JURNAL
TEKNOLOGI INFORMASI
DAN KOMUNIKASI

Vol.6 No. 2, Desember 2017 : 80 - 91

PERBANDINGAN BEBERAPA MODEL UNTUK KINERJA ALGORITMA BACKPROPAGATION

COMPARISON OF SOME MODEL FOR PEFORMANCE IMPROVEMENT IN BACKPROPAGATION ALGORITHM

Mulia Dhamma[1], Erna Budhiarti Nababan[2], Muhammad Zarlis[3]
[1, 2, 3]Program Studi Ilmu Komputer,
Fakultas Ilmu Komputer Dan Teknologi Informasi,
Universitas Sumatera Utara,
Medan, Indonesia, 20155
mulia.dhamma@students.usu.ac.id[1], ernabrn@usu.ac.id[2], m.zarlis@usu.ac.id[3]

Diterima : 11 Oktober 2017 Direvisi : 18 November 2017 Disetuji : 29 November 2017

ABSTRAK

Backpropagation merupakan salah satu metode dari jaringan syaraf tiruan yang bisa digunakan untuk memprediksi harga saham. Backpropagation memiliki kelemahan pada proses pelatihannya yang lambat. Penelitian ini bertujuan untuk memperbaiki kinerja dari algoritma backpropagation dengan cara membagi data secara acak ke sejumlah thread. Data yang dibagikan akan dinormalisasi terlebih dahulu dengan metode min-max yang memiliki nilai 0 sampai 1. Proses pelatihan akan ditingkatkan lagi dengan merubah nilai bobot dan bias pada thread yang berjalan lambat pada setiap epoch yang ditentukan. Penelitian ini akan menggunakan 8 jenis model yang berbeda di jumlah thread dan epoch untuk perubahan nilai bobot dan bias ke thread yang lambat. Setiap model yang diteliti akan dilakuan pengujian sebanyak 5 kali. Berdasarkan penelitian dengan menggunakan 239 data, peneliti mendapatkan perbandingan antara model 1&2 terjadi pengecilan epoch di 5 pengujian, model 3&4 terjadi pada 4 pengujian, model 5&6 terjadi pada 2 pengujian, model 7&8 terjadi pada 4 pengujian dimana masing – masing perbandingan melakukan perubahan nilai bobot ke thread lain dengan jumlah epoch yang berbeda - beda

Kata Kunci *: Jaringan Syaraf Tiruan, Backpropagation, Thread, Normalisasi Min-Max, Epoch.*

ABSTRACT

Backpropagation is one of artificial neural network that can be used to predict stock price. Backpropagation has a problem in learning for a data. The learning is slow. This research is intended to improve the performance of backpropagation algorithm by separate the data to the existing thread. The separated data will be normalized using min-max normalization with a range value from 0 – 1. The process of learning will be improved by updating the slowest thread for the weight and bias value. This research consists of 8 models with different number of thread and epoch. Each model has 5 times experiment. Base on the research for 239 data, research found epoch reduce on 5 experiment in model 1&2, 4 experiment in model 3&4, 2 experiment in model 5&6, 4 experiment in model 7&8. All comparison has its own epoch number to change the weight to the other thread.

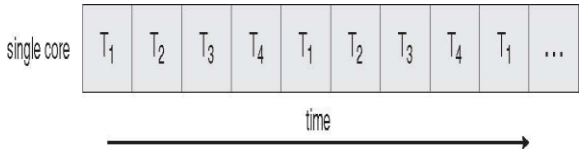
Keywords *: Neural Network, Backpropagation, Thread, Min-Max Normalization, Epoch.*

PENDAHULUAN

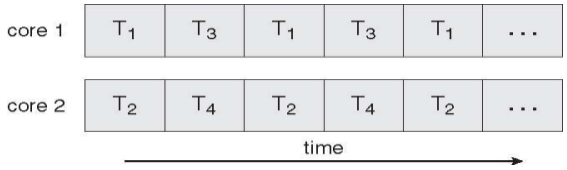
Backpropagation merupakan salah satu dari banyaknya algoritma jaringan syaraf tiruan (JST) yang dapat digunakan untuk memprediksi dengan cara mempelajari data secara terbimbing. *Backpropagation* memiliki 3 proses *layer* yang dimulai dari *input layer* menuju ke *hidden layer* dan yang terakhir ke *output layer*. *Backpropagation* dapat juga digunakan untuk memprediksi harga saham yang mengartikan bahwa *backpropagation* mempelajari data yang terdahulu dan tidak beraturan. Pembelajaran dengan nilai bobot dan bias yang tepat di mana nantinya akan dikalkulasi ke nilai *learning rate* akan mempengaruhi performa *backpropagation*. *Error* yang dihasilkan oleh *backpropagation* juga akan mempengaruhi kecepatan dari pelatihannya karena hasil error yang didapatkan akan mempengaruhi nilai bobot dan bias yang baru yang terjadi pada setiap *epoch*. *Epoch* disebut sebagai suatu rangkaian langkah dalam pembelajaran JST, oleh karena itu 1 *epoch* akan mengartikan bahwa telah terjadinya pembelajaran sebanyak 1 kali. Faktor lain yang akan mempengaruhi performa *backpropagation* adalah jumlah data yang diberikan untuk dilatih. Rodriguez [1] mengungkapkan teknik *parallel processing* adalah salah satu dari banyaknya teknik yang bisa digunakan untuk meningkatkan performa *backpropagation* karena teknik ini memproses menggunakan komputer yang memiliki prosesor lebih dari satu.

Thread

Multithreading adalah suatu proses eksekusi yang dapat melakukan banyak proses pada suatu waktu. Silberschatz [6] mengatakan *multithreading* memiliki 2 konsep yaitu *concurrency* dan *parallelism*. *Concurrency* adalah konsep *multithreading* di mana proses bergantung pada 1 prosesor saja. *Parallelism* adalah konsep *multithreading* yang melakukan banyak proses menggunakan lebih dari 1 prosesor. Gambar 1 merupakan contoh proses eksekusi *thread* dengan sistem yang hanya terdapat 1 *core* dimana *sistem* secara bergantian mengganti proses eksekusi *thread* yang satu dengan yang lainnya. Gambar 2 merupakan contoh eksekusi *thread* dengan 2 core dimana sistem secara otomatis membagi eksekusi thread ke 2 tempat/*core* sehingga tugas untuk setiap *core* tidak begitu banyak seperti yang ada pada Gambar 1.



Gambar 1. Eksekusi Sistem Dengan 1 *Core*



Gambar 2. Eksekusi Sistem Dengan *Core* Lebih Dari 1

Thread Model

Thread dapat digunakan untuk tingkat *user* yaitu *user thread* ataupun tingkat *kernel* yaitu *kernel thread*. User *thread* dijalankan diatas *kernel* dan diatur tanpa bantuan *kernel* di mana *kernel thread* dijalankan dan diatur oleh sistem operasi secara langsung termasuk system operasi *Windows, Linux, Os x* dan *Solaris*.

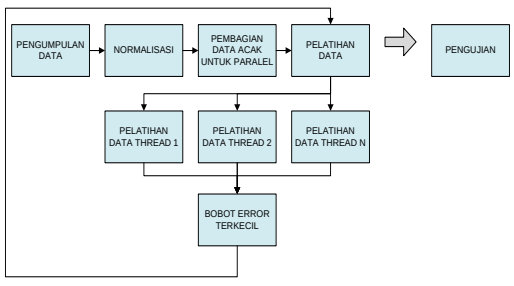
Penelitian Terdahulu

Rangkuman peneliti terdahulu yang menggunakan paralel dan normalisasi seperti yang diteliti oleh Krpan dkk [2], Rodriguez [1], Tsaregorodev [3] dan chamidah [4] dijadikan sebagai landasan untuk melakukan penelitian dengan membagi proses pelatihan yang tidak hanya bergantung pada satu proses saja. Tetapi juga bisa melakukan banyak proses dengan cara perpindahan proses thread yang satu ke yang lainnya dalam waktu yang sangat cepat dan dalam hal ini peneliti mencoba membagi data ke beberapa proses untuk dilatih. Peneliti peneliti tersebut adalah Chamidah [4] yang mendapatkan jumlah rata – rata *epoch* yang

dihasilkan *backpropagation* antara lain *decimal scaling* = 40, *sigmoid* = 584, *softmax* = 49, *min-max* = 21, *statistical column* = 45 dan *z-score* = 38. Jumlah rata – rata *epoch* terkecil diantara 6 metode normalisasi adalah metode normalisasi *min-max*. Krpan dkk [2] mengatakan pembelajaran *backpropagation* dengan paralel akan lebih efisien apabila jumlah jaringan yang digunakan banyak. Rodriguez [1] mengatakan pembelajaran *backpropagation* akan menghabiskan banyak waktu pada setiap iterasinya dan *GPU parallelization* dapat mengurangi waktu tersebut. Tsaregorodev [3] mengatakan penggunaan paralel dapat meningkatkan kecepatan 1.5 lebih cepat dibandingkan dengan penggunaan 1 *thread* dan 1.2 – 1.6 apabila jumlah jaringan dan datanya banyak.

METODE PENELITIAN

Backpropagation mempunyai kelemahan dalam melakukan pelatihan secara terbimbing sehingga untuk memperbaiki performanya peneliti mencoba menggunakan pembagian data ke sejumlah *thread* yang ditentukan agar pelatihan bisa berjalan tanpa harus bergantung pada 1 proses saja dan juga penggantian nilai bobot dan bias setiap 5 *epoch* sekali. *Error* terkecil yang dihasilkan oleh masing – masing *thread* akan dipilih untuk menggantikan nilai bobot dan bias pada *thread* lainnya. Penelitian ini akan memiliki tahapan dari pengumpulan data, normalisasi data, penyusunan data, pelatihan data dan pengujian data seperti yang ditunjukan pada Gambar 3.



Gambar 3. Tahapan Penelitian

Data yang digunakan untuk penelitian ini adalah nilai harga saham perusahaan adaro energy yang didapatkan dari website www.finance.yahoo.com dengan jumlah data sebanyak 239 dari januari – desember 2016. Data tersebut akan dinormalisasi dengan menggunakan metode normalisasi *min-max* dengan rumus (1).

$$x' = \frac{(x-a)}{b-a} \qquad (1)$$

Keterangan :
x' = Hasil Normalisasi
x = Data Awal
a = Nilai Minimal
b = Nilai Maksimal

Metode normalisasi *min-max* adalah metode yang menskalakan data dari suatu *range* ke *range* baru lain yaitu 0 sampai dengan 1 seperti yang dikatakan oleh chamidah [4]. Data yang telah dinormalisasi disebar secara acak ke sejumlah *thread*. Setiap *thread* yang sudah berisi data akan dijalankan proses pelatihannya untuk mencari nilai bobot yang tepat untuk melakukan pengujian pada data baru.

Tabel 1. Nilai Harga Saham Adaro Energy Company

No	Tanggal	X1	X2	X3	Y
1	2016-12-30	1660	1705	1660	1695
2	2016-11-29	1680	1700	1630	1630
3	2016-10-19	1485	1545	1485	1530
4	2016-09-16	1180	1195	1155	1165
⋮	⋮	⋮	⋮	⋮	⋮
239	2016-01-06	499	510	495	500

www.finance.yahoo.com

Penelitian ini dilakukan dengan mencoba 8 jenis model yang berbeda seperti yang ditunjukan pada Tabel 2 di mana masing – masing model akan dicoba diberikan jumlah *thread* yang berbeda dan juga jumlah *epoch*nya dalam pengambilan nilai error terkecil yang akan

digunakan untuk melakukan perubahan nilai bobot dan bias seperti yang ditunjukan pada Tabel 2.

Model 1 dan 2 akan dibandingkan untuk melihat perbedaan waktu antara pelatihan yang menggunakan 1 *thread* dengan 2 *thread* yang menukar nilai bobot ke *thread* lainnya setiap 1 *epoch* sekali. Model 3 dan 4 akan dibandingkan untuk melihat adanya perbedaan pada penggunaan jumlah *thread* yang sama sebanyak 4 buah tetapi perubahan nilai bobot ke *thread* lainnya setiap 1 *epoch* sekali untuk model 3 dan 5 *epoch* sekali untuk model 4. Model 5 dan 6 akan dibandingkan untuk melihat adanya perbedaan pada penggunaan jumlah *thread* yang sama sebanyak 5 buah tetapi perubahan nilai bobot ke *thread* lainnya setiap 1 *epoch* sekali untuk model 5 dan 5 *epoch* sekali untuk model 6. Model 7 dan 8 akan dibandingkan untuk melihat adanya perbedaan pada penggunaan jumlah *thread* yang sama sebanyak 6 buah tetapi perubahan nilai bobot ke *thread* lainnya setiap 1 *epoch* sekali untuk model 7 dan 5 *epoch* sekali untuk model 8.

Semua model yang akan diteliti akan diberikan nilai bobot & nilai bias acak yang berkisar diantara -0.5 sampai dengan 0.5 di mana v adalah nilai bobot yang akan berjalan dari *input layer* menuju *hidden layer* seperti persamaan (2).

$$z\_in_j' = b1 + \sum_{i=1}^{n} x_i v_{ij} \qquad (2)$$

w adalah bobot yang akan berjalan dari *hidden layer* menuju *output layer* seperti persamaan (3).

$$y\_in_j' = b2 + \sum_{j=1}^{p} z_j w_{jk} \qquad (3)$$

b1 adalah nilai bias yang ada pada *hidden layer* dan b2 adalah nilai bias yang ada pada *output* layer seperti yang dijelaskan oleh Setiawan [5]. Jumlah *neuron* di bagian *input layer* ada sebanyak 3 buah *input neuron*, bagian *hidden layer* ada sebanyak 4 buah *hidden neuron* dan bagian *output layer* ada sebanyak 1 *output neuron*. Nilai *error* terkecil untuk menghentikan pelatihannya adalah 0.001 beserta dengan jumlak *epoch* maksimalnya yang sebanyak 100000 *epoch*. Nilai *learning rate/alfa* untuk penelitian ini adalah 1.

Tabel 2. 8 Jenis Model Penelitian Dengan Attribut Yang Berbeda

Model	Jumlah Thread	Pe ngambilan Nilai Error Terkecil
1	1	Setiap 1 *epoch* sekali
2	2	Setiap 1 *epoch* sekali
3	4	Setiap 1 *epoch* sekali
4	4	Setiap 5 *epoch* sekali
5	5	Setiap 1 *epoch* sekali
6	5	Setiap 5 *epoch* sekali
7	6	Setiap 1 *epoch* sekali
8	6	Setiap 5 *epoch* sekali

Nilai yang didapatkan dinormalisasi dengan metode *min-max* agar bisa memperkecil jumlah *epoch* seperti yang telah diteliti oleh Chamidah [4]. Nilai terendah yang didapatkan sebelum dinormalisasi dengan metode *min-max* adalah 437 dan nilai tertinggi yang didapatkan adalah 1850 seperti yang ada pada Tabel 1.

Parameter - parameter yang ada pada Tabel 1 terdiri dari X1 = open, X2 = high, X3 = low dan Y = target. Hasil dari normalisasi dapat dilihat pada Tabel 3.

Hasil normalisasi bisa dilihat pada Tabel 3 untuk data No. 1 yang terdapat pada kolom X1 di mana harga sebelum dinormalisasi adalah 1660 dikurang 437 (Nilai Terendah Dari Semua Kumpulan Harga Saham Tahun 2016) dibagi dengan 1850 (Nilai Tertinggi Dari Semua Kumpulan Data Harga Saham Tahun 2016) – dengan 437 untuk nilai terendah. Proses perhitungan normalisasi untuk nilai terendah dan tertinggi juga dilakukan untuk kolom X2, X3 dan Y.

Tabel 3. Hasil Normalisasi Dengan *Min-Max*

No.	Tanggal	X1	X2	X3	Y
1	2016-12-30	$x' = \frac{(1660-437)}{1850-437}$	$x' = \frac{(1705-437)}{1850-437}$	$x' = \frac{(1660-437)}{1850-437}$	$x' = \frac{(1695-437)}{1850-437}$
2	2016-11-29	0.8797	0.8938	0.8443	0.8443
3	2016-10-19	0.7417	0.7841	0.7417	0.7735
4	2016-09-16	0.5258	0.5364	0.5082	0.5152
⋮	⋮	⋮	⋮	⋮	⋮
239	2016-01-06	0.0439	0.0517	0.0410	0.0446

Semua data yang berjumlah sebanyak 239 yang sudah dinormalisasi akan disebarkan secara acak ke sejumlah *thread* yang ada. Data akan disebarkan secara merata ke setiap *thread* yang ada. Apabila jumlah data yang disebarkan ke sejumlah *thread* tidak memiliki jumlah data yang sama maka *thread* yang kekurangan data tersebut akan mengambil data dari *thread* lainnya sehingga setiap *thread* akan memiliki jumlah data yang sama.

Proses pelatihan pengambilan nilai error terkecil di antara setiap *thread* yang melakukan perubahan nilai bobot dan bias setiap 1 *epoch* sekali adalah model 1, 2, 3, 5, 7 dapat dilihat pada Tabel 4 dan perubahan nilai bobot dan bias setiap 5 *epoch* sekali adalah model 4, 6, 8 dapat dilihat pada Tabel 5.

Pada Tabel 4 bisa dilihat *epoch* ke 1 error terkecil yang didapatkan pada kolom T1 sesudah adalah 0.099, kolom T2 sesudah adalah 0.026, kolom T3 sesudah adalah 0.096 dari perbandingan tiga thread yaitu T1, T2 dan T3 dan nilai *error* terkecil yang didapatkan adalah 0.026 yang ada pada kolom T2 sesudah. *Thread* T2 dipilih karena *thread* ini menghasilkan nilai *error* terkecil yaitu 0.026, semua nilai bobot dan bias yang ada pada *thread* T2 akan diubah ke T1 dan T3. Untuk *epoch* ke 2 nilai T1, T2 dan T3 sebelum akan memiliki nilai bobot dan bias yang sama. Proses pelatihan pada *epoch* ke 2 akan sama dengan *epoch* ke 1 hingga proses pelatihannya berhenti apabila nilai *error* yang dihasil sudah dibawah nilai 0.001 seperti yang sudah ditentukan pada bagian metode penelitian sebelumnya. Pada Tabel 5 proses pelatihannya sama dengan proses pelatihan yang ada pada Tabel 4, namun perbedaanya pada perubahan nilai bobot setiap 5 *epoch* sekali. *Epoch* tersebut akan berjalan mulai dari *epoch* ke 1 lalu berlanjut ke *epoch* ke 6, 11 hingga proses pelatihannya berhenti apabila nilai *error* yang dihasilkan sudah dibawah nilai *error* yang ditentukan pada bagian metode penelitian sebelumnya.

Tabel 4. Perubahan Bobot Model 1, 2, 3, 5, 7

Epoch Ke	T1 Sebelum	T1 Sesudah	T2 Sebelum	T2 Sesudah	T3 Sebelum	T3 Sesudah
1	v_{11} =0.110	v_{11} =0.120	v_{11} =0.213	v_{11} =0.216	v_{11} =0.311	v_{11} =0.013
	v_{12} =0.121	v_{12} =0.132	v_{12} =0.214	v_{12} =0.217	v_{12} =0.313	v_{12} =0.014
	v_{13} =0.131	v_{13} =0.136	v_{13} =0.215	v_{13} =0.218	v_{13} =0.312	v_{13} =0.015
		e =0.099		e =0.026		e 0.096
	Epoch = 1	Epoch = 2	Epoch = 1	Epoch = 2	Epoch = 1	Epoch = 2
2	v_{11} =0.216	v_{11} =0.222	v_{11} =0.216	v_{11} =0.217	v_{11} =0.216	v_{11} =0.223
	v_{12} =0.217	v_{12} =0.221	v_{12} =0.217	v_{12} =0.219	v_{12} =0.217	v_{12} =0.226
	v_{13} =0.218	v_{13} =0.223	v_{13} =0.218	v_{13} =0.219	v_{13} =0.218	v_{13} =0.220
	e =0.026	e =0.021	e =0.026	e =0.024	e =0.026	e =0.027
	Epoch = 2	Epoch = 3	Epoch = 2	Epoch = 3	Epoch = 2	Epoch = 3
3	v_{11} =0.222	v_{11} =0.223	v_{11} =0.222	v_{11} =0.225	v_{11} =0.222	v_{11} =0.223
	v_{12} =0.221	v_{12} =0.225	v_{12} =0.221	v_{12} =0.226	v_{12} =0.221	v_{12} =0.223
	v_{13} =0.223	v_{13} =0.224	v_{13} =0.223	v_{13} =0.225	v_{13} =0.223	v_{13} =0.227
	e = 0.021	e = 0.019	e = 0.021	e = 0.018	e = 0.021	e = 0.020
	Epoch = 3	Epoch = 4	Epoch = 3	Epoch = 4	Epoch = 3	Epoch = 4
⋮		⋮	⋮	⋮	⋮	⋮

Akan terus melakukan pelatihan hingga error sudah lebih kecil dari yang ditentukan

Tabel 5. Perubahan Bobot Model 4, 6, 8

Epoch Ke	T1 Sebelum	T1 Sesudah	T2 Sebelum	T2 Sesudah	T3 Sebelum	T3 Sesudah
1	v_{11} =0.110	v_{11} =0.120	v_{11} =0.213	v_{11} =0.216	v_{11} =0.311	v_{11} =0.013
	v_{12} =0.121	v_{12} =0.132	v_{12} =0.214	v_{12} =0.217	v_{12} =0.313	v_{12} =0.014
	v_{13} =0.131	v_{13} =0.136	v_{13} =0.215	v_{13} =0.218	v_{13} =0.312	v_{13} =0.015
		e =0.099		e =0.026		e =0.096
	Epoch = 1	Epoch = 6	Epoch = 1	Epoch = 6	Epoch = 1	Epoch = 6
6	v_{11} =0.216	v_{11} =0.222	v_{11} =0.216	v_{11} =0.217	v_{11} =0.216	v_{11} =0.223
	v_{12} =0.217	v_{12} =0.221	v_{12} =0.217	v_{12} =0.219	v_{12} =0.217	v_{12} =0.226
	v_{13} =0.218	v_{13} =0.223	v_{13} =0.218	v_{13} =0.219	v_{13} =0.218	v_{13} =0.220
	e =0.026	e =0.021	e =0.026	e =0.024	e =0.026	e =0.027
	Epoch = 6	Epoch = 11	Epoch = 6	Epoch = 11	Epoch = 6	Epoch = 11
11	v_{11} =0.222	v_{11} =0.223	v_{11} =0.222	v_{11} =0.225	v_{11} =0.222	v_{11} =0.223
	v_{12} =0.221	v_{12} =0.225	v_{12} =0.221	v_{12} =0.226	v_{12} =0.221	v_{12} =0.223
	v_{13} =0.223	v_{13} =0.224	v_{13} =0.223	v_{13} =0.225	v_{13} =0.223	v_{13} =0.227
	e =0.021	e =0.019	e =0.021	e =0.018	e =0.021	e =0.020
	Epoch = 11	Epoch = 16	Epoch = 11	Epoch = 16	Epoch = 11	Epoch = 16
⋮		⋮	⋮	⋮	⋮	⋮

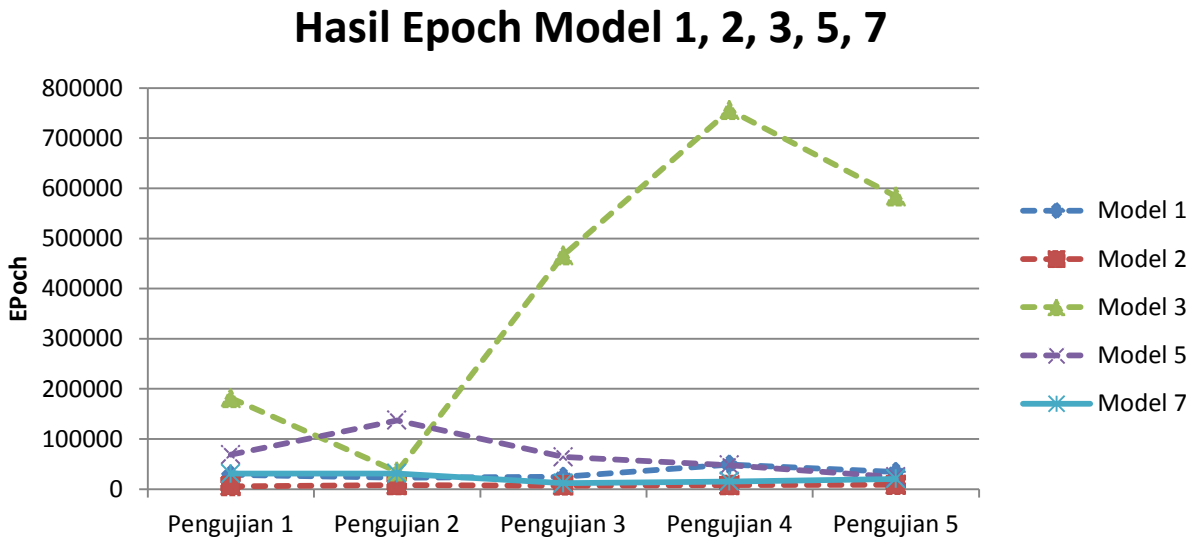
Akan terus melakukan pelatihan hingga error sudah lebih kecil dari yang ditentukan

HASIL DAN PEMBAHASAN

Berdasarkan hasil penelitian untuk model 1 & 2 *epoch* pada pengujian ke 1 berkurang 22511 *epoch*, ke 2 berkurang 15176, ke 3 berkurang 17780, ke 4 berkurang 40691 dan ke 5 berkurang 25619 yang juga dapat dilihat pada gambar 6. Model 3 & 4 *epoch* pada pengujian ke 1 berkurang 10799 *epoch*, ke 2 10, ke 3 berkurang 84450, ke 4 berkurang 15658 dan ke 5 bertambah 251347 yang juga dapat dilihat pada gambar 7. Model 5 & 6 *epoch* pada pengujian ke 1 bertambah 8082 *epoch*, ke 2 berkurang 27142, ke 3 bertambah 296, ke 4 berkurang 2211 dan ke 5 bertambah 8503 yang juga dapat dilihat pada gambar 8. Model 7 & 8 *epoch* pada pengujian ke 1 bertambah 7508 *epoch*, ke 2 berkurang 6689, ke 3 berkurang 3, ke 4 berkurang 5186 dan ke 5 berkurang 2893 yang juga dapat dilihat pada gambar 9. Hasil *epoch* yang dijelaskan sebelumnya untuk model 1, 2, 3, 4, 5, 6, 7, 8 untuk 5 pengujian dapat dilihat pada Tabel 6.

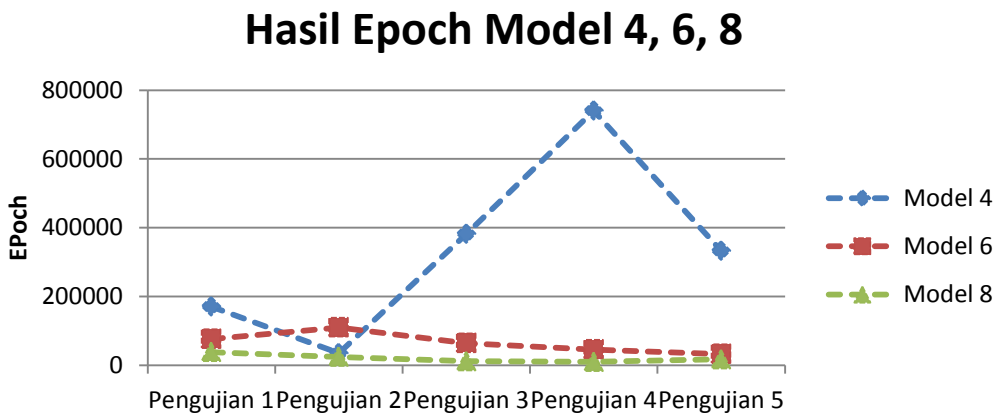
Tabel 6. Hasil *Epoch* Untuk 8 Model

Model	Pengujian				
	1	2	3	4	5
	Epoch				
1	28379	23085	24658	48679	34153
2	5868	7909	6878	7988	8534
3	181594	35035	466230	755993	583947
4	170795	35045	381780	740335	332600
5	68643	137262	64174	47896	24377
6	76725	110120	64470	45685	32880
7	30907	31124	11912	15351	20408
8	38415	24435	11895	10165	17515



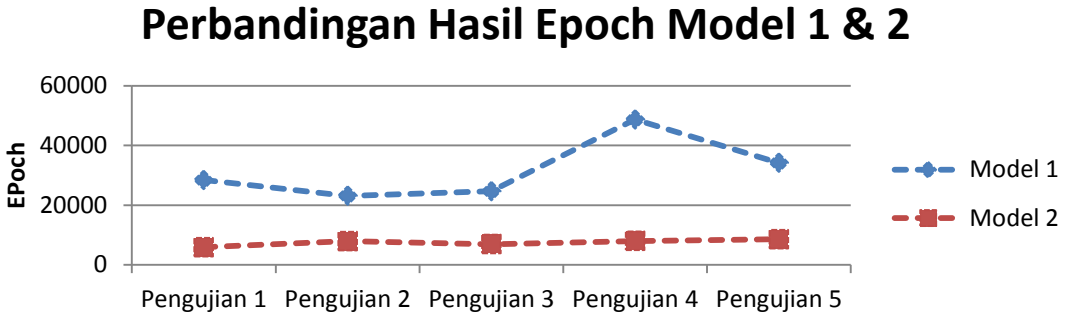
Gambar 4. Grafik Hasil *Epoch* Untuk Model 1, 2, 3, 5, 7

Gambar 4 menunjukan grafik hasil *epoch* untuk model 1, 2, 3, 5, 7 yang melakukan perubahan nilai bobot ke *thread* yang lambat setiap satu *epoch* sekali.



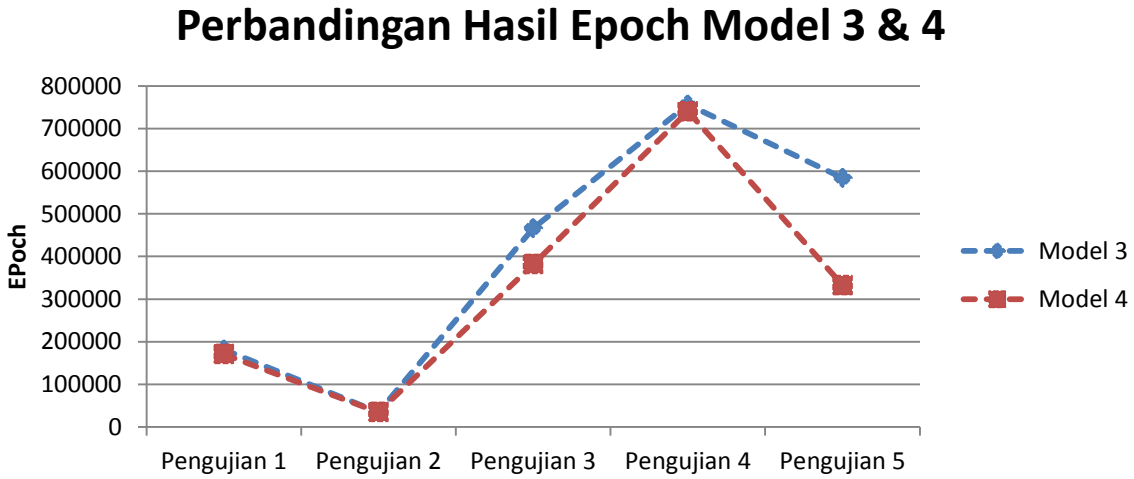
Gambar 5. Grafik Hasil Epoch Untuk Model 4, 6, 8

Gambar 5 menunjukan grafik hasil *epoch* untuk model 4, 6, 8 yang melakukan perubahan nilai bobot ke *thread* yang lambat setiap lima *epoch* sekali.



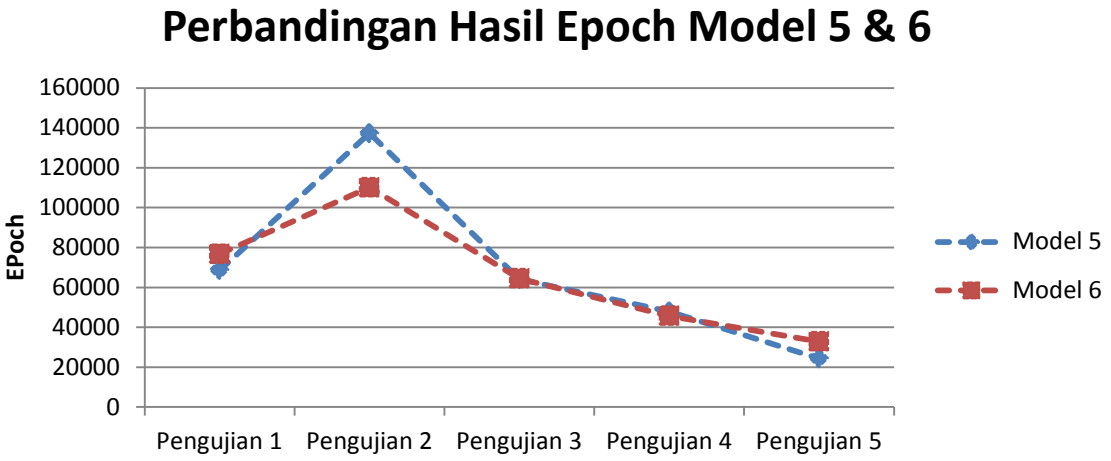
Gambar 6. Grafik Perbandingan Hasil Epoch Model 1 & 2

Gambar 6 menunjukan grafik hasil *epoch* yang membandingkan 2 model yang memiliki jumlah thread yang berbeda dimana model 1 menggunakan 1 buah thread dan model 2 menggunakan 2 buah thread.



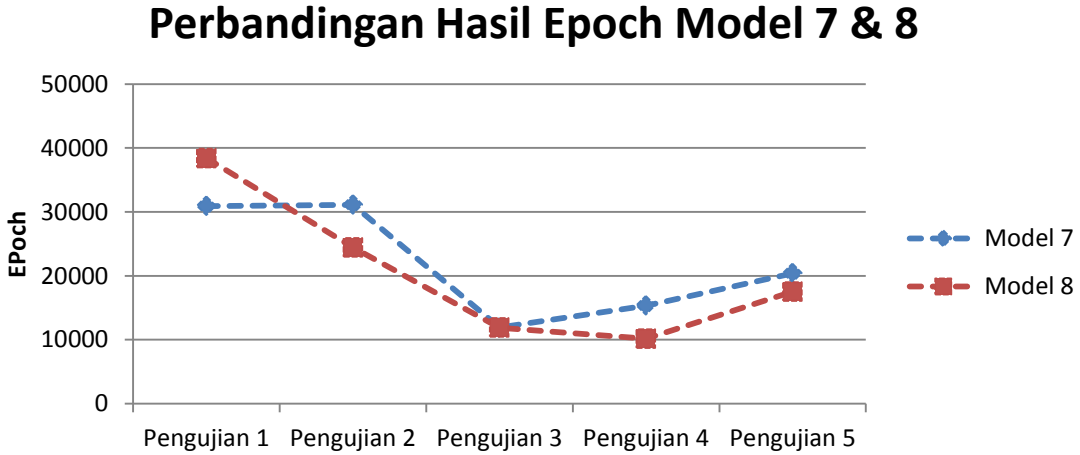
Gambar 7. Grafik Perbandingan Hasil Epoch Model 3 & 4

Gambar 7 menunjukan grafik hasil *epoch* yang membandingkan perubahan bobot ke *thread* lainnya setiap 1 epoch sekali untuk model 3 dan setiap 5 epoch sekali untuk model 4 dengan menggunakan 4 buah *thread*.



Gambar 8. Grafik Perbandingan Hasil Epoch Model 5 & 6

Gambar 8 menunjukan grafik hasil *epoch* yang membandingkan perubahan bobot ke *thread* lainnya setiap 1 epoch sekali untuk model 5 dan setiap 5 epoch sekali untuk model 6 dengan menggunakan 5 buah *thread*.



Gambar 9. Grafik Perbandingan Hasil Epoch Model 7 & 8

Gambar 9 menunjukan grafik hasil *epoch* yang membandingkan perubahan bobot ke *thread* lainnya setiap 1 *epoch* sekali untuk model 7 dan setiap 5 epoch sekali untuk model 8 dengan menggunakan 6 buah *thread*.

Hasil nilai *error* yang didapatkan di setiap *epoch*nya bisa berbeda – beda karena adanya perbedaan nilai bobot dan bias yang didapatkan dengan cara acak pada setiap *thread*. Proses pelatihan dengan menggunakan lebih dari satu *thread* akan sangat berguna apabila terdapat *thread* yang berjalan dengan lambat dalam proses pelatihannya, selain itu juga akan ada *thread* yang dapat berjalan dengan cepat, oleh karena itu perubahan nilai bobot dan bias yang diambil pada suatu *epoch* dari *thread* tercepat dan diubah ke *thread* yang berlatih lambat akan memungkinkan *thread*

yang berjalan lambat menjadi yang tercepat sehingga bisa diambil nilai bobot dan biasnya untuk diubah lagi ke *thread* yang berjalan lambat.

Hasil yang didapatkan untuk perbandingan nilai *error* terkecil yang akan dipilih untuk model ke 3 pengujian ke 2 dapat dilihat pada Tabel 7. Pada Tabel 7 proses pelatihan berhenti pada *epoch* ke 35035. *Epoch* ke 1 nilai *error* terkecil terdapat pada *thread* ke 3, *epoch* ke 2 nilai *error* terkecil terdapat pada *thread* ke 3, *epoch* ke 3 nilai error terkecil terdapat pada *thread* ke 3 dan *epoch* ke 35035 nilai error terkecil terdapat pada *thread* ke 4.

Tabel 7. Perbandingan nilai error terkecil model ke 3 pengujian ke 2.

Epoch	Thread				Error Terkecil
	1	2	3	4	
	Epoch				
1	1.76949405	1.53865860	1.38908388	1.65274527	T3=1.38908388
2	1.24209339	1.18472281	1.12649140	1.27504320	T3=1.12649140
3	0.89342828	0.87532233	0.80190988	0.93866225	T3=0.80190988
⋮	⋮	⋮	⋮	⋮	⋮
35035	0.00311680	0.00328185	0.00451140	0.00099999	T4=0.00099999

Hasil yang didapatkan untuk perbandingan nilai *error* terkecil yang akan dipilih untuk model ke 4 pengujian ke 2 dapat dilihat pada Tabel 8. Pada Tabel 8 proses pelatihan berhenti pada *epoch* ke 35045. *Epoch* ke 1 nilai *error* terkecil terdapat pada *thread* ke 3, *epoch* ke 6 nilai *error* terkecil terdapat pada *thread* ke 4, *epoch* ke 11 nilai *error* terkecil terdapat pada *thread* ke 4 dan *epoch* ke 35045 nilai error terkecil terdapat pada *thread* ke 4. Proses pelatihan pada Tabel 8 juga sama dengan proses pelatihan yang ada pada Tabel 7, hanya pada Tabel 8 proses pelatihan yang melakukan perubahan nilai bobot dan nilai bias ke *thread* yang lebih lambat dilakukan setiap 5 *epoch* sekali.

Tabel 8. Perbandingan nilai error terkecil model ke 4 pengujian ke 2.

Epoch	Thread				Error Terkecil
	1	2	3	4	
	Epoch				
1	0.69364301	0.54869375	0.29540435	0.68766450	T3=0.29540435
6	0.04492452	0.04661984	0.04082662	0.03535855	T4=0.03535855
11	0.02729451	0.02595853	0.02707452	0.01873484	T4=0.01873484
⋮	⋮	⋮	⋮	⋮	⋮
35045	0.00299913	0.00317476	0.00442119	0.00099998	T4=0.00099998

SIMPULAN

Pada penelitian ini dilakukan perbandingan antara model 1 dan 2 dimana model 1 hanya terdapat 1 proses *thread* pelatihan sedangkan model 2 terdapat 2 proses *thread* pelatihan dan hasil dari pelatihan dengan menggunakan 2 proses *thread* pelatihan bisa memperkecil jumlah *epoch* proses pelatihan seperti yang ditunjukan pada Tabel 5. Selain menggunakan jumlah *thread* lebih dari 1 seperti model 1 dan 2 pengubahan nilai bobot dan bias setiap 5 *epoch* sekali juga dapat meningkatkan performa dari *backpropagation*. Berdasarkan hasil penelitian untuk model 3 & 4 dimana model

sama-sama memiliki 4 proses *thread* pelatihan namun model 3 hanya melakukan perubahan bobot setiap 1 *epoch* sekali sedangkan model 4 melakukan perubahan bobot setiap 5 *epoch* sekali dan untuk model 4 performanya meningkat pada 4 pengujian. Model 5 & 6 dengan jumlah proses *thread* pelatihanya sebanyak 5 namun model 5 melakukan proses perubahan bobot setiap 1 *epoch* sekali sedangkan model 6 melakukan perubahan bobot setiap 5 *epoch* sekali dan untuk model 5 performanya meningkat pada 2 pengujian. Model 7 & 8 dengan jumlah proses *thread* pelatihan sebanyak 6 namun model 7 melakukan proses perubahan bobot setiap 1 *epoch* sekali sedangkan model 8 melakukan perubahan bobot setiap 5 *epoch* sekali dan untuk model 8 performanya meningkat pada 4 pengujian. Sesuai dengan hasil penelitian yang telah diuji ini maka dapat disimpulkan bahwa proses pelatihan *backpropagation* dengan menggunakan proses *thread* yang semakin banyak dan juga melakukan perubahan bobot ke proses *thread* yang lambat setiap 5 *epoch* sekali bisa meningkatkan performa *backpropagation* dengan mengecilnya hasil *epoch* dari pelatihan.

DAFTAR PUSTAKA

[1] Rodriguez D D H 2012 *Optimization of back-propagation learning algorithm on mlp networks* (Germany: Universitat Stuttgart).

[2] Krpan N & Jakobovic 2010 *Parallel neural network training with opencl* (Croatia: University of Zagreb).

[3] Tsaregorodtsev V G 2005 *Parallel implementation of backpropagation neural network software on smp computer.* V. Malyshkin (Ed.): PaCT 2005, LNCS 3606, pp. 186–192.

[4] Chamidah N, Wiharto & Salamah 2015 Pengaruh Normalisasi data pada jaringan syaraf tiruan backpropagasi gradient descent adaptive gain(bpgdag). Jurnal Itsmart 1: 2301 – 7201.

[5] Setiawan W 2008 Prediksi Harga Saham Menggunakan Jaringan Syaraf Tiruan Multilayer Feedforward Network Dengan Algoritma Backpropagation KNS&I08-020.

[6] Silberschatz A, Galvin P B & Gagne G 2013 *Operating System Concepts* (New Jersey: John Wiley & Sons).