Perbandingan Model FFNN Dan Garch Pada Data IHSG Bursa efek jakarta[1]

Oleh :
Budi Warsito[2], Suparti[3] dan Subanar[4]
[1]Dibiayai oleh Hibah Pekerti Dikti tahun 2010
[2,3]Program Studi Statistika Jurusan Matematika FMIPA Universitas Diponegoro
[4]Program Studi Statistika Jurusan Matematika FMIPA Universitas Gadjah Mada

ABSTRAK

Tulisan ini membahas perbandingan model Feed Forward Neural Network (FFNN) dengan model Generalized Autoregressive Conditional Heteroscedasticity (GARCH) pada data time series. Pada model FFNN, metode pelatihan yang digunakan adalah Levenberg-Marquardt dengan fungsi aktivasi logistic sigmoid. Metode yang digunakan untuk mendapatkan arsitektur jaringan optimal dari model FFNN adalah metode pruning Optimal Brain Damage (OBD). Pada metode OBD bobot yang mempunyai *saliency* (pengaruh terhadap perubahan error) kecil dihapus dari jaringan. Dengan menghapus bobot yang tidak penting, diharapkan dapat meningkatkan performansi jaringan meliputi error pelatihan dan pengujian. Studi kasus dilakukan pada data time series Indek Harga Saham Gabungan (IHSG) Bursa Efek Jakarta.
Kata Kunci : FFNN, Optimal Brain Damage, GARCH, IHSG

1. Pendahuluan

Kondisi perekonomian suatu negara dapat mengalami suatu fluktuasi yang dipengaruhi oleh faktor internal maupun eksternal perekonomian itu sendiri, sehingga dapat mempengaruhi perilaku para pasar modal dalam menganalisis dan memprediksi pendapatannya. Data time series khususnya pada kasus data bursa saham, data keuangan, umumnya mempunyai model tertentu karena adanya suatu kondisi heteroskedastisitas, hal ini disebabkan adanya sifat volatilitas dalam datanya. Salah satu model yang kemudian berkembang untuk mengatasi masalah ini adalah model ARCH (Autoregressif Conditional Heteroscedastic) yang dikembangkan oleh Engle (1982) dan kemudian digeneralisir menjadi model GARCH (Generalized Autoregressif Conditional Heteroscedastis) yang diusulkan oleh Borreslav (1986). Beberapa penelitian yang membahas tentang pemodelan GARCH diantaranya dilakukan oleh Subanar (2004) serta Warsito dkk (2006).

Pada perkembangan pemodelan statistika yang lain, berkembang pula model-model non parametrik yang mengabaikan berbagai asumsi sebagaimana pada model parametrik. Salah satu model yang banyak dikembangkan adalah model Neural Network. Model ini mempunyai fungsi aktivasi atau semacam fungsi transfer nonlinear yang memungkinkan untuk melakukan berbagai analisis data dengan output yang diinginkan, termasuk analisis data time series. Algoritma model juga memungkinkan untuk memilih berbagai metode optimasi untuk mendapatkan output dengan kesalahan yang kecil. Konstruksi arsitektur (model) Neural Network juga sangat fleksibel yang menjadikan model ini sangat terbuka untuk dikaji lebih lanjut. Fleksibilitas yang dimiliki diantaranya adalah pemilihan input model dan jumlah unit hidden yang digunakan. Secara umum terdapat dua metode yang sering digunakan baik untuk pemilihan unit input maupun jumlah unit hidden yang digunakan yaitu top-down yang termasuk ke dalam kelompok *General to Specific* dan bottom-up yang termasuk ke dalam kelompok *Specific to General*.

Tulisan ini membahas penerapan model Neural Network khususnya Feed Forward Neural Network (FFNN) pada data time series yang mengandung efek GARCH. Metode

yang digunakan untuk mendapatkan arsitektur jaringan adalah Pruning Optimal Brain Damage (OBD) yang termasuk dalam kelompok *General to Specific*. Kemudian dilakukan analisis lanjutan mengenai keakuratan prediksi yang diperoleh dan dibandingkan kekuatan dari kedua model. Data yang digunakan adalah Indeks Harga Saham Gabungan pada Bursa Efek Jakarta.

2. Model GARCH

Pada pemodelan time series menggunakan metode ARIMA Box-Jenkins, diasumsikan bahwa variansi bersyarat dari residual adalah konstan atau homoskedastis, yaitu $E\left[\varepsilon_{t+1}^2\right]=\sigma^2$. Jika variansi bersyarat tidak konstan atau $\mathrm{E}\left(\varepsilon_{t+1}^2\right)=\sigma_t^2$ maka variansi bersyarat dimodelkan sebagai proses AR(q):

$$\varepsilon_t^2 = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \cdots + \alpha_q\varepsilon_{t-q}^2 + a_t \qquad (1)$$

Persamaan (1) disebut model Autoregressive Conditional Heteroscedasticity (ARCH). Model ARCH dibedakan atas dua jenis persamaan, yaitu persamaan mean bersyarat dan variansi bersyarat. Persamaan mean bersyarat dapat berupa model AR, ARMA atau model regresi biasa, misalnya $Y_t = \mathbf{x}_t'\boldsymbol{\beta} + \varepsilon_t$ dengan $\mathbf{x}_t$ adalah vektor dari variabel independen yang dapat juga berupa lag dari variabel dependen Y_t. Sedangkan $\boldsymbol{\beta}$ adalah vektor dari parameter yang tidak diketahui. Persamaan variansi bersyarat merupakan fungsi dari konstanta (α_0) dan variansi dari periode lalu, yang diukur sebagai lag dari error kuadrat persamaan mean yaitu ε_{t-1}^2.

Untuk melakukan uji adanya efek ARCH pada data time series ada dua macam cara yaitu dengan uji Lagrange Multiplier atau dengan memeriksa fungsi Autokorelasi (ACF) dari error kuadrat. Pada data yang mengandung efek ARCH errornya tidak berkorelasi namun error kuadratnya berkorelasi.

Bentuk umum dari model ARCH adalah model Generalized Autoregressive Conditional Heteroscedasticity (GARCH). Model GARCH(p,q) didefinisikan sebagai :

$$\varepsilon_t \left| \psi_{t-1} \right. \sim N(0, h_t)$$

$$h_t = \alpha_0 + \sum_{k=1}^{q}\alpha_k\varepsilon_{t-k}^2 + \sum_{j=1}^{p}\lambda_j h_{t-j}$$

$$= \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \alpha_2\varepsilon_{t-2}^2 + \cdots + \alpha_q\varepsilon_{t-q}^2 + \lambda_1 h_{t-1} + \lambda_2 h_{t-2} + \cdots + \lambda_p h_{t-p} \qquad (2)$$

dimana $p \geq 0$, q>0, $\alpha_0 > 0$, $\alpha_k > 0$, k=1,2,…,q dan $\lambda_j \geq 0$, j=1,2,…,p

Proses GARCH(p,q) memungkinkan komponen Autoregressive dan Moving Average dengan variansi yang heteroskedastis terdapat didalamnya. Jika p = 0 dan q $\neq 0$ model pada (2) merupakan proses ARCH(q). Jadi proses ARCH merupakan bentuk khusus dari GARCH. Kelas dari model ARCH membolehkan untuk mengestimasi variansi bersyarat yang bervariasi sepanjang waktu. Model GARCH memasukkan lag-lag dari variansi bersyarat untuk mengestimasi variansi bersyarat dari model.

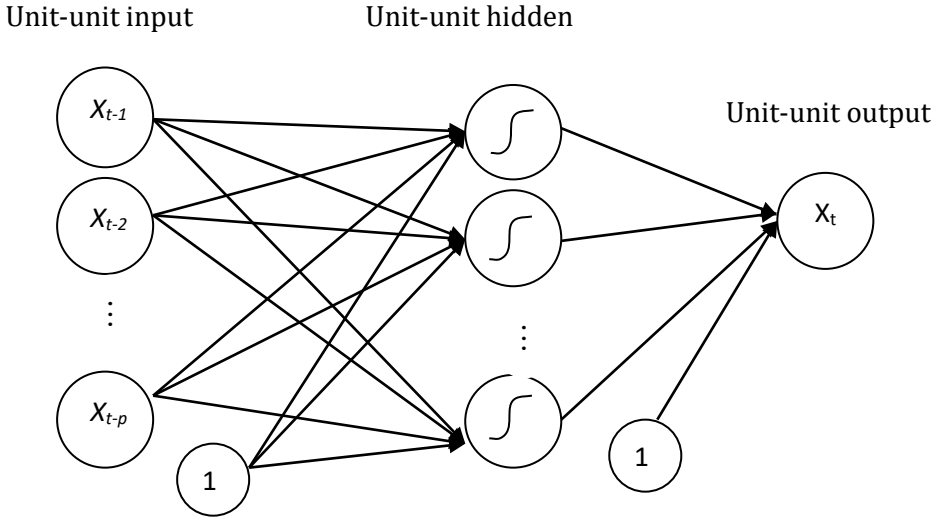
3. Model Neural Network

Model Neural Network yang akan diamati secara khusus dalam tulisan ini adalah model Feed Forward Neural Network (FFNN). Arsitektur jaringan FFNN untuk peramalan data time series dengan konfigurasi unit input X_1 sampai X_p dan satu unit konstan (bias), sebuah *hidden layer* yang terdiri dari n neuron dan 1 unit output

diilustrasikan pada gambar 1. Model FFNN dengan satu hidden layer dengan input $X_{t-1}, X_{t-2}, \ldots, X_{t-p}$ ditulis dalam bentuk

$$\hat{X}_t = \psi_0 \left\{ w_{co} + \sum_n w_{no} \psi_n \left(w_{cn} + \sum_i w_{in} X_{t-j_i} \right) \right\} \quad (3)$$

dengan $\{w_{cn}\}$ adalah bobot antara unit konstan dan neuron dan w_{co} adalah bobot antara unit konstan dan output. Notasi $\{w_{in}\}$ dan $\{w_{no}\}$ masing-masing menyatakan bobot koneksi input dengan neuron dan antara neuron dengan output. Kedua fungsi ψ_n dan ψ_o masing-masing merupakan fungsi aktivasi yang digunakan pada neuron dan output. Notasi yang digunakan untuk model neural network adalah NN(j_1, …, j_k, n) untuk menyatakan NN dengan variabel input pada lag j_1, …, j_k dan n neuron dalam satu hidden layer.



Gambar 1 Arsitektur FFNN untuk peramalan data time series dengan satu hidden layer yang terdiri n neuron dan variabel input nilai pada lag $X_{t-1}, X_{t-2}, \ldots, X_{t-p}$

Algoritma pembelajaran yang digunakan pada jaringan FFNN adalah backpropagation yang meliputi tiga tahap yaitu umpan maju (*feedforward*) dari pola input, penghitungan dan propagasi balik dari error serta penyesuaian bobot-bobot. Pada tahap umpan maju setiap unit input (sensor) menerima sebuah sinyal input (x_i) dan menyebarkan sinyal tersebut pada setiap unit-unit tersembunyi $z_1, \ldots, z_p$. Setiap unit tersembunyi kemudian menghitung aktivasinya dan menghitung jumlah terboboti dari input-inputnya dalam bentuk

$$z\_in_j = \sum_i w_{ji} x_i + w_{bj} \quad (4)$$

dengan x_i adalah aktivasi dari unit input ke-i yang mengirimkan sinyalnya ke unit hidden ke-j dan w_j adalah bobot dari sinyal yang terkirim tersebut serta j = 1,2, …, q adalah jumlah unit pada hidden layer. Sedangkan w_{bj} adalah bobot dari bias ke unit hidden ke-j. Hasil penjumlahan tersebut ditransformasi dengan fungsi aktivasi nonlinear f(·) untuk memberikan aktivasi z_j dari unit j dalam bentuk

$$z_j = f\left(z\_in_j\right) \quad (5)$$

Setelah semua unit tersembunyi menghitung aktivasinya kemudian mengirimkan sinyalnya (z_j) pada unit output. Kemudian unit output menghitung aktivasinya untuk memberikan respon dari jaringan atas pola input yang dimasukkan dalam bentuk

$$g(w,z) = \sum_j w_{jo} z_j + w_{bo} \tag{6}$$

Fungsi pada (6) merupakan nilai output dari jaringan yaitu

$$y = \sum_j w_{jo} f(a_j) + w_{bo} \tag{7}$$

dengan w_{bo} adalah bobot dari bias ke unit output. Selama pelatihan unit output membandingkan aktivasi terhitung y_k dengan nilai targetnya t untuk menentukan error pada pola dengan unit tersebut (Fausett, 1994).

4. Pruning OBD

Konsep dasar dari *pruning* adalah mengurangi bobot ataupun *neuron* pada jaringan yang memiliki pengaruh terkecil pada perubahan error (*saliency*). Nilai *saliency* yang kecil berarti suatu bobot tidak terlalu penting bagi jaringan. Dengan dilakukannya *pruning*, diharapkan tingkat generalisasi model FFNN akan meningkat. Akan tetapi, seringkali metode *pruning* tidak meningkatkan generalisasi, hanya mengurangi *neuron* atau *link* pada jaringan, sehingga efeknya hanya pada mempersingkat waktu yang diperlukan untuk pelatihan dan pengujian. Beberapa metode dikembangkan untuk melakukan proses pruning terhadap jaringan, diantaranya dilakukan oleh Kashoek dan van Dick (1998) dengan metode kontribusi incremental dan Hagiwara (1993) dengan metode *Magnitude Base Pruning*. Le Chun, et.al (1990) mengembangkan metode Optimal Brain Damage (OBD) untuk mereduksi ukuran pembelajaran jaringan dengan penghapusan bobot secara selektif. Metode OBD ini membentuk analisis sensitifitas pada error pelatihan untuk menghapus bobot. Kelebihan dari metode ini yaitu sebagai prosedur minimisasi jaringan secara otomatis dan sebagai alat interaktif untuk mendapatkan arsitektur jaringan yang lebih baik.

Ide dasar *OBD* adalah bagaimana mendapatkan jaringan yang sempurna dan logis, dengan melakukan seleksi dan menghapus separuh atau lebih dari bobot sehingga menjadikan jaringan bekerja dengan baik atau lebih baik. Teknik pada metode ini menggunakan turunan kedua dari fungsi *error* terhadap bobot untuk menghitung perubahan *error*. Dalam menghitung perubahan *error*, OBD menggunakan diagonal dari matriks Hessian. Matriks Hessian adalah matriks yang berisi semua nilai turunan kedua dari *error* terhadap bobot jaringan (Haykin, 1994). Dalam hal ini saliency dari bobot dapat dihitung dengan algoritma Gauss Newton dan Levenberg-Marquardt.

Untuk suatu arsitektur dari sebarang jaringan FFNN, MSE E didefinisikan sebagai :

$$E = \frac{1}{N}\sum_{i=1}^{N}(t - z(x_i, w_i))^2 \tag{8}$$

dengan t dan z masing-masing adalah vektor target dan output jaringan sedangkan N adalah jumlah unit input. Output jaringan adalah fungsi dari input x_i dan bobot w_i yang diupdate selama proses pelatihan. Turunan pertama dari error adalah :

$$\frac{\partial E}{\partial w_i} = \frac{2}{N}\sum_{i=1}^{N}\left\{(t - z)\left(\frac{-\partial z}{\partial w_i}\right)\right\} \tag{9}$$

Sedangkan turunan kedua dari E adalah

$$h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \approx \frac{2}{N}\sum_{i=1}^{N}\left\{(t-z)\left(\frac{-\partial^2 z}{\partial w_i \partial w_j}\right) + \left(-\frac{\partial z}{\partial w_i}\right)\left(-\frac{\partial z}{\partial w_j}\right)\right\} \quad (10)$$

Ada dua bagian dalam tanda kurung dari persamaan (10). Bagian pertama dapat menyebabkan masalah ketidakstabilan pada proses komputasi jika H diinverskan (Samarasinghe, 2006). Untuk menghindari masalah ini, matriks Hessian didekati dengan bagian kedua dari persamaan (10) :

$$h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \approx \frac{2}{N}\sum_{i=1}^{N}\left\{\left(-\frac{\partial z}{\partial w_i}\right)\left(-\frac{\partial z}{\partial w_j}\right)\right\} \quad (11)$$

Fungsi perubahan *error* $(\Delta \mathrm{E})$ diaproksimasi dengan :

$$\delta E = \sum_i g_i\, \delta w_i + \frac{1}{2}\sum_i h_{ii}\, \delta w_i^2 + \frac{1}{2}\sum_{i \neq j} h_{ij}\, \delta w_i \delta w_j + O(||\delta w||^3 \quad (12)$$

dengan

$$g_i = \frac{\partial E}{\partial w_i}$$

$$h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$$

Dari persamaan tersebut g_i merupakan komponen gradien dari *E (error)* terhadap w sedangkan h_{ij} merupakan elemen dari matriks *Hessian H* dari *E* terhadap *w* sedangkan w_i merupakan elemen ke-i dari *w*. Diagonal dari matriks hessian dihitung dengan persamaan :

$$h_{ii} = \frac{\partial^2 E}{\partial w_i \partial w_i} \quad (13)$$

Tujuan dari metode OBD adalah menemukan himpunan bobot yang penghapusannya akan menyebabkan peningkatan nilai E yang terkecil.

Beberapa asumsi dari metode OBD :

1) *Error* fungsi jaringan mencapai minimum setelah dilakukan pelatihan sehingga sisi kanan pertama pada persamaan (12) dapat diabaikan.
2) Matriks Hessian H merupakan matriks yang sangat besar sehingga elemen non diagonal dari matriks Hessian diasumsikan bernilai nol. Hal ini mengakibatkan turunan silang pada sisi kanan ketiga dari persamaan (12) dapat diabaikan.
3) *Error* fungsi jaringan sangat kecil sehingga sisi kanan terakhir pada persamaan (12) dapat diabaikan.

Dari asumsi tersebut, persamaan (12) dapat direduksi menjadi:

$$\delta E = \frac{1}{2}\sum_i h_{ii}\, \delta w_i^2 \quad (14)$$

Saliency s_i dari bobot w_i dapat dinyatakan sebagai :

$$s_i = \frac{h_{ii} w_i^2}{2} \quad (15)$$

Prosedur OBD dapat dituliskan sebagai berikut (Le Chun, et.al, 1990) :

1) Pilih arsitektur jaringan yang logis (*reasonable*)
2) Lakukan pelatihan terhadap jaringan sampai diperoleh solusi yang logis
3) Hitung derivatif kedua h_{kk} untuk tiap-tiap parameter
4) Hitung saliency untuk tiap-tiap parameter : $s_i = h_{ii} w_i^2/2$

5) Urutkan parameter-parameter tersebut berdasarkan saliency-nya dan hapus beberapa parameter dengan saliency yang rendah
6) Iterasikan langkah 2

Penghapusan parameter didefinisikan sebagai pengaturan nilainya diubah menjadi nol .

5. Penerapan pada Data IHSG

Prosedur pemodelan GARCH dan FFNN dengan metode OBD ini diterapkan pada data harian Indeks Harga Saham Gabungan (IHSG) Bursa Efek Jakarta (BEJ) mulai tanggal 2 Januari 2007 sampai 12 Mei 2010 yang diperoleh dari situs www.bi.go.id. Data dibagi dua, sebanyak 767 data digunakan untuk membangun model sedangkan 66 data untuk menguji model. Hasil pengolahan data diperoleh model terbaik untuk conditional mean adalah ARIMA ((1,5,13),1,(0,0,0)) tanpa konstanta. Correlogram residual kuadrat selain menunjukkan varian dari data saling berkorelasi juga mengindikasikan adanya efek ARCH. Hal ini juga dapat dilihat dari statistic Q Ljung Box berikut.

Hipotesis

$H_0 : \rho_1 = \rho_2 = ... = \rho_q = 0$ (ε_t^2 tidak saling berkorelasi atau tidak ada efek ARCH)

$H_1 : \rho_k \neq 0$, paling tidak satu k, k=1,2,...,q (ε_t^2 saling berkorelasi atau ada efek ARCH)

Statistik uji

Q = $T(T+2)\sum_{k=1}^{q} \hat{\rho}_k^2$; taraf signifikansi α = 5%.

Kriteria uji

H_0 ditolak jika Q > $\chi^2_{q,0.05}$ atau *Prob.sig* < α (0,05)

Dengan mengambil q = 4, maka diperoleh nilai Q = 166.52 > $\chi^2_{4,0.05}$ = 9.488 maka H_0 ditolak sehingga dapat disimpulkan bahwa terdapat korelasi antar residual kuadrat. Untuk menguatkan indikasi ini dilakukan uji efek ARCH dengan uji Lagrange Multiplier (LM).

Heteroskedasticity Test: ARCH

F-statistic	57.75614	Prob. F(2,750)	0.0000
Obs*R-squared	100.4963	Prob. Chi-Square(2)	0.0000

Hipotesis

$H_0 : \alpha_1 = \alpha_2 = ... = \alpha_q = 0$ (tidak terdapat efek ARCH)

$H_1 : \alpha_k \neq 0$, paling tidak satu k, k=1,2,...,q (terdapat efek ARCH)

Statistik uji

LM = TR^2 (T=banyaknya residual) ; taraf signifikansi α = 5%.

Kriteria uji

H_0 ditolak jika LM > $\chi^2_{q,0.05}$ atau *Prob.sig* < α (0,05)

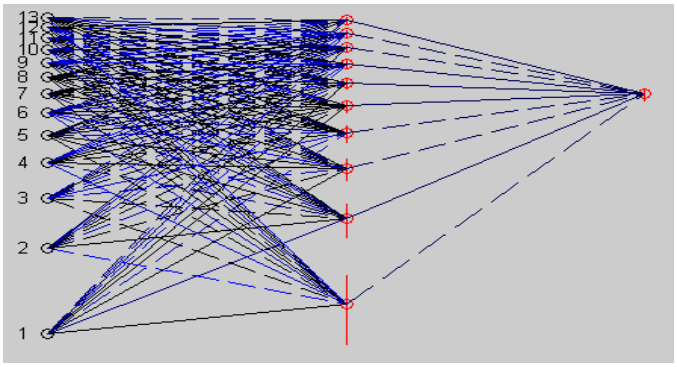
Dengan mengambil q = 2, maka diperoleh nilai LM = 102.4963 > $\chi^2_{2,0.05}$ = 5.991 maka H_0 ditolak yang mengindikasikan adanya efek ARCH. Statistik uji F dan LM test menunjukkan terdapat efek ARCH secara signifikan sampai lag 2 sehingga diperlukan model ARCH untuk analisis lebih lanjut. Masing-masing model GARCH diestimasi dan

hasilnya menunjukkan bahwa model terbaik adalah GARCH(1,0) atau ARCH(1). Hasil plot secara visual menunjukkan bahwa model relative baik hanya untuk prediksi *in sample*, sedangkan untuk prediksi *out of sample* kurang baik.

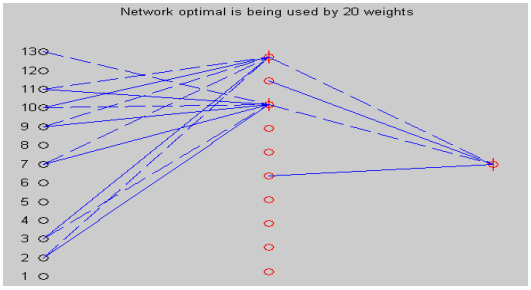
Gambar 1. Data asli dan prediksi model GARCH(1,0)

Pada pemodelan FFNN jumlah data sebanyak 833 data dibagi menjadi dua. Bagian pertama sebagai training sebanyak 767 data sedangkan bagian kedua sebanyak 66 data sebagai testing. Input awal ditetapkan sebanyak 13 unit yaitu dari lag 1 sampai 13, jumlah unit pada hidden layer sebanyak 10. Fungsi aktivasi pada hidden layer adalah logistic sigmoid, f(x) = 1/(1+exp(-x)). Jumlah epoch untuk setiap proses adalah 1000 dengan pencarian bobot optimal menggunakan metode Levenberg-Marquardt. Arsitektur jaringan FFNN hasil pelatihan tanpa proses pruning disajikan pada gambar 2. Garis lurus tanpa terputus menunjukkan bobot hasil pelatihan memberikan hasil positif sedangkan garis putus-putus menunjukkan bobot bernilai negatif. Tanda bulat dengan garis vertikal menunjukkan adanya bobot dari bias ke unit hidden atau ke output layer. Dengan arsitektur tersebut jumlah bobot yang diestimasi sebanyak 151.

Arsitektur jaringan FFNN setelah dilakukan pruning OBD disajikan pada gambar 3. Tanda bulat tanpa garis pada hidden layer menunjukkan bahwa bobot bias ke unit yang bersangkutan termasuk yang direduksi karena mempunyai saliency rendah. Sekarang jumlah bobot yang tersisa tinggal 20 buah.

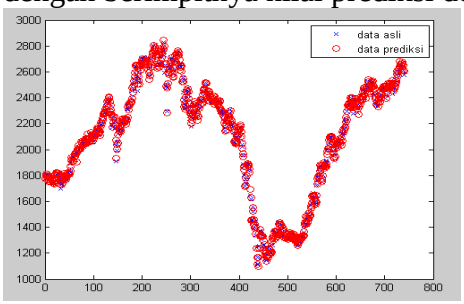


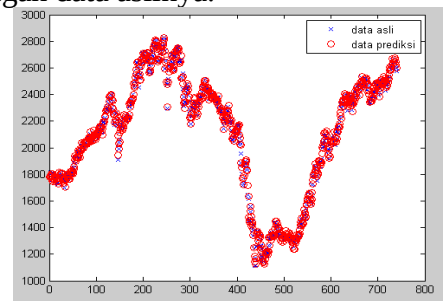
Gambar 2. Arsitektur jaringan FFNN pada data IHSG BEJ hasil pelatihan dengan input lag 1-13 sebelum dilakukan proses pruning



Gambar 3. Arsitektur jaringan FFNN pada data IHSG BEJ hasil pelatihan dengan input lag 1-13 dengan proses pruning OBD

Dari gambar 2 nampak bahwa arsitektur jaringan menjadi lebih sederhana. Neuron input direduksi dari 13 menjadi 7 dengan lag-lag yang signifikan meliputi lag 2, 3, 7, 9, 10, 11, dan 13. Jumlah neuron pada hidden layer juga berkurang dari 10 menjadi 4 yaitu neuron ke 1, 2, 3, dan 6. Neuron ke 1 dan 3 mempunyai bobot yang signifikan dengan bias, sedangkan bobot dari bias ke neuron ke 2 dan 4 tidak signifikan. Bobot dari bias ke output juga signifikan. Hasil prediksi pada data training (*predict in-sample*) dengan dan tanpa pruning OBD disajikan pada gambar 4a dan 4b. Secara visual hasil kedua metode tidak terlalu menunjukkan perbedaan yang berarti. Kedua metode memberikan hasil prediksi yang cukup baik, yaitu error yang relatif kecil, ditandai dengan berimpitnya nilai prediksi dengan data aslinya.

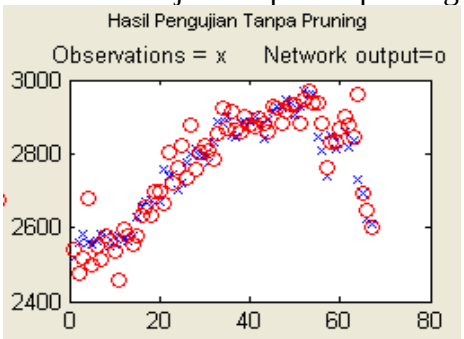


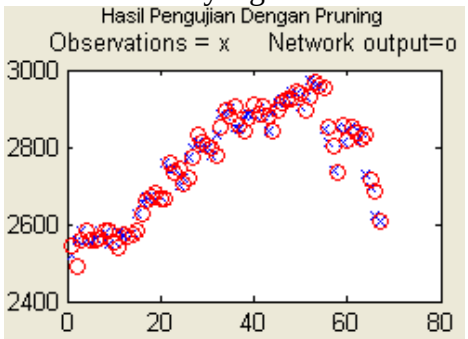


(4a) (4b)

Gambar 4a Prediksi pada data training IHSG BEJ tanpa pruning OBD
Gambar 4b Prediksi pada data training IHSG BEJ dengan pruning OBD

Hasil pengujian dengan data testing (*predict out of sample*) dengan dan tanpa pruning OBD disajikan pada gambar 5a dan 5b. Pengujian dengan data testing secara visual menunjukkan proses pruning OBD memberikan hasil yang lebih teliti.





(5a) (5b)

Gambar 5a Prediksi *out of sample* data testing IHSG BEJ tanpa pruning OBD
Gambar 5b Prediksi *out of sample* data testing IHSG BEJ dengan pruning OBD

Berikut ini adalah tabel perbandingan nilai keakuratan prediksi model GARCH dengan model FFNN untuk data pelatihan (RMSE training), data pengujian (RMSE testing), arsitektur jaringan setelah dilakukan beberapa kali running program.

Tabel 1. Perbandingan keakuratan prediksi model GARCH dengan FFNN

	Model GARCH(1,0)	Model FFNN	
		Non Pruning	Pruning OBD
RMSE training	36,28397	26,57481	26,55789
RMSE testing	222,2522	54,283264	41,976691
Jumlah neuron	-	25	13
Jumlah parameter	5	151	20

Dari tabel 1 nampak bahwa model FFNN menghasilkan prediksi yang lebih baik daripada model GARCH. Sementara proses pruning OBD secara efektif meningkatkan performansi jaringan FFNN meliputi penurunan nilai RMSE training maupun testing, serta penurunan jumlah neuron dan bobot sehingga arsitektur menjadi lebih sederhana.

6. Penutup

Metode OBD dapat menyederhanakan arsitektur jaringan FFNN, dalam hal pemilihan jumlah unit hidden optimal. Metode ini juga dapat secara otomatis memilih lag-lag sebagai unit input yang sesuai. Hasil analisis terhadap data IHSG pada Bursa Efek Jakarta menunjukkan model FFNN dengan metode pruning OBD memberikan hasil prediksi yang lebih baik dari FFNN tanpa pruning maupun model GARCH.

Daftar Pustaka

Bollerslev, T., *Generalized Autoregressive Conditional Heteroskedasticity*, 1986, Journal of Econometrics, Vol. 31, pp 307-327

Gorodkin, J., Hansen, L.K., Krogh, A., Svarer, C., and Winther, O., 1993, *A Quantitative Study of Pruning by Optimal Brain Damage*, Working Paper, Niels Bohr Institute and Technical University of Denmark

Hagiwara, M., 1993, *Removal of Hidden Units and Weights for Backpropagation Networks*, Proceedings of the International Joint Conference on Neural Networks, Vol. 1, IEEE, Los Alamitos, Cam 351

Haykin, S., 1999, *Neural Networks a Comprehensive Foundation*, Prentice-Hall Inc., New Jersey

Kaashoek, J.F. and van Dick, H.K., 1998, *Neural Network Analysis of Varying Trends in Real Exchange Rates*, Report EI9915/A Econometric Institute Rotterdam

Le Cun, Y., Denker, J.S. and Solla, S.A., 1990, *Optimal Brain Damage; on Advances in Neural Information Processing Systems (NIPS)*, vol. 2 page 598–605. San Matteo: Morgan Kaufman

Samarasinghe, S., 2006, *Neural Networks for Applied Sciences and Engineering; from Fundamentals to Complex Pattern Recognition*, Auerbach Publications, Taylor and Francis Group, Boca Raton, New York

Subanar, 2004, *Pemodelan Runtun Waktu ARCH dan GARCH*, Jurnal MIPA, Edisi April, vol. 25 no. 1

Warsito, B., Sichah, I.A. dan Prasetyawati, R.A., 2006, *Model GARCH pada Data Nilai Tukar Rupiah terhadap Dolar Singapura*, Jurnal Math-Info Volume I No 7