SPEECH RECOGNITION USING ELMAN RECURRENT NEURAL NETWORK FOR SPOKEN WORDS IN BAHASA INDONESIA

Wildan Kurniadi[1], Irfan Maliki[2]

[1,2] Informatics Engineering - Universitas Komputer Indonesia
Jl. Dipatiukur No 112-116 Bandung, 40132
E-mail : kurniadiwildan@email.unikom.ac.id [1], irfan.maliki@email.unikom.ac.id [2]

ABSTRACT

Speech Recognition is one popular field of computer science. In a speech recognition algorithm is needed to determine the meaning of the voice signal. One of the algorithms that can be used in speech recognition is Elman Recurrent Neural Network (ERNN). ERNN has been tested on several speech recognition research like in English, Arabic, and Bahasa Indonesia in different case, and the results are quite good, the accuracy resulted by ERNN are between 87.5% to 99.47%. In this research, ERNN is used to recognize ten highest-ranked verb words according to Leipzig Corpora Collection. The dataset consist of 40 male speakers each speakers will say 10 words and repeat it for 4 times to produce 1600 datasets (40 × 10 × 4 = 1600). The dataset is stored as a wav file with 16000 Hz sampling frequency. The training data used were 800 data samples (20 speakers × 10 words × 4 reps = 800 dataset) and the test data were 800 samples of data from 20 different speakers from the training data. The method of feature extraction used in this study was MFCC, the best accuracy produced by this study was 83.88%..

Kata kunci : Speech Recognition, Elman Recurrent Neural Network, MFCC, Bahasa Indonesia

1. INTRODUCTION

Speech Recognition is the process of interpreting human speech to be recognized by the computer [1]. In an Automated Speech Recognition (ASR) system a classification algorithm is needed to recognize the meaning of the voice signal. There are many kind of classification algorithms that can be used for ASR, one of them is the Elman Recurrent Neural network (ERNN).

Elman Recurrent Neural network (ERNN) can work well in speech recognition due to its ability to recognize different patterns based on time [2]. ERNN has been used as classification algorithm in some research about Automatic Speech Recognition (ASR). Some of the research that used ERNN are the research done by Robert B. Allen that used ERNN to recognize speech in English words that resulting accuracy of 87.5% [3]. In another research conducted by Yousef Alotaibi [4], ERNN yielded 99.47% accuracy and 96.46% to recognize spoken digits in Arabic. In the same language but different case of study, the research conducted by M. M. El Choubassi stated that ERNN is capable of classifying spoken words in Arabic, in that research accuracy resulted by ERNN reach 90%, Automatic Speech Recognition (ASR) in that research can recognize 6 words in Arabic [5]. In a research conducted by Muhammad Fachrie [6], ERNN was able to produce the highest accuracy of 99.30% in the case of digit 0 to 9 speech recognition. From some of these studies, ERNN appears able to give good results when used for speech recognition in several different cases, but the performance of ERNN for speech recognition with Indonesian words as the case is not yet known.

Based on description above, this research will use ERNN as classification algorithm on Automatic Speech Recognition to recognize 10 verbs words in Bahasa Indonesia. The purpose of this research is to find out how good accuracy resulted by ERNN to recognize spoken verb words in Bahasa Indonesia.

2. METHOD

The Automatic Speech Recognition system to be built consists of several main processes, consisting of Pre-Processing, Feature Extraction, Normalization and Thresholding, and ERNN section (Training or testing depending on which mode used). The block diagram of Automatic Speech Recognition in this research can be seen in Figure 1.

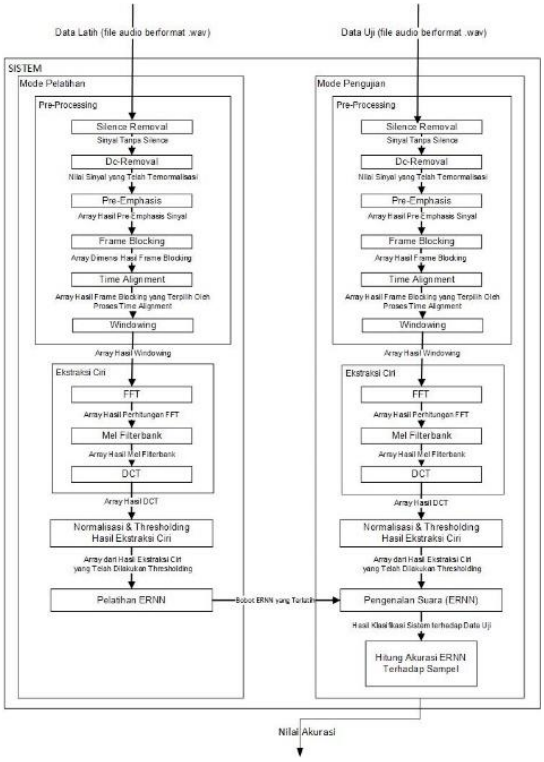


Figure 1. Proposed System Block Diagram

Input data (training data and test data) is an audio file from recording a speaker mentioning a word in duration of 1 second with format .wav

In the training mode, the choosen audio file will preprocessed. Preprocessing stage have some phase such as silence removal, DC removal, pre emphasis, frame blocking and windowing. After the choosen audio file is preprocessed, we will get the important characteristic of those audio signal on feature extraction stage. To get the important features of those audio file we use DFT, Mel Frequency Warping, and DCT. The DCT would resulted important number of features from the audio that we just choose to be extracted. The number of coefficient that resulted by DCT is called by Mel Frequency Cepstrum Coefficient (MFCC). The output (which in the form of mel frequency cepstrum coefficient) will be normalized to get the value with the spread of 0 to 1. After normalization phase, we will done thresholding to the value of normalization results, so now the value that resulted by extraction feature process be a value of 0 or 1 so the training of neural network could be much easier because a neural network would be much easier to be trained by discrete value of input rather than continue value of input [1].

In testing mode preprocessing, feature extraction, normalization and thersholding would be done to the choosen testing data, then ERNN would try to recognize the pattern of the testing data based on the training phase that has been done before, here ERNN would have a role of classifying algorithm. This testing mode would show the result of accuracy resulted by ERNN combined with MFCC feature extraction to recognize spoken verbs in Bahasa Indonesia..

2.1 Pre-processing

Pre-processing aims to produce a good signal value for feature extraction from the input data. The process of pre-processing consists of six stages consisting Silence Removal, DC Removal, Pre Emphasis, Blocking Frame, Time Alignment, Windowing.

a. Silence Removal

Silence removal is a process to remove silence signal (speechless signal). The silence removal process carried out in this research is adapted from other research[2]. Silence removal is quite important because it will affect the amount of data that will be processed at a later stage. Silence removal is done manually, which divides the signal into several parts (frames) that have a certain length, then if the maximum value of the sample on the frame is below the threshold that has been determined that frame of signal will be removed.

In figure 2, we can see the audio signal of a speaker mentioning the word "adalah" (is), the signal is seen using audacity.

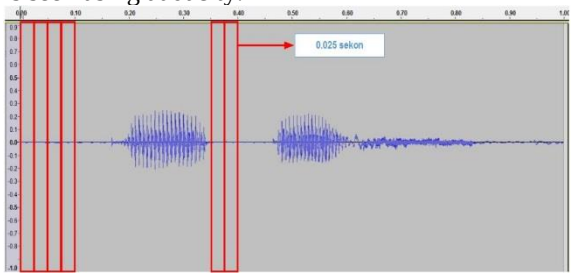


Figure 2. Dividing Signal Into Several Parts

The signal on figure 2 wil be divided into some frame block (for example, 0.025 sec each frame). Then each frame will be checked if those blocked frame content a signal that has value over determined threshold, then those frame would be stored in a new array as a non-silence signal, differently if the frame doesn content any signal over the determined threshold, then it would not stored in the new non-silence signal array (as it is a silence frame signal), and will not be processed for the next phase. After choosing which the silence frame an whic is not, then the new array contain non-silence signal would be reunited in a one new array. The result of silence removal of audio signal inf Figure 2 could be seen in Figure 3 (the picture is shown with audacity).

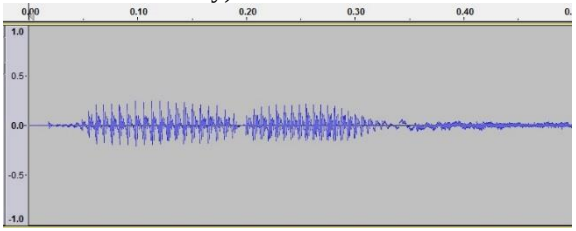


Figure 3. Silence Removal Result

b. DC *Removal*

After doing the silent removal then the next stage that will be done is DC removal. The DC removal process is performed by calculating the mean of the averages [3]. To calculate the result of DC Removal we used equation (1).

$dc_i = s_i - \bar{x}$, $0 \leq i \leq N-1$ (1)

where

dc_i = signal result of *DC removal*
s_i = i-signal point
$\bar{x}$ = input signal average value.
i = 0,1,2,…, N-1
N = panjang sinyal (banyaknya titik sinyal)

Where to get the average value of input signal point we use the equation as follows.

$\bar{x} = \frac{s_0+s_1+s_2+\cdots+s_{N-1}}{N-1}$ (2)

which :

$\bar{x}$ = average value of input signal
s_i = i-signal point
i = 0,1,2,…, N-1

c. *Pre Emphasis*

After doing DC Removal, the next step that will be done is pre-emphasis stage. Pre-emphasis is a filter used to improve the signal from noise interference [3].

$p_i = dc_i + pf_i$ (3)

which:

p_i = new i-signal point
dc_i = sample value before i-*pre-emphasis* (DC Removal results)
pf_i = *pre emphasis filter* on i-signal sample
i = 0,1,2,…, N-1
N = signal length (the number of signal point)

Where to get the value of pf_i we used equation below.

$pf_i = dc_i - (dc_{i-1} \times \alpha)$ (4)

which:

pf_i = i-pre-emphasize filter signal
dc_i = i-signal dc removal
dc_{i-1} = i-signal dc removal before
α = pre-emphasize filter alpha
i = 0,1,2,…, N-1
N = signal length (the number of signal point)

d. *Frame Blocking*

The process performed at the frame blocking stage is dividing the signal points into multiple frames, where each frame has the same sample point size, in addition each frame overlaps with other frames. Overlapping distance between frames that can be used ranges from 1/2 to 1/3 frame length (sample point). In this study the overlapping to be used is 1/2 of the frame length (sample point) and will be symbolized by the letter M. As for to get the SP value is done by multiplying the sample rate with the duration of the frame in seconds (s). In this study the duration of the frame to be used is 0.025 seconds while the sample rate is 16000. As for calculating the number of frame blocking is as follows.

I = ((NPE-SP)/M)+1) (5)

which:

I = frame number
NPE = sample number(the number of sample going to segmented)
SP = *Sample point* in each *frame* (*Sample rate* × frame length in seconds)
M = SP/2 (*Overlapping*)

e. *Time Alignment*

After the speech signal has been inserted into the frame form, the next step that must be done is to do time alignment (selection of frames to be used). The purpose of time alignment is to determine the number of frames to be obtained by its frequency coefficient according to the number of CFs. Time alignment will be done by specifying the CF (desired frame), SF (beginning of desired frame), and EF (desired frame end). The SF value is obtained by multiplying the value of SFD (the desired SF value in decimal) by the total of the existing frames, while EF is derived from the EFD value (total desired EF value in decimal) with the total existing frame, where the value of SFD is the decimal number worth 0 to 1 with terms of SFD plus EFD value will be worth 1 [2].

Here is the calculation if let CF is 5, the number of frames owned is 35 frames, and SFD is 0.06, (so that the value of EFD = 0.94)

CF = 5

SF = F × SFD (6)

= 35 × 0.06 = 1.4 = 1 (rounded to the nearest integer)

EF = F × EFD (7)

= 35 × 0.94 = 33.6 = 34 (rounded to the nearest integer)

f. *Windowing*

The selected frame at the time alignment stage will be the 0th frame up to the CF frame on the new frame. After the time alignment step is done it will be windowing. Windowing aims to overcome spectral leaks. The windowing calculation is done to the sample point on each frame by multiplying the hamming window function with the sample point value. The calculation can be seen in the following equation.

$w_i = p_i \,.\, fw_i$ (8)

which :

w_i = the result of *windowing*
p_i = i-result of the frame
fw_i = i-*window* function of the frame
NPE = the number of *sample point* each *frame*
i = i-*sample point* each *frame* (where n numbers are 0,1,…,NPE-1)

to get the value of fw_i we use the following equation.

$fw_i = 0.54 - 0.46\ cos\frac{2\pi i}{NSP-1}$ (9)

where :

fw_i = *hamming window* function for i-signal point on i-*frame*

SP = *sample point* (can be called as *frame)*

l = 0,1,...,SP-1

2.2 Feature Extraction

Signals that have passed the pre-processing stage will be processed to obtain the value of extraction of the character. The purpose of feature extraction is to get important features that will later be processed for recognition by ERNN. the process performed at the feature extraction stage comprises DFT, Mel Frequency Warping, DCT.

1. DFT

After the windowing of the selected frame, the next process that will be done is Discrete Fourier Transform (DFT). Discrete Fourier Transform will transform discrete signals at time domain into discrete signals on frequency domain. The equation on DFT can be seen as follows[3]

$$ft_k = \sum_{i=0}^{SP-1}\left(w_i\, cos\frac{2\pi ik}{SP}\right) - j\sum_{i=0}^{SP-1}\left(w_i sin\frac{2\pi ik}{SP}\right) \quad (10)$$

Where to get the result from DFT is used the following equation.

$$|ft_k| = [R^2 + I^2]^{1/2} \quad (11)$$

The result of dft will be called a magnitude dft and will be mapped onto the scale of mel.

2. Mel Frequency Warping

The magnitude that the DFT process has generated will go through the filterbank stage. At this stage a filter will be filtered to the incoming sound, where the filterbank used will follow the scale of the mel (human hearing scale). To obtain the value of Mel Frequency Warping result, the following equation is used. [4]

$$MF_m = \sum_{k=0}^{SP-1} ft_k \times H_{mk} \quad (12)$$

Dimana :

MF_m = Result Mel Frequency Warping ke-*m*

ft_k = Result DFT ke-*k*

H_{mk} = koefisien dari mel *filterbank*

SP = Banyaknya data hasil DFT

m = 0,1,2,..,*Nfbank*

Nfbank = Jumlah *Mel Filterbank*

Where the coefficient mel filterbank obtained by making a triangular filterbank, using the mel scal equatione[3].

$$mel\,(f) = 2595 \times log10(1 + \frac{f}{700}) \quad (13)$$

and mel scale inverse

$$mel^{-1}(f) = 700 \times (10^{\frac{f}{2595}} - 1) \quad (14)$$

Then specify the boundary filterbank to be used, ie as many as the number of triangles on the filterbank to be created. To calculate the value of boundary filterbank can be used the following equation[4].

$$fb_m = \left(\frac{SP}{FS}\right) \times mel^{-1}\left(mel(Nlow) + m\frac{mel(Nhigh)-mel(Nlow)}{Nfbank}\right) \quad (15)$$

where :

fb_m = *boundary point filter* results

SP = the number of sample point

FS = sampling frequency

After boundary filterbank value obtained, then by using the rule on the equation below it will generate triangular filterbank as much as the number of filterbank boundary.

$$H_{mk} = \begin{cases} 0,\ untuk\ k < fb_m \\ \frac{k - fb_{m-1}}{fb_m - fb_{m-1}},\ untuk\ fb_{m-1} \le k < fb_m \\ \frac{fb_{m+1} - k}{fb_{m+1} - fb_{m-1}},\ untuk\ fb_m \le k < fb_{m+1} \\ k > fb_{m+1} \end{cases} \quad (16)$$

Dimana :

fb_m = *boundary point filter* results

H_{mk} = mel *filterbank* coefficient

So if the steps on making triangular filterbank done then, will be generated a triangular filterbank, in this study triangular filterbank is diguanakan is 40, it refers to previous research [2], [5], thus triangular filterbank in this study can be seen as in the picture berikut.

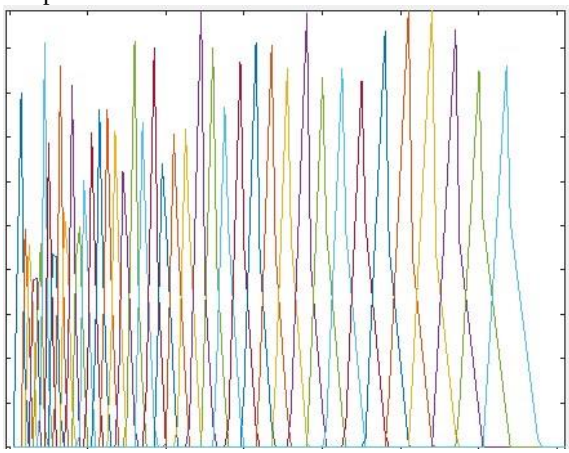


Figure 4 *Triangular* Mel *Filterbank* (seen using matlab)

3. DCT

DCT is the last process performed on the feature extraction section. DCT will compress the Mel Frequency Warping signal, DCT is well known for compressing the signal values.

$$C_{koef} = \sqrt{\frac{2}{Nfbank}}\sum_{m=0}^{Nfbank-1}(\log MF_m)\cos\left[koef\left(\frac{2m-1}{2}\right)\frac{\pi}{Nkoef}\right] \quad (17)$$

2.3 Normalization and Thresholding

The extraction of traits that have been obtained needs to be normalized and thresholded to be processed by ERNN. this is because neural networks will be easier to train with discrete rather than continuous values.

1. Normalization

Normalization is scaling of values into a certain range. This is done so that the input and output target values correspond to the range of activation functions used in the network. Normalization is done using the following equation [6].

$$normal_{koef} = \frac{0.8(C_{koef} - a)}{b - a} + 0.1 \quad (18)$$

2. Thresholding

Thresholding is done to change the value of normalization results to be 0 or 1. This is done because the neural network is more easily trained with discrete data than continuous [1]. The way it works is to give a threshold value if the value of the normalized result data is above the threshold value then the value will be 1, otherwise, if the normalized result value is below the threshold then the value will be 0.

2.4 *Elman Recurrent Neural network* (ERNN)

In the Elman Recurrent Neural network (ERNN) section there are two types of processes that can be done, namely, training and testing.

1. Training

The training used in this study refers to Boden [7]. Simply a recurrent neural network can be described as in Figure 5. Where every hidden neuron / state neuron will be used as input to the next timestep (in this case, the previous hidden neuron is called the context neuron and the layer is called the context layer). Function used in elman recurrent neural network is actually almost the same as the other neural network is obtained from the number of multiplication between the value of neuron input with weight, only on the recurrent neural network elman there is a difference that is by adding the value of the number of multiplication between the value of hidden neurons in the previous timestep (context neuron) with its context weight.

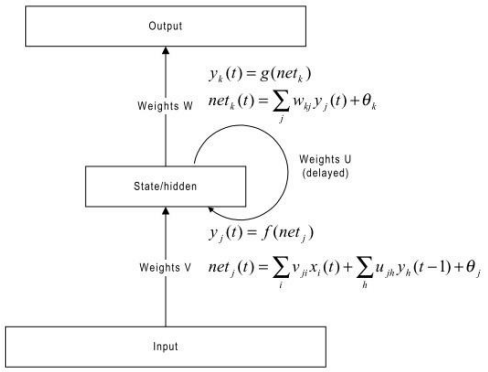


Figure 5 A Simple *ERNN*

If imagined, actually in Figure 5 there are several input layer and hidden layer. The number of input layers and hidden layers owned by ERNN depends on the number of timesteps owned by the ERNN architecture. Sehinggga if the timestep owned by ERNN is 3 then on the neural network architecture built there will be 3 input layer and 3 hidden layer, this can be seen in Figure 6.

Training on ERNN is done by making backpropagtion errors on each weight ranging from W weights, and also the weight of V and U on each timestep (from the last timestep to first). Since backpropagation of the recurrent neural network passes through certain timesteps, this process is called backpropagation through time (BPTT). The basic principle of BPTT is that "unfolding" [7] (which occurs in figure 5 is the unfolding result).

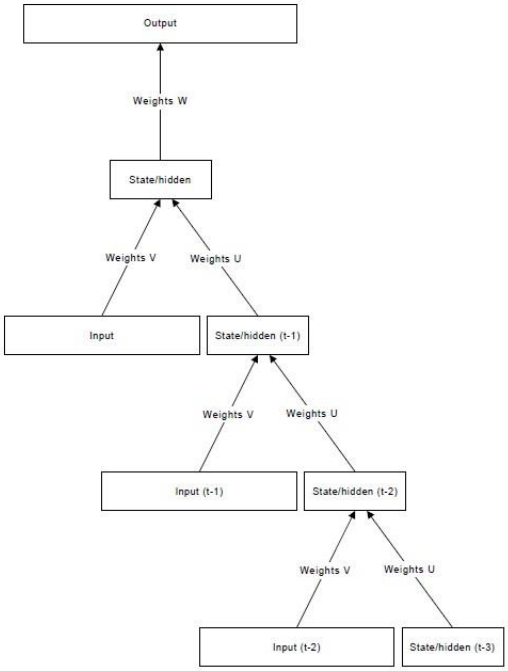


Figure 6 ERNN After *Unfolding*

Given the principle of unfolding, each weight of the recurrent (weight V and U) needs to be stored in memory in order to be trained (error training results will be stored in memory).Calculations that occur within the ERNN are described as follows. To calculate the value in the hidden neuron in the timestep we use equation 19.

$$net_j(t) = \sum_i v_{ji} x_i(t) + \sum_h u_{jh} y_h(t-1) + \theta_j \quad (19)$$

where :

$net_j(t)$ = *hidden layer vector* in t-*timestep*

v_{ji} = weight of i-*input* neuron to j-*hidden neuron*

x_i = value of i-*input neuron*

u_{jh} = weight of h-*context* neuron to j-*hidden neuron*

y_h = value of h-*context neuron* (the value of h-*hidden neuron* in the previous *timestep*)

t = *timestep*

Furthermore the value on the hidden neuron needs activated using activation function to be used. In this study the activation function used is tanh (hyperbolic tangent). The function of tanh can be seen in equation 20.

$$net_j(t) = \frac{e^{2 \times net_{j-1}}}{e^{2 \times net_{j+1}}} \quad (20)$$

The derivatives are as follows.

$$\text{dnet}_j = (1 - net_j) \times (1 + net_j) \quad (21)$$

After the activation of the value of each hidden neuron in the timestep, if the timestep is not the last

timestep then the activation of the hidden neuron will be the input value in the next timestep (into a context neuron). If timestep is the last timestep then the value of the activated hidden neuron will be used to obtain the output value of the neuron in the output layer. In the output layer the following equation is used to obtain the value at the neuron output.

$$net_k(t) = \sum_j w_{kj}\, y_j(t) + \theta_k \quad (22)$$

As with hidden neurons, neurons in the output layer also need to be activated, only in the output layer type activation function used instead of tanh again, but softmax. This is because the activation function of softmax is an activation function that can represent categorical distributions. The funmax activation softmax is as follows.

$$net_k = \frac{e^{net_k}}{\sum_{k=1}^{K} e^{net_k}} \quad (23)$$

As for getting derived values from softmax activation function used equation 24.

$$dnet_k = (net_k \times (1 - net_k)) \times (target_k - net_k) + \theta_k \quad (24)$$

Using the equations previously mentioned, ERNN's ERNN training using Backpropagation Through Time is described as follows.

1. Initialize weights with random values for each weights between input-hidden, context-hidden, hidden-output, and biases.
2. Determine the value of hyperparameter such as learning_rate, max_epoch.
3. Do step no. 4 to no. 8 as much as max_epoch times.
4. Calculate the feedforward.
   .In the feedforward calculation process use equation 20 to compute the value on the hidden neuron, then use equation 21 to compute the activation result of each hidden neuron, if the calculation process for each hidden neuron value has been done then calculate the value of each neuron output by using equation 22 then activate each output of the neuron using equation 23.
5. Backward Phase
   Calculate the error value of each neuron output, (the errror value at the neuron output is obtained from the difference in expected output value with the output produced in each neuron) Use equation 24.
   $Error_k = dnet_k$
   Calculate the value of error / gradient in each hidden neuron, in each timestep. The error value of each hidden neuron can be calculated using equation 25.
   $Error_j = dnet_j \times sum_j$ (25)
6. To get the value from used equation 21.
   While to get sum value used equation 26 if at last timestep.
   $sum_j = \sum_{k=0}^{k} Error_k \times w_{jk}$ (26)
   Or use equation 27 if not on the last timestep.
   $sum_h = \sum_{j=0}^{j} j \times u_{jh}$ (27)
7. After the error value of each hidden neuron in each timestep is obtained then calculate the delta for each weight using equation 28
   $delta_{ij} = \alpha \times dgradient \times input$ (28)
   which :
   α = Learning rate
   dgradient = gradient value (error) of j-layer ($Error_k$ atau $Error_j$)
   input = input value of i-neuron in the input layer
8. Update the weight value by means of the delta value + the weight value of origin.
   $v_{ji} = delta_{ji} + v_{ji}$ (29)
   $u_{jh} = delta_{jh} + u_{jh}$ (30)
   $w_{kj} = delta_{kj} + w_{kj}$ (31)
9. Finish.

2. Testing

The derivatives are as followsThe process undertaken at the stage of testing is done by calculating the value of inputs to the ERNN architecture that has been trained. That is by counting feedforward, as in process no. 4 at the training stage. If the recognition results match.

3. RESULTS

In this research the test is done by changing some variables that influence the timestep, the number of hidden neurons in one layer, and the number of epoch used.

3.1 Result of Testing with *Timestep* 3

In this section the test results of speech recognition using ERNN will be displayed. As for the number of hidden neurons that are attempted ranging from 7 neurons per layer to 18 neurons per layer. And the epoch used is 80000, 100000, 120000, and 150000.

a. *Epoch* 80000

The value of the test results with timestep 3 and epoch 80000 can be seen in Figure 7. In the graph it is seen that the highest accuracy value for timestep 3 and epoh 80000 is achieved by the architecture with the number of hidden neurons 16 for each layer (12-16-10), while the minimum accuracy found in the number of hidden neurons 9 (12-9-10).

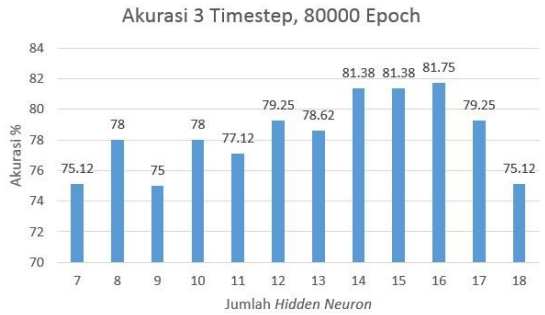


Figure 7 Graphic of Testing Result *Timestep* 3 *Epoch* 80000

b. *Epoch* 100000

the value of the test results with timestep 3 and epoch 100000 can be seen in Figure 8. In the graph it is seen that the highest accuracy value for timestep 3 and epoh 100000 is achieved by the architecture with the number of hidden neurons 15 for each layer (12-15-10), while the accuracy minimums earned on the number of hidden neurons 7 (12-7-10).

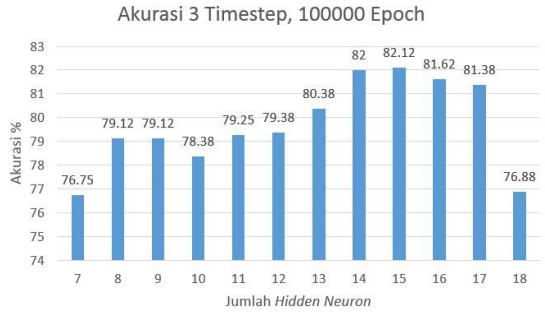


Figure 8 Graphic of Testing Result *Timestep* 3 *Epoch* 100000

c. *Epoch* 120000

the value of the test result with timestep 3 and epoch 120000 can be seen in Figure 9. In the graph it is seen that the highest accuracy value for timestep 3 and epoh 120000 is achieved by the architecture with the number of hidden neurons 14 for each layer (12-14-10), while the accuracy minimums earned on the number of hidden neurons 9 (12-18-10).

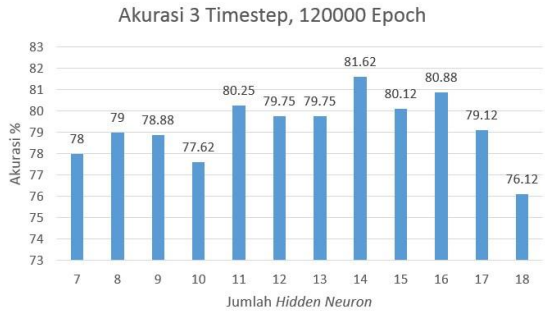


Figure 9 Graphic of Testing Result *Timestep* 3 *Epoch* 120000

d. *Epoch* 150000

the value of the test results with timestep 3 and epoch 120000 can be seen in Figure 10. In the graph it is seen that the highest accuracy value for timestep 3 and epoh 120000 is achieved by the architecture with the number of hidden neurons 14 for each layer (12-14-10), while the accuracy minimums earned on the number of hidden neurons 10 (12-10-10).

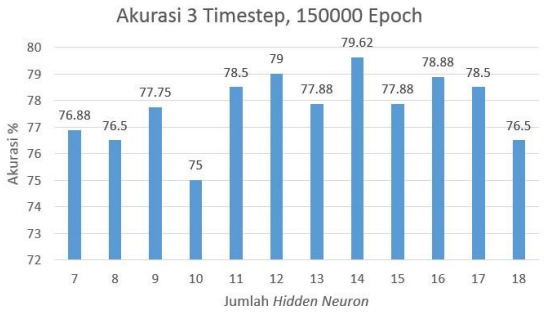


Figure 10 Graphic of Testing Result *Timestep* 3 *Epoch* 150000

3.2 Result of Testing with *Timestep* 4

Similar to test with timestep 4, testing on timestep 4 will test how the number of hidden neuron per layer impacts against ERNN's resulting accuracy.

1. *Epoch* 80000

The value of the test results with timestep 4 and epoch 80000 can be seen in the image xx. On the graph shows that the highest accuracy value for timestep 3 and epoh 80000 is achieved by the architecture with the number of hidden neurons 16 for each layer (12-16-10), while the minimum accuracy obtained on the number of hidden neurons 8 (12-8-10).

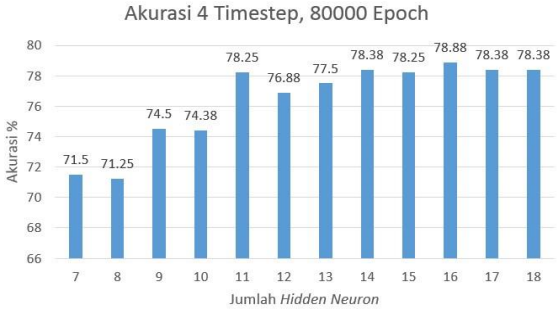


Figure 11 Graphic of Testing Result *Timestep* 4 *Epoch* 80000

2. *Epoch* 100000

The value of the test results with timestep 4 and epoch 80000 can be seen in Figure 12. In the graph it is seen that the highest accuracy value for timestep 4 and epoh 100000 is achieved by the architecture with the number of hidden neurons 13 for each layer (12-13-10), while the accuracy minimums earned on the number of hidden neurons 7 (12-7-10).

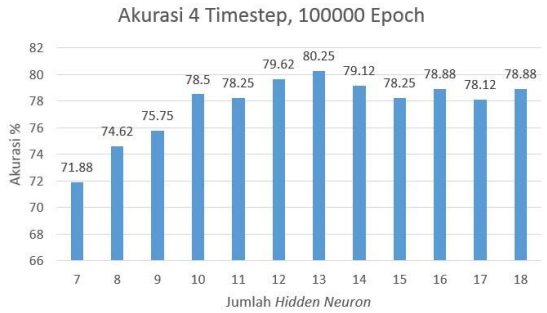


Figure 12 Graphic of Testing Result *Timestep* 4 *Epoch* 100000

3. *Epoch* 120000

The value of the test results with timestep 4 and epoch 80000 can be seen in Figure 13. In the graph it is seen that the highest accuracy value for timestep 4 and epoh 120000 is achieved by the architecture with the number of hidden neurons 12 for each layer (12-12-10), while the accuracy minimums earned on the number of hidden neurons 7 (12-7-10).

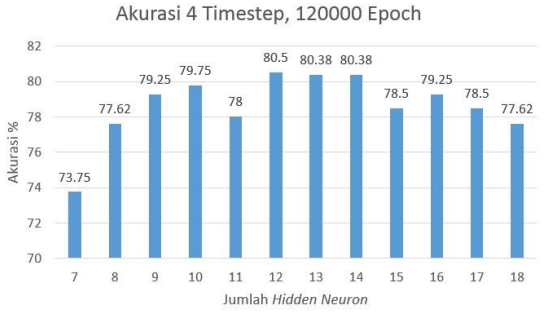


Figure 13 Graphic of Testing Result *Timestep* 4 *Epoch* 120000

4. *Epoch* 150000

The value of test result with timestep 4 and epoch 150000 can be seen in Figure 14.

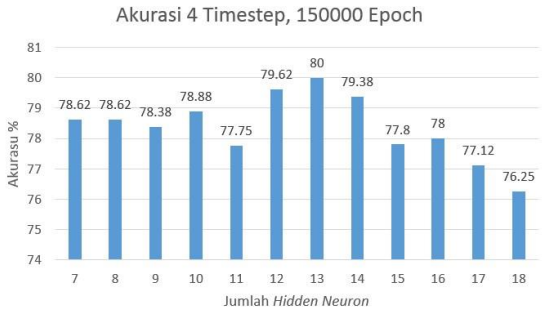


Figure 14 Graphic of Testing Result *Timestep* 4 *Epoch* 150000

On the graph shows that the highest accuracy value for timestep 4 and epoh 150000 is achieved by the architecture with the number of hidden neurons 13 for each layer (12-13-10), while the minimum accuracy obtained on the number of hidden neurons 18 (12-18-10).

3.3 Result of Testing with *Timestep* 5

1. *Epoch* 80000

The value of the test results with timestep 5 and epoch 80000 can be seen in Figure 15. In the graph it is seen that the highest accuracy value for timestep 5 and epoh 80000 is achieved by the architecture with the number of hidden neurons 17 for each layer (12-17-10), while the accuracy minimums earned on the number of hidden neurons 11 (12-11-10).

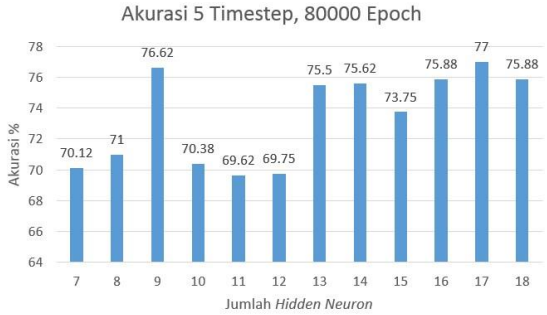


Figure 15 Graphic of Testing Result *Timestep* 5 *Epoch* 80000

2. *Epoch* 100000

The value of the test results with timestep 5 and epoch 80000 can be seen in Figure 16. In the graph it is seen that the highest accuracy value for timestep 5 and epoh 80000 is achieved by the architecture with the number of hidden neurons 17 for each layer (12-17-10), while the accuracy minimums earned on the number of hidden neurons 7 (12-7-10).

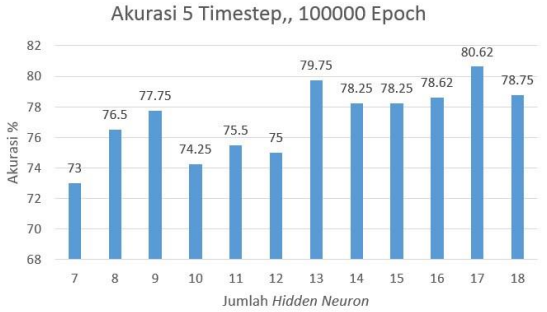


Figure 16 Graphic of Testing Result *Timestep* 5 *Epoch* 100000

3. *Epoch* 120000

The value of the test result with timestep 4 and epoch 120000 can be seen in Figure 17. In the graph it is seen that the highest accuracy value for timestep 3 and epoh 120000 is achieved by architecture with the number of hidden neurons 12,13 and 17, while the minimum accuracy obtained on the number hidden neurons 7 (12-7-10).

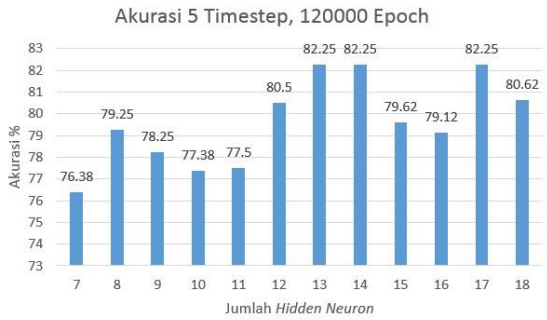


Figure 17 Graphic of Testing Result *Timestep* 5 *Epoch* 120000

4. *Epoch* 150000

The value of the test results with timestep 4 and epoch 150000 can be seen in figure 18. In the graph it is seen that the highest accuracy value for timestep 3 and epoh 150000 is achieved by the architecture with the number of hidden neurons 14 for each layer (12-14-10) with an accuracy of 83.88 %, while the minimum accuracy obtained on the number of hidden neurons 11 (12-11-10) while the accuracy is 76.62% .

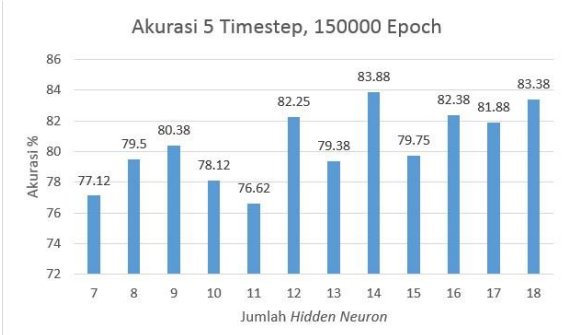


Figure 18 Graphic of Testing Result *Timestep* 5 *Epoch* 150000

3.4 Best Testing Result

The best accuracy test in this research was ERNN with 5 timestep architecture, with 14 hidden neurons trained as many as 150000 epoch (12-14-10). The accuracy generated by the architecture is 83.88%, for more details the best recognition results can be seen in the form of confussion matrix as in figure 17.

		Kata Hasil Pengenalan										
	Kelas	Ada	Adalah	Beri	Bisa	Guna	Jadi	Laku	Milik	Rupa	Sebut	Akurasi
Label Kata dari Dataset	Ada	65	9	0	0	1	0	1	0	3	1	81.25%
	Adalah	8	60	0	0	4	0	6	1	1	0	75.00%
	Beri	0	0	63	0	1	5	1	10	0	0	78.75%
	Bisa	0	1	0	78	0	0	0	1	0	0	97.50%
	Guna	6	3	1	1	65	0	0	0	4	0	81.25%
	Jadi	0	0	5	0	0	71	0	4	0	0	88.75%
	Laku	0	0	1	1	0	0	71	0	5	2	88.5%
	Milik	0	0	2	0	2	3	0	72	1	0	90.00%
	Rupa	2	4	0	1	6	0	2	0	65	0	81.25%
	Sebut	0	4	1	0	1	2	8	1	2	61	76.25%
Total Akurasi												83.85%

Figure 19 Confussion Matrix of Best Accuracy Produced

3.5 Analysis of Testing Result

From the test result, the best result is 83.88% obtained from the test with timestep 5 with 150000 epoch, then the other result is 82.12% if the timestep used is 4 and the epoch 120000, as well as the use of 3 timestep which is able to produce the best accuracy above 80% although the number of epoch is not too much. From these observations it can be seen that the highest accuracy results are obtained by the most timestep (ie timestep 5) but, it is also accompanied by the need for large epoch numbers for optimal system accuracy. In contrast, the smallest number of timesteps (timestep 3) is always capable of producing the best accuracy above 80% in each epoch, but the accuracy that the timestep is capable of generating is less developed even as the epoch increases.

4. Conclusion

After testing ERNN to recognize the word verbs in Indonesian, it can be concluded that the accuracy of ERNN is good enough, reaching 83.88%, it is not possible that accuracy can be better if other characteristic extraction method is used, or modification to the method feature extraction used.

The suggestion of this research is to try other characteristic extraction methods such as LPC or RASTA-PLP, can also add delta and delta logs into MFCC as input on ERNN (which in this research is not used), or with the use of momentum while training ERNN , or use different words. In addition, other types of recurrent neural networks such as LSTM (Long-Short Term Memory) can also be tried for the same case.

REFERENCES

[1] S. Muis, *Teknik Jaringan Syaraf Tiruan*. Yogyakarta: Penerbit Graha Ilmu, 2006.

[2] Y. A. Alotaibi, "A Simple Time Alignment Algorithm for Spoken Arabic Digit Recognition," *JKAU*, vol. 20, pp. 29–43, 2009.

[3] A. D. Andriana, "METODE MEL FREQUENCY CEPSTRUM COEFFICIENTS Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)," *J. Ilm. Komput. dan Inform.*, vol. 2, no. 1, pp. 21–26, 2013.

[4] S. Chen and Y. Luo, "Speaker verification using MFCC and support vector machine," *Proc. Int. MultiConference ...*, vol. I, pp. 18–21, 2009.

[5] M. Fachrie, "Robust Indonesian Digit Speech Recognition using Elman Recurrent Neural Network," pp. 49–54, 2015.

[6] Z. Ramadhan and S. N. Endah, "Perintah Suara Berbahasa Indonesia untuk Membuka dan Menutup Aplikasi dalam Sistem Operasi Windows Menggunakan Metode Mel

Frequency Cepstrum Coefficient dan Metode Backpropagation," no. Snik, pp. 33–41, 2016.

[7] M. Boden, "A guide to recurrent neural networks and backpropagation," *Electr. Eng.*, no. 2, pp. 1–10, 2001.