

ANALISIS PENGGUNAAN PARALLEL PROCESSING MULTITHREADING PADA RESILIENT BACKPROPAGATION

Kelvin Onggrono, Tulus, Erna Budhiarti Nababan
Magister Teknik Informatika, Universitas Sumatera Utara
Jl. Universitas No. 9A Kampus USU, Medan, Sumatera Utara-Indonesia
kelvin_onggrono@yahoo.co.id

Abstrak—Proses pembelajaran neural network merupakan hal yang penting, bertujuan untuk mengenali lingkungan. Proses pembelajaran neural network membutuhkan waktu untuk dapat mengenali lingkungan. Terutama pada salah satu algoritma neural network yaitu resilient backpropagation. Proses untuk mempercepat pembelajaran resilient backpropagation pada penelitian ini adalah menggunakan teknik parallel processing. Teknik parallel processing yang digunakan adalah multithreading. Teknik parallel ini diterapkan pada bagian hidden layer yaitu membagi jumlah neuron pada hidden layer menjadi beberapa subproses yang dikerjakan secara bersamaan, pembagian yang dilakukan berdasarkan pada jumlah thread. Hasil yang didapatkan dalam penerapan parallel processing menggunakan teknik multithreading ke dalam algoritma resilient backpropagation membantu mempercepat waktu proses pembelajaran resilient backpropagation dengan thread yang digunakan sebanyak 3 buah thread.

Keyword — Neural Network, Resilient Backpropagation, Parallel, Hybrid Partition, Multithreading.

I. PENDAHULUAN

Artificial Neural Network (ANN) merupakan komputasi yang mengambil sistem biologi yaitu jaringan saraf, jaringan saraf buatan ini berfungsi untuk melakukan komputasi klasifikasi, pengenalan pola, kontrol, *forecasting*, dll [1]. Algoritma ANN sebelum diuji pada sebuah lingkungan, ANN perlu melakukan *training* terlebih dahulu agar algoritma ANN dapat mengenali lingkungan tersebut. Pada saat algoritma ANN di *training* maka memakan waktu yang cukup lama tergantung dari besarnya *dataset*, nilai *epoch*, kecepatan pembelajaran mencapai titik konvergen [2].

Proses training pada *artificial neural network* membutuhkan waktu untuk menyelesaikan masalah tersebut salah satu metode yang dapat digunakan adalah *adaptive learning rate*. *Adaptive learning rate* adalah teknik yang digunakan untuk mengubah *learning rate* berdasarkan pada perubahan nilai error yang dihasilkan dari setiap iterasi [3]. Selain penggunaan metode *adaptive learning rate* untuk mempercepat waktu training pada ANN, ada metode lain yang dapat digunakan yaitu memasukan teknik *parallel* kedalam arsitektur *neural network* tersebut. Teknik *parallel* adalah membagi proses menjadi subproses yang dapat dijalankan secara bersamaan. Penerapan *parallel* komputer terhadap *backpropagation* sangat membantu dalam proses training yaitu dapat memangkas waktu training [4].

Resilient Propagation / Resilient Backpropagation (RPROP) adalah salah satu algoritma *neural network* yang dilengkapi dengan kemampuan *adaptive learning* yaitu kemampuan untuk menentukan *learning rate* yang digunakan berdasarkan nilai *error gradien* [5]. Algoritma RPROP dalam kasus untuk mengklasifikasikan gigitan ular menunjukkan tingkat kecepatan pembelajaran untuk mencapai titik konvergen berdasarkan pada nilai batasan MSE 0.03 dimana standar *backpropagation* membutuhkan pembelajaran sebanyak 378 epoch sedangkan *resilient backpropagation* hanya membutuhkan sebanyak 73 epoch untuk mencapai titik konvergen berdasarkan pada nilai MSE [6]. Algoritma RPROP memiliki kecepatan dalam mencapai nilai konvergen dibandingkan dengan *backpropagation*. Pada tingkat akurasi algoritma RPROP berada pada rata 76,88%, sedangkan rata akurasi *backpropagation* 69,264% [7]. Kemampuan algoritma RPROP menjadikan pengujian terhadap semua kasus *dataset* yang diuji sangat stabil dalam pembelajaran [3]. Pada penelitian terdahulu menunjukkan kemampuan *resilient propagation* memiliki keunggulan kecepatan dalam training untuk mencapai titik konvergen dibandingkan dengan *backpropagation* standar. Pengujian data yang digunakan adalah data kasus klasifikasi aktifitas manusia berdasarkan sensor gyroscope dan accelerometer.

II. LANDASAN TEORI

A. ResilientBackpropagation

Resilient backpropagation (RPROP) adalah algoritma neural network yang bersifat supervised dan adaptive learning. RPROP merupakan perkembangan dari backpropagation. Pada resilient backpropagation parameter sudah ditetapkan jadi tidak diperlukan penentuan learning rate lagi. Kemampuan RPROP adalah untuk menghindari perubahan nilai gradien yang terlalu kecil ini bertujuan agar mempercepat laju pembelajaran RPROP dibandingkan dengan backpropagation, ini ditunjukkan pada jaringan RPROP memiliki sebuah nilai delta yang mengikuti perubahan nilai weight. Jika perubahan nilai weight kecil maka nilai delta membesar, sebaliknya ketika perubahan weight aktif maka nilai delta mengecil.

RPROP memiliki langkah yang sama dengan backpropagation yang membedakannya adalah pada waktu backwardnya yaitu pada bagian updateweight. Untuk melakukan updateweight harus dilakukan pemilihan berdasarkan gradien error yang dihasilkan, lalu penentuan learningrate yang dipakai untuk melakukan updateweight.

Pelatihan feedforward pada algoritma resilient propagation sama dengan pelatihan feedforward pada algoritma backpropagation, yang membedakan pada waktu melakukan updateweight dengan learningrate pada pelatihan backward. Perhitungan learningrateakan dijabarkan pada langkah – langkah sebagai berikut (Riedmiller & Braun, 1993):

1) Perhitungan nilai $\Delta_i^{(t)}$

Untuk menentukan nilai $\Delta_i^{(t)}$ ada beberapa aturan yang harus dipenuhi untuk mendapat nilai delta tersebut yaitu :

$$\Delta_i^{(t)} = \begin{cases} \eta^+ * \Delta_i^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_i} * \frac{\partial E^{(t)}}{\partial w_i} > 0 \\ \eta^- * \Delta_i^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_i} * \frac{\partial E^{(t)}}{\partial w_i} < 0 \\ \Delta_i^{(t-1)}, & \text{if } e_i \end{cases}$$

Untuk mendapatkan $\Delta_i^{(t)}$ untuk pertama kali pembelajaran $\Delta_i^{(t-1)} = \Delta_0$. Nilai $\Delta_0 = 0.1$ nilai pada delta 0 dapat saja ditetapkan lebih besar atau lebih kecil dari 0.1, karena nilai delta 0 tidak memiliki pengaruh besar terhadap laju proses pembelajarannya (Martin Riedmiller, 1994). Untuk pembelajaran berikutnya maka Δ_i yang terdahulu akan dikalikan dengan learningrate. Pada perkalian $\Delta_i^{(t-1)}$ dengan learningrate ada aturan yang harus dicapai yaitu

error gradien pada hasil terdahulu $\frac{\partial E^{(t-1)}}{\partial w_i}$ dikalikan dengan errorgradien $\frac{\partial E^{(t)}}{\partial w_i}$. Dari hasil perkalian error tersebut maka didapatkan hasil jika hasil lebih besar dari 0 maka $\Delta_i^{(t-1)}$ dikalikan dengan η^+ , dan jika hasil lebih kecil dari 0 maka $\Delta_i^{(t-1)}$ dikalikan dengan η^- . Nilai η^+ dan η^- merupakan learningrate yang membedakan adalah pada η^+ memiliki nilai yang lebih besar daripada η^- . Marthin Riedmiller (1994) menetapkan nilai standar pada $\eta^+ = 1.2$ sedangkan $\eta^- = 0.5$ untuk algoritma resilient propagation.

2) Penentuan nilai $\Delta w_i^{(t)}$

Setelah ditentukan $\Delta_i^{(t)}$ maka akan dilanjutkan ke dalam aturan untuk menentukan fungsi operator yang akan dipakai untuk melakukan updateweight dengan aturan sebagai berikut:

$$\Delta w_i^{(t)} = \begin{cases} -\Delta_i^{(t)}, & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ +\Delta_i^{(t)}, & \text{if } \frac{\partial E^{(t)}}{\partial w_i} < 0 \\ 0, & \text{if } e_i \end{cases}$$

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t)}$$

Dengan menggunakan error function maka diketahui $\Delta_i^{(t)}$ yang akan digunakan apakah akan dikurangi atau ditambah atau tidak terjadi perubahan bobot. Untuk menentukan fungsi operator yang dipakai pada $\Delta_i^{(t)}$ maka dibandingkan nilai error gradien terhadap nol. Jika nilai error gradien lebih besar daripada nol maka $\Delta w_i^{(t)}$ akan menerima nilai $-\Delta_i^{(t)}$, dan jika nilai error gradien lebih kecil daripada 0 maka $\Delta w_i^{(t)}$ akan menerima $+\Delta_i^{(t)}$. Setelah mendapatkan nilai delta w maka dilanjutkan ke updateweightnya secara langsung.

Pada aturan pertama untuk penentuan learningrate memiliki masalah pada aturan kedua yaitu terjadi peningkatan pembelajaran yang melebihi batas minimum maka dilakukan pengurangan weight secara langsung dengan weight terdahulunya.

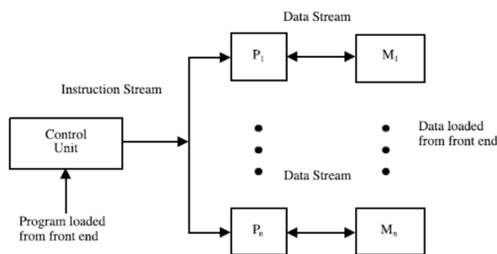
$$\Delta w_i^{(t)} = -\Delta w_i^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_i} * \frac{\partial E^{(t)}}{\partial w_i} < 0$$

Untuk menghindari terjadi keadaan tersebut untuk terjadi kedua kalinya maka pada $\frac{\partial E^{(t)}}{\partial w_i} = 0$.

B. Parallel Processing

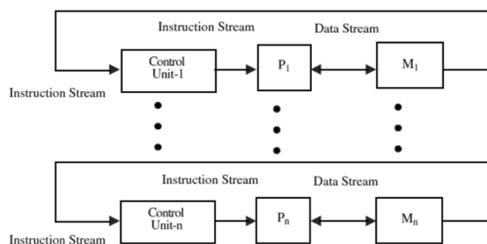
Parallel processing adalah sebuah kegiatan menjalankan beberapa proses sekaligus dalam waktu yang bersamaan. Teknik *parallel processing* ada 2 macam yaitu SIMD (*Single Instruction Multiple Data streams*) dan MIMD (*Multiple Instruction Multiple Data streams*).

SIMD (*Single Instruction Multiple Data streams*) adalah teknik *parallel* yang menggunakan sebuah *control unit* dan *processor* melakukan eksekusi terhadap instruksi yang sama ini ditunjukkan pada Gbr 1



Gbr. 1 *Single Instruction Multiple Data streams* (Sumber :El-Rewini & Abd-El-Barr, 2005)

MIMD (*Multiple Instruction Multiple Data streams*) adalah teknik *parallel* yang menggunakan *multiple processor* dan *multiple memory* yang tergabung dalam sebuah koneksi. Pada MIMD setiap processor memiliki *control unit* sendiri yang dapat memberikan instruksi yang berbeda. Arsitektur dari MIMD akan ditunjukkan pada Gbr 2.



Gbr. 2 *Multiple Instruction Multiple Data streams* (Sumber : El-Rewini & Abd-El-Barr, 2005)

Implementasi *parallel processing* maka digunakan *multithreading* untuk mengimplementasikan *parallel processing* tersebut.

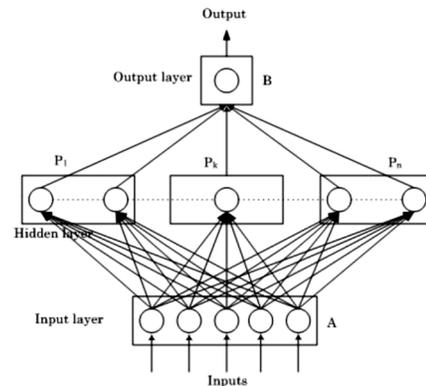
C. Thread & Multithreading

Thread adalah sebuah proses yang berukuran kecil yang dibuat oleh sebuah program untuk dijalankan bersamaan dengan *thread – thread* lainnya. Tujuan *thread* ini adalah agar proses – proses yang dapat dikerjakan bersamaan dapat dijalankan secara bersamaan tanpa memerlukan waktu tunggu untuk proses berikutnya. Ini dapat dicontohkan dari perangkat lunak pengolah kata yang membentuk beberapa *thread* untuk melakukan fungsi tampilan, proses pengetikan, dan proses pemeriksaan jumlah kata, dengan pembagian fungsi tersebut kedalam *thread* maka semua proses tersebut dapat dijalankan secara bersamaan tanpa terjadinya *delay*, sehingga dapat disebut berjalan secara *parallel*.

Multithreading adalah kumpulan beberapa *thread* yang dijalankan bersamaan pada sebuah *processor*. Pada *multithreading* ini merupakan proses eksekusi dari kumpulan *thread* dimana kumpulan *thread* tersebut diproses secara berulang – ulang dengan perpindahan dalam waktu *nanosecond*.

D. Parallel Arsitektur Neural Network

Parallel arsitektur neural network mengikuti penelitian dari Ganeshamoorthy & Ranasinghe, 2008. *Parallel* yang digunakan adalah *Hybrid partition* pada metode *parallel* ini tidak membutuhkan siklus pembelajaran per pola hanya melakukan penyimpanan nilai - nilai *weight* pada masing – masing neuron *hidden* unit. Bentuk arsitektur dari *parallel neural network* tersebut ditunjukkan pada Gbr. 3



Gbr. 3 Arsitektur *NeuralNetwork* dengan *parallel* (Ganeshamoorthy & Ranasinghe, 2008)

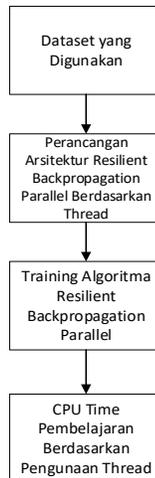
Gbr 3 adalah sebuah teknik *parallel* yang bernama *hybrid partition* yaitu membagi arsitektur proses pada *neuralnetwork* menjadi subproses – subproses. Subproses tersebut dikerjakan pada waktu yang bersamaan dengan menggunakan teknik *multithreading*. Proses yang akan dilakukan adalah dari *inputlayer* (A) akan mengirimkan data *input* yang telah dikalikan dengan *weight* ke neuron (P1) dan nilai tersebut akan dihasilkan nilai function yang akan diteruskan ke *output*

(B). Proses di *hiddenlayer* yaitu P1, Pk, dan Pn dijalankan bersamaan dengan memproses nilai dari A dan diteruskan ke B.

III. METODE PENELITIAN

A. Tahapan – Tahapan Penelitian

Tahapan – tahapan penelitian bertujuan penelitian yang dilaksanakan tidak keluar dari jalur yang ditentukan. Tahapan penelitian yang dilakukan dijabarkan pada Gbr. 3.1.



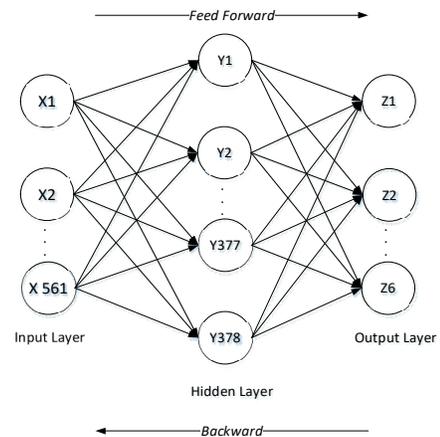
Gbr. 4 Diagram Tahapan – Tahapan Penelitian

B. Data yang Digunakan

Dataset adalah kumpulan data yang digunakan untuk pembelajaran dan ujicoba pada *artificialneural network*. Dataset yang digunakan berasal dari UCI Machine Learning dengan data HAR (Human Activity Recognition). Dataset HAR ini berisi data – data sensor dari accelerometer dan gyroscope menggunakan smartphone jenis Samsung Galaxy S II yang diletakan pada pinggang. Penelitian yang dilakukan menggunakan data dari sensor *accelerometer* dan *gyroscope*. Dataset HAR terdiri dari dua jenis data sensor dan data yang diambil adalah sensor accelerometer dan gyroscope, pada dataset ini telah disediakan dataset untuk training dan testing.

Dataset sensor accelerometer dan gyroscope didapatkan tiga buah nilai *axial* yaitu nilai x, y, dan z. Jumlah kasus pada masing – masing nilai axial tersebut adalah 7352 kasus. Kasus tersebut adalah aktifitas yang dilakukan oleh *volunteers* berjumlah 30 *volunteers*. Aktifitas yang dilakukan oleh *volunteers* yaitu berjalan, menaiki tangga, menuruni tangga, duduk, berdiri, dan tidur. Dari setiap aktifitas yang dilakukan tersebut menghasilkan nilai axial – nya yaitu x, y, dan z. Nilai triaxial tersebut diekstrak menjadi 561 feature data.

C. Rancangan Arsitektur *Resilient BackpropagationParallel*
 Rancangan arsitektur resilient backpropagation parallel dibagi menjadi tiga bagian utama yaitu input layer, hidden layer, dan output layer. Pada masing – masing layer ada memiliki node, pada input layer node yang dimiliki sebanyak 561 input, hidden layer memiliki node sebanyak 378 node, dan output layer memiliki 6 node. Gambar arsitektur resilient backpropagation adalah sebagai berikut



Gbr. 5 Arsitektur Neural Network Resilient Backpropagation

Arsitektur neural network pada bagian hidden layer node dibagi menjadi beberapa bagian yang akan diproses secara bersamaan menggunakan teknik parallel multithreading. Pada pembagian node pada hidden layer untuk penelitian ini dijelaskan pada tabel berikut ini :

TABEL I
 PEMBAGIAN NEURON HIDDEN LAYER BERDASARKAN JUMLAH THREAD

Thread yang terbentuk	Hidden Neuron pada sebuah Thread	Total Neuron pada HiddenLayer
1	378	378
2	189	378
3	126	378
4	94	378
5	75	378
6	63	378
7	54	378

D. CPU Time untuk *ParallelProcessing*

Analisis CPU Time jaringan arsitektur *neuralnetworkparallelprocessing* ini difokuskan pada setiap iterasi yang diproses oleh jaringan *neuralnetwork*. Hasil CPU Time yang dihasilkan adalah waktu per iterasi, dan total CPU Time pembelajaran akan didapatkan dari total dari seluruh iterasi.

$$T = \sum_{i=1}^n I_i$$

T adalah total waktu jaringan *neuralnetwork*. I_i adalah waktu iterasi. Waktu yang dihasilkan sebanyak 7 total waktu dari masing – masing *thread* yang diuji. Pengujian yang dilakukan adalah sebanyak 3 kali ini dilakukan karena hasil CPU Time yang dihasilkan tidaklah sama pada setiap waktu karena tergantung pada proses komputer tersebut, sehingga harus dilakukan tiga kali pengujian untuk memastikan jumlah *thread* yang cepat melakukan pembelajaran *neuralnetwork*. Pengujian yang dilakukan sebanyak tiga kali maka harus dicari rata – rata nilai tersebut dengan menggunakan rumus sebagai berikut:

$$R = \frac{\sum_{i=1}^3 T_i}{3}$$

RT adalah merupakan rata – rata waktu, dan $\sum_{i=1}^3 T_i$ merupakan jumlah total dari 3 pengujian tersebut dan dibagi dengan banyak pengujian tersebut dilakukan yaitu 3.

IV. HASIL DAN PEMBAHASAN

A. Hasil

Pengujian dilakukan sebanyak tujuh kali pengujian berdasarkan jumlah *thread* yang telah ditentukan pada metode penelitian. Pengujian yang dilakukan ini berdasarkan banyak *thread* yang digunakan pada arsitektur *neuralnetwork*. Pengujian yang diberikan dilakukan sebanyak tiga kali testing untuk mendapatkan waktu rata – rata pada ketujuh pengujian tersebut.

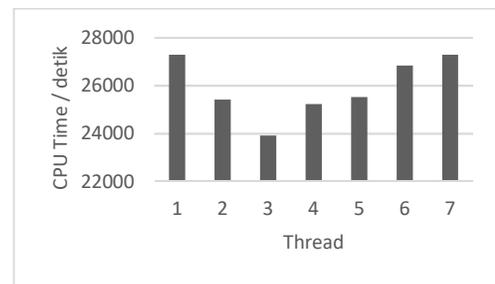
Hasil waktu dari algoritma *resilientbackpropagation* dengan menggunakan teknik *parallelprocessing* menggunakan *threading* ditampilkan dalam bentuk tabel, hasil pengujian dilakukan sebanyak tiga kali sehingga ditampilkan tiga bentuk tabel pengujian dengan bentuk pengujian yang sama yaitu pada setiap tabel akan menampilkan tujuh bagian yang diselesaikan oleh masing – masing *thread* bertujuan untuk mengetahui waktu yang diselesaikan pada masing – masing *thread* dan dibandingkan *thread* manakah yang memiliki waktu proses yang paling cepat selesai.

Pengujian pada pertama kali *resilientbackpropagation* disajikan dalam Tabel 2 sebagai lanjutannya.

TABEL II
HASIL PENGUJIAN I *RESILIENT BACKPROPAGATION PARALLEL MULTITHREADING*

Thread	Waktu Selesai/detik	Rata – Rata Waktu Iterasi/detik
1	27293.694772998733	1.3646847386499366
2	25413.755037002018	1.270687751850101
3	23914.01629200012	1.195700814600006
4	25231.32637700154	1.261566318850077
5	25519.277693998476	1.2759638846999237
6	26835.739719997993	1.3417869859998997
7	27292.983101998594	1.3646491550999298

Tabel 2 menunjukkan pada pengujian pertama kali mendapatkan hasil yaitu menggunakan 3 buah *thread* pada arsitektur *neuralnetworkresilientbackpropagation* pada dapat diselesaikan dalam waktu 23914,01629200012 / detik atau 6,643782 jam. Sedangkan pada pada proses yang menggunakan satu buah *thread* dan tujuh buah *thread* menunjukkan hasil yang hampir sama yaitu 27293,694772998733 / detik untuk satu buah *thread* dan 27292,983101998594 / detik untuk tujuh buah *thread*, hanya menunjukkan perbedaan 1 detik saja. Pada tbl 2 pada pengujian pertama ini menunjukkan terjadi kecepatan penyelesaian yang dimulai dari penggunaan satu buah *thread* sampai tiga buah *thread* tapi setelah empat buah *thread* menunjukkan perlambatan waktu penyelesaian dan ini terjadi pada terus pada lima buah *thread* sampai tujuh buah *thread*. Untuk hasil grafik dari peningkatan kecepatan pada penggunaan *thread* ditunjukkan pada Gbr. 6



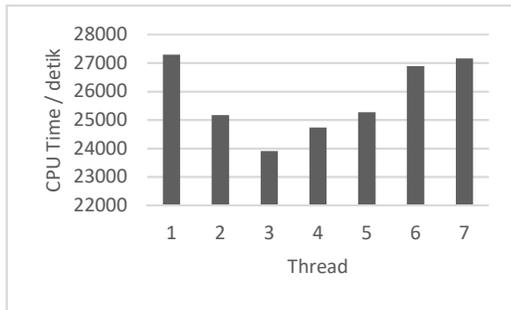
Gbr. 6 Grafik Peningkatan Kecepatan Waktu Berdasarkan Peningkatan Jumlah *Thread* Pengujian I

Pada pengujian yang kedua menunjukkan hasil sama pada pengujian pertama yaitu 3 buah *thread* cepat dalam penyelesaian tapi berbeda waktu penyelesaiannya pada pengujian yang pertama yaitu pada pengujian yang pertama menunjukkan CPU time yang diselesaikan 23914,01629200012 / detik pada pengujian yang kedua waktu selesai 23910,218218001297 /

detik, lebih cepat 4 detik dalam penyelesaian waktunya. Hasil dari pengujian yang kedua ini ditampilkan dalam Tbl 3 dan untuk grafik ditampilkan pada Gbr. 7

TABEL III
HASIL PENGUJIAN II RESILIENT BACKPROPAGATION PARALLEL MULTITHREADING

Thread	Waktu Selesai / detik	Rata – Rata Waktu Iterasi/detik
1	27290.439604999618	1.3645219802499808
2	25172.112700000416	1.2586056350000208
3	23910.218218001297	1.1955109109000648
4	24732.658940000627	1.2366329470000312
5	25274.041188002	1.2637020594001
6	26885.634780998025	1.3442817390499013
7	27160.287975998206	1.3580143987999103



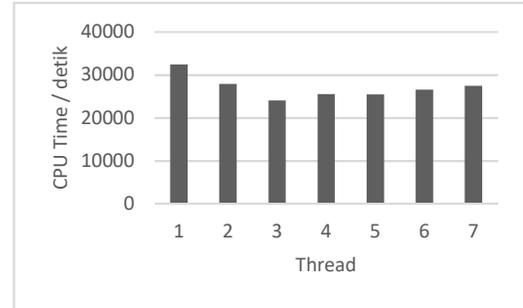
Gbr. 7 Grafik Peningkatan Kecepatan Waktu Berdasarkan Peningkatan Jumlah Thread Pengujian II

Pada pengujian yang ketiga menampilkan hasil sama yaitu dengan menggunakan 3 buah *thread* menunjukkan kecepatan waktu penyelesaian yang lebih cepat daripada *thread* lainnya, tetapi waktu penyelesaian berbeda yaitu pengujian ketiga waktu diselesaikan dalam 24067.489326000406 / detik, pengujian kedua waktu selesai 23910.218218001297 / detik, dan pengujian pertama waktu selesai 23914.01629200012 / detik. Hasil pengujian ini akan ditampilkan pada Tbl 4 dan untuk grafik waktu akan ditampilkan pada Gbr. 8.

TABEL IV
HASIL PENGUJIAN III RESILIENT BACKPROPAGATION PARALLEL MULTITHREADING

Thread	Waktu Selesai/detik	Rata – Rata Waktu Iterasi/detik
1	32389.488616999854	1.6194744308499927
2	27906.764599999868	1.3953382299999935
3	24067.489326000406	1.2033744663000203
4	25553.457462000133	1.2776728731000067

5	25500.42951299844	1.2750214756499219
6	26590.765166998073	1.3295382583499036
7	27500.32026299811	1.3750160131499054



Gbr. 8 Grafik Peningkatan Kecepatan Waktu Berdasarkan Peningkatan Jumlah Thread Pengujian II

Hasil pengujian pada Tbl 2, Tbl 3, dan Tbl 4 menunjukkan hasil waktu yang bervariasi. Karena hasil yang bervariasi maka ketiga pengujian tersebut di-average waktunya selesai dari hasil pengujian tersebut.

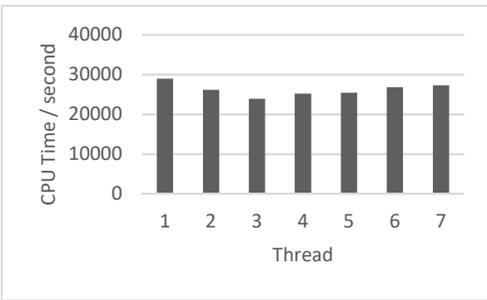
$$R = \frac{\sum_{i=1}^n W_i}{n}$$

Hasil rata – rata pengujian tersebut ditunjukkan pada Tabel 5 sebagai berikut

TABEL V
HASIL PENGUJIAN RATA – RATA
RESILIENT BACKPROPAGATION PARALLEL MULTITHREADING

Thread	Waktu Selesai / detik	Rata – Rata Waktu Iterasi / detik
1	28991,20766	1,449560383
2	26164,21078	1,308210539
3	23963,90795	1,198195397
4	25172,48093	1,258624046
5	25431,24946	1,271562473
6	26770,71322	1,338535661
7	27317,86378	1,365893189

Pada pengujian pertama sampai pengujian ketiga bahwa dengan menggunakan 3 *thread* pada arsitektur *neuralnetwork* menunjukkan waktu proses pembelajaran pada jaringan tersebut lebih dibandingkan dengan enam pengujian *thread* lainnya. Pada penggabungan waktu berfungsi untuk melihat hasil rata – rata waktu penyelesaian masing – masing *thread* pada tiga pengujian tersebut. Hasil tampilan grafik waktu rata – rata waktu tersebut ditampilkan pada Gambar 4.4.



Gbr. 9 Grafik Rata – Rata Waktu Selesai Pembelajaran Resilient Backpropagation

Hasil penelitian yang telah dilakukan pada bagian diatas menunjukkan 3 buahthread membantu dalam mempercepat waktu training, dengan jumlah neuroan pada hiddenlayer sebanyak 378 neuron.

B. Pembahasan

Hasil pengujian yang telah dilakukan mendapatkan bahwa jaringan *neuralnetwork* yaitu *resilientbackpropagation* dengan menambahkan teknik *parallelprocessing* yang diimplementasikan menggunakan *multithreading* menunjukkan membantu kinerja pembelajaran dari jaringan *neuralnetworkresilientbackpropagation*. Ini ditunjukkan dari hasil rata – rata waktu pengujian kecepatan pembelajaran yaitu penurunan waktu ini menunjukkan kecepatan pembelajaran *thread* tiga lebih cepat dari pada *thread* satu dan *thread* dua. Tetapi *thread* empat menunjukkan waktu yang berangsur – angsur naik sampai *thread* ke tujuh, ini menunjukkan waktu pembelajaran menjadi lebih lambat.

Penurunan kecepatan yang dimulai dari penggunaan empat buah *thread* ini dapat disebabkan oleh kemampuan *hardware* yang digunakan dan pergantian antar *thread* yang terjadi. Penggunaan *hardware* yaitu Core i3 dengan dua *core* pada masing – masing *core* dapat menjalankan duathread sekaligus, karena kemampuan ini dapat memproses empatthread pada waktu yang bersamaan. Proses pada jaringan *neuralnetwork* menggunakan *parallelprocessing* dengan teknik *multithreading*, pada satuthread maka yang terjadi ada adanya penggunaan duathread yaitu satuthread untuk memproses semua *neuron* pada *hidden layer* dan satuthread sebagai melakukan pengecekan terhadap proses *neuron* pada *hidden layer* apakah sudah selesai semua. Pada saat duathread maka yang terjadi adalah pembentukan sebanyak duathread untuk memproses *neuron* pada *hidden layer* dan satuthread sebagai pengecekannya. Pada saat tigathread maka yang berjalan adalah empatthread yaitu tigathread memproses *neuron* pada *hidden layer* dan satuthread sebagai pengecekan. Pada setiap *thread* yang digunakan maka ditambah satu *thread* sebagai

pengecekan proses pada *hidden layer*. Oleh sebab itu pemanfaatan *thread* yang paling optimal adalah tigathread karena penggunaan seluruh kemampuan *hardware*. Tetapi pada saat penggunaan empatthread maka yang terjadi adalah pengantian proses antar *thread* yaitu *thread* diproses secara bergantian pada *hardware* secara berulang – ulang sampai proses tersebut selesai. Oleh sebab itu terjadi kenaikan waktu proses karena perpindahan antar *thread* tersebut.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Kesimpulan dari hasil penelitian yang dilakukan yaitu:

1. Parallel processing menggunakan multithreading yang diterapkan ke dalam arsitektur *resilient backpropagation* membantu kinerja waktu pembelajaran *resilient backpropagation* dengan hanya memanfaatkan 3 *thread* pada kasus pengenalan aktifitas manusia.
2. Penggunaan lebih dari 3 *thread* pada arsitektur *resilient backpropagation* menyebabkan penurunan waktu pembelajaran yaitu lebih lambat, ini disebabkan oleh terjadi proses perpindahan antar *thread* sehingga terjadi penambahan waktu pada setiap perpindahan *thread* tersebut.

B. Saran

Saran yang diberikan oleh penulis adalah sebagai berikut:

1. Penggunaan *parallel* dengan *multithreading* terhadap arsitektur *resilient backpropagation* harus memperhatikan jumlah *thread* yang dapat diproses pada waktu bersamaan pada sebuah CPU karena ini berpengaruh terhadap eksekusi *thread* yang dapat menyebabkan waktu delay pada perpindahan antar *thread* tersebut.
2. Untuk meningkatkan kemampuan *parallel processing* pada arsitektur *resilient backpropagation* dapat menggunakan *multiprocessor* yaitu menggunakan beberapa *processor* pada waktu bersamaan.

REFERENCES

- [1] K. Ganeshamoorthy and D. N. Ranasinghe, "On the Performance of Parallel Neural Network Implementations on Distributed Memory Architectures," in *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, Lyon, 2008.
- [2] R. Gu, F. Shen and Y. Huang, "A Parallel Computing Platform for Training Large Scale Neural Networks," in *Big Data, 2013 IEEE International Conference on*, Silicon Valley, CA, 2013.
- [3] M. Moreira and E. Fiesler, "Neural Networks with Adaptive Learning Rate and Momentum Terms," *IDIAP Technical Report*, 1995.
- [4] J. Torresen, *Parallelization of Backpropagation Training fo Feed - Forward Neural Networks*, Norwegian: The University of Trondheim, 1996.

- [5] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning : The RPROP Algorithm," in *IEEE International Conference*, San Francisco, CA, 1993.
- [6] S. A. Halim, A. Ahmad, N. M. Noh, M. S. B. M. A. Safudin and R. Ahmad, "A Comparative Study between Standard Back Propagation and Resilient Propagation," in *IT in Medicine and Education (ITME), 2011 International Symposium on*, Cuangzhou, 2011.
- [7] N. Prasad, R. Singh and S. P. Lal, "Comparison of Back Propagation and Resilient Propagation Algorithm for Spam Classification," in *2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation*, Seoul, 2013.
- [8] J. Stubbemann, O. Kramer and N. A. Treiber, "Resilient Propagation for Multivariate Wind Power Prediction," 2015. [Online]. Available: http://www.researchgate.net/profile/Oliver_Kramer2/publication/268966338_Resilient_Propagation_for_Multivariate_Wind_Power_Prediction/links/55360e290cf20ea35f10f924.pdf.