
3	1.3086
4	1.3086
5	1.3086
6	1.3086
7	1.3086
8	1.3086
9	1.3086
10	1.3086

1.4 Hitung nilai *hessian* total pada bobot dari *hidden layer* ke *output layer* $\left\langle \frac{\partial^2 E}{\partial w_{jk}^2} \right\rangle$.

$$\left\langle \frac{\partial^2 E}{\partial w_{jk}^2} \right\rangle = (1\text{-}0.2) \left\langle \frac{\partial^2 E}{\partial w_{jk}^2} \right\rangle \text{lama} + 0.2 \frac{\partial^2 E}{\partial w_{jk}^2},$$

jika diketahui $\frac{\partial^2 E}{\partial w_{jk}^2} = \frac{\partial^2 E}{\partial y_k^2}(z_j)^2$.

Dalam Tabel 3.12 hanya ditampilkan sepuluh nilai *hessian* total pertama saja dari 30 nilai.

Tabel 3.12 Nilai *hessian* total pada bobot dari *hidden layer* ke *output layer*

$\left\langle \frac{\partial^2 E}{\partial w_{jk}^2} \right\rangle$	Z_1	Z_2
Y_1	0.0012	0.0007
Y_2	0.0012	0.0007
Y_3	0.0012	0.0007
Y_4	0.0012	0.0007
Y_5	0.0012	0.0007
Y_6	0.0012	0.0007
Y_7	0.0012	0.0007
Y_8	0.0012	0.0007
Y_9	0.0012	0.0007
Y_{10}	0.0012	0.0007

1.5 Hitung nilai turunan kedua fungsi *error* terhadap unit *hidden* ($\frac{\partial^2 E}{\partial x_j^2}$).

$$\frac{\partial^2 E}{\partial x_j^2} = \sum_k \frac{\partial^2 E}{\partial y_k^2} W_{jk}^2.$$

Nilai turunan kedua fungsi *error* terhadap unit *hidden* dapat dilihat pada Tabel 3.13.

Tabel 3.13 Nilai turunan kedua fungsi *error* terhadap unit *hidden*

J	$\frac{\partial^2 E}{\partial x_j^2}$
1	0.0189
2	0.0409

1.6 Hitung nilai turunan kedua fungsi *error* terhadap sinyal *hidden* terbobot ($\frac{\partial^2 E}{\partial y_j^2}$).

$$\frac{\partial^2 E}{\partial y_j^2} = \frac{\partial^2 E}{\partial x_j^2} (f'(z\_in_j))^2.$$

Nilai turunan kedua fungsi *error* terhadap sinyal *hidden* terbobot dapat dilihat pada Tabel 3.14.

Tabel 3.14 Nilai turunan kedua fungsi *error* terhadap sinyal *hidden* terbobot

J	$\frac{\partial^2 E}{\partial y_j^2}$
1	0.0247
2	0.0534

1.7 Hitung nilai *hessian* total pada bobot dari *input layer* ke *hidden layer* $\left\langle \frac{\partial^2 E}{\partial w_{ij}^2} \right\rangle$.

$$\left\langle \frac{\partial^2 E}{\partial w_{ij}^2} \right\rangle = (1\text{-}0.2) \left\langle \frac{\partial^2 E}{\partial w_{ij}^2} \right\rangle \text{lama} + 0.2 \frac{\partial^2 E}{\partial w_{ij}^2},$$

jika diketahui $\frac{\partial^2 E}{\partial w_{ij}^2} = \frac{\partial^2 E}{\partial y_j^2} (x_i)^2$.

Dalam Tabel 3.15 hanya ditampilkan sepuluh nilai *hessian* total pada bobot dari *input layer* ke *hidden layer* pertama saja dari 30 nilai.

Tabel 3.15 Nilai *hessian* total pada bobot dari *input layer* ke *hidden layer*

$\left\langle \frac{\partial^2 E}{\partial w_{ij}^2} \right\rangle$	Z_1	Z_2
X_1	0	1
X_2	0	1

$\left(\frac{\partial^2 E}{\partial w_{ij}^2}\right)$	Z_1	Z_2
X_3	0	1
X_4	0	1
X_5	0	1
X_6	0	1
X_7	0	1
X_8	0	1
X_9	0	1
X_{10}	0	1

b. Setelah mendapatkan nilai *hessian* total, dilanjutkan proses pembelajaran untuk tiap-tiap kata. Untuk tiap-tiap seri *window* pada semua kata, lakukan perhitungan :

1. Hitung nilai *output* dari unit-unit di *hidden layer*

   $z\_in_j = \sum_{i=1}^{n} x_i v_{ji}$

   $z\_in_1 = 0.0597$

   $z\_in_2 = 0.0436$

   $z_j = f(z\_in_j)$

   $z_j = 1.7159 \tanh(\frac{2}{3} x)$

   $z_1 = 0.0682$

   $z_2 = 0.0499$

2. Hitung nilai *output* dari unit-unit di *output layer*.
   Dalam Tabel 3.16 hanya ditampilkan sepuluh nilai *output* pertama saja dari 30 nilai *output*.

   $y\_in_k = \sum_{j=1}^{m} z_j w_{kj}$

   $y_k = f(y\_in_k) = 1.7159 \tanh(\frac{2}{3} y\_in_k)$

Tabel 3.16 Nilai *output* dari unit-unit di *output layer*

K	Y_in	Y
1	0.0039	0.0045
2	0.0026	0.0030
3	0.0031	0.0035
4	0.0002	0.0003

K	Y_in	Y
5	-0.0006	-0.0007
6	0.0014	0.0016
7	-0.0015	-0.0017
8	0.0022	0.0025
9	0.0025	0.0028
10	0.0006	0.0007

c. Hitung nilai informasi *error* (δ) di unit-unit *output* (y).
Dalam Tabel 3.17 nilai δ yang dihitung hanya sepuluh pertama dari total nilai δ sebanyak 30.
$\delta_k = (\text{target} - y)f'(y\_in_k)$

Tabel 3.17 Nilai informasi *error* di unit-unit *output* (δ)

K	δ
1	1.1388
2	-0.0034
3	-0.0040
4	-0.0003
5	0.0008
6	-0.0018
7	0.0020
8	-0.0029
9	-0.0032
10	-0.0007

d. Hitung nilai informasi *error* (δ) di unit-unit *hidden* (z) (Tabel 3.18).
$\delta_j = \delta\_in_j f'(z\_in_j)$

Tabel 3.18 Nilai informasi *error* di unit-unit *hidden* (δ)

Z	δ
1	0.0263
2	0.0631

e. Hitung nilai *learning rate* untuk bobot dari *input layer* ke *hidden layer* (η_{ij}) (Tabel 3.19).

$$\eta_{ij} = \frac{\alpha}{\left(\frac{\partial^2 E}{\partial w_{ij}^2}\right) + \mu}$$

Tabel 3.19 Nilai *learning rate* untuk bobot dari *input layer* ke *hidden layer*

η_{ij}	Z_1	Z_2
X_1	0.0095	0.0089
X_2	0.0100	0.0100
X_3	0.0100	0.0100
X_4	0.0098	0.0095
X_5	0.0097	0.0095
X_6	0.0100	0.0100
X_7	0.0097	0.0093
X_8	0.0100	0.0100
X_9	0.0100	0.0100
X_{10}	0.0100	0.0100

f. Hitung nilai *learning rate* untuk bobot dari *hidden layer* ke *output layer* (η_{jk}) (Tabel 3.20).

$$\eta_{jk} = \frac{\alpha}{\left(\frac{\partial^2 E}{\partial w_{jk}^2}\right) + \mu}$$

Tabel 3.20 Nilai *learning rate* untuk bobot dari *hidden layer* ke *output layer*

η_{jk}	Z_1	Z_2
Y_1	0.0098	0.0095
Y_2	0.0098	0.0095
Y_3	0.0098	0.0095
Y_4	0.0098	0.0095
Y_5	0.0098	0.0095
Y_6	0.0098	0.0095

η_{jk}	Z_1	Z_2
Y_7	0.0098	0.0095
Y_8	0.0098	0.0095
Y_9	0.0098	0.0095
Y_{10}	0.0098	0.0095

g. Hitung nilai bobot baru dari *hidden layer* ke *output layer* (W_{jk}) (Tabel 3.21).
$W_{jk} = W_{jk}$ lama + $\eta_{jk}\ \delta_k\ z_j$

Tabel 3.21 Nilai bobot baru dari *hidden layer* ke *output layer*

	Z_1	Z_2
Y_1	0.0219	0.0504
Y_2	0.0275	0.0142
Y_3	0.0121	0.0446
Y_4	0.0188	-0.0210
Y_5	0.0067	-0.0210
Y_6	0.0048	0.0212
Y_7	-0.0254	0.0047
Y_8	-0.0007	0.0452
Y_9	0.0170	0.0265
Y_{10}	0.0123	-0.0054

h. Hitung nilai bobot baru dari *input layer* ke *hidden layer* (W_{ij}) (Tabel 3.22).
$W_{ij} = W_{ij}$ lama + $\eta_{ij}\ \delta_j\ x_i$

Tabel 3.22 Nilai bobot baru dari *input layer* ke *hidden layer*

	Z_1	Z_2
X_1	-0.0493	-0.0490
X_2	0.0441	-0.0060
X_3	0.0102	0.0008
X_4	0.0286	0.0107
X_5	0.0077	0.0259

	Z_1	Z_2
X_6	-0.0357	-0.0098
X_7	-0.0277	-0.0150
X_8	-0.0117	0.0343
X_9	-0.0495	0.0443
X_{10}	-0.0082	0.0100

i. Setiap satu *epoch*, dilakukan perhitungan nilai MSE dan nilai *average error*. Setiap selesai melatih 120.000 kata, dilakukan proses pengurangan nilai *learning rate* dengan cara mengalikannya dengan nilai 0.7941833. Proses pengurangan nilai *learning rate* berhenti dilakukan apabila nilai *learning rate* mencapai nilai 0.00005. Pengurangan nilai *learning rate* perlu dilakukan agar nilai *error* tetap konvergen menurun. Pelatihan dilakukan sampai nilai MSE telah memenuhi atau *max epoch* telah tercapai.

3.4. Contoh Proses Pengenalan Fonem

Pada contoh proses pengenalan fonem dimisalkan kata yang akan dikenali adalah kata 'absensi'. Berikut adalah langkah-langkah pada proses pengenalan fonem :

1. Kata 'absensi' diubah menjadi beberapa seri *window* sesuai dengan jumlah huruf yang ada (Tabel 3.23).

Tabel 3.23 Tabel seri *window*

Seri *window*	*Input*						
1	-	a	b	s	e	n	s
2	a	b	s	e	n	s	i
3	b	s	e	n	s	i	-
4	s	e	n	s	i	-	-
5	e	n	s	i	-	-	-
6	n	s	i	-	-	-	-
7	s	i	-	-	-	-	-

2. Setiap seri *window* kemudian dikonversikan menjadi deretan bit seperti pada contoh proses pembelajaran (sub bab 3.3). Untuk

setiap seri *window*, setiap huruf pada tiap *window* dikonversikan menjadi deretan bit sesuai dengan data pada tabel abjad. Berikut adalah contoh seri *window* pertama yang dikonversikan menjadi deretan bit (Tabel 3.24).

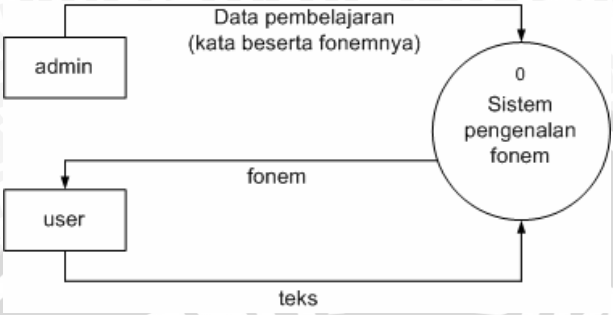
Tabel 3.24 Hasil konversi satu seri *window*

Seri *window*						
-	a	b	s	e	n	s
Deretan Bit						
00000000000000000000000000100100000000000000000000000000 00000100000000000000000000000000000000000000000000000100 00000000000010000000000000000000000000000000000001000 00000000000000000000000000000010000000000						

3. Deretan bit dari setiap seri *window* kemudian dijadikan *input* untuk proses *feedforward* (persamaan 2.2 sampai 2.5). Bobot yang dipakai pada proses *feedforward* adalah bobot yang diperoleh setelah proses pembelajaran fonem. Contoh proses *feedforward* dapat dilihat pada contoh proses pembelajaran (sub bab 3.3).
4. Proses *feedforward* untuk setiap seri *window* menghasilkan *output* berupa deretan bit yang kemudian dikonversikan menjadi sebuah fonem. Proses konversi deretan bit menjadi fonem dilakukan sesuai dengan data pada tabel fonem.
5. Setiap fonem yang diperoleh setelah proses *feedforward* dan konversi deretan bit kemudian disusun sesuai dengan urutan tiap seri *window*.

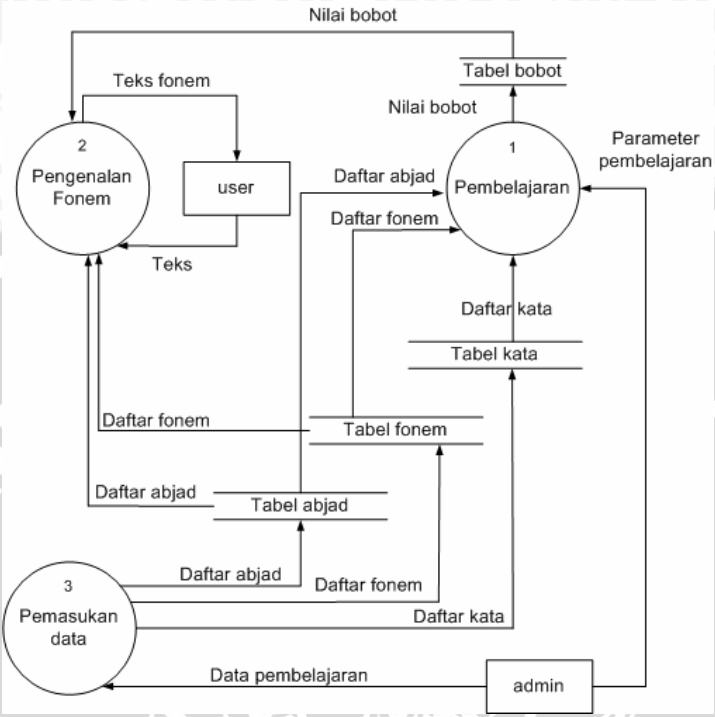
3.5. Perancangan *Data Flow Diagram* (DFD)

Dari perancangan proses yang telah dijelaskan pada sub bab 3.2, kemudian dilakukan perancangan DFD. Pada DFD level 0 (Gambar 3.6) dari sistem pengenalan fonem ini terdapat dua entitas yaitu *admin* dan *user*. Entitas *admin* memberikan *input* berupa data pembelajaran. Entitas *user* memberikan *input* berupa teks yang kemudian diolah oleh sistem pengenalan fonem sehingga menghasilkan fonem dari *input* teks tersebut.



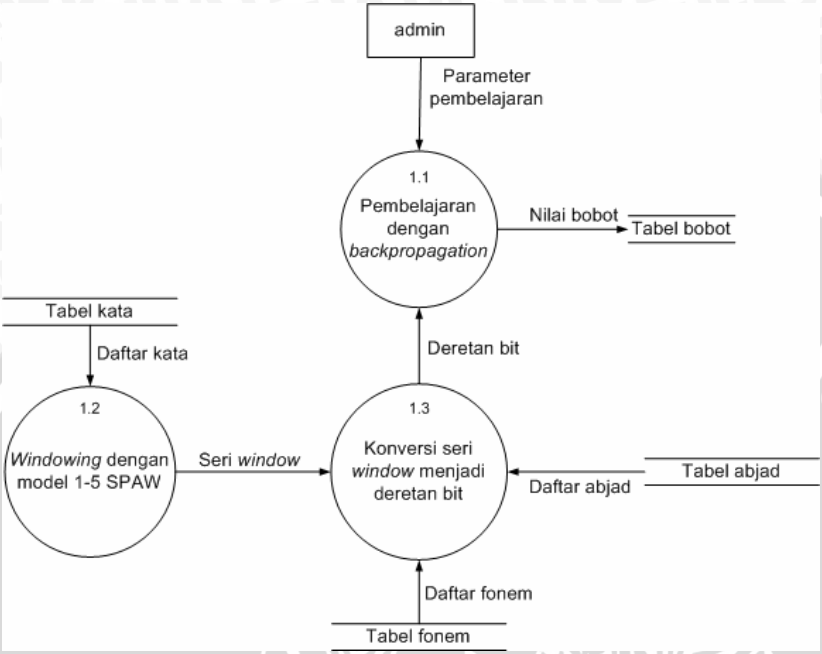
Gambar 3.6 DFD level 0 (*Context Diagram*) sistem pengenalan fonem

Pada DFD level 1 (Gambar 3.7) dari sistem pengenalan fonem ini terdapat dua entitas yaitu *admin* dan *user* serta serta tiga proses yaitu proses pengenalan fonem, proses pembelajaran dan proses pemasukan data. Entitas *admin* berperan dalam proses pembelajaran dan proses pemasukan data pembelajaran. Sedangkan entitas *user* berperan dalam proses pengenalan fonem. Proses pemasukan data bertujuan untuk menyimpan semua data pembelajaran dalam media penyimpanan yang berupa tabel. Proses pembelajaran bertujuan untuk mendapatkan nilai bobot yang kemudian disimpan pada tabel bobot. Proses pengenalan fonem bertujuan untuk mengolah teks masukan dari *user* menjadi teks fonem yang akan ditampilkan ke *user*.



Gambar 3.7 DFD level 1 sistem pengenalan fonem

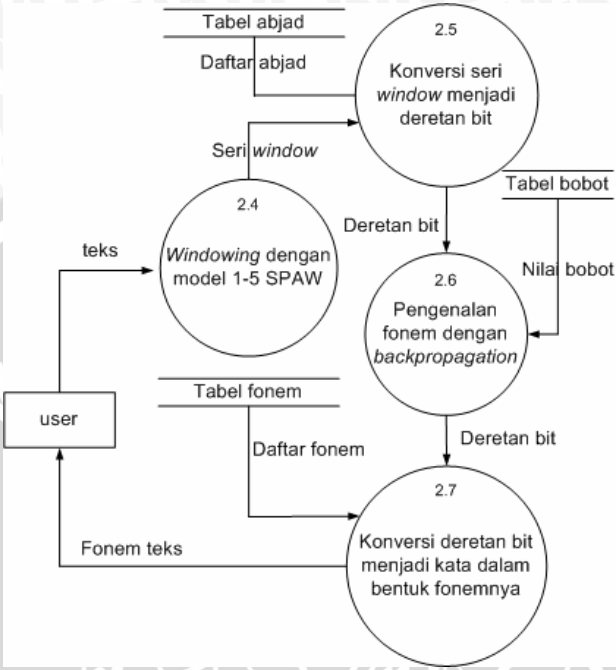
Pada DFD level 2 proses pembelajaran (Gambar 3.8) terdapat satu entitas *admin* dan tiga sub proses yaitu sub proses *windowing* dengan model 1-5 SPAW, sub proses pembelajaran dengan jaringan syaraf tiruan *backpropagation* dan sub proses konversi seri *window* menjadi deretan bit. Sub proses *windowing* dengan model 1-5 SPAW bertujuan untuk mengubah tiap kata menjadi beberapa seri *window.* Tiap seri *window* kemudian dijadikan deretan bit melalui sub proses konversi seri *window* menjadi deretan bit. Deretan bit tersebut kemudian dijadikan *input* pada sub proses pembelajaran dengan jaringan syaraf tiruan *backpropagation.* Hasil dari sub proses pembelajaran dengan jaringan syaraf tiruan *backpropagation* adalah nilai bobot yang kemudian disimpan pada tabel bobot.



Gambar 3.8 DFD level 2 proses pembelajaran

Pada DFD level 2 proses pengenalan fonem (Gambar 3.9) terdapat satu entitas *user*, empat sub proses yaitu :

1. *Windowing* dengan model 1-5 SPAW.
2. Konversi seri *window* menjadi deretan bit .
3. Pengenalan fonem dengan jaringan syaraf tiruan *backpropagation.*
4. Konversi deretan bit menjadi kata dalam bentuk fonemnya.



Gambar 3.9 DFD level 1 proses pengenalan fonem

3.6. Perancangan Tabel

Dari perancangan proses dan perancangan DFD yang telah dipaparkan pada sub bab 3.2 dan sub bab 3.5, maka diperlukan beberapa tabel pada sistem pengenalan fonem ini. Tabel-tabel tersebut adalah :

Tabel 3.25 Tabel TKata

Field	Type	Keterangan
Kata	Text (*primary key*)	Contoh: belajar
Fonem	Text	Contoh: bêlajar

Tabel TKata (Tabel 3.25) berisi daftar kata beserta fonem-fonem yang menyusunnya. Tabel TKata diperlukan pada saat proses pembelajaran.

Tabel 3.26 Tabel TFonem

Field	Type	Keterangan
Fonem	Text (*primary key*)	Contoh: ê
Bit	Text	Contoh: 00000100000000000000000000000000

Tabel TFonem (Tabel 3.26) berisi daftar fonem yang dikenali dalam sistem pengenalan fonem ini beserta representasinya dalam bentuk deretan bit. Tabel TFonem diperlukan pada saat proses pembelajaran dan pengenalan fonem.

Tabel 3.27 Tabel TAbjad

Field	Type	Keterangan
Abjad	Text (*primary key*)	Contoh: a
Bit	Text	Contoh: 10000000000000000000000000

Tabel TAbjad (Tabel 3.27) berisi daftar huruf abjad yang dikenali dalam sistem pengenalan fonem ini beserta representasinya dalam bentuk deretan bit. Tabel TAbjad diperlukan pada saat proses pembelajaran dan pengenalan fonem.

Tabel 3.28 Tabel TPembelajaran

Field	Type	Keterangan
LearningRate	Text	Nilai *learning rate*
HiddenUnit	Integer	Jumlah unit pada *hidden layer*
MaxEpoch	Integer	Jumlah perulangan
TargetError	Double	*Error* yang minimum

Tabel TPembelajaran (Tabel 3.28) digunakan untuk menyimpan parameter-parameter pembelajaran. Tabel TPembelajaran diperlukan pada saat proses pembelajaran dan pengenalan fonem.

Tabel 3.29 Tabel TBobot

Field	Type	Keterangan
IDBobot	Integer (*primary key*)	Contoh: 1
IDLayer	Integer	Contoh: 1
DariNeuron	Integer	Contoh: 1
KeNeuron	Integer	Contoh: 2
Bobot	Double	Contoh: 0.56

Tabel TBobot (Tabel 3.29) berisi nilai bobot pada tiap-tiap *layer* yang ada dalam sistem pengenalan fonem ini. Tabel TBobot diperlukan pada saat pengenalan fonem dan datanya diisi pada saat proses pembelajaran.

3.7. Perancangan Antarmuka

Antarmuka yang akan dibangun dibagi menjadi dua bagian yaitu antarmuka pada tahapan pembelajaran dan penambahan kata.

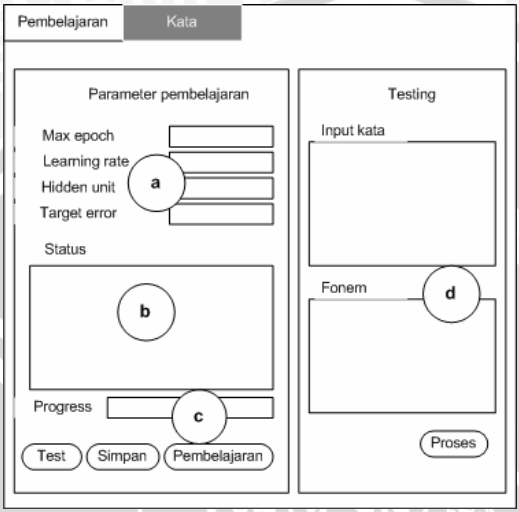
Pada *form* pembelajaran (Gambar 3.10) terdiri atas beberapa bagian antara lain :

a : Untuk mengatur parameter-parameter pembelajaran diantaranya besar *max epoch*, nilai *learning rate*, jumlah unit pada *hidden layer*, nilai target *error* yang ingin dicapai.
b : Untuk mengetahui status pembelajaran pada tiap *epoch*.
c : Untuk menjalankan proses pembelajaran dan menyimpan bobot.
d : Untuk mengetes hasil pembelajaran.

Pada *form* penambahan kata (Gambar 3.11) terdiri atas beberapa bagian antara lain:

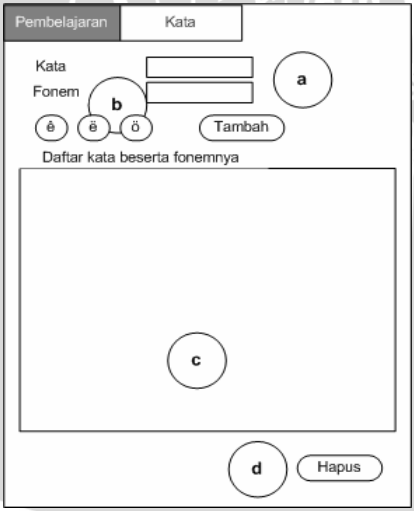
a : Untuk memasukkan kata beserta fonemnya ke dalam tabel kata.
b : Untuk memberi simbol fonem.
c : Untuk menampilkan data kata beserta fonemnya dari tabel kata.
d : Untuk menghapus kata beserta fonemnya dari tabel kata.

***Form* pembelajaran**



Gambar 3.10 Rancangan *form* pembelajaran

***Form* penambahan kata**



Gambar 3.11 Rancangan *form* penambahan kata

3.8. Perancangan Uji Coba

Tujuan dari uji coba sistem pengenalan fonem ini adalah untuk mencari arsitektur jaringan syaraf tiruan yang optimal dalam proses pengenalan fonem suatu kata. Arsitektur jaringan syaraf tiruan yang dimaksudkan adalah jumlah unit pada *hidden layer* dan nilai *learning rate*.

3.8.1. Skenario Evaluasi

Pencarian arsitektur jaringan syaraf tiruan yang optimal dilakukan dengan cara mencoba jumlah unit pada *hidden layer* antara 20 sampai 100 dengan selang 20 unit. Percobaan ini memerlukan daftar kata beserta fonemnya sebanyak 500 buah untuk proses pembelajaran. Besar *max epoch* yang dipakai selama pembelajaran adalah 2000 *epoch.* Nilai keakuratan didapat dari perbandingan antara jumlah kata yang berhasil dikenali fonemnya dengan benar dengan jumlah kata yang tidak berhasil dikenali fonemnya. Hasil dari uji coba jumlah unit pada *hidden layer* disimpan pada Tabel 3.30.

Tabel 3.30 Tabel hasil uji jumlah unit pada *hidden layer*

Jumlah unit pada *hidden layer*	MSE minimum	*Average Error* minimum	Keakuratan

Jumlah unit optimal pada *hidden layer* yang didapatkan dari pengujian pertama kemudian dipakai pada uji coba nilai *learning rate*. Besar nilai *learning rate* yang diujicobakan adalah diantara 0.001 sampai 0.005 dengan selang 0.001. Hasil dari uji coba nilai *learning rate* disimpan pada Tabel 3.31.

Tabel 3.31 Tabel hasil uji nilai *learning rate*

Nilai *Learning rate*	MSE minimum	*Average Error* minimum	Keakuratan

Setelah mendapat arsitektur jaringan yang optimal, kemudian dilanjutkan dengan proses pencarian nilai bobot optimal. Pencarian nilai bobot dilakukan lewat proses pembelajaran pada kata beserta fonemnya sebanyak 500 buah. Nilai bobot optimal tersebut kemudian dipakai untuk uji coba pengenalan fonem dari kata-kata yang belum pernah dilakukan proses pembelajaran sebelumnya. Jumlah kata yang dipakai uji coba adalah sebanyak 50 kata. Uji coba dilakukan sebanyak 10 kali percobaan. Hasil dari uji coba pengenalan fonem disimpan pada Tabel 3.32.

Tabel 3.32 Tabel hasil uji pengenalan fonem terhadap kata-kata yang belum pernah dilakukan pembelajaran

Percobaan ke	Keakuratan

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai implementasi dari perancangan pada bab 3. Dari implementasi tersebut kemudian dibahas mengenai hasil evaluasi dari perangkat lunak yang dihasilkan.

4.1. Lingkungan Implementasi

Lingkungan implementasi meliputi lingkungan perangkat keras serta lingkungan perangkat lunak.

4.1.1. Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem pengenalan fonem ini adalah:

1. *Processor* Intel(R) Celeron(R) 4 – 2.26 GHz.
2. RAM 256 MB.
3. *Harddisk* dengan kapasitas 40 GB.
4. *Monitor*.
5. *Keyboard*.
6. *Mouse*.
7. *Speaker*.
8. *Microphone*.

4.1.2. Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pengenalan fonem ini adalah :

1. Sistem Operasi *Microsoft Windows XP Professional Service Pack 2.*
2. *Microsoft Visual C++ 6.*
3. *Microsoft Access 2007.*

4.2. Implementasi Program

Berdasarkan perancangan proses pada sub bab 3.2 maka pada sub bab ini akan dibahas mengenai implementasi dari perancangan

tersebut. Dari perancangan proses diketahui bahwa dalam sistem pengenalan fonem ini terdapat beberapa sub modul diantaranya :

1. Sub modul *Second Position Asymmetric Windowing*, bertugas mengubah kata menjadi sejumlah seri *window* dan kemudian mengubahnya menjadi deretan bit. Selain itu juga terdapat alur utama proses pembelajaran dengan menggunakan jaringan syaraf tiruan *backpropagation* dan proses pengubahan teks ke bentuk fonemnya.
2. Sub modul jaringan syaraf tiruan *backpropagation*, bertugas melakukan proses pembelajaran dengan algoritma *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt.* Sub modul jaringan syaraf tiruan *backpropagation* mempunyai *input* berupa deretan bit dari tiap seri *window* dan target berupa deretan bit dari target fonem dari tiap seri *window* yang berasal dari sub modul *Second Position Asymmetric Windowing.*

Sub modul-sub modul di atas diimplementasikan ke dalam bentuk kelas-kelas untuk mempermudah dalam proses pemrograman. Sehingga akan terdapat tiga kelas utama yaitu :

1. Kelas CSPAW yang merepresentasikan sub modul *Second Position Asymmetric Windowing* dan bertugas menyimpan daftar kata yang akan dijadikan *input* pada proses pembelajaran.
2. Kelas CBackPro yang merepresentasikan sub modul jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt.*

Struktur data utama yang ada dalam sistem pengenalan fonem ini direpresentasikan pada Gambar 4.1.

```
typedef struct tag_FONEM
{
        vector<double> binaryfeature;
        CString sFonem;
}FONEM;

typedef struct tag_ABJAD
{
        vector<double> binaryfeature;
        CString sAbjad;
}ABJAD;
typedef struct tag_FONEMKATA
```

```
{
        CString sKata;
        CString sFonem;
}FONEMKATA;
typedef struct tag_INPUT
{
        vector<double> input;
        vector<double> output;
        int id;
}INPUTPATTERN;
typedef struct tag_PATTERN
{
        int start,jumhuruf;
}PATTERNKATA;
```

Gambar 4.1 Struktur data utama

Berikut ini adalah penjelasan dari masing-masing struktur data :

a. FONEM, merupakan tipe data *struct* yang mempunyai anggota yaitu binaryfeature dan sFonem. sFonem merupakan variabel yang menyimpan satu simbol fonem sedangkan binaryfeature merupakan sebuah *vector* yang menyimpan deretan bit yang mewakili fonem tersebut.
b. ABJAD, merupakan tipe data *struct* yang mempunyai anggota yaitu binaryfeature dan sAbjad. sAbjad merupakan variabel yang menyimpan satu simbol abjad sedangkan binaryfeature merupakan sebuah *vector* yang menyimpan deretan bit yang mewakili abjad tersebut.
c. FONEMKATA, merupakan tipe data *struct* yang mempunyai anggota yaitu sKata dan sFonem. Tipe data FONEMKATA digunakan untuk menyimpan sebuah kata beserta fonemnya.
d. INPUTPATTERN, merupakan tipe data *struct* yang mempunyai anggota yaitu input dan output. Input merupakan variabel yang digunakan untuk menyimpan *input* dari satu pola pada proses pembelajaran dengan jaringan syaraf tiruan *backpropagation.* sedangkan output merupakan variabel yang digunakan untuk menyimpan target yang diinginkan. Kedua variabel ini merupakan suatu *vector* yang menyimpan deretan bit.
e. PATTERN, merupakan tipe data *struct* yang digunakan untuk mengelompokkan INPUTPATTERN berdasarkan kata yang mewakilinya. Mempunyai anggota start dan jumhuruf, start digunakan untuk menyimpan posisi dari INPUTPATTERN yang

pertama dari suatu kata dalam *vector* pembelajaran, sedangkan jumhuruf digunakan untuk menyimpan jumlah dari INPUTPATTERN yang mewakili suatu kata.

Struktur data yang utama dalam kelas CSPAW adalah :

a. m_BpFonem, merupakan variabel dengan tipe data kelas CBackPro. Variabel m_BpFonem digunakan dalam proses pembelajaran dengan jaringan syaraf tiruan *backpropagation.*
b. m_vFonem, merupakan variabel dengan tipe data *vector* dari FONEM. Variabel m_vFonem digunakan untuk menyimpan data fonem beserta deretan bit yang mewakilinya.
c. m_vAbjad, merupakan variabel dengan tipe data *vector* dari ABJAD. Variabel m_vAbjad digunakan untuk menyimpan data abjad beserta deretan bit yang mewakilinya.
d. m_vKata, merupakan variabel dengan tipe data *vector* dari FONEMKATA. Variabel m_vKata digunakan untuk menyimpan data kata beserta fonemnya.

Struktur data yang utama dalam kelas CBackPro adalah :

a. m_vWeightInput2Hidden, merupakan variable dengan tipe data *vector* dua dimensi yang berfungsi untuk menyimpan bobot dari unit-unit yang berada pada *input layer* ke unit-unit yang berada pada *hidden layer.*
b. m_vWeightHidden2Output, merupakan variabel dengan tipe data *vector* dua dimensi yang berfungsi untuk menyimpan bobot dari unit-unit yang berada pada *hidden layer* ke unit-unit yang berada pada *output layer.*
c. m_vLearningData, merupakan variabel dengan tipe data *vector* INPUTPATTERN yang berfungsi untuk menyimpan pasangan *input* beserta targetnya. Variabel m_vLearningData digunakan sebagai data pembelajaran.
d. m_vBestWeightInput2Hidden dan m_vBestWeightHidden2Output, merupakan variabel-variabel yang digunakan untuk menyimpan bobot terbaik selama pembelajaran.

4.2.1. Implementasi Proses Pembelajaran

Proses pembelajaran yang meliputi proses *windowing* dengan model 1-5 SPAW, proses konversi *string* pada seri *window* menjadi deretan bit, proses konversi fonem menjadi deretan bit dan proses pembelajaran fonem menggunakan jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* diimplementasikan pada sebuah fungsi Training dari kelas CSPAW (Gambar 4.2).

```
void CSPAW::Training(CProgressCtrl * pProg,CEdit * pStatus)
{
    int * posx = new int[m_uiWinSize];
    int pos = abs((m_uiWinSize/2)-m_pos);
    vector<char> vWindowx;
    vWindowx.resize(m_uiWinSize);
    m_BpFonem->Reset();
    m_BpFonem->InitWeight();
    CADORecordset * pRs = new CADORecordset(m_pDb);
    CString sCmd;
    sCmd.Format("SELECT distinct kata, fonem FROM TKata
      order by kata");
    if(pRs->Open(sCmd,CADORecordset::openQuery))
    {
       int index = 0;
       while(!pRs->IsEOF())
    {
       if(m_bStop)break;
       CString sKata,sFonem;
       pRs->GetFieldValue("fonem", sFonem);
       pRs->GetFieldValue("kata",sKata);
       sKata.TrimLeft();sKata.TrimRight();
       sFonem.TrimLeft();sFonem.TrimRight();
       int nLength = sKata.GetLength();
       for(int w=0;w<m_uiWinSize;w++)
       {
         posx[w] = w - pos;
       }
       CString sWindow="";
       CString sBuffer,sText;
       for(int i=0;i<nLength;i++)
       {
         for(int j=0;j<m_uiWinSize;j++)
         {
           if((posx[j]<0) ||(posx[j]>=nLength))
           {
               vWindowx[j] = '-';
           }else{
              vWindowx[j] = sKata.GetAt(posx[j]);
           }
```

```
            vector<double> vBinaryInput  =
              AbjadToBinary(vWindowx);
            char cfonem = sFonem.GetAt(i);
            vector<double> vBinaryTarget =
              FonemToBinary(cfonem);
            INPUT input;
            input.input = vBinaryInput;
            input.output = vBinaryTarget;
            input.id = index;
            m_BpFonem->AddData(input);
             for(w=0;w<m_uiWinSize;w++)
             {
                 posx[w]++;
             }
          }
          index++; pRs->MoveNext();
        }
         pRs->Close();
     }else
     {
        MessageBox(NULL,
          "Error on opening table TKata","NETtalk2",MB_OK);
     }
     delete pRs;
     m_BpFonem->Learning(pProg,pStatus);
     delete[] posx;
}
```

Gambar 4.2 *Source code* fungsi Training

Fungsi Training (Gambar 4.2) merupakan fungsi yang bertugas melakukan proses pembelajaran terhadap kata-kata yang terdapat dalam tabel TKata. Fungsi Training mula-mula mengubah tiap kata menjadi beberapa seri *window*, kemudian untuk tiap seri *window* dipilih target fonemnya. Tiap seri *window* kemudian dikonversikan ke dalam bentuk deretan bit melalui fungsi AbjadToBinary, demikian juga untuk target fonemnya melalui fungsi FonemToBinary. Deretan bit dari tiap seri *window* dijadikan *input* sedangkan deretan bit dari target fonemnya dijadikan sebagai target. Pasangan *input* dengan target tersebut dijadikan data masukan pada kelas CBackPro melalui fungsi AddData. Setelah deretan bit dari tiap seri *window* semua kata selesai dijadikan *input*, kemudian dilanjutkan proses pembelajaran dengan metode jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* melalui pemanggilan fungsi Learning dari kelas CBackPro (Gambar 4.3).

```
void CBackPro::Learning(CProgressCtrl * pProg,
  CEdit * pStatus) {
  m_uiIterations = 0;
  double dEtaDecay=0.7941833,dEtaMin=0.00005;
  Normalize();  InitWeight();
  int sample=0,start,jumhuruf,index;
  m_dMinMSE = 99;  m_dMinErrAbs = 99;
  random_shuffle(m_vOrder.begin(),m_vOrder.end());
  push_heap(m_vOrder.begin(),m_vOrder.end());
  GetTotalHessian();
  int step=0;
  while (1)
  {
     if((step%120000==0) && (step!=0))
     {
       m_dEta*=dEtaDecay;
       if(m_dEta<dEtaMin)
         m_dEta = dEtaMin;
     }
     if (sample == m_iJumKata){
          TotalError(pStatus);
          sample = 0;  m_uiIterations++;
          random_shuffle(m_vOrder.begin(),m_vOrder.end());
          push_heap(m_vOrder.begin(),m_vOrder.end());
          GetTotalHessian();
     }
     int kata = m_vOrder[sample];
     start = m_vPatternKata[kata].start;
     jumhuruf = m_vPatternKata[kata].jumhuruf;
     ResetDeltaWeight();
     //tiap seri window
     for (index=start;index<(start+jumhuruf);index++)
     {
       INPUTPATTERN inputx = m_vLearningData[index];
       FeedForward(inputx);
       CalculateError(inputx);
       step++;
       UpdateWeight();
     }
     sample++;
   }
}
```

Gambar 4.3 *Source code* fungsi Learning

Sebelum proses pembelajaran dengan metode jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* dimulai, terlebih dahulu dilakukan proses inisialisasi bobot melalui fungsi InitWeight (Gambar 4.4).

```
#define UNIFORM_PLUS_MINUS_ONE
  ( (double)(2.0 * rand())/RAND_MAX - 1.0 )

void CBackPro::InitWeight()
{
  int i,j;
  for(j=0;j<m_uiNumHidden;j++)
  {
    for(i=0;i<m_uiNumInput;i++)
    {
       m_vWeightInput2Hidden[i][j] = 0.05 *
         UNIFORM_PLUS_MINUS_ONE;
    }
  }
  for(j=0;j<m_uiNumOutput;j++)
  {
    for(i=0;i<m_uiNumHidden;i++)
    {
       m_vWeightHidden2Output[i][j] = 0.05 *
         UNIFORM_PLUS_MINUS_ONE;
    }
  }
}
```

Gambar 4.4 *Source code* fungsi InitWeight

Fungsi InitWeight (Gambar 4.4) merupakan fungsi yang bertugas menginisialiasai nilai awal bobot dari *input layer* ke *hidden layer* dan bobot dari *hidden layer* ke *output layer*. Bobot diinisialisasi secara acak dengan nilai berkisar antara -0,05 sampai 0,05 sehingga nilai rata-rata bobot yang masuk pada tiap unit *neuron* mendekati nol.

Setelah proses inisialisasi bobot, kemudian dilanjutkan dengan proses normalisasi (persamaan 2.22) nilai *input* melalui fungsi Normalize (Gambar 4.5). Proses normalisasi diperlukan untuk mempercepat proses pembelajaran dan nilai *error* bisa diminimumkan.

```
void CBackPro::Normalize()
{
  vector<INPUTPATTERN>::iterator iterLearningData;
  int i;
  for(iterLearningData = m_vLearningData.begin();
    iterLearningData != m_vLearningData.end();
    iterLearningData++)
  {
    INPUTPATTERN & inputx = *iterLearningData;
    for(i=0;i<m_uiNumInput;i++)
    {
```

```
      inputx.input[i] = (0.8 * inputx.input[i])+0.1;
    }
   }
}
```

Gambar 4.5 *Source code* fungsi Normalize

Dalam proses pembelajaran dengan metode jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt* (fungsi Learning) terdapat beberapa fungsi yang dijalankan diantaranya fungsi FeedForward (Gambar 4.6), fungsi Activation (Gambar 4.7), fungsi ActivationDerivative (Gambar 4.8), fungsi CalculateError (Gambar 4.9), fungsi CalculateHessian (Gambar 4.10) dan fungsi UpdateWeight (Gambar 4.11). Berikut akan dijelaskan masing-masing dari fungsi tersebut.

```
void CBackPro::FeedForward()
{
  int i,j;
  //operasi pada hidden layer
  for(i=0;i<m_uiNumHidden;i++)
  {
    z_in[i] = 0;
    for( j=0;j<m_uiNumInput;j++)
    {
      z_in[i] += m_vWeightInput2Hidden[j][i] *
        m_currinput.input[j];
    }
    //pengaktifan
    z[i] = Activation(z_in[i]);
  }
  //operasi pada output layer
  for(i=0;i<m_uiNumOutput;i++)
  {
    y_in[i] = 0;
    for(j=0;j<m_uiNumHidden;j++)
    {
      y_in[i]+= m_vWeightHidden2Output[j][i]*z[j];
    }
    //pengaktifan
    y[i] = Activation(y_in[i]);
  }
}
```

Gambar 4.6 *Source code* fungsi FeedForward

Fungsi FeedForward (Gambar 4.6) merupakan fungsi yang bertugas mengaktifkan unit-unit yang berada pada *hidden layer* dan *output layer* dengan menggunakan fungsi aktivasi pada persamaan 2.15.

```
double CBackPro::Activation(double z_in)
{
  return (1.7159*tanh(0.66666667*z_in));
}
```

Gambar 4.7 *Source code* fungsi Activation

Fungsi Activation (Gambar 4.7) merupakan fungsi yang bertugas mengaktifasi tiap-tiap unit dengan persamaan 2.15.

```
double CBackPro::ActivationDerivative( double y_in)
{
  double x = cosh(0.66666667*y_in);
  return (0.66666667*1.7159/(x*x));
}
```

Gambar 4.8 *Source code* fungsi ActivationDerivative

Fungsi ActivationDerivative (Gambar 4.8) merupakan implementasi dari persamaan 2.16.

```
void CBackPro::CalculateError(INPUTPATTERN inputx)
{
  int i,j;
  double err,selisih;
  for(i=0;i<m_uiNumOutput;i++)
  {
    //error pada tiap output
    selisih = inputx.output[i]-y[i];
    m_vOutputError[i]= selisih *
      ActivationDerivative(y_in[i]);
    err = m_vOutputError[i];
    for(j=0;j<m_uiNumHidden;j++)
    {
      m_vErrorWeightHidden2Output[j][i] = err*z[j];
    }
  }
  //hitung error pada hidden layer
  for(i=0;i<m_uiNumHidden;i++)
  {
    errorhiddenlayer[i] = 0;
```

```
    for(j=0;j<m_uiNumOutput;j++)
    {
        errorhiddenlayer[i] += m_vOutputError[j] *
          m_vWeightHidden2Output[i][j];
    }
    m_vHiddenError[i] = errorhiddenlayer[i] *
     ActivationDerivative(z_in[i]);
  }
  //hitung delta weight input 2 hidden
  for(j=0;j<m_uiNumHidden;j++)
  {
    err = m_vHiddenError[j];
    for(i=0;i<m_uiNumInput;i++)
    {
       m_vErrorWeightInput2Hidden[i][j] = err *
        inputx.input[i];
    }
  }
}
```

Gambar 4.9 *Source code* fungsi CalculateError

Fungsi CalculateError (Gambar 4.9) merupakan fungsi yang bertugas mendapatkan nilai koreksi bobot yang kemudian digunakan untuk memperbaiki nilai bobot (persamaan 2.7 dan persamaan 2.11).

```
void CBackPro::CalculateHessian(INPUTPATTERN inputx)
{
  int i,j;
  double yk,xi,wki;
  vector<double> d2E_d2Yn;
  vector<double> d2E_d2X;
  d2E_d2Yn.resize(m_uiNumOutput);
  d2E_d2X.resize(m_uiNumHidden);
  //hitung d2E/d2Yin = f'(Yn)^2* d2E/d2X ->1
  for(i=0;i<m_uiNumOutput;i++)
  {
    yk = ActivationDerivative(y_in[i]);
    d2E_d2Yn[i] = yk*yk;
  }
  //hitung d2E/d2Wki = d2E/d2Yin * x^2
  for(i=0;i<m_uiNumOutput;i++)
  {
    for(j=0;j<m_uiNumHidden;j++)
    {
       xi = z[j];
       m_v2WeightHidden2Output[j][i] = (0.8 *
         m_v2WeightHidden2Output[j][i] ) + (0.2 *
         d2E_d2Yn[i] * xi*xi);
```

```
    }
  }
  //hitung d2E/d2x = sigma(d2E/d2Yn * wki^2)
  for(j=0;j<m_uiNumHidden;j++)
  {
    d2E_d2X[j] = 0;
    for(i=0;i<m_uiNumOutput;i++)
    {
      wki = m_vWeightHidden2Output[j][i];
      d2E_d2X[j]+= d2E_d2Yn[i]*wki*wki;
    }
  }
  vector<double> d2E_d2Yn2;
  vector<double> d2E_d2X2;
  d2E_d2Yn2.resize(m_uiNumHidden);
  d2E_d2X2.resize(m_uiNumInput);
  //hitung d2E/d2Yin = f'(Yn)^2* d2E/d2X
  for(i=0;i<m_uiNumHidden;i++)
  {
    yk = ActivationDerivative(z_in[i]);
    d2E_d2Yn2[i] = d2E_d2X[i]*yk*yk;
  }
  //hitung d2E/d2Wki = d2E/d2Yin * x^2
  for(i=0;i<m_uiNumHidden;i++)
  {
    for(j=0;j<m_uiNumInput;j++)
    {
      xi = inputx.input[j];
      m_v2WeightInput2Hidden[j][i] = (0.8 *
        m_v2WeightInput2Hidden[j][i])+ (0.2 * d2E_d2Yn2[i] *
        xi*xi)
     }
   }
}
```

Gambar 4.10 *Source code* fungsi CalculateHessian

Fungsi CalculateHessian (Gambar 4.10) merupakan fungsi yang bertugas menghitung nilai *hessian* total yang kemudian digunakan untuk mendapatkan nilai *learning rate* pada tiap-tiap bobot (persamaan 2.17 sampai persamaan 2.21).

```
void CBackPro::UpdateWeight()
{
  //lakukan perubahan bobot untuk hidden layer
  int i,j;
  double divisor,epsilon;
  for(j=0;j<m_uiNumHidden;j++)
  {
    for(i=0;i<m_uiNumInput;i++)
```

```
    {
        divisor = m_v2WeightInput2Hidden[i][j] + 0.1;
        if(divisor<0.1)
          divisor = 1.0;
        epsilon = m_dEta / divisor;
        m_vWeightInput2Hidden[i][j] +=  epsilon *
           m_vErrorWeightInput2Hidden[i][j];
    }
  }
  //lakukan perubahan bobot untuk output layer
  for(i=0;i<m_uiNumOutput;i++)
  {
    for(j=0;j<m_uiNumHidden;j++)
    {
        divisor = m_v2WeightHidden2Output[j][i] + 0.1;
        if(divisor<0.1)
        divisor = 1.0;
        epsilon = m_dEta / divisor;
        m_vWeightHidden2Output[j][i] +=  epsilon *
           m_vErrorWeightHidden2Output[j][i];
    }
  }
}
```

Gambar 4.11 *Source code* fungsi UpdateWeight

Fungsi UpdateWeight (Gambar 4.11) merupakan fungsi yang bertugas mengubah nilai bobot berdasarkan nilai *hessian* total dan nilai koreksi bobot (persamaan 2.23 sampai persamaan 2.28).

4.2.2. Implementasi Proses Pengenalan Fonem

Proses pengenalan fonem dilakukan melalui pemanggilan fungsi Text2Fonem (Gambar 4.12) dari kelas CSPAW.

```
CString CSPAW::Text2Fonem(CString sText)
{
  int * posx = new int[m_uiWinSize];
  int pos = (m_uiWinSize/2)-m_pos;
  vector<char> vWindowx;
  vWindowx.resize(m_uiWinSize);
  UINT uiTrue=0,uiFalse=0;
  int j=0;
  int nLength = sText.GetLength();
  for(int w=0;w<m_uiWinSize;w++)
  {
    posx[w] = w - pos;
  }
  CString sFonem = "";
  for(int i=0;i<nLength;i++)
```

```
  {
    for(int j=0;j<m_uiWinSize;j++)
    {
      if((posx[j]<0) ||(posx[j]>=nLength))
      {
        vWindowx[j] = '-';
      }else{
        vWindowx[j] = sText.GetAt(posx[j]);
      }
    }
      vector<double> vBinaryInput = AbjadToBinary(vWindowx);
      vector<double> vBineryOutput =
        m_BpFonem->Test(vBinaryInput);
      CString sFonemx=BinaryFonemToString(vBineryOutput);
      if(sFonemx=="")
      {
         sFonemx = sText.GetAt(i);
      }
      sFonem += sFonemx;
      for(w=0;w<m_uiWinSize;w++)
      {
         posx[w]++;
      }
  }
  delete[] posx;
  return sFonem;
}
```

Gambar 4.12 *Source code* fungsi Text2Fonem

Fungsi Text2Fonem (Gambar 4.12) merupakan fungsi yang bertugas mengubah suatu teks ke dalam bentuk fonemnya. Proses yang ada dalam fungsi Text2Fonem adalah proses pengubahan teks masukan menjadi beberapa seri *window* kemudian untuk tiap seri *window* dikonversikan ke dalam deretan bit oleh fungsi AbjadToBinary dan dilakukan pengenalan fonem dengan metode jaringan syaraf tiruan *backpropagation* melalui pemanggilan fungsi Test (Gambar 4.13) dari kelas CBackPro. Hasil dari pengenalan dengan metode jaringan syaraf tiruan *backpropagation* kemudian dikonversikan ke dalam bentuk fonem yang mewakilinya melalui fungsi BinaryFonemToString.

```
vector<double> CBackPro::Test(vector<double> input)
{
  int i,j;
  for(i=0;i<m_uiNumInput;i++)
  {
    input[i] = (0.8 * input[i])+0.1;
```

```
    }
    for(i=0;i<m_uiNumHidden;i++)
    {
      z_in[i] = 0;
      for( j=0;j<m_uiNumInput;j++)
      {
         z_in[i] += m_vWeightInput2Hidden[j][i]*input[j];
      }
      //pengaktifan
      z[i] = Activation(z_in[i]);
    }
    double max;
    int sel=0;
    for(i=0;i<m_uiNumOutput;i++)
    {
      y_in[i] = 0;
      for(j=0;j<m_uiNumHidden;j++)
      {
         y_in[i]+= m_vWeightHidden2Output[j][i]*z[j];
      }
     y[i] = Activation(y_in[i]);
     if(i==0)
     {
         max = y[0];
         sel = 0;
     }else
     {
         if(y[i] > max)
         {
           max = y[i];
           sel = i;
         }
     }
    }
    //winner takes all
    for(i=0;i<m_uiNumOutput;i++)
    {
      if(i!=sel)
      {
         y[i] = 0;
      }else{
         y[i] = 1;
      }
    }
    return y;
}
```

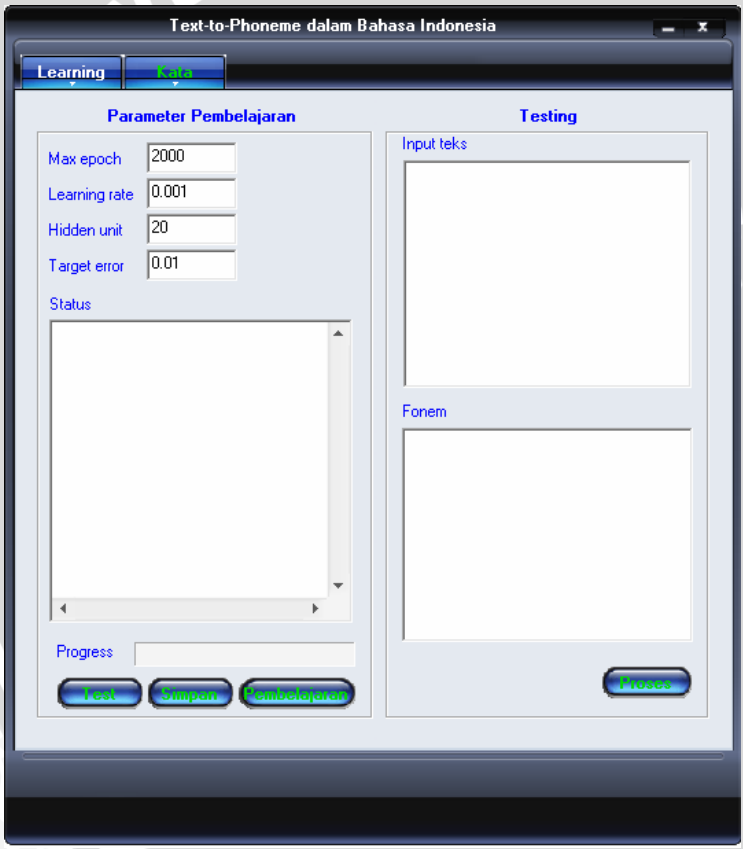
Gambar 4.13 *Source code* fungsi Test.

Fungsi Test (Gambar 4.13) merupakan fungsi yang bertugas mendapatkan deretan bit yang merepresentasikan suatu fonem dari

input yang berupa deretan bit suatu seri *window*. Proses pengenalan yang dilakukan hampir sama seperti pada fungsi FeedForward. Pengenalan didasarkan atas nilai bobot yang diperoleh pada proses pembelajaran.

4.3. Implementasi Antarmuka

Berdasarkan rancangan antarmuka pada sub bab 3.7 maka dihasilkan antarmuka yang ditunjukkan pada Gambar 4.14 dan Gambar 4.15. Gambar 4.14 adalah antarmuka proses pembelajaran dan Gambar 4.5 adalah antarmuka tahap penambahan kata.



Gambar 4.14 Antarmuka tahap pembelajaran

Dalam antarmuka tahap pembelajaran (Gambar 4.14), *user* dapat mengatur parameter pembelajaran yang terdiri atas besar *max epoch*, nilai *learning rate*, jumlah unit pada *hidden layer* dan nilai *target error*. Proses pembelajaran dimulai saat *user* menekan tombol Pembelajaran. Proses pembelajaran dapat dihentikan dengan menekan tombol Stop. Untuk menyimpan bobot hasil pembelajaran, *user* dapat menekan tombol Simpan. *User* juga dapat menguji coba pengenalan fonem beberapa kata dengan cara memasukkan teks pada *text box* Input Teks dan menekan tombol Proses. Hasil dari proses pengenalan dapat dilihat pada *text box* Fonem.

Keterangan Status yang ada pada Gambar 4.14 adalah :

- Total data : 500 kata, 3165 seri *window* adalah keterangan mengenai jumlah data yang akan diproses dalam pembelajaran dengan jaringan syaraf tiruan *backpropagation* dengan metode *stochastic diagonal Levenberg-Marquardt.* Data dalam hal ini adalah jumlah kata sebanyak 500 kata yang terdiri atas 3165 seri *window.*
- 1 MSE : 0.1962 AvgErr : 1.3583 menunjukkan nilai MSE dan nilai *Average Error* pada *epoch* pertama.
- 3 Best MSE : 0.0773 AvgErr : 0.7637 menunjukkan kombinasi terbaik antara nilai MSE dan nilai *Average Error* selama 3 *epoch.*
- Kata dalam database benar : 302, salah : 198, keakuratan : 60.40 percent menunjukkan hasil pengenalan fonem terhadap kata yang telah dilakukan proses pembelajaran. Nilai keakuratan diperoleh dari perbandingan jumlah kata yang berhasil dikenali fonemnya dengan jumlah kata yang tidak berhasil dikenali fonemnya.
- Kata tidak ada dalam database benar : 35, salah : 15, keakuratan : 70.00 percent menunjukkan hasil pengenalan fonem terhadap kata yang belum pernah dilakukan proses pembelajaran.

Gambar 4.15 Antarmuka tahap penambahan kata

Form pengaturan daftar kata (Gambar 4.15) digunakan untuk menambah atau menghapus kata-kata yang dipakai selama proses pembelajaran. Proses penambahan kata dilakukan dengan cara memasukkan kata melalui *text box* Kata dan cara pengucapan (fonem) kata tersebut melalui *text box* Fonem, kemudian dilanjutkan dengan menekan tombol Tambah. Untuk menulis simbol fonem dapat dilakukan dengan menekan tombol ê (misal : bandêng), ë (misal : rëmëh) atau ö (misal : tököh). Penghapusan suatu kata dilakukan dengan cara memilih kata yang akan dihapus pada *list view*, kemudian dilanjutkan dengan menekan tombol Hapus.

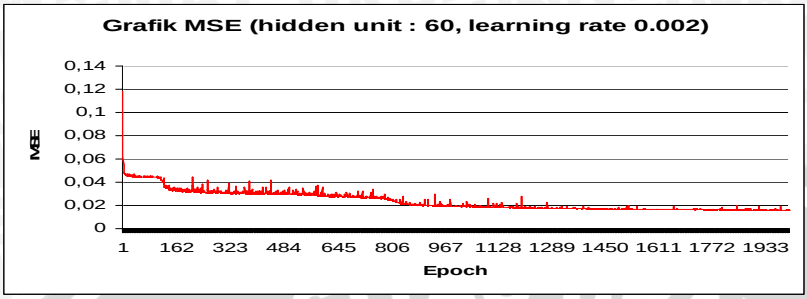
4.4. Analisa Hasil

Dari pengujian terhadap jumlah unit pada *hidden layer*, didapatkan hasil seperti pada Tabel 4.1. Dalam percobaan tersebut menunjukkan bahwa jumlah unit pada *hidden layer* sebanyak 60 unit dapat menghasilkan nilai MSE yang minimum yaitu 0.0145 namun mempunyai keakuratan sebesar 89.20%. Sedangkan dengan jumlah unit pada *hidden layer* sebanyak 80 unit dapat menghasilkan nilai MSE sebesar 0.0168 dan mempunyai keakuratan sebesar 90.60%.

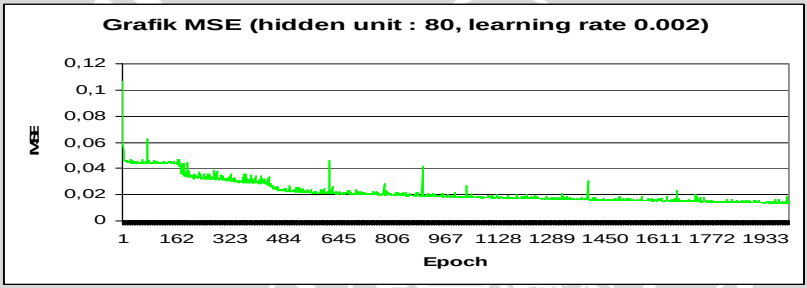
Tabel 4.1 Hasil pengujian jumlah unit pada *hidden layer*

Jumlah unit pada *hidden layer*	MSE minimum	*Average Error* minimum	Keakuratan
20	0.0457	0.2834	66.60% benar : 333 kata salah : 167 kata
40	0.0181	0.1818	87.60% benar : 438 kata salah : 62 kata
60	0.0145	0.1461	89.20% benar : 446 kata salah : 54 kata
80	0.0168	0.1709	90.60% benar : 453 kata salah : 47 kata
100	0.0211	0.1786	85.20% benar : 426 kata salah : 74 kata

Karena perbedaan keakuratan yang kecil inilah maka akan diujicobakan nilai *learning rate* sebesar 0.002 dengan jumlah unit pada *hidden layer* sebanyak 60 dan 80 unit. Hasil dari percobaan ini dapat dilihat pada Gambar 4.16, Gambar 4.17 dan Tabel 4.2. Dari percobaan ini didapatkan kesimpulan bahwa jumlah *hidden* unit yang optimal adalah sebanyak 80 unit.



Gambar 4.16 Grafik nilai MSE untuk jumlah unit sebanyak 60 unit pada *hidden layer*



Gambar 4.17 Grafik nilai MSE untuk jumlah unit sebanyak 80 unit pada *hidden layer*

Tabel 4.2 Hasil pengujian jumlah unit pada *hidden layer* dengan nilai *learning rate* sebesar 0.002

Jumlah unit pada *hidden layer*	MSE minimum	*Average Error* minimum	Keakuratan
60	0.0157	0.1680	90.00% benar : 450 kata salah : 50 kata
80	0.0133	0.1691	92.20% benar : 461 kata salah : 39 kata

Dari pengujian terhadap jumlah unit pada *hidden layer,* kemudian dilanjutkan dengan pengujian terhadap nilai *learning rate* yang optimal. Jumlah unit pada *hidden layer* yang dipakai dalam percobaan ini adalah 80 unit. Dari hasil pengujian didapatkan kesimpulan bahwa nilai *learning rate* yang optimal adalah 0.005 untuk jumlah unit pada *hidden layer* sebesar 80 unit. Hasil uji coba nilai *learning rate* dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil pengujian nilai *learning rate* dengan jumlah unit pada *hidden layer* sebesar 80 unit

Nilai *Learning rate*	MSE minimum	*Average Error* minimum	Keakuratan
0.002	0.0133	0.1691	92.20% benar : 461 kata salah : 39 kata
0.003	0.0113	0.1361	93.20% benar : 466 kata salah : 34 kata
0.004	0.0078	0.1220	96.20% benar : 481 kata salah : 19 kata
0.005	0.0075	0.1249	96.40% benar : 482 kata salah : 18 kata

Setelah dilakukan pengujian terhadap nilai *learning rate*, kemudian dilakukan uji coba jumlah *epoch.* Jumlah *epoch* yang diujikan adalah sejumlah 3000 *epoch.* Hasil dari pengujian jumlah *epoch* dapat dilihat pada Table 4.4.

Dari tabel 4.4 dapat disimpulkan bahwa struktur jaringan syaraf dengan jumlah *hidden* unit sebanyak 80 unit dan nilai *learning rate* sebesar 0.005 masih dapat belajar meskipun sedikit, hal ini dapat dilihat dari nilai MSE yang lebih kecil pada jumlah *epoch* sebesar 3000 dibandingkan dengan nilai MSE pada jumlah *epoch* sebesar 2000.

Tabel 4.4 Hasil pengujian jumlah *epoch* dengan jumlah unit pada *hidden layer* sebesar 80 unit

Nilai *Learning rate*	MSE minimum	*Average Error* minimum	Keakuratan
0.005	0.0054	0.0992	97.80% benar : 489 kata salah : 11 kata

Setelah mendapatkan arsitektur jaringan yang optimal kemudian dilanjutkan dengan pengujian terhadap kata-kata yang belum pernah dilakukan pembelajaran sebelumnya. Pengujian dilakukan dengan cara menguji sebanyak 50 buah kata yang belum pernah dilakukan pembelajaran dan pengujian dilakukan sebanyak 10 kali percobaan. Bobot yang dipakai adalah bobot dari hasil pembelajaran dengan jumlah unit pada *hidden layer* sebesar 80 unit, nilai *learning rate* sebesar 0.005 dan *max epoch* sebesar 3000. Hasil dari pengujian terhadap kata-kata yang belum pernah dilakukan pembelajaran dapat dilihat pada Tabel 4.5. Contoh kata-kata yang tidak dapat dikenali selama proses pembelajaran terdapat pada Tabel 4.6. Sedangkan contoh kata-kata yang tidak melalui proses pembelajaran dan tidak dapat dikenali dapat dilihat pada Tabel 4.7.

Tabel 4.5 Hasil pengujian terhadap kata-kata yang belum pernah dilakukan pembelajaran

Percobaan ke	**Keakuratan**
1	60.00% benar : 30 kata salah : 20 kata
2	78.00% benar : 39 kata salah : 11 kata
3	56.00% benar : 28 kata salah : 21 kata

4	62.00% benar : 31 kata salah : 19 kata
5	62.00% benar : 31 kata salah : 19 kata
6	66.00% benar : 33 kata salah : 17 kata
7	66.00% benar : 33 kata salah : 17 kata
8	66.00% benar : 33 kata salah : 17 kata
9	60.00% benar : 30 kata salah : 20 kata
10	60.00% benar : 32 kata salah : 18 kata
Rata-rata	63.60%

Tabel 4.6 Contoh kata-kata yang tidak dapat dikenali selama proses pembelajaran

Kata	**Dikenali sebagai**
bandêng	bandëng
bëda	beda
fonem	fonêm
ide	idê
indonesia	indonêsia
mönyët	mönyêt
öngkös	ongkoën
përak	pêrak
töpëng	töpêng

Tabel 4.7 Contoh kata-kata yang tidak melalui proses pembelajaran dan tidak dapat dikenali

Kata	Dikenali sebagai
kêpêrcayaan	këpêrcayaan
öpêr	öpër
sêmêntara	sêmëntara
deflasi	dêflasi
körëksi	korëksi
cömöt	comöt
sêharusnya	sëharusnya
tönjök	tonjok
têrtimpa	tërtimpa
sëkring	sêkring
ëtalasê	êtalasê
ëlëmën	ëlêmën
këmah	kêmah
intêrlökal	intërlökal

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan yang didapat selama pengerjaan skripsi ini adalah :

1. Sistem pengenalan fonem yang dibangun dengan menggunakan metode jaringan syaraf tiruan *backpropagation* dan metode *Second Position Asymmetric Windowing* mampu mengenali perbedaan pengucapan pada vokal 'e' dan 'o' dengan baik.
2. Sistem pengenalan fonem ini dapat mengenali perbedaan pengucapan pada vokal 'e' dan 'o' dengan nilai keakuratan sebesar 97.80% untuk kata-kata yang pernah dilakukan proses pembelajaran dan menghasilkan nilai keakuratan rata-rata sebesar 63.6% untuk kata-kata yang belum pernah dilakukan proses pembelajaran.
3. Pembelajaran dilakukan menggunakan metode *Second Position Asymmetric Windowing* dengan model 1-5 dan metode jaringan syaraf tiruan *backpropagation* dengan jumlah unit pada *input layer* sebanyak 189 unit, jumlah unit pada *hidden layer* sebanyak 80 unit, jumlah unit pada *output layer* sebanyak 30 unit, nilai *learning rate* sebesar 0.005 dan *max epoch* sebesar 3000 pada kata sebanyak 500 buah. Sedangkan pengujian untuk kata-kata yang belum pernah dilakukan pembelajaran melibatkan kata sebanyak 50 buah.

5.2. Saran

Saran yang dapat diberikan setelah pengerjaan skripsi ini adalah :

1. Kata yang digunakan sebagai obyek pembelajaran lebih banyak dan beragam, sehingga jaringan syaraf tiruan *backpropagation* dapat mengenali fonem pada kata lebih banyak lagi.
2. Perlu dilakukan pengujian model *window* yang lainnya pada metode *Second Position Asymmetric Windowing.*

DAFTAR PUSTAKA

Alwi, Hasan dan Soenjono Dardjowidjojo. 2000. *Tata bahasa baku Bahasa Indonesia edisi ketiga.* Balai Pustaka. Jakarta.

Arciniegas, Fabio dan Mark J. Embrechts. 2000. *Text-to-speech with staged neural networks.* Departement of Decision Sciences and Engineering Systems. Rensselaer Polytechnic Institute.

Bakiri, Ghulum dan Thomas G.D. 1997. *Achieving high-accuracy text-to-speech with machine learning.* Departement of Computer Science. University of Bahrain.

Hendessi, F., A. Ghayoori dan T.A. Gulliver. 2004. *A speech synthesizer for Persian text using a neural network with a smooth ergodic HMM.* Dept of Electrical and Computer Engineering. Isfahan University of Technology.

Kusumadewi, Sri. 2003. *Artificial intelligence (teknik dan aplikasinya).* Graha Ilmu. Jakarta.

LeCun, Yann., Leon Bottou., Genevieve B. Orr. dan Klaus-Robert Müller. 1998. *Efficient backprop.* IEEE volume 86 no 11.

O'Neill, Mike. 2006. *Neural network for recognition of handwritten digits.* http://www.codeproject.com/library/NeuralNetRecognition/NeuralNetRecognition.asp. Diakses pada tanggal 10 Desember 2006.

Sejnowski, Terrence J. dan Charles R. Rosenberg. 1986. *NETtalk: a parallel network that learns to read aloud*. The Johns Hopkins University Electrical Engineering and Computer Science Technical Report.

Siang, JJ. 2005. *Jaringan syaraf tiruan dan pemrogramannya menggunakan MATLAB*. Penerbit ANDI Yogyakarta.

Tjiputra, Sudarmono. 2003. *Perangkat lunak text-to-speech dalam bahasa Indonesia dengan metode phoneme synthesis*. Universitas Kristen Petra. Surabaya.

Lampiran 1

Daftar kata untuk proses pembelajaran

No	Kata	Fonem	No	Kata	Fonem
1	absensi	absënsi	34	belum	bêlum
2	adegan	adêgan	35	bentuk	bêntuk
3	adopsi	adöpsi	36	berbasiskan	bêrbasiskan
4	agen	agën	37	berbeda	bêrbeda
5	agenda	agënda	38	berdasarkan	bêrdasarkan
6	agresif	agrësif	39	berupa	bêrupa
7	akademi	akadêmi	40	besar	bêsar
8	akomodasi	akömödasi	41	bodoh	bödöh
9	akreditasi	akrëditasi	42	calon	calön
10	akrobat	akröbat	43	capek	capëk
11	akronim	akrönim	44	cebur	cêbur
12	akses	aksës	45	cegah	cêgah
13	aktor	aktör	46	cek	cëk
14	algoritma	algoritma	47	cela	cêla
15	alkohol	alköhöl	48	celah	cêlah
16	almamater	almamatêr	49	celaka	cêlaka
17	alofon	alofon	50	celana	cêlana
18	aneh	anëh	51	cemar	cêmar
19	aneka	anëka	52	cemas	cêmas
20	anggota	anggöta	53	cemburu	cêmburu
21	ayo	ayo	54	cenderung	cêndêrung
22	bandeng	bandêng	55	cepat	cêpat
23	bebas	bëbas	56	cerah	cêrah
24	bebek	bëbëk	57	cerai	cêrai
25	beberapa	bêbêrapa	58	cerdas	cêrdas
26	becak	bëcak	59	cerewet	cêrëwët
27	beda	bëda	60	cerita	cêrita
28	begini	bêgini	61	ceroboh	cêröböh
29	begitu	bêgitu	62	cocok	cöcök
30	bekerja	bêkêrja	63	contoh	cöntöh
31	belajar	bêlajar	64	debar	dêbar
32	belakang	bêlakang	65	debat	dêbat
33	belanja	bêlanja	66	debu	dêbu

No	Kata	Fonem	No	Kata	Fonem
67	dedikasi	dëdikasi	103	emosi	emosi
68	definisi	definisi	104	enak	ënak
69	dehidrasi	dehidrasi	105	engkau	ëngkau
70	dek	dëk	106	entah	êntah
71	dekade	dekadê	107	faktor	faktör
72	dekat	dêkat	108	fase	fasê
73	deklarasi	dëklarasi	109	favorit	favörit
74	dekorasi	dëkörasi	110	feminin	fëminin
75	delapan	dêlapan	111	filosofi	filosofi
76	demam	dêmam	112	fleksibel	flëksibel
77	demi	dêmi	113	flora	flöra
78	demo	demo	114	fokus	fökus
79	demokrasi	demokrasi	115	fonem	fonem
80	dengan	dêngan	116	fonemnya	fonemnya
81	dengar	dêngar	117	formulir	förmulir
82	dikarenakan	dikarênakan	118	forum	förum
83	dikembangkan	dikêmbangkan	119	fosil	fösil
84	dikenal	dikênal	120	foto	foto
85	diperlukan	dipêrlukan	121	fotografi	fotografi
86	disebut	disêbut	122	fotokopi	fotoköpi
87	dompet	dömpët	123	frekuensi	frëkuënsi
88	dosa	dösa	124	fundamental	fundamëntal
89	ecer	ëcër	125	fungsional	fungsiönal
90	edan	ëdan	126	gandeng	gandëng
91	edar	ëdar	127	gebuk	gêbuk
92	edisi	ëdisi	128	gedung	gêdung
93	edit	ëdit	129	gejala	gêjala
94	efek	ëfëk	130	gejolak	gêjolak
95	efisien	ëfisiën	131	gel	gël
96	ejek	ëjëk	132	gelandang	gêlandang
97	ekonomi	ekonomi	133	gelang	gêlang
98	ekor	ëkör	134	gelap	gêlap
99	eksperimen	ëkspërimën	135	gelar	gêlar
100	ekspor	ëkspör	136	gelas	gêlas
101	elit	ëlit	137	geledah	gêlëdah
102	emas	êmas	138	gelembung	gêlêmbung

No	Kata	Fonem	No	Kata	Fonem
139	gelincir	gêlincir	176	ideologi	idëologi
140	gelisah	gêlisah	177	idola	idöla
141	gelombang	gêlömbang	178	impor	impör
142	gelora	gêlöra	179	indeks	indëks
143	gema	gêma	180	indonesia	indonesia
144	gemar	gêmar	181	infeksi	infëksi
145	gembira	gêmbira	182	insiden	insidën
146	gempa	gêmpa	183	integrasi	intêgrasi
147	gendut	gêndut	184	intelek	intêlëk
148	gerak	gêrak	185	intensif	intënsif
149	gerbang	gêrbang	186	intensitas	intënsitas
150	goncang	göncang	187	interaksi	intêraksi
151	goreng	görëng	188	inventaris	invëntaris
152	gosok	gösök	189	iseng	isêng
153	hakekat	hakëkat	190	istimewa	istimëwa
154	harmonis	harmonis	191	jago	jago
155	hebat	hëbat	192	jahe	jahe
156	heboh	hëböh	193	jaket	jakët
157	hektar	hëktar	194	jebak	jêbak
158	hela	hêla	195	jeda	jêda
159	helm	hëlm	196	jejak	jêjak
160	hemat	hëmat	197	jelajah	jêlajah
161	hembus	hêmbus	198	jelang	jêlang
162	hempas	hêmpas	199	jelas	jêlas
163	hendak	hêndak	200	jelek	jêlëk
164	hening	hêning	201	jeli	jêli
165	henti	hênti	202	jembatan	jêmbatan
166	heran	hëran	203	jempol	jêmpöl
167	hewan	hëwan	204	jemput	jêmput
168	hobi	höbi	205	jemu	jêmu
169	honor	hönör	206	jemur	jêmur
170	hormat	hörmat	207	jenaka	jênaka
171	hotel	hötël	208	jendela	jêndëla
172	ide	ide	209	jenis	jênis
173	idem	idêm	210	kabel	kabêl
174	identifikasi	idëntifikasi	211	kabupaten	kabupatën
175	identitas	idëntitas	212	kaget	kagët

No	Kata	Fonem	No	Kata	Fonem
213	kakek	kakëk	250	lengan	lêngan
214	kaleng	kalëng	251	lokasi	lökasi
215	kalkulator	kalkulatör	252	lolos	lölös
216	kamera	kamëra	253	lompat	lömpat
217	kampanye	kampanyê	254	macet	macêt
218	kanker	kankêr	255	mangkok	mangkök
219	kantong	kantöng	256	matematika	matêmatika
220	kantor	kantör	257	material	matërial
221	kare	kare	258	megah	mêgah
222	karena	karêna	259	meja	mëja
223	kaset	kasët	260	membuat	mêmbuat
224	kategori	kategori	261	membutuhkan	mêmbutuhkan
225	ke	kê	262	memerlukan	mêmêrlukan
226	keadaan	kêadaan	263	memiliki	mêmiliki
227	keakuratan	kêakuratan	264	memproduksi	mêmproduksi
228	keamanan	kêamanan	265	mempunyai	mêmpunyai
229	kemudian	kêmudian	266	memungkinkan	mêmungkinkan
230	keunggulan	kêunggulan	267	menang	mênang
231	komputer	komputêr	268	mencapai	mêncapai
232	lapor	lapör	269	mengambil	mêngambil
233	lebar	lëbar	270	mengatasi	mêngatasi
234	lebat	lêbat	271	mengenali	mêngênali
235	lebih	lêbih	272	menggantikan	mênggantikan
236	ledak	lêdak	273	menggunakan	mênggunakan
237	lega	lêga	274	menghasilkan	mênghasilkan
238	leher	lëhër	275	mengkonversi	mêngkonvërsi
239	lekat	lêkat	276	mengusulkan	mêngusulkan
240	lelah	lêlah	277	menjadi	mênjadi
241	lelaki	lêlaki	278	menurut	mênurut
242	lelang	lëlang	279	menyimpan	mênyimpan
243	leleh	lëlëh	280	merah	mërah
244	leluasa	lêluasa	281	merak	mêrak
245	lem	lëm	282	mereka	mêrëka
246	lemah	lêmah	283	meriah	mêriah
247	lemas	lêmas	284	mesin	mêsin
248	lembab	lêmbab	285	meski	mêski
249	lempar	lëmpar	286	meskipun	mêskipun

No	Kata	Fonem	No	Kata	Fonem
289	metode	metodê	326	obsesi	öbsësi
290	mewakili	mêwakili	327	oceh	öcëh
291	mikroskop	mikroskop	328	oksigen	ongkos
292	mobil	möbil	329	olah	ölah
293	modal	mödal	330	olahraga	ölahraga
294	model	mödël	331	oleh	ölëh
295	mohon	möhön	332	oles	ölës
296	monitor	mönitör	333	ombak	ömbak
297	monyet	mönyët	334	omel	ömël
298	moral	möral	335	omong	ömöng
299	motivasi	mötivasi	336	onar	önar
300	motor	mötor	337	ongkos	öngkös
301	multimedia	multimëdia	338	operasi	öpêrasi
302	nasehat	nasëhat	339	opini	öpini
303	nasional	nasiönal	340	optimal	öptimal
304	negara	nêgara	341	optimis	öptimis
305	negatif	nëgatif	342	orang	örang
306	nelayan	nêlayan	343	pamer	pamër
307	nenek	nënëk	344	panen	panën
308	neon	nëön	345	pasien	pasiën
309	neraka	nêraka	346	paten	patën
310	ngobrol	ngöbröl	347	pecah	pêcah
311	ngompol	ngömpöl	348	pedang	pêdang
312	noda	nöda	349	pembelajaran	pêmbêlajaran
313	nol	nöl	350	pemenggalan	pêmênggalan
314	nominal	nöminal	351	pengenalan	pêngênalan
315	nomor	nömör	352	pengetahuan	pêngêtahuan
316	nona	nöna	353	pengucapan	pêngucapan
317	norma	nörma	354	penulis	pênulis
318	normal	nörmal	355	perak	përak
319	nota	nöta	356	perbedaan	pêrbedaan
320	nyonya	nyönya	357	perkembangan	pêrkêmbangan
321	obat	öbat	358	perkembangannya	pêrkêmbangannya
322	objek	öbjëk	359	pertanyaan	pêrtanyaan
323	obor	öbör	360	peserta	pêsêrta
324	obral	öbral	361	peti	pêti
325	obrol	öbröl	362	pidato	pidato

No	Kata	Fonem	No	Kata	Fonem
363	pot	pöt	400	seberang	sêbêrang
364	presentasi	prësêntasi	401	sebuah	sêbuah
365	produk	produk	402	sebutan	sêbutan
366	profesi	profësi	403	sedangkan	sêdangkan
367	program	program	404	sedap	sêdap
368	propinsi	propinsi	405	sedot	sêdot
369	proses	prösës	406	segala	sêgala
370	radio	radio	407	sejagat	sêjagat
371	rajalela	rajalëla	408	sejak	sêjak
372	rampok	rampök	409	sejuk	sêjuk
373	reaksi	rëaksi	410	sekolah	sêkölah
374	realisasi	rëalisasi	411	selain	sêlain
375	rebah	rêbah	412	selamat	sêlamat
376	rebus	rêbus	413	selasa	sêlasa
377	rebutan	rêbutan	414	seleksi	sêlëksi
378	rekening	rëkêning	415	selesai	sêlêsai
379	rekor	rëkör	416	selimut	sêlimut
380	relasi	rëlasi	417	seluruh	sêluruh
381	remaja	rêmaja	418	semakin	sêmakin
382	remeh	rëmëh	419	sembunyi	sêmbunyi
383	rencana	rêncana	420	semua	sêmua
384	rendah	rêndah	421	semula	sêmula
385	repot	rëpot	422	semut	sêmut
386	resah	rêsah	423	senam	sênam
387	resep	rêsëp	424	seni	sêni
388	resiko	rësiko	425	sepakbola	sëpakbola
389	restoran	rëstöran	426	seperti	sêpêrti
390	restu	rêstu	427	sesuai	sêsuai
391	roda	röda	428	setelah	sêtêlah
392	roh	röh	429	sistem	sistêm
393	rokok	rökök	430	solusi	sölusi
394	roti	röti	431	sore	sore
395	saldo	saldo	432	soto	soto
396	samudera	samudêra	433	teks	tëks
397	sate	sate	434	telah	têlah
398	sebab	sêbab	435	tembak	tëmbak
399	sebentar	sêbêntar	436	tembus	têmbus

No	Kata	Fonem	No	Kata	Fonem
437	tenang	tênang	474	violet	violët
438	tentang	têntang	475	vokal	vokal
439	terdapat	têrdapat	476	volt	völt
440	terima	têrima	477	voltase	völtasê
441	terjalin	têrjalin	478	volume	völumê
442	terlebih	têrlêbih	479	vonis	vönis
443	teror	tërör	480	vulkanologi	vulkanologi
444	tersebut	têrsêbut	481	wafer	wafêr
445	tersedia	têrsêdia	482	walikota	waliköta
446	tertentu	têrtêntu	483	warganegara	warganêgara
447	tertib	têrtib	484	wartel	wartël
448	tetapi	têtapi	485	waskom	wasköm
449	tipe	tipê	486	wastafel	wastafël
450	toko	toko	487	wesel	wësël
451	tokoh	tökòh	488	wewangian	wêwangian
452	tong	töng	489	wewenang	wêwênang
453	topeng	töpeng	490	wijen	wijën
454	topi	töpi	491	wol	wöl
455	tropis	tröpis	492	wortel	wörtel
456	tumor	tumör	493	yodium	yodium
457	ulet	ulêt	494	yoga	yöga
458	ultrasonik	ultrasönik	495	yoyo	yoyo
459	universal	univêrsal	496	yudo	yudo
460	universitas	univêrsitas	497	zebra	zëbra
461	upeti	upêti	498	zionis	ziönis
462	uretra	uretra	499	zodiak	zodiak
463	urine	urinê	500	zona	zöna
464	variabel	variabêl			
465	vasektomi	vasëktomi			
466	vegetarian	vëgëtarian			
467	ventilasi	vëntilasi			
468	verifikasi	vërifikasi			
469	versi	vërsi			
470	vertikal	vërtikal			
471	veteran	vëtêran			
472	veto	vetö			
473	video	videö			