Kompresi Data Teks Menggunakan Autoencoder Neural Network (AENN)

M Nur Alfiansyah, Muhammad Soleh, Achmad Fanany Onnilita Gaffar, Rheo Malani
Politeknik Negeri Samarinda
Teknik Informatika
Samarinda, Indonesia
mnuralfiansyah@gmail.com,Muhammadsoleh1997@gmail.com,onnygaffar212@gmail.com,anaogie@gmail.com

Haviluddin
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Mulawarman
haviluddin@unmul.ac.id

Abstract—salah satu jenis data adalah data teks. Data teks biasanya dikodekan dalam bentuk kode ASCII yang memiliki panjang binner 8. Kompresi data teks bertujuan untuk mengurangi pemakaian ruang penyimpanan pada memori. Penelitian ini menggunakan Auto-Encoder Neural Network (AENN). Hasil dari penelitian ini menghasilkan rasio sebesar 88,62%, namun masih ada keterbasan jika panjang asli ganjil maka setelah didekompresi menghasilkan panjang yang genap dan keterbasan pembagi hidden yang menyebabkan data tidak kembali ke aslinya.

Keywords—kompresi,dekompresi,AENN

I. PENDAHULUAN

Kompresi data adalah teknik untuk mengurangi redundansi dalam representasi data guna mengurangi kebutuhan penyimpanan data dan biaya komunikasi. Cara efisien untuk menyimpan banyak data dan keterbatasan bandwidth, maka data harus dikompres sebelum dikirim dan disimpan. Kemudian untuk mengetahui data asli dapat dilakukan dengan proses dekompresi. Dekompresi adalah proses untuk mendapatkan kembali data asli sebelum di kompresi. Kompresi data dicapai bila satu atau lebih redundansi ini dikurangi atau dihilangkan [1]. Beberapa data yang bisa dikompresi adalah data teks, data audio, data citra, dan data video.

Di dalam sistem komputasi modern, karakter atau simbol biasanya dituliskan ke dalam kode ASCII. Setiap simbol yang muncul pada layer komputer mempunyai kode ASCII yang berbeda-beda. Karena panjang setiap kode ASCII dalam biner adalah 8, maka ada 28 simbol unik dalam tabel ASCII untuk setiap karakter [2].

Representasi pembelajaran adalah seperangkat metode yang memungkinkan mesin diberi umpan dengan data mentah dan untuk secara otomatis menemukan representasi yang diperlukan untuk deteksi atau klasifikasi. Metode deep learning adalah metode representasi-pembelajaran dengan berbagai tingkat representasi, diperoleh dengan membuat modul sederhana namun tidak linier yang masing-masing mengubah representasi pada satu tingkat (dimulai dengan input mentah) menjadi representasi pada tingkat yang lebih tinggi dan sedikit lebih abstrak. Dengan komposisi transformasi yang cukup, fungsi yang sangat kompleks bisa dipelajari. Untuk tugas klasifikasi, lapisan representasi yang lebih tinggi memperkuat aspek input yang penting untuk diskriminasi dan menekan variasi yang tidak relevan [3].

Auto-Encoder adalah struktur pembelajaran tanpa pengawasan termasuk tiga lapisan: lapisan data masukan, lapisan kode dan lapisan data restrukturisasi. Model mengkodekan data masukan ke sekumpulan kode dan kemudian menerjemahkannya untuk mendapatkan data restrukturisasi. [4].

Berdasarkan acuan penelitian yang berkaitan tentang kompresi data teks, maka pada penelitian ini dilakukan beberapa hal sebagai berikut: tahap pertama melakukan kompresi data teks dengan menggunakan AENN. Tahap kedua melakukan dekompresi data teks dari hasil kompresi.

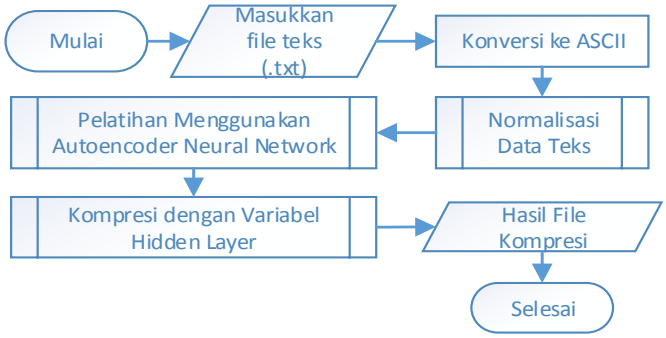
Manfaat dari penelitian ini adalah inovasi dari metode AENN yang diterapkan pada proses kompresi yang bertujuan untuk mengurangi pemakaian ruang penyimpanan pada memori dan data yang dilakukan pada proses kompresi menggunakan data teks.

II. METODOLOGI PENELITIAN

Metodologi penelitian yang akan digunakan pada penelitian ini terdiri dari dua tahapan, yaitu kompresi dan dekompresi.

A. Metode Penelitian

Tahapan dalam melakukan kompresi data teks terdiri dari tiga tahapan utama, yaitu normalisasi data teks, pelatihan, dan kompresi. Tahapan penelitian untuk proses kompresi ditunjukkan dalam bentuk flowchart dalam Gambar 1.



Gambar 1. Flowchart Kompresi

Pada Gambar 1 merupakan proses kompresi terdapat beberapa rangkaian prosedur. Berikut adalah rangkaian tahapan prosedur dalam proses kompresi

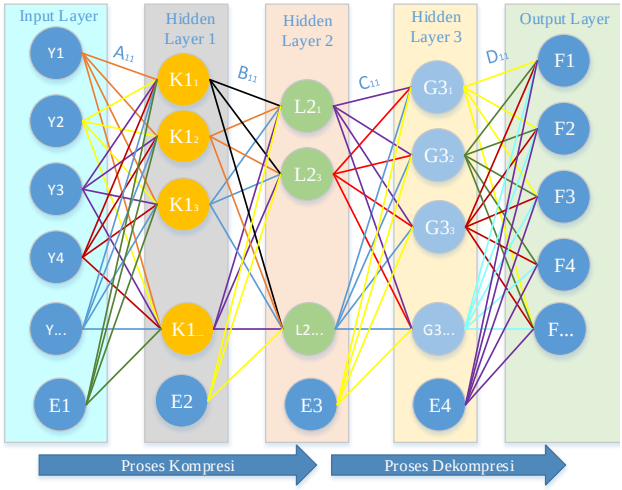
1) Proses Normalisasi Data Teks

Normalisasi bertujuan agar hanya memiliki nilai input kecil terdiri 0.000000 hingga 1.0000000. penelitian ini menggunakan kode ASCII yang paling kecil 32 hingga 122. Kode ASCII 32 adalah spasi hingga 122 adalah huruf Z.untuk melakukan normalisasi digunakan persamaan 1.

$$x_n = \frac{x - 32}{90} \tag{1}$$

Dimana xn adalah ASCII yang sudah dinormalisasi dan x adalah ASCII

2) Proses Pelatihan



Gambar 2. Graph Autoencoder Neural Network

Graph yang ditunjukkan dalam Gambar 2 terdiri input layer, hidden layer 1, hidden layer 2, hidden layer 3, dan output layer. Input layer disini adalah xn dari hasil normalisasi. Proses pelatihan ini jumlah hidden layer 1 dan hidden layer 2 kurang dari jumlah input layer. menentukan hidden layer maka digunakan persamaan yang ditunjukkan berikut ini

$$K1 = \frac{Y}{2} \tag{2}$$

Jika banyaknya input layer ganjil maka menggunakan persamaan 2

$$K1 = \frac{Y+1}{2} \tag{3}$$

Menentukan banyaknya neuron pada hidden layer 2

$$L2 = \frac{K1}{2} \tag{4}$$

Menentukan banyaknya neuron pada hidden layer 3

$$G3 = L1 \times 2 \tag{5}$$

Menentukan banyaknya neuron pada output layer

$$F = G3 \times 2 \tag{6}$$

dimana *Y* adalah banyaknya *neuron* pada *Input Layer*, *K*1 adalah banyaknya *neuron* pada *hidden layer* 1,*L*2 adalah banyaknya *neuron* pada *hidden layer 2*, G3 adalah banyaknya *neuron* pada *hidden layer* 3, dan F adalah banyaknya *neuron* pada *output layer* 3

Forward

Hidden layer 1

$$\mathrm{k1}_{(M,1)} = A_{(M,N)}\, \mathrm{g}Y_{(N,1)} + E1_{h1(M,1)}$$

$$\mathrm{K1}_{(M,1)} = \frac{2}{1+e^{-2\times k1}} - 1 \tag{7}.$$

dimana *A* adalah pembobot, *E1* adalah pembobot, *Y* adalah nilai pada tiap *neuron input layer*, *k1* adalah nilai pada tiap *neuron hidden layer* 1 sebelum diaktivasi, *K1* adalah nilai pada tiap *neuron hidden layer* 1 setelah diaktivasi, e adalah basis algoritma natural.

Hidden layer 2

$$\mathrm{l2}_{(M,1)} = B_{(M,N)}\, \mathrm{g}K1_{(N,1)} + E2_{h1(M,1)}$$

$$L2_{(M,1)} = \frac{2}{1+e^{-2\times l2}} - 1 \tag{8}.$$

dimana *B* adalah pembobot, *E2* adalah pembobot, *K1* adalah nilai pada tiap *neuron hidden layer* 1, *l2* adalah nilai pada tiap *neuron hidden layer* 2 sebelum diaktivasi, *L2* adalah

nilai pada tiap *neuron hidden layer* 2 setelah diaktivasi, *e* adalah basis algoritma natural.

Hidden layer 3

$$g3_{(M,1)} = C_{(M,N)}\, gL2_{(N,1)} + E3_{h1(M,1)}$$

$$G3_{(M,1)} = \frac{2}{1+e^{-2\times g3}} - 1 \quad (9).$$

dimana *C* adalah pembobot, *E3* adalah pembobot, *L2* adalah nilai pada tiap *neuron hidden layer* 2, *g3* adalah nilai pada tiap *neuron hidden layer* 3 sebelum diaktivasi, *G3* adalah nilai pada tiap *neuron hidden layer* 3 setelah diaktivasi, *e* adalah basis algoritma natural.

Output Layer

$$f_{(M,1)} = D_{(M,N)}\, gG3_{(N,1)} + E4_{h1(M,1)}$$

$$F_{(M,1)} = \frac{2}{1+e^{-2\times f}} - 1 \quad (10).$$

dimana *D* adalah pembobot, *E4* adalah pembobot, *G3* adalah nilai pada tiap *neuron hidden layer* 3, *f* adalah nilai pada tiap *neuron output layer* sebelum diaktivasi, *F* adalah nilai pada tiap *neuron output layer* setelah diaktivasi, *e* adalah basis algoritma natural.

Backward

Output layer

$$\Delta F_{(N,1)} = \boldsymbol{e}_{(N,1)} = F_{(N,1)} - Y_{(N,1)} \quad (11)$$

$$\Delta Y_{(N,1)} = \boldsymbol{e}_{(N,1)} .* F_{(N,1)} .* \left(1 - F_{(N,1)}\right) \quad (12)$$

$$\Delta \boldsymbol{D}_{(N,M)} = \Delta Y_{(N,1)}\, g\left(G3_{(M,1)}\right)^T \quad (13)$$

$$\Delta E4_{(N,1)} = \Delta Y_{(N,1)} \quad (14)$$

dimana ΔF ,e adalah selisih nilai tiap neuron output layer dan neuron input hidden layer, F adalah nilai tiap neuron output layer, G3 adalah nilai tiap neuron hidden layer 3, ΔD adalah akan digunakan untuk memperbarui pembobot D, $\Delta E4$ akan digunakan untuk memperbarui pembobot, E4, ΔY akan digunakan untuk memperbarui neuron input layer

Hidden layer 3

$$\Delta G_{(M,1)} = \left(\boldsymbol{D}_{(N,M)}\right)^T g\Delta Y_{(N,1)} \quad (15)$$

$$\Delta H3_{(M,1)} = \Delta G_{(M,1)} .* G3_{(M,1)} .* \left(1 - G3_{(M,1)}\right) \quad (16)$$

$$\Delta \boldsymbol{C}_{(M,L)} = \Delta H3_{(M,1)}\, g\left(\mathrm{L2}_{(L,1)}\right)^T \quad (17)$$

$$\Delta E3_{(M,1)} = \Delta H3_{(M,1)} \quad (18)$$

dimana ΔG adalah nilai yang akan digunakan untuk memperbarui pembobot, *F* adalah nilai tiap *neuron output layer*, *G3* adalah nilai tiap *neuron hidden layer* 3, $\Delta E3$ adalah nilai yang akan digunakan untuk memperbaharui pembobot *E3*, ΔY adalah nilai yang akan digunakan untuk memperbarui pembobot, $\Delta H3$ adalah akan digunakan untuk memperbarui $\Delta E3$, D adalah pembobot.

Hidden layer 2

$$\Delta \boldsymbol{L}_{(L,1)} = \left(\boldsymbol{C}_{(M,L)}\right)^T g\Delta G_{(M,1)} \quad (19)$$

$$\Delta H2_{(L,1)} = \Delta L_{(L,1)} .* \boldsymbol{L}2_{(L,1)} .* \left(1 - L2_{(L,1)}\right) \quad (20)$$

$$\Delta B_{(L,M)} = \Delta H2_{(L,1)}\, g\left(\mathrm{K1}_{(M,1)}\right)^T \quad (21)$$

$$\Delta E2_{(L,1)} = \Delta H2_{(L,1)} \quad (22)$$

dimana ΔG adalah nilai yang akan digunakan untuk memperbarui pembobot, *F* adalah nilai tiap *neuron output layer*, *G3* adalah nilai tiap *neuron hidden layer* 3, ΔB adalah nilai yang akan digunakan untuk memperbaharui pembobot *D*, $\Delta E2$ adalah nilai yang akan digunakan untuk memperbaharui pembobot *E*2, C adalah pembobot

Hidden layer 1

$$\Delta K_{(M,1)} = \left(B_{(L,M)}\right)^T g\Delta L_{(L,1)} \quad (23)$$

$$\Delta H1_{(M,1)} = \Delta K_{(M,1)} .* K1_{(M,1)} .* \left(1 - K1_{(M,1)}\right) \quad (24)$$

$$\Delta \boldsymbol{W}_{(M,N)} = \Delta H1_{(M,1)}\, g\left(\boldsymbol{x}_{(N,1)}\right)^T \quad (25)$$

$$\Delta E1_{(M,1)} = \Delta K_{(M,1)} \quad (26)$$

dimana ΔK adalah nilai yang akan digunakan untuk memperbarui pembobot,*B* = pembobot, *K*1= nilai tiap *neuron hidden layer* 1, ΔW adalah nilai yang akan digunakan untuk memperbarui pembobot, $\Delta E1$ nilai yang akan digunakan

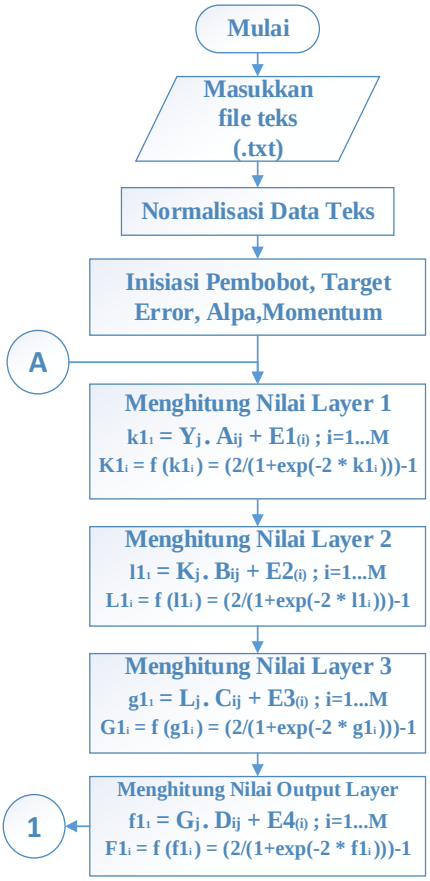
untuk memperbarui pembobot $E1$, ΔL adalah nilai yang akan digunakan untuk memperbarui pembobot, $\Delta H2$ adalah nilai yang akan digunakan untuk memperbarui pembobot.

Update Pembobot

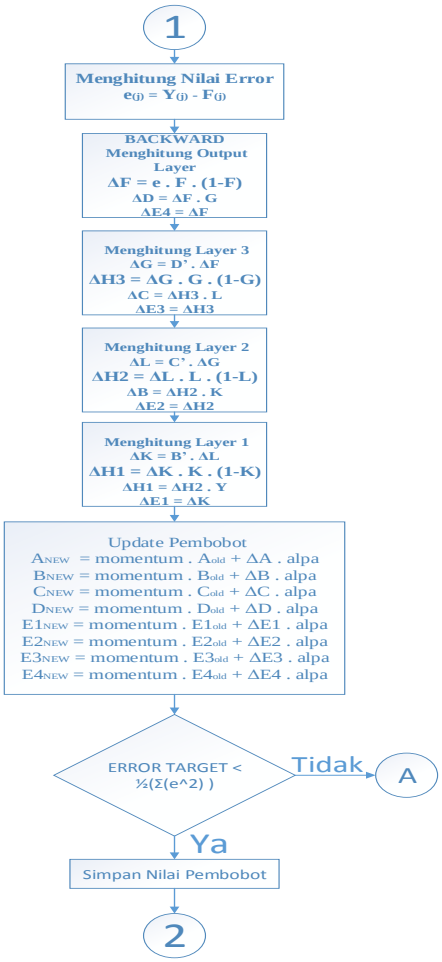
$$\begin{aligned}
A_{ij(new)} &= m.A_{ij(old)} + \alpha.\Delta A_{ij} \\
E1_{(i)(new)} &= E1_{(i)(old)} + \alpha.\Delta E1_{(i)} \\
B_{ki(new)} &= m.B_{ki(old)} + \alpha.\Delta B_{ki} \\
E2_{(k)(new)} &= E2_{(k)(old)} + \alpha.\Delta E2_{(k)} \\
C_{ik(new)} &= m.C_{ik(old)} + \alpha.\Delta C_{ik} \\
E3_{(i)(new)} &= E3_{(i)(old)} + \alpha.\Delta E3_{(i)} \\
D_{ji(new)} &= D_{ji(old)} + \alpha.\Delta D_{ji} \\
E4_{(j)(new)} &= E4_{(j)(old)} + \alpha.\Delta E4_{(j)}
\end{aligned} \quad (27)$$

dimana Y adalah input layer, K1 adalah hidden layer 1, L2 adalah hidden layer 2, G3 adalah hidden layer 3, F adalah output layer, α adalah alpa, m adalah momentum dan A,B,C,D,E1,E2,E3,E4 adalah pembobot, $\Delta A, \Delta B, \Delta C, \Delta D, \Delta E1, \Delta E2, \Delta E3, \Delta E4$ adalah nilai yang digunakan untuk memperbarui pembobot.

Proses pelatihan pada tahap ini dimulai setelah normalisasi data teks yaitu proses inisiasi pembobot, kemudian menghitung nilai pada hidden layer 1, hidden layer 2, hidden layer 3 dan output layer, kemudian menghitung nilai error dan perbaikan pembobot sehingga nilai error mendekati target error. Proses pelatihan ditunjukkan pada Gambar 3.

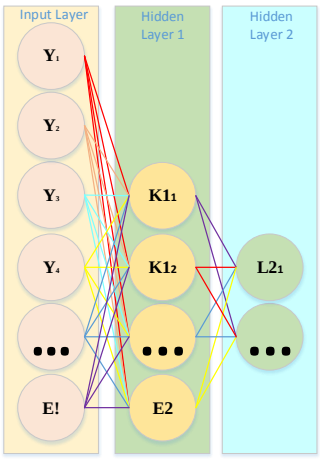


Gambar 3 Flowchart pelatihan 1



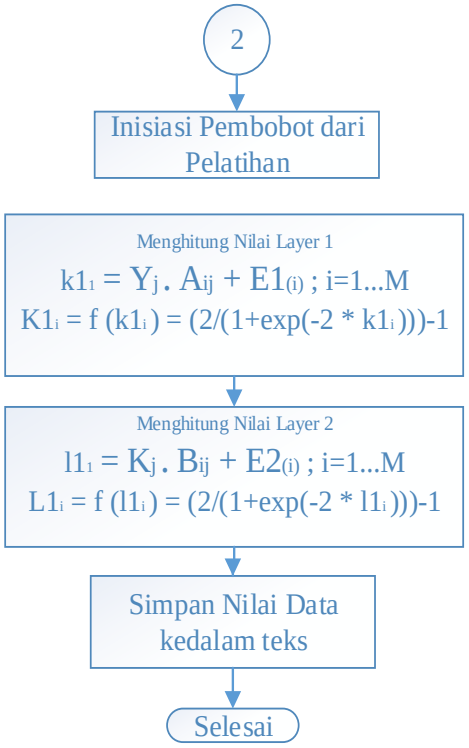
Gambar 4 Flowchart Pelatihan 2

3) Proses Kompresi



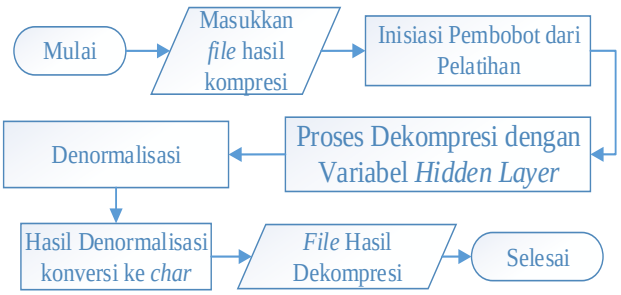
Gambar 5 Graph untuk kompresi

Gambar 5 merupakan graph yang akan digunakan pada proses kompresi. Graph pada Gambar 5 berasal dari graph yang ditunjukkan pada Gambar 2. Proses kompresi hanya menggunakan jumlah neuron pada masing-masing input layer, hidden layer 1, dan hidden layer 2. Hidden layer 2 adalah hasil kompresi. Hidden layer 1 menggunakan (7) dan hidden layer 2 menggunakan (8). Setelah melakukan proses pelatihan maka mendapatkan nilai pembobot yang akan digunakan pada tahap kompresi. Proses kompresi ditunjukkan dalam Gambar 6.



Gambar 6 Flowchart Kompresi

Hasil dari hidden layer 2 merupakan hasil kompresi dalam variabel kompresi. Tahapan selanjutnya adalah proses dekompresi yang menggunakan nilai inisiasi pembobot yang berasal dari proses pelatihan kemudian dilakukan dekompresi dan didenormalisasi. Proses dekompresi ditunjukkan dalam Gambar 2.

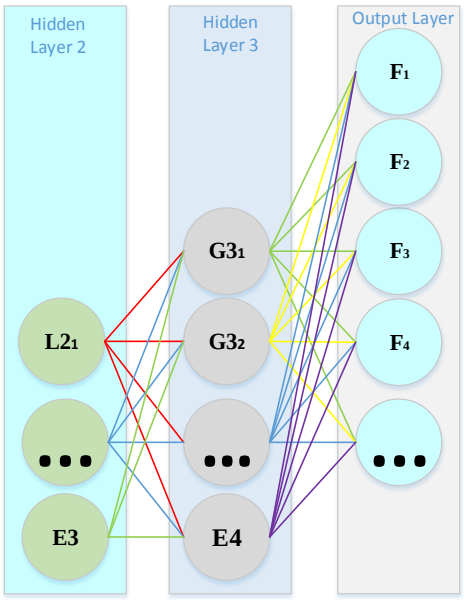


Gambar 7. Flowchart Dekompresi

Terdapat beberapa rangkaian prosedur ditunjukkan dalam Gambar 7. Berikut adalah rangkaian tahapan prosedur dalam proses dekompresi

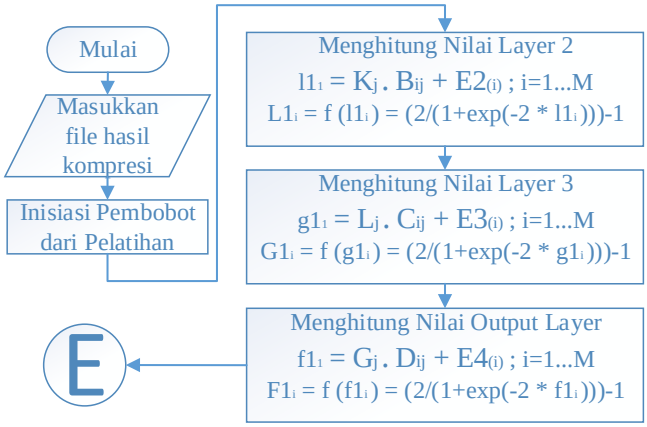
4) Proses dekompresi

Proses dekompresi merupakan proses mengembalikan data yang telah dikompresi menjadi data aslinya. Graph proses dekompresi ditunjukkan dalam Gambar 8.



Gambar 8 Graph untuk Dekompresi

Gambar 8 merupakan graph yang digunakan untuk proses dekompresi dimulai dari hidden layer 2, hidden layer 3 dan output layer. Hidden layer 2 berisi data hasil kompresi dan output layer berisi hasil dekompresi. Untuk menghitung hidden layer 3 menggunakan persamaan yang ditunjukkan dalam (9) dan untuk menghitung nilai output layer menggunakan persamaan yang digunakan dalam (10). Proses dekompresi ditunjukkan dalam Gambar 10.



Gambar 10 Flowchart Proses Dekompresi

Hasil yang diperoleh pada Gambar 10 digunakan untuk proses selanjutnya

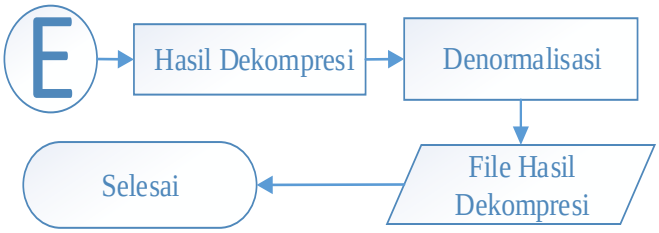
5) Proses Denormalisasi

Setelah mendapatkan hasil dekompresi maka tahap selanjutnya adalah denormalisasi. Tujuan denormalisasi merupakan inversi dari proses normalisasi yaitu mengembalikan nilai ASCII yang telah dinormalisasi kembali menjadi nilai aslinya. Dalam penelitian ini, denormalisasi dilakukan dengan menggunakan persamaan

$$x = x_n \times 122 \tag{28}$$

dimana x adalah hasil denormalisasi dan xn adalah hasil dari output layer.

Setelah didenormalisasi maka data yang telah dinormalisasi akan kembali seperti data aslinya. Proses denormalisasi ditunjukkan dalam Gambar 11.



Gambar 11 Proses Denormalisasi

B. Persamaan untuk pengujian

Rasio Kompresi

$$Rasio = \frac{(uAsli - uKompresi)}{uAsli} \times 100\% \tag{29}$$

dimana uAsli adalah ukuran file asli, dan uKompresi adalah ukuran file setelah kompresi.

III. HASIL DAN PEMBAHASAN

A. Kompresi

Data teks yang digunakan "M Nur Alfiansyah". Memiliki ukuran 16 bytes. Data teks kemudian diubah kedalam kode ASCII seperti ditunjukkan dalam Tabel 1.

TABEL 1. DATA TEKS DENGAN ASCII

Huruf	ASCII	Huruf	ASCII
M	77	f	102
	32	i	105
N	78	a	97
u	117	n	110
r	114	s	115
	32	y	121
A	65	a	97
l	108	h	104

Data teks yang sudah berupa kode ASCII kemudian dinormalisasi dengan menggunakan (1). Hasil yang diperoleh ditunjukkan dalam Tabel 2.

TABEL 2 DATA YANG SUDAH DINORMALISASI

Huruf	ASCII	Normalisasi

M	77	0.5000
	32	0.0000
N	78	0.5111
u	117	0.9444
r	114	0.9111
	32	0.0000
A	65	0.3667
l	108	0.8444
f	102	0.7778
i	105	0.8111
a	97	0.7222
n	110	0.8667
s	115	0.9222
y	121	0.9889
a	97	0.7222
h	104	0.8000

Data hasil normalisasi kemudian digunakan sebagai data pelatihan AENN. Penentuan banyaknya neuron pada tiap *hidden layer* yang akan digunakan adalah sebagai berikut :

Jumlah *neuron* pada *hidden layer* 1 menggunakan (2), sehingga diperoleh:

$$\frac{16}{2}=8$$

sehingga jumlah *neuron* pada *hidden layer* 1 adalah 8. Jumlah *neuron* pada *hidden layer* 2 menggunakan (4), diperoleh:

$$\frac{8}{2}=4$$

sehingga jumlah *neuron* pada *hidden layer* 2 banyaknya adalah 4. Jumlah *neuron* pada *hidden layer* 3 menggunakan (5), diperoleh:

$$4\times 2=8$$

sehingga jumlah *neuron* pada *hidden layer* 3 banyaknya adalah 8.Jumlah *neuron* pada *output layer* menggunakan (6), diperoleh;

$$8\times 2=16$$

sehingga *neuron* pada *hidden layer* 3 banyaknya adalah 1.

Setelah diketahui masing-masing *neuron hidden layer* maka proses pelatihan dapat dilakukan. Pembobot diisi secara acak mulai dari 0.00001 hinga 1.00000. kemudian target *error* diisi dengan 0.0000010, momentum =1, dan alpa 1.4. Berikut hasil dari pelatihan.

TABEL 3 HASIL ITERASI PERTAMA

Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Output layer
0.5000	2.0000	0.0002	0.0000	0.0000
0.0000	2.0000	0.0003	0.0000	0.0000
0.5111	2.0000	0.0002	0.0000	0.0000
0.9444	2.0000	0.0003	0.0000	0.0000
0.9111	2.0000	-	0.0000	0.0000
0.0000	2.0000	-	0.0000	0.0000
0.3667	2.0000	-	0.0000	0.0000
0.8444	2.0000	-	0.0000	0.0000
0.7778	-	-	-	0.0000
0.8111	-	-	-	0.0000
0.7222	-	-	-	0.0000
0.8667	-	-	-	0.0000
0.9222	-	-	-	0.0000
0.9889	-	-	-	0.0000
0.7222	-	-	-	0.0000
0.8000	-	-	-	0.0000
Error				4.960258

TABEL 4 HASIL ITERASI 2

Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Output layer
0.5000	2.0000	0.0002	0.0000	0.0000
0.0000	2.0000	0.0003	0.0000	0.0000
0.5111	2.0000	0.0002	0.0000	0.0001
0.9444	2.0000	0.0003	0.0000	0.0000
0.9111	2.0000	-	0.0000	0.0001
0.0000	2.0000	-	0.0000	0.0001
0.3667	2.0000	-	0.0000	0.0001
0.8444	2.0000	-	0.0000	0.0000
0.7778	-	-	-	0.0001
0.8111	-	-	-	0.0000
0.7222	-	-	-	0.0000
0.8667	-	-	-	0.0001
0.9222	-	-	-	0.0000
0.9889	-	-	-	0.0000
0.7222	-	-	-	0.0000
0.8000	-	-	-	0.0001
Error				4.959991

Kemudian iterasi terakhir dimana nilai error mendekati nilai target error.

TABEL 5 HASIL ITERASI TERAKHIR

Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Output layer
0.5000	2.0000	0.00024311	0.0000	0.6311
0.0000	2.0000	0.00025382	0.0000	0.2623
0.5111	2.0000	0.00020379	0.0000	0.6393
0.9444	2.0000	0.00027995	0.0000	0.9587
0.9111	2.0000	-	0.0000	0.9344
0.0000	2.0000	-	0.0000	0.2623
0.3667	2.0000	-	0.0000	0.5328
0.8444	2.0000	-	0.0000	0.8852
0.7778	-	-	-	0.8361
0.8111	-	-	-	0.8607
0.7222	-	-	-	0.7951
0.8667	-	-	-	0.9016
0.9222	-	-	-	0.9426
0.9889	-	-	-	0.9874
0.7222	-	-	-	0.7951
0.8000	-	-	-	0.8525
Error				0.000010

Kemudian setelah nilai error mendekati target error maka nilai pembobot disimpan untuk digunakan untuk proses kompresi.

TABEL 6 HASIL PROSES KOMPRESI

Input layer	Hidden layer 1	Hidden layer 2
0.5000	2.0000	0.00024311
0.0000	2.0000	0.00025382
0.5111	2.0000	0.00020379
0.9444	2.0000	0.00027995

0.9111	2.0000	-
0.0000	2.0000	-
0.3667	2.0000	-
0.8444	2.0000	-
0.7778	-	-
0.8111	-	-
0.7222	-	-
0.8667	-	-
0.9222	-	-
0.9889	-	-
0.7222	-	-
0.8000	-	-
0.5000	-	-

Proses kompresi menghasilkan 4 hasil yaitu 0.00024311, 0.00025382, 0.00020379, 0.00027995.

Menghitung hasil rasio kompresi menggunakan (29)

$$Rasio = \frac{16-4}{16} \times 100\% = 75\%$$

panjang asli adalah 16 dan panjang setelah didekompresi adalah 4 maka hasil rasio kompresi sebesar 75%.

B. Dekompresi

Data yang akan digunakan pada proses dekompresi merupakan data hasil kompresi sebelumnya. Datanya berupa 0.00024311, 0.00025382, 0.00020379, 0.00027995.

TABEL 7 HASIL DEKOMPRESI

Hasil Kompresi	*Hidden layer* 3	*Output layer*
0.00024311	0.0000	0.6311
0.00025382	0.0000	0.2623
0.00020379	0.0000	0.6393
0.00027995	0.0000	0.9587
-	0.0000	0.9344
-	0.0000	0.2623
-	0.0000	0.5328
-	0.0000	0.8852
-	-	0.8361
-	-	0.8607
-	-	0.7951
-	-	0.9016
-	-	0.9426
-	-	0.9874
-	-	0.7951
-	-	0.8525
-	-	0.6311

Setelah diperoleh hasil output layer maka tahap selanjutnya adalah denormalisasi dengan menggunakan (28) kemudian langsung dikonversi kedalam bentuk character. Hasil denormalisasi dan konversi ditunjukkan dalam Tabel 8.

TABEL 8 HASIL DENORMALISASI DAN KONVERSI DARI DENORMALISASI MENJADI CHARACTER

Denormalisasi	Hasil Dekompresi
77	M
32	
78	N
117	u
114	r
32	
65	A
108	l
102	f
105	i
97	a
110	n
115	s
121	y
97	a
104	h

Maka hasil dekompresi kembali seperti data asli.

TABEL 9 HASIL PENGUJIAN DENGAN PEMBAGI HIDDEN 2

Ukuran Asli	Ukuran Kompresi	Ukuran Dekompresi	Rasio Kompresi	Keberhasilan Kembali
934	234	934	75	100.0
940	235	940	75	000.0
930	233	1056	75	100.0
787	233	788	75	099.9
714	197	714	75	100.0
612	153	612	75	100.0
586	153	586	75	100.0
268	67	268	75	0
134	34	134	75	100.0
67	17	68	75	098.5
33	9	34	73	097.1
16	4	16	75	100.0
5	2	6	60	083.3
Pembagi *hidden*				2
Rata-rata ukuran asli				464
Rata-rata ukuran kompresi				116
Rata-rata ukuran dekompresi				464
Rata-rata rasio kompresi				73.59%
Rata-rata keberhasilan kembali				82.98%

Dari Tabel 9 terdapat kegagalan data kembali ketika ukuran data sebesar 940 dan beberapa data tidak kembali 100%

TABEL 10 HASIL PENGUJIAN DENGAN PEMBAGI HIDDEN 4

Ukuran Asli	Ukuran Kompresi	Ukuran Dekompresi	Rasio Kompresi	Keberhasilan Kembali
1876	117	1876	94	0
1875	117	1876	94	099.0
1056	66	1056	94	100.0
528	33	528	94	100.0
264	17	264	94	100.0
132	8	132	94	100.0
66	4	66	94	100.0
33	2	34	94	99.7
Pembagi *hidden*				4
Rata-rata ukuran asli				729
Rata-rata ukuran kompresi				46
Rata-rata ukuran dekompresi				729
Rata-rata rasio kompresi				94.00%
Rata-rata keberhasilan kembali				87.12%

Dari Tabel 10 terdapat kegagalan data kembali ketika ukuran data sebesar 1876 dan satu pengujian data tidak kembali 100%.

TABEL 11 HASIL PENGUJIAN DENGAN PEMBAGI HIDDEN 8

Ukuran Asli	Ukuran Kompresi	Ukuran Dekompresi	Rasio Kompresi	Keberhasilan Kembali
2144	34	2144	98	0.0%
2139	34	2140	98	99.9%
1069	17	1070	98	99.9%
534	8	534	98	100.0%
267	4	267	98	99.6%
133	2	133	98	99.3%
Pembagi *hidden*				8
Rata-rata ukuran asli				1048
Rata-rata ukuran kompresi				17
Rata-rata ukuran dekompresi				1048
Rata-rata rasio kompresi				98.00%
Rata-rata keberhasilan kembali				83.12%

Dari Tabel 11 terdapat kegagalan data kembali ketika ukuran data sebesar 2144 dan beberapa data tidak kembali 100%.

TABEL 12 HASIL RATA-RATA DARI SEMUA PENGUJIAN

Pembagi *Hidden*	Rata-Rata Ukuran asli	Rata-Rata Ukuran Kompresi	Rata-Rata Ukuran Dekompresi	Rata-Rata Rasio Kompresi	Rata-Rata Keberhasilan Kembali
8	1048	17	1048	98	83.1
4	729	46	729	94	87.1
2	464	116	464	74	83.0
Rata-rata pembagi *hidden*					5
Rata-rata ukuran asli					747
Rata-rata ukuran kompresi					59
Rata-rata ukuran dekompresi					547
Rata-rata rasio kompresi					88.62%
Rata-rata keberhasilan kembali					84.41%

Analisa

Dari hasil pengujian dianalisa bahwa jika ukuran file aslinya ganjil maka ukuran setelah didekompresi menjadi genap yang menyebabkan keberhasilannya tidak 100 %. Selain itu jika file pembaginya memiliki keterbatasan jika terlalu besar maka keberhasilan kembalinya 0 %.

IV. KESIMPULAN

Dari hasil penelitian dan pembahasan dapat disimpulkan bahwa proses kompresi dengan menggunakan Auto-Encoder Neural Network menghasilkan rasio kompresi sebesar 88.62%. Namun masih terdapat kekurangan jika file asli panjangnya ganjil maka file hasil didekompresi panjangnya genap dan juga memiliki keterbatasan masing-masing pembagi *hidden* yang menyebabkan hasil pengujian tidak kembali sama sekali.

REFERENCES

[1]. Dipalee Gupta, S.C., Discrete Wavelet Transform for Image Processing. International Journal of Emerging Technology and Advanced Engineering (IJETAE), 2015. 4(3): p. 598-602.

[2]. M A Budiman, D.R., On Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression on Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression. IOP Conf. Series: Materials Science and Engineering, 2017. 180: p. 1-5.

[3]. LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. Nature, 2015. 521(7553): p. 436-44.

[4]. Jingfei Jiang, R.H., Dongsheng Wang, Jinwei Xu, Yong Dou Performance of the fixed-point autoencoder. Tehnicki vjesnik - Technical Gazette, 2016. 23(1): p. 77-82.