PERANCANGAN PERANGKAT LUNAK UNTUK PENGENALAN KARAKTER ASCII DARI GAMBAR BITMAP MENGGUNAKAN JARINGAN SARAF TIRUAN METODE PROPAGASI BALIK

Ema Utami[1] dan Sri Hartati[2]

ABSTRACT

ANN is a method in soft computing that imitate the structure of biological nervous, where multiple nodes communicate with each other through synapses that interconnect them. That method can be utilized for pattern recognition processes, such as ASCI Icharacter recognition.

This paper is aimed to develop ASCII character recognition using ANN method. Multilayer neural network is adopted in ANN with back propagation algorithm. A mapping method is used for preprocessing to segment a character image to be processed by ANN.

The experimental results show that among all of successful segmented characters of all the training data having the different size of font, the system successfully recognizes characters up to 81.74 %. For the training data with different sizes and fonts data, the system recognizes the characters with an accuracy close to 55.74%. Among the unsuccessful segmented characters of all training data having different sizes of font, the system recognizes characters with an accuracy 67.53%. For the different sizes and fonts data, the system recognizes the characters with an accuracy close to 46.65%.

***Keywords** : ASCII character recognition, artificial neural network, back propagation, Testing*

INTISARI

Pengenalan karakter ASCII merupakan salah satu bidang dalam ilmu komputer yang dapat membantu proses pengolahan data. Salah satu teknik pengenalan karakter adalah metode Jaringan Saraf Tiruan (JST), dimana metode ini menggunakan prinsipotak manusia yang terdiri dari *neuron* sebagai pemrosesan *input* untuk menghasilkan *output* berdasarkan bobot yang ada.

Tujuan dari paper ini adalah membuat perangkat lunak yang dapat melakukan pengenalan karakter ASCII dengan menggunakan metode JST. Arsitektur jaringan saraf tiruan yang digunakan adalah *multilayer neural network*, dengan algoritma pembelajaran propagasi balik. Untuk membantu pengenalan karakter ASCII, dilakukan proses pemotongan gambar menggunakan metode pemetaan.

Hasil uji coba menunjukkan bahwa ujicoba dari seluruh karakter yang berhasil disegmentasi untuk jenis font yang sama dengan pelatihan, dengan ukuran font yang berbeda berhasil dikenali sebanyak 81,74%, sedangkan dari seluruh karakter yang gagal disegmentasi berhasil dikenali sebanyak 67,53%. Ujicoba dari seluruh karakter yang berhasil disegmentasi untuk jenis font yang berbeda dengan pelatihan, dengan ukuran font yang berbeda berhasil dikenali sebanyak 55,47%, sedangkan dari seluruh karakter yang gagal disegmentasi berhasil dikenali sebanyak 46,65%.

Kata kunci : pengenalan karakter ASCII, jaringan saraf tiruan, propagasi balik, pengujian.

PENDAHULUAN

Kemajuan teknologi membuat sebuah perangkat komputer memiliki kemampuan komputasi yang tinggi untuk meningkatkan kinerja dalam pengolahan data menjadi informasi. Hal yang dapat dilakukan oleh manusia untuk mengimbangi kemajuan teknologi adalah dengan mengembangkan sistem yang dapat memanfaatkan teknologi tersebut untuk membantunya memasukkan data kedalam komputer. Salah satu dari sistem tersebut adalah sistem pengenalan karakter (*character recognition system*). Pengenalan karakter ASCII (*American Standard Code for Information Interchange*) adalah suatu sistem dimana data yang akan menghasilkan gambar pada komputer a-

[1] Jurusan Sistem Informasi STMIK AMIKOM Yogyakarta
[2] Program Studi Ilmu Komputer Sekolah Pascasarjana Universitas Gadjah Mada

kan dikenali sebagai titik-titik (*bitmap*).*Bitmap* inilah yang kemudian akan diproses lebih lanjut dengan menggunakanalgoritma tertentu menjadi karakter sehingga dapat dikenali dan diolah menjadi informasi (Rodrigues, 2000).

Sistem pengenalan karakter ASCII akan diterapkan pada perangkat lunak yang nantinya menghasilkan karakter berdasarkan gambar yang dimasukkan oleh pengguna. Salah satu algoritma yang dapat diterapkan untuk pengenalan karakter ASCII adalah jaringan saraf tiruan (*artificial neural network*).Berdasarkan fungsinya, JST bertujuan untuk memecahkan sebuah masalah dengan cara belajar dari pengalaman. Dalam hal ini gambar *bitmap* dan karakter yang akan dihasilkan dari gambar tersebut akan diberikan kepada sistem JST sebagai pengalaman. Dari pengalaman ini diharapkan nantinya sistem JST dapat melakukan pengenalan karakter ASCII untuk gambar yang lainnya.

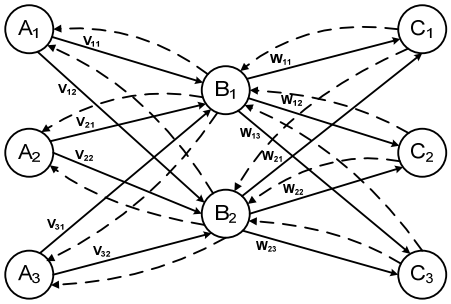
Adapun ruang lingkup untuk pembuatan perangkat lunak ini meliputi :

a. Gambar yang dibacakan adalah gambar *bitmap (file bmp)*.
b. Karakter ASCII yang akan dikenali adalah karakter ASCII cetak yang meliputi 52 karakter alphabet untuk huruf besar dan huruf kecil, 10 karakter numeris dan 32 karakter simbol-simbol khusus termasuk tanda baca. Karakter-karakter tersebut menggunakan berbagai jenis font seperti *Times New Roman*, *Comic Sans MS*, *Microsoft Sans Serif*, *Tahoma*, *Verdana*, *Arial*, *Book Antiqua*, *Courier New*, *Garamond* dan *Lucida Unicode.*
c. Menggunakan bahasa pemrograman Visual Basic 6.
d. Menggunakan metode propagasi balik.

Metode propagasi balik adalah metode pelatihan yang terawasi dan digunakan untuk melatih *multilayer neural network* untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang terdapat pada *hidden layer* (Bernacki, 2005). Metode propagasi balik menggunakan error output untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan error ini, tahap umpan maju (*feed forward input training pattern*) harus dikerjakan terlebih dahulu. Pada saat umpan maju neuron-neuron diaktifkan menggunakan fungsi aktivasi sigmoid.

$$Y = f(x) = \frac{1}{1 + e^{-x}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Arsitektur jaringan propagasi balik ditunjukkan pada gambar 1 :



Gambar 1. Arsitektur jaringan propagasi balik

Secara umum proses pelatihan untuk metode propagasi balik terdiri dari 3 tahap yaitu:

Feedforward input training pattern (tahap umpan maju)

Merupakan tahap untuk melakukan perhitungan dengan mencari nilai *output* pada setiap *layer* yang akan menjadi nilai *input* pada *layer* selanjutnya, dan nilai *output* pada *layer* terakhir merupakan *output* dari sistem tersebut.

a. Back propagation associated error (tahap pemropagasibalikan error)

Merupakan tahap untuk melakukan perhitungan *error* dan nilai koreksi pada *layer* terakhir berdasarkan target yang telah diberikan dalam pelatihan dibandingkan. *Error* pada *layer* sebelumnya akan menggunakan *error* pada *layer* di atasnya.

b. *Adjustment weight* (tahap peng-update-an bobot)

Merupakan tahap untuk mengubah bobot yang dimiliki oleh setiap node pada tiap *layer*, mulai dari *layer output* sampai pada *hidden layer* yang pertama.

Tahap-tahap pelatihan di atas menghasilkan algoritma pelatihan *neural network* sebagai berikut (Karray, 2004) :

Langkah 0
Menginisialisasi bobot (setiap bobot diberi nilai acak antara 0 - 1)
Langkah 1
Selama kondisi berhenti belum terpenuhi lakukan langkah 2-13
Langkah 2
Melakukan langkah 3-13 sebanyak jumlah pelatihan yang diinginkan
Langkah 3
Untuk *output layer* dan setiap *hidden layer* lakukan langkah 4-13
Tahap umpan maju :
Langkah 4
Menghitung *input* setiap *node* pada *hidden layer*

$$z\_in_j = \sum_{i=1}^{n} X_i * w_{ij} \quad \ldots\ldots\ldots\ldots\ldots (2)$$

dimana :
z_in_j : jumlah total input untuk node ke-j
n : jumlah node pada *hidden layer* sebelumnya, untuk *hidden layer* yang pertama n adalah jumlah *node input*
x_i : nilai *node* ke-i yang dikeluarkan oleh *hidden layer* sebelumnya, untuk *hidden layer* yang pertama x_i adalah *input* yang diterima oleh sistem
w_{ij} : bobot yang menghubungkan antara *node* ke-i dan *node* ke-j

Langkah 5
Menghitung *output* setiap *node* pada *hidden layer* dengan fungsi aktifasi.

$$z_i = f(z\_in_{nj}) \quad \ldots\ldots\ldots\ldots\ldots (3)$$

dimana :
z_j : *output* dari *node* ke-j

$$f(x) = \frac{1}{1 + e\ x\ p\ (-x)} \quad \ldots\ldots\ldots\ldots (4)$$

Langkah 6
Menghitung *input* setiap *node* pada *output layer*

$$y\_in_1 = \sum_{j=1}^{n} z_i * w_{j1} \quad \ldots\ldots\ldots\ldots\ldots (5)$$

dimana :
y_in_1 : jumlah total *input node* ke-k
n : jumlah *node* pada *hidden layer* sebelum *output layer*
w_{j1} : bobot yang menghubungkan antara *node* ke-j dan *node* ke-k

Langkah 7
Menghitung *output* pada setiap *node* pada *output layer.*

$$y_i = f(z\_i_{nj}) \quad \ldots\ldots\ldots\ldots\ldots (6)$$

dimana:
y_i : *output* dari *node* ke-k

Tahap pemropagasibalikan error :
Langkah 8
Menghitung *error* setiap *node* pada *output layer* dengan fungsi deaktifasi

$$\delta_i = (t_i - y_i) * f^1(y\_i_{ni}) \quad \ldots\ldots\ldots\ldots (7)$$

dimana :
δ_i : *error* pada *node* ke-k
t_i : target ke-k
$f^1(x)$: $f(x)[1-f(x)]$

Tahap peng-update-an bobot :
Langkah 9
Menghitung perubahan bobot pada setiap *node* pada setiap *hidden layer*

$$\Delta W_{ji} = \alpha * \delta_i \quad \ldots\ldots\ldots\ldots (8)$$

dimana :
Δw_{ji} : perubahan bobot yang menghubungkan *node* ke-j dan *node* ke-k
α : *learning rate* yang merupakan nilai 0 – 1

Langkah 10
Menghitung *error* setiap *node* pada *output layer* dengan fungsi deaktifasi

$$\delta_j = (\sum_{i=1} \delta_i * wji) * f'(z\_inj) \quad \ldots\ldots\ldots\ldots (9)$$

dimana :
δ_j : *error* pada *node* ke-j

Langkah 11
Menghitung perubahan bobot pada setiap *node* pada setiap *hidden layer*

$$\Delta W_y = \alpha * \delta_j \quad \ldots\ldots\ldots\ldots (10)$$

dimana :
Δw_y : perubahan bobot yang menghubungkan *node* ke-j

Langkah 12
Me-update bobot pada setiap *node* pada *output layer*

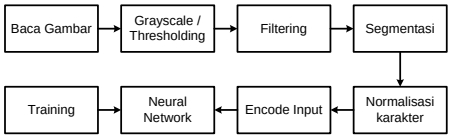
$$w_{ji}(new) = w_{ji}(old) + \Delta w_{ji} \quad \ldots\ldots\ldots\ldots\ldots (11)$$

Langkah 13
Me-update bobot pada setiap *node* pada setiap *hidden layer*

$$w_y(new) = w_y(old) + \Delta w_y \quad \ldots\ldots\ldots\ldots\ldots (12)$$

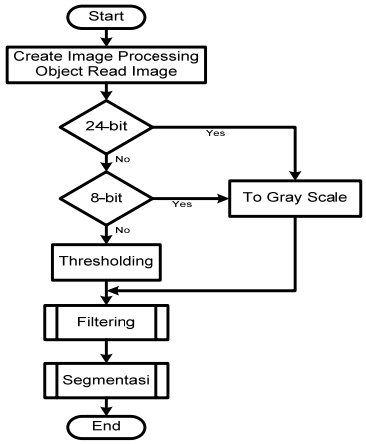
PEMBAHASAN

Rangkaian proses yang akan dilakukan oleh perangkat lunak pengenalan karakter ASCII ini adalah baca gambar, *gray scale/thresholding*, *filtering,* segmentasi, normalisasi karakter input, *encode input*, dan *neural network*. Proses *neural network* sendiri membutuhkan proses *training* agar *output* yang dihasilkan dapat sesuai dengan yang diinginkan. Proses ini dapat digambarkan dengan blok diagram sebagai berikut :



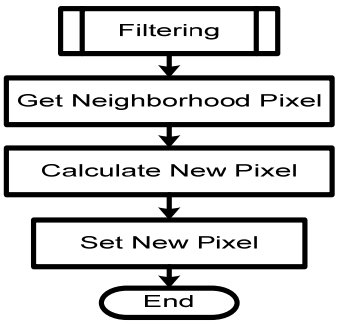
Gambar 2. Blok diagram perangkat lunak pengenalan karakter ASCII

Gambar yang diolah berupa rangkaian karakter ASCII akan dibaca dan setiap karakter ASCII pada gambar akan dicari posisi dan ukuran (segmentasi). Sebelum gambar tersebut disegmentasi terdapat proses *thresholding* untuk memudahkan pengolahan data dan proses *filtering* untuk menghasilkan gambar dengan kualitas yang lebih baik.



Gambar 3. Flowchart pengolahan gambar

Proses *filtering* dilakukan dengan menghitung nilai baru pada sebuah titik pada gambar berdasarkan nilai dari titik itu dan nilai dari titik-titik yang berada disekitarnya.



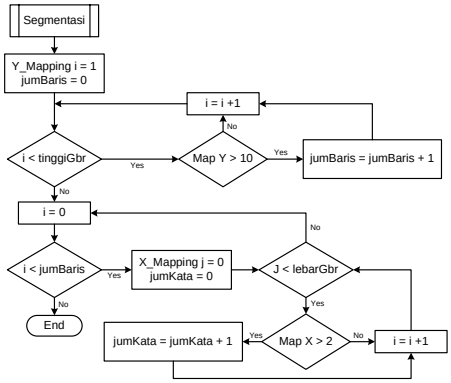
Gambar 4. Flowchart *filtering*

Jenis filter untuk melakukan perhitungan *pixel* baru (*calculate new pixel*) adalah modifikasi dari *lowpass filtering* yaitu dengan menggunakan *convolution window* yang berisi 2 untuk *pixel* disekitarnya dan bernilai 5 untuk pixel itu sendiri. Hasil dari perkalian nilai sekitar (berukuran 3x3) dengan *convolution window* akan di-*threshold* dengan nilai 10.

2	2	2
2	5	2
2	2	2

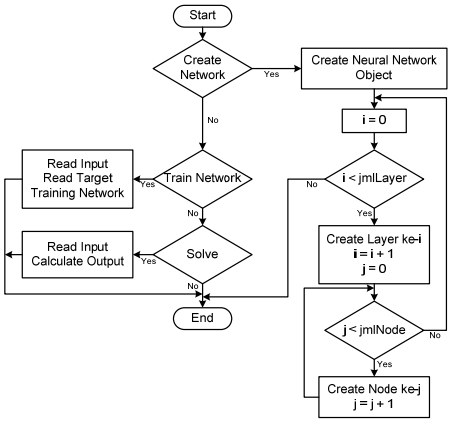
Gambar 5. *Filter convolution window* yang digunakan

Proses segmentasi dilakukan dengan memetakan jumlah titik hitam pada setiap baris dari gambar ke sumbu Y, untuk mendapatkan tinggi karakter pada setiap baris (*Y-Mapping*), dimana pemetaan dengan jumlah yang lebih sedikit dari 10 akan menjadi batas untuk setiap baris karakter (Rodrigues, 2000). Setiap baris karakter dari hasil pemetaan tersebut dipetakan lagi ke sumbu X, untuk mendapatkan lebar dari masing-masing karakter untuk setiap baris (*X-Mapping*), dimana pemetaan dengan jumlah yang lebih sedikit dari 2 akan menjadi batas untuk setiap lebar dari masing-masing karakter.



Gambar 6. Flowchart Segmentasi

Proses *neural network* dijalankan apabila pengguna memilih menu *image – recognize* atau memilih menu *network*. Proses ini meliputi pembuatan, pelatihan, dan pengenalan *neural network*. Proses pembuatan *neural network* meliputi pembuatan *layer* dan *node*. Untuk melakukan pelatihan diperlukan proses untuk membaca input dan target, sedangkan untuk melakukan pengenalan diperlukan proses untuk membaca input yang akan dikenali.



Gambar 7. Flowchart *neural network*

Proses *neural network* akan membuat sebuah obyek *neural network* yang terdiri dari 5 buah *object layer* (empat *hidden layer* dan satu *output layer*) dan sebuah *input layer* berupa array dinamis dengan ukuran 576 sebagai masukan bagi *object layer* yang pertama. *Output layer* pada *neural network* ini terdiri dari 7 buah *node* sehingga jumlah maksimal karakter yang dapat dikenali adalah 128 karakter (2^7). Jumlah *output node* ini ditentukan berdasarkan jumlah karakter yang harus dikenali yaitu 94 karakter yang masing-masing terdiri dari :

a. 52 karakter untuk huruf besar dan kecil
b. 10 karakter untuk angka
c. 32 karakter untuk tanda-tanda baca

Pada saat pembuatan sebuah *object layer* dibutuhkan parameter yang akan menjadi daftar input (*input list*) bagi *layer* tersebut. Dalam hal ini parameter bagi objek pertama adalah array dinamis yang merupakan *input layer*. Parameter bagi *object layer* kedua sampai dengan *output list* (juga berupa array dinamis). Dalam setiap *object layer* memiliki 500 *object node*, sedangkan untuk *layer* terakhir yang menjadi *output layer* hanya akan memiliki 7 *object node.*

Object node sendiri mempunyai ciri yang sama dengan *artificial neuron* yaitu bahwa masing-masing *node* akan terhubung dengan setiap *node* pada *layer* di bawahnya dan nilai *output* pada *node* yang di bawahnya akan menjadi nilai *input* bagi *node* tersebut (*input list*), dan setiap *input* akan memiliki bobotnya masing-masing (*weight list*).

Setiap *node* juga akan memiliki *output* yang dihitung berdasarkan fungsi aktifasi. Fungsi sig-moid biner memiliki nilai range 0 sampai 1. Oleh karena itu, fungsi ini sering digu-nakan untuk *neural network* yang mem-butuhkan nilai output yang terletak pada interval 0 sampai 1. Fungsi sigmoid biner dirumuskan sebagai berikut (Negnevit-sky, 2002):

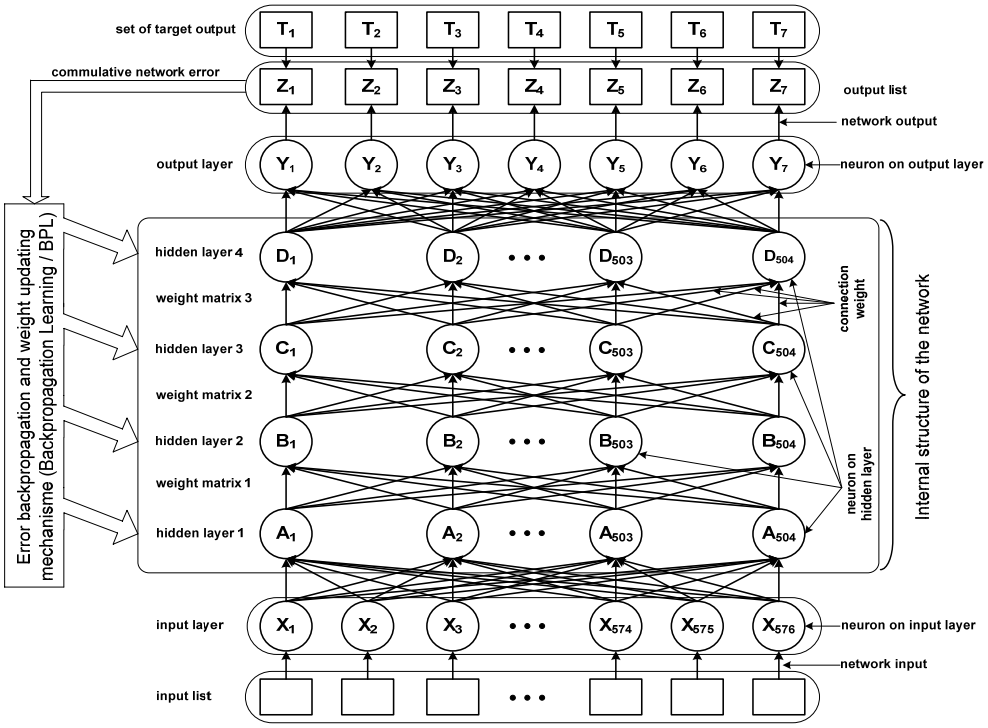
$$Y = f(x) = \frac{1}{1 + e^{-\sigma x}} \quad \ldots\ldots\ldots\ldots\ldots (13)$$

$$\text{dengan } f^1(x) = \sigma f(x)[1-f(x)] \quad \ldots\ldots\ldots\ldots\ldots (14)$$

Dan untuk menghasilkan *binary output* yaitu 0 dan 1 diperlukan fungsi *thresholding* yang dirumuskan sebagai :

$$Y = f(x) = \begin{cases} 0 \text{ , jika } x < \theta \\ 1 \text{ , jika } x \geq \theta \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (15)$$

Secara garis besar modul ini akan membentuk sebuah *neural network* dengan satu *input layer* (506 node), empat *hidden layer* (masing-masing 504 node), dan satu *output layer* (7 node). Setiap citra karakter dibagi menjadi matriks berukuran 24x24 (24 baris, 24 kolom). Setiap elemen matriks bernilai 1 atau 0.



Gambar 8. Jaringan propagasi balik dengan empat buah *hidden layer*

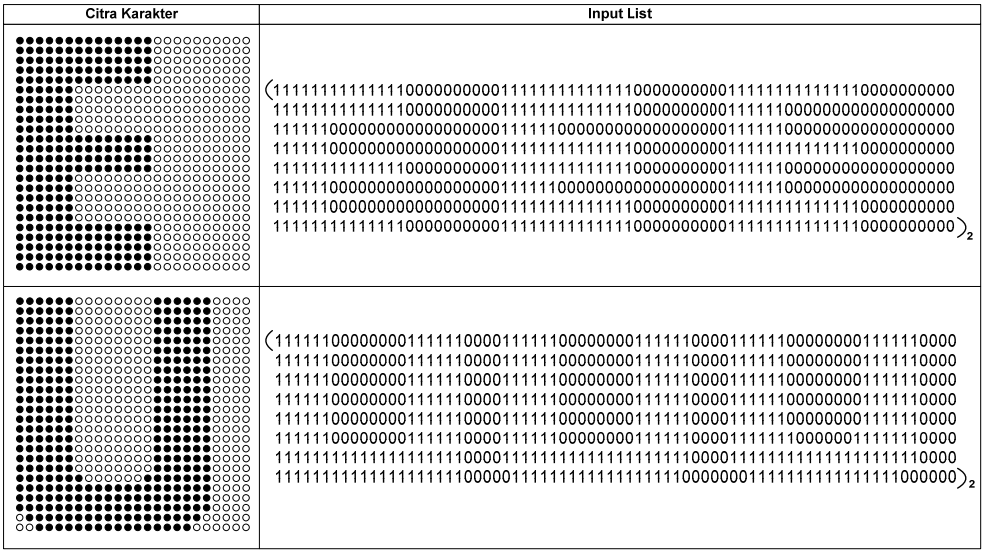
Selanjutnya matriks tersebut dibawa ke bentuk vektor dengan ukuran 24x24 (576 elemen). Penentuan 576 *node* pada *input layer* didasarkan pada hasil dari proses segmentasi yaitu posisi, lebar dan tinggi untuk setiap karakter. Setiap karakter hasil segmentasi diterjemahkan ke dalam bentuk matriks berukuran 24x24 pixel (atau =576) yang berisikan bilangan-bilangan biner yang melambangkan citra *pixel* (*picture element*) pemetaan bit (*bit-mapped pixel image*) dari sebuah karakter ASCII ke dalam bentuk kode ASCII 8 bit. Angka 1 menunjukkan *pixel* berisi citra, sedangkan angka 0 menunjukkan kosong.

Setiap karakter berupa vektor berukuran 576 elemen, sedangkan karakter ASCII yang akan dikenali adalah 94 karakter, sehingga digunakan matriks berukuran 94x576. Dari matriks berukuran 24x24 ini digunakan 3 baris sebagai bobot. Ukuran matriks bobot 24x3 (72 elemen) sehingga untuk pembelajaran hanya dilakukan pada elemen-elemen sisanya, yaitu matriks berukuran 21x24 (504 elemen).

Jumlah node untuk *output layer* adalah sebanyak 7 *node* dengan pertimbangan bahwa perlambangan 94 karakter ASCII (52 karakter alphabet untuk huruf besar dan huruf kecil, 10 karakter numeris dan 32 karakter simbol-simbol khusus termasuk tanda baca) dalam bentuk biner akan menghabiskan tempat sebanyak 7 digit. Pada pengolahan data output, kesembilanpuluhempat karakter tersebut diurutkan sesuai dengan urutan binernya.

JST diuji dengan menggunakan 2 set data, yaitu set pelatihan (set training) dan set testing.

1. Pengujian JST Menggunakan Set Pelatihan



Gambar 9. Citra karakter E dan U yang dipetakan sebagai input jaringan

Sebelum sistem *neural network* dapat melakukan pengenalan, sistem tersebut harus dilatih terlebih dahulu. Sistem dilatih untuk mengenali 5 jenis font berbeda yaitu *Times New Roman, Comic Sans MS, Microsoft Sans Serif, Tahoma* dan *Verdana* dengan font berukuran 12 pt dan setiap jenis font akan memiliki 790 sampel yang terdiri dari :

a. 10 sampel huruf besar dan huruf kecil = 10 x 52 = 520
b. 10 sampel angka = 10 x10 = 100
c. 10 sampel koma dan titik = 10 x 2 = 20
d. 5 sampel tanda-tanda baca (tanpa koma dan titik) = 5 x 30 = 150

Sehingga total karakter ASCII dari pelatihan tersebut adalah 3950 karakter.

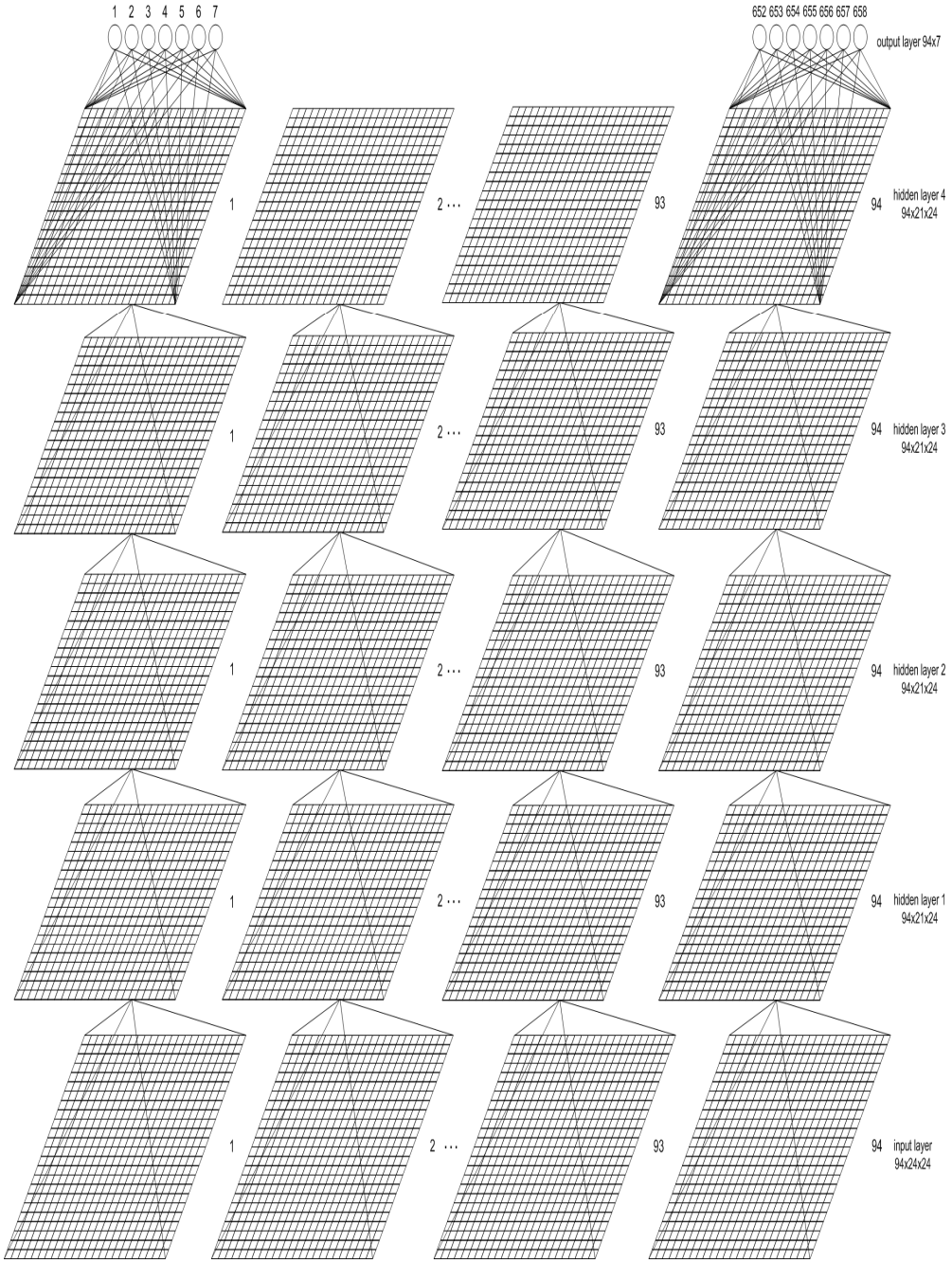
Proses melatih *network* berdasarkan gambar karakter yang dimasukkan sebagai target. Pada saat pelatihan, selain gambar karakter yang akan dilatih sebagai input bagi *object neural network* diperlukan juga sebuah nilai yang menjadi target bagi *object neural network* ini. Nilai target ini akan diberikan berdasarkan nilai ASCII dari karakter yang akan dilatih, misalnya pelatihan karakter 'B' akan memberikan nilai 66 sebagai target bagi *object neural network* ini. Setelah proses pelatihan berhasil atau dihentikan oleh pengguna, maka perangkat lunak ini akan meminta pengguna untuk menyimpan bobot dari *network* yang telah dilatih pada file yang diinginkan.

Setelah menyimpan bobot tersebut, maka file yang berisikan bobot tersebut dapat dibaca kembali menggunakan perangkat lunak ini untuk melakukan pengenalan karakter ASCII. Ketika sistem diuji dengan memasukkan set pelatihan, yang diuji adalah ingatan jaringan, sebab kasus-kasus yang dimasukan telah dipelajari sebelumnya.

Jumlah set pelatihan (3950 kasus) dianggap cukup mampu melatih jaringan untuk mengenal karakter-karakter yang telah dipelajarinya setelah melakukan 280 iterasi belajar.

2. Pengujian JST Menggunakan Set Testing

Dari pengalaman selama pelatihan diharapkan jaringan akan mampu mengeneralisasikan kasus yang dihadapi dan kemudian menarik kesimpulan yang cenderung ke output tertentu.



Gambar 10. Rancangan arsitektur jaringan saraf propagasi balik untuk pengenalan karakter ASCII

JST diuji menggunakan set testing berjumlah 22460 kasus yang meliputi kasus-kasus yang pernah dipelajari jaringan sebanyak 3950 kasus dan kasus-kasus yang belum pernah dipelajari jaringan sebanyak 18510 kasus.

2.1 Pengujian JST menggunakan set testing sama dengan karakter yang dilatihkan

Set testing pertama berupa set gambar yang telah dilatihkan kepada sistem yang dipisahkan berdasarkan jenis font masing-masing. Berdasarkan hasil pengenalan tersebut, berikut ini adalah prosentase hasil pengenalan karakter ASCII dari gambar yang dilatih dibandingkan hasil yang seharusnya didapatkan.

Tabel 1. Output dan karakter yang diwakilinya

Output List	Target	
1234567	Nilai	Karakter
100001	33	!
100010	34	"
100011	35	#
100100	36	$
100101	37	%
100110	38	&
100111	39	`
101000	40	(
101001	41	)
101010	42	*
101011	43	+
101100	44	,
101101	45	-
101110	46	.
101111	47	/
110000	48	0
110001	49	1
110010	50	2
110011	51	3
110100	52	4
110101	53	5
110110	54	6
110111	55	7
111000	56	8
111001	57	9
111010	58	:
111011	59	;
111100	60	<
100001	61	=
111110	62	>
111111	63	?
1000000	64	@
1000001	65	A
1000010	66	B
1000011	67	C
1000100	68	D
1000101	69	E
1000110	70	F
1000111	71	G
1001000	72	H
1001001	73	I
1001010	74	J
1001011	75	K
1001100	76	L
1001101	77	M
1001110	78	N
1001111	79	O
1010000	80	P
1010001	81	Q
1010010	82	R
1010011	83	S
1010100	84	T
1010101	85	U
1010110	86	V
1010111	87	W
1011000	88	X
1011001	89	Y
1011010	90	Z
1011011	91	[
1011100	92	\
1011101	93	]
1011110	94	^
1011111	95	_
1100000	96	`
1100001	97	a
1100010	98	b
1100011	99	c
1100100	100	d
1100101	101	e
1100110	102	f
1100111	103	g
1101000	104	h
1101001	105	i
1101010	106	j
1101011	107	k
1101100	108	l
1101101	109	m
1101110	110	n
1101111	111	o
1110000	112	p
1110001	113	q
1110010	114	r

1110011	115	s
1110100	116	t
1110101	117	u
1110110	118	v
1110111	119	w
1111000	120	x
1111001	121	y
1111010	122	z
1111011	123	{
1111100	124	|
1111101	125	}
1111110	126	`

Tabel 2. Prosentase hasil pengenalan menggunakan set testing sama dengan karakter yang dilatihkan

Jenis Font	Jumlah Karakter ASCII	Jumlah yang Benar	Prosentase Kebenaran
Times New Roman	790	785	99.37
Comis Sans Serif	790	781	98.86
Microsoft Sans Serif	790	773	97.85
Tahoma	790	776	98.23
Verdana	790	783	99.11
Rata-rata	790	779.6	98.68

2.2 Pengujian JST menggunakan set testing dengan jenis font yang sama dengan karakter yang dilatihkan dan ukuran font yang tidak sama dengan karakter yang dilatihkan

Set testing kedua berupa set gambar yang berisi teks yang berbeda dengan yang dilatihkan pada sistem namun dengan jenis font yang sama: *Times New Roman, Comic Sans MS, Microsoft Sans Serif, Tahoma,* dan *Verdana*. Setiap jenis font akan diuji menggunakan 3 kombinasi yaitu berdasarkan ukuran font (10pt, 12pt dan 14pt).

Karena proses segmentasi gambar menggunakan metode *mapping* maka pemotongan yang dihasilkan dapat menjadi tidak valid, dimana karakter ASCII yang saling berpotongan akan dianggap sebagai satu karakter ASCII dan karakter ASCII yang tersambung dengan sangat tipis akan dianggap sebagai dua karakter ASCII. Hasil pemotongan yang tidak valid ini akan menimbulkan kesalahan pada saat pengenalan, oleh karena itu perhitungan prosentase pengenalan karakter ASCII akan dibagi menjadi 2 yaitu berdasarkan hasil pemotongan yang valid dan berdasarkan hasil pemotongan yang tidak valid.

Tabel 3. Prosentase kebenaran pengenalan menggunakan set testing dengan jenis font yang sama dengan karakter yang dilatihkan dan ukuran font yang tidak sama dengan karakter yang dilatihkan

Jenis Font	Ukuran Font	Jumlah Kata	Jumlah Valid	Jumlah Benar	Prosentase Benar pada Hasil Pemotongan yang Valid	Prosentase Benar pada Hasil Pemotongan yang tidak Valid
Times New Roman	10	617	355	270	78.06	43.76
	12	617	387	321	82.95	52.03
	14	617	499	411	82.36	66.61
Comic Sans MS	10	617	384	329	85.68	53.32
	12	617	492	409	83.13	66.29
	14	617	582	492	84.54	79.74
Microsoft Sans Serif	10	617	522	428	81.99	69.37
	12	617	530	453	85.47	73.42
	14	617	604	488	80.79	79.09
Tahoma	10	617	436	359	82.34	58.18
	12	617	496	416	83.87	67.42
	14	617	579	468	80.19	75.85
Verdana	10	617	562	436	77.58	70.66
	12	617	607	479	78.91	77.63
	14	617	611	491	80.36	79.58
Rata-rata		617	509.73	416.67	81.74	67.53

Tabel 4. Prosentase kebenaran pengenalan menggunakan set testing dengan jenis dan ukuran font yang tidak sama dengan karakter yang dilatihkan

Jenis Font	Ukuran Font	Jumlah Kata	Jumlah Valid	Jumlah Benar	Prosentase Benar pada Hasil Pemotongan yang Valid	Prosentase Benar pada Hasil Pemotongan yang tidak Valid
Times New Roman	10	617	513	377	73.49	61.10
	12	617	534	419	78.46	67.91
	14	617	579	417	81.35	76.34
Comic Sans MS	10	617	416	245	58.89	39.71
	12	617	442	267	60.41	43.27
	14	617	449	252	56.12	40.84
Microsoft Sans Serif	10	617	602	137	22.76	22.20
	12	617	605	134	22.15	21.72
	14	617	605	123	20.33	19.94
Tahoma	10	617	394	118	29.95	19.12
	12	617	416	129	31.01	20.19
	14	617	422	121	28.67	19.61
Verdana	10	617	599	458	76.46	74.23
	12	617	603	517	85.74	83.79
	14	617	603	549	91.04	88.98
Rata-rata		617	548.25	411.25	55.47	46.65

2.3Pengujian JST menggunakan set testing dengan jenis dan ukuran font yang tidak sama dengan karakter yang dilatihkan

Set testing kedua berupa set gambar dengan jenis font yang dipilih adalah *Arial, Book Antiqua, Courier New, Garamond,* dan *Lucida Sans Unicode*. Seperti pada uji coba dengan jenis font yang sama, uji coba untuk kelima font ini

didasarkan pada 3 kombinasi yaitu font dengan ukuran 10pt, 12pt, dan 14pt.

KESIMPULAN

Pada proses pembelajaran adanya kesalahan pada satu karakter akan menyebabkan kesalahan pengenalan karakter tersebut semakin besar.

JST diuji dengan menggunakan 2 set data, yaitu set pelatihan dan set testing.

Set pelatihan berjumlah 3950 kasus, sedangkan set testing berjumlah 22460 kasus yang meliputi kasus-kasus yang pernah dipelajari jaringan sebanyak 3950 kasus dan kasus-kasus yang belum pernah dipelajari jaringan sebanyak 18510 kasus.

Hasil pengujian JST menggunakan set testing sama dengan karakter yang dilatihkan menghasilkan pengenalan sebesar 98.68%.

Pengujian JST menggunakan set testing dengan jenis font yang sama dengan karakter yang dilatihkan dan ukuran font yang tidak sama dengan karakter yang dilatihkan menghasilkan pengenalan sebesar 81,74% untuk pemotongan yang sukses dan 67,53% untuk yang gagal.

Pengujian JST menggunakan set testing dengan jenis dan ukuran font yang tidak sama dengan karakter yang dilatihkan menghasilkan pengenalan sebesar 55,47% untuk pemotongan yang sukses dan 46,65% untuk yang gagal.

Rata-rata hasil pengenalan untuk setiap jenis karakter yang tidak dilatihkan bergantung pada jenis karakter tersebut. Apabila model karakter tersebut mirip dengan salah satu model karakter yang dilatihkan maka rata-rata hasil pengenalan karakter tersebut akan tinggi.

DAFTAR PUSTAKA

Bernacki, Mariusz & Wlodarczyk, Przemyslaw, 2005, *Principles of Training Multi-Layer Neural Network Using Backpropagation Algorithm*(pdf), http://www.galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/-Backprop.html/downloads/

Karray, Fakhreddine & De Silva, Clarence, 2004, *Soft Computing and Intelligent Systems Design*, Pearson Addison-Wesley Publishing Company Negnevitsky,

Michael, 2002, *Artificial Intelligence A Guide to Intelligent Systems*, Addison-Wesley Publishing Company

Rodrigues, J. R, & Thome, A. C, 2000, *Cursive CharacterRecognition – A Character Segmentation Method Using Projection Profile Based Technique* (pdf),http:// www.nce.ufrj.br/labic/downloads/