Training Performance of Recurrent Neural Network using RTRL and BPTT for Gamelan Onset Detection

Dian Kartika Sari¹, Diah Puspito Wulandari¹, and Yoyon Kusnendar Suprapto¹

¹ Departemen of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

E-mail : dian.kartika.sari16@mhs.ee.its.ac.id

Abstract. Gamelan is one of Indonesia's traditional musical instruments. Signal variations in gamelan music are caused by differences in style and the process of making gamelan. Gamelan music analysis usually using supervised learning method like Recurrent Neural Network (RNN). This paper will compare the performance of Simple Recurrent Neural Network training process using a gradient-based algorithm Backpropagation Through Time (BPTT) and Real-Time Recurrent Learning (RTRL) algorithm. The performance of the algorithm during training process was necessary to be evaluated, in order to know which algorithm has better performance and faster process to approach convergences on the training method of the recurrent neural network. The performance results of the algorithm training process will be compared and evaluated by the means of a Normalized Negative Log-likelihood (NNL). BPTT resulted better and faster forming convergence in terms of the number of epoch parameter with NNL 0.0121. In terms of the value of learning rate, BPTT perform better at learning rate 0.1 with NNL 0.0174 and RTRL performs better at learning rate 0.4 with NNL 0.0382.

1. Introduction

Gamelan is one of Indonesia's traditional musical instruments. A Gamelan set consists of several groups, about fifteen, of different instruments. Their groups are Balungan, kenong, kempul, kendang, bonang, etc. The Balungan group is constructed by three instruments namely Demung, Saron and Peking. Gamelan has two fundamental music notations, called Slendro and Pelog. While Slendro is a pentatonic, Pelog is a heptatonic[1]. The process of making gamelan instruments only uses traditional methods that rely on the experience and estimates of gamelan-making experts without any specific notation standards such as the making of modern Western musical instruments. The style of play and the force of the punch which applied to the gamelan instruments also cause several variations, such as the force of the punch and the location of the area being attacked (edge, middle). This condition induces variations in terms of basic frequencies, signal envelopes, and harmonic contents of instrument sounds between different gamelan sequences[2].

The temporal evolution of musical instrument sounds plays a conspicuous role on the perception of their most important features. Based on temporal evolution of audio signal, envelope model of signal divided into ADSR (Attack, Decay, Sustain, Release) area [3]. The attack corresponds to the initial excitation of the instrument, the decay supposedly corresponds to a decrease of energy after the attack, the sustain part usually corresponds to the region where the system is constantly exited with external energy and the release area can refer to the release of the excitation [4]. The onset of the note is a
single instant chosen to mark the start of the attack area can be reliably detected [5]. Onset detection and onset location determination are very important roles for the music signal processing process, especially for the process of segmentation and music transcription or the process of notifying music signals. This onset feature determines the beginning of the rhythm of a music. Basically the onset detection method includes three stages, pre-processing, reduction, and peak-picking. In the pre-processing stage, the signal is transformed into a domain that emphasizes relevant features. The reduction phase produces an onset detection function and is a stage that distinguishes the method from the others.

Recurrent neural network (RNN) is one of the method that used to develop onset reduction function. Recurrent neural network, in contrast to the classical feedforward network, better handle input that has space-time structure, but very poorly and slowly to satisfactory result when being solved with the classic gradient optimisation method on longer input sequences[6]. Research of [7] using a classifier-based approach to onset detection. Two-way repetitive neural networks with short-term memory are used because they have the ability to map the entire previous input memory and the context of future events for network output. The research of [2] used neural network method using Elman network to detect gamelan music onset. Short-time Fourier Transform (STFT) is used to convert a signal into a time frequency domain. The results show that performance is very dependent on the length of the window used and the variation in the signal. The research of simple recurrent neural network training using RTRL and extended Kalman filter has been introduced by [8]. He performed experiment on two artificial languages, one is a complex center-embedding recursion (CER) that cannot be generated by a finite state machine, and the other one is a right-branching recursion (RBR) that can be produced by simple iterative process carried out by finite state machine Whereas, the study of [6] using recurrent neural network training with extended Kalman filter. He compared the modified Kalman filter, called Unscented Kalman filter, with the extended Kalman filter to training sequences of symbol prediction of Reber’s language.

In this paper, authors focused on the performance comparison of RTRL and BPTT algorithm to training onset detection function of gamelan signal. The performance of algorithm during training process was necessary to know before we used the algorithm into a complex task. BPTT algorithm consists of unfolding a recurrent network in time and applying the well-known backpropagation algorithm directly. RTRL algorithm represents the gradient-based approach, where estimates of derivatives needed for evaluating error gradient, calculated in every time step [8]. In this study, we will compare two algorithm in order to know which algorithm has better performance and faster process to approach convergences for better training method in the recurrent neural network.

2. Proposed Method

This section describes the main flow of the proposed method of comparison simple recurrent neural network training sections. First, the audio data were generated by mixing several single hit gamelan sound and set the distance between them. Then, the output audio was transformed to the time-frequency representation using short time Fourier transform (STFT). The mechanism of STFT is to make non-stationary signal becomes stationary signal representation using window function. A non-stationary signal is a signal that has the frequency which changes in a certain time, like a music signal that has a frequency variation in each time. The output of STFT then processed using triangular filterbank to extract the features of Mel spectrogram and the positive first-order difference. The obtained Mel spectrogram and the positive first-order difference then used as an input of RNN training.

2.1. Feature extraction

As input, we used the audio signal with sampling rate 48kHz. STFT window length 1024 was chosen to maintain better time resolution. The same scheme like as [2], STFT with 480 hop size and 10 ms time resolution was used in this paper. The output of STFT was complex spectrogram $S(n,k)$ with $n$ is frame index and $k$ is frequency bin index, then the output STFT was converted to the power spectrogram [7].
The matrix dimensionality of the power spectrogram reduced by using a filterbank with 40 triangular filters to transform the power spectrogram to the Mel spectrogram. Then transform Mel spectrogram into its logarithmic value to match human perception of loudness.

\[ M_{\log}(n,m) = \log((n,m) + 1.0)) \]  

The next feature is the positive first order difference which can be extracted by applying half-wave rectifier function to two consecutive Mel spectra [7].

\[ D^+(m,n) = H(M_{\log}(n,m) - M_{\log}(n-1,m)) \]  

2.2. Elman Neural Network (BPTT algorithm)

In this study, author used the Elman recurrent neural network as a simple recurrent neural network which trained by BPTT as a standard training function. The representation of simple recurrent using Elman network can be seen in Figure 1.

![Figure 1. Representation of Elman recurrent neural network](image)

Input layer defined by I, the recurrent layer R and the output layer O. They are fully connected layer with weight \( W^{RI} \) and \( W^{OR} \). Given input pattern time \( t, I^{(t)} = (I_{1}^{(t)}, \ldots, I_{I}^{(t)}, \ldots, I_{I}^{(t)}) \) and recurrent activities network \( R^{(t)} = (R_{1}^{(t)}, \ldots, R_{j}^{(t)}, \ldots, R_{n}^{(t)}) \). The recurrent net input and output activity calculated as [8]:

\[ \tilde{R}_{i}^{(t)} = \sum_{j} W^{RI} I_{j}^{(t)} + \sum_{j} W^{RC} \tilde{R}_{j}^{(t-1)} \]  

\[ R_{i}^{(t)} = f(\tilde{R}_{i}^{(t)}) \]  

Where output unit \( k \), calculates its input \( \tilde{O}_{i}^{(t)} \) and output activity \( O_{i}^{(t)} \) as:
\[ \tilde{O}_i^{(t)} = \sum_j W_{ij}^{OR} R_j^{(t)} \]  
\[ O_i^{(t)} = f(\tilde{O}_i^{(t)}) \]  

Where \(|I|, |R|\) and \(|O|\) are the number of input units, hidden units and output units, and \(f\) is the activation function which used the logistic sigmoid function [8].

### 2.3. RTRL Algorithm

RTRL was based on approximate on-line gradient computation and was described in [9]. But, for Elman RNN we use modification of RTRL, network weight are updated in each time step to minimize the output error. In time \(t\), modification of weight connecting output and recurrent unit are calculate as [8]:

\[ \Delta W_{ij}^{OR} = \alpha(D_i^{(t)} - O_i^{(t)}) f'(\tilde{O}_i^{(t)}) R_j^{(t)} \]  

Where \(D_i^{(t)} = (D_1^{(t)}, ..., D_\alpha^{(t)}, ..., D_{|O|}^{(t)})\) is the desired output pattern and \(\alpha\) is the learning rate. Modification of weight connecting recurrent and input unit :

\[ \Delta W_{ji}^{RI} = \alpha \sum_k (D_k^{(t)} - O_k^{(t)}) f'(\tilde{O}_i^{(t)}) \sum_{l=1}^{R_{\text{out}}} W_{lh}^{\text{RC}} \frac{\partial R_i^{(t)}}{\partial W_{ji}^{RI}} \]  

Then, modifications of weights connecting context unit and recurrent unit :

\[ \Delta W_{ji}^{RC} = \alpha \sum_k (D_k^{(t)} - O_k^{(t)}) f'(\tilde{O}_i^{(t)}) \sum_{l=1}^{R_{\text{out}}} W_{lh}^{\text{RC}} \frac{\partial R_i^{(t)}}{\partial W_{ji}^{RC}} \]  

Where \(\hat{\delta}_{ij}^{kron}\) is the Kronecker’s delta and \(\hat{\delta}_{ij}^{kron} = 1\) if \(h = i\), otherwise \(\hat{\delta}_{ij}^{kron} = 0\).

### 3. Experimental Result

#### 3.1. Dataset

Dataset consist of 105 audio data of semi-synthetic gamelan sound, 70 audio data were used for the training process and the rest data were used for the testing process. We used the same scheme as [2], but we modified the time interval of the consecutive hits as 20 ms, 40ms and 50 ms. The input data for the network were 40 features of log Mel spectra and 40 features of its positive first order difference. Before we trained the features, we changed them to be a vector’s form as an input of the Elman network. We used two scheme for the training process with BPTT and RTRL. The main goal of this study was to compare the performance of training process with BPTT and RTRL, it was necessary to know the
algorithm which has better performance and faster process to approach convergences for better training method in the recurrent neural network.

3.2 Training Method
BPTT and RTRL were used to train Elman recurrent neural network. Network weight and starting activation were randomly initialized from the interval (-0.5, 0.5). In this training process, we used three different scheme to be compared. The first scheme, we used different training epoch as a parameter to be compared, and we set other parameters to be the same. We used 50 training epoch, 100 training epoch and 125 training epoch to be compared. The second scheme, we used the different number of hidden layer units to be compared. We used 10 units, 30 unit and 50 units of the hidden layer. The third scheme, we used different learning rate, learning rate varied from 0.1 to 0.5 and the other parameters were the same.

3.3 NNL (Normalized Negative Log-likelihood)
The performance of the trained algorithm was evaluated by the means of a normalized negative log-likelihood (NNL), which was calculated over input features sequence from time step \( t = 1 \) to \( T \) as [8]:

\[
\text{NNL} = -\frac{1}{T} \sum_{t=1}^{T} \log_p|A| p^{(t)}(s^{(t)})
\]

(14)

Where the base of the logarithm \(|A|\) is the number of input sequence and the \( p^{(t)}(s^{(t)}) \) is the probability of predicting features \( s^{(t)} \) in the time step \( t \). Value \( p^{(t)}(s^{(t)}) \) is obtained by normalizing activities of output units and choosing normalized output activity corresponding to the input features \( s^{(t)} \). With NNL = 0 corresponding to the 100% correct prediction [10].

3.4 Training Result

a) Different epoch parameter:
In this scheme, we used different epoch parameter to know which number of epoch produce the best result on training process. We used 80 input, 20 number of hidden units, 0.0 momentum rate and 0.5 learning rate for each dataset as standard parameters. Otherwise, we used the different number of epoch parameters to be evaluated, such as 50 training epoch, 100 training epoch, and 125 training epoch. The result of this scheme can be seen in Figure 2.

![Figure 2](image)

**Figure 2.** a) NNL performance of the output layer of 50 training epoch. b) NNL performance of the output layer of 125 training epoch
The data of comparison of NNL of different number of epochs can also be presented as in Table 1.

| Epoch | NNL of BPTT (%) | Correct Prediction BPTT (%) | NNL of RTRL (%) | Correct Prediction RTRL (%) |
|-------|-----------------|-----------------------------|-----------------|-----------------------------|
| 50    | 0.0211          | 97.89                       | 0.0321          | 96.79                       |
| 100   | 0.0121          | 96.79                       | 0.462           | 53.74                       |
| 125   | 0.0125          | 98.75                       | 0.0491          | 95.09                       |

From the results of the comparison at Table 1, it can be seen that the higher the epoch value used to train the network, the NNL value tends to be smaller in BPTT, but for network trained using RTRL, the higher the epoch value, the NNL value also tends to be higher. The ideal epoch value at BPTT is 100 training epoch. Whereas, the ideal epoch value at RTRL is 50 training epoch. RTRL tends to be better at smaller epoch values, because RTRL calculates network weights every time step and then updates the network weights, with the aim of minimizing errors gradient at every time step.

b) Different hidden unit parameter:

In this scheme, we used different hidden units parameters to know which number of hidden units which produce the best result in the training process. We used the different number of hidden units parameters to be evaluated, such as 10 hidden units, 20 hidden units, 30 hidden units, 40 hidden units, and 50 hidden units. We used 80 input, 50 training epoch, 0.0 momentum rate, and 0.5 learning rate for each dataset as standard parameters. The result of this scheme can be seen in Figure 3.

![Figure 3](image)

**Figure 3.** a) NNL performance of the output layer of 10 hidden units. b) NNL performance of the output layer of 30 hidden units

The comparison NNL of different number of hidden units can also be presented as in Table 2.

| Hidden units | NNL of BPTT (%) | Correct Prediction BPTT (%) | NNL of RTRL (%) | Correct Prediction RTRL (%) |
|--------------|-----------------|-----------------------------|-----------------|-----------------------------|
| 10           | 0.0257          | 97.43                       | 0.3451          | 65.49                       |
| 20           | 0.0258          | 97.42                       | 0.4598          | 54.02                       |
From the results of the comparison at Table 2, it can be seen that the higher the number of hidden units used to train the network, the NNL value tends to be higher in BPTT, the same thing also happened to the network training using RTRL, the higher the number of hidden units, the NNL value also tends to be higher. The ideal number of hidden units at BPTT is 10 hidden units. Whereas, the ideal number of hidden units at RTRL is 10 hidden units. In this parameter, RTRL and BPTT produce the same result of performance, they produce optimal performance at the smallest number of hidden units.

c) **Different learning rate parameter:**

In this scheme, we used different learning rate parameters to know which number of learning rate which produce the best result in the training process. We used the different number of learning rate parameters to be evaluated, the learning rate was set varied from 0.1 to 0.5. Otherwise, we used 80 input, 50 training epoch, 0.0 momentum rate, and 20 hidden units for each dataset as standard parameters. The result of this scheme can be seen in Figure 4.

![Figure 4. a) NNL performance of the output layer of 0.1 learning rate. b) NNL performance of the output layer of 0.4 learning rate](image)

The data of comparison NNL of different learning rate can also be presented as in Table 3.

| Learning rate | NNL of BPTT | Correct Prediction BPTT (%) | NNL of RTRL | Correct Prediction RTRL (%) |
|---------------|-------------|-----------------------------|-------------|-----------------------------|
| 0.1           | 0.0174      | 98.26                       | 0.3484      | 65.16                       |
| 0.2           | 0.0305      | 96.95                       | 0.1420      | 85.80                       |
| 0.3           | 0.0313      | 96.87                       | 0.1672      | 83.28                       |
| 0.4           | 0.0445      | 95.55                       | 0.0382      | 96.18                       |
| 0.5           | 0.0275      | 97.25                       | 0.2760      | 72.40                       |
From the results of the comparison at Table 3 it can be seen that the higher the value of learning rate used to train the network, the NNL value tends to fluctuate in BPTT. The smaller NNL of BPTT is 0.0174 at learning rate 0.1 , if we set the value of learning rate at 0.4, the NNL value will be higher, but after we set the learning rate value at 0.5, the NNL value will be back turned smaller to 0.0275.

The same thing also happened to the network training using RTRL, the value of NNL is fluctuate at RTRL, the smaller NNL of RTRL is 0.0382 at learning rate 0.4 and the higher value NNL of RTRL is 0.3484 at learning rate 0.1. The ideal value of the learning rate at BPTT is 0.1. Whereas, the ideal value of the learning rate at RTRL is 0.4.

4. Conclusion

In this paper, we have experimented two algorithms BPTT and RTRL to trained the Elman recurrent neural network for reduction function on gamelan music signal. In terms of the number of epochs, BPTT resulted better with the higher correct prediction 98.79% . In terms of the number of hidden layers, BPTT resulted better with correct prediction 97.43% and faster forming convergence rather than RTRL. But, in terms of the value of learning rate, both BPTT and RTRL have fluctuated value of NNL. BPTT perform better at learning rate 0.1 with correct prediction 98.26 %, an RTRL performs better at learning rate 0.4 with correct prediction 96.18 %. From this comparison performance, we have known that BPTT has higher robustness and better performance rather than RTRL in term of training process. In this experiment, we used HP Probook 430 G4 Notebook PC with specification Intel Core i5-7200U Intel HD Graphics 620. The result might be different in other PC specification.

References

[1] Sumarsam 1992-1995 Cultural Interaction and Musical Development in Central Java The University of Chicago Press ISBN 0226780112

[2] Wulandari D P, Suprapto Y K and Purnomo M H 2012 Gamelan Music Onset Detection Using Elman Network Proc. 2012 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Tianjin China, pp 91-96

[3] Pinch T and Trocco F 2002 Analog Days: The Invention and Impact of the Moog Synthesizer (Cambridge: MA: Harvard University Press)

[4] Caetano M, Burred J J and Rodet X 2010 Automatic Segmentation of the Temporal Evolution of Isolated Acoustic Musical Instrument Sounds Using Spectro-Temporal Cues 13th International Conference on Digital Audio Effects (DAFX-10) Graz Austria pp 1-8

[5] Bello J P, Daudet L, Abdallah S, Duxbury C, Davies M, and Sandler M B 2005 A Tutorial on Onset Detection in Music Signals IEEE Trans. on Speech and Audio Processing Vol 13 No 5 pp 1-13

[6] Trebaticky P and Bielikova M 2005 Recurrent Neural Network Training with Extended Kalman Filter JIT.SRC pp 57-64

[7] Eyben F, Böck S, Schuller B and Graves A 2010 Universal Onset Detection with Bidirectional Long Short-term Memory Neural Networks 11th International Society for Music Information Retrieval Conference (ISMIR) pp 589-594

[8] Cermansky M and Benuskova L 2003 Simpe Recurrent Neural Network Trained By RTRL and Extended Kalman Filter Algorithms Neural Network World Vol 13 No 3 pp 223-234

[9] William R J and Zipser D 1989 A Learning Algorithm for Continually Running Fully Recurrent Neural Networks Neural Computation Vol 1 No 2 pp 270-280

[10] Cermansky M and Benuskova L 2001 Finite-State Reber Automaton and the Recurrent Neural Networks Trained in Supervised and Unsupervised Manner In: Dorffner G., Bischof H., Hornik K. (eds) Artificial Neural Networks — ICANN 2001 Lecture Notes in Computer Science vol 2130 (Berlin : Springer)
Acknowledgments
I would like to express my gratitude to my lecturer in Electrical Engineering ITS for their guidance and encouragement in this work.