TESIS

PREDIKSI DATA *TIME SERIES* MULTIVARIAT MENGGUNAKAN *ECHO STATE NETWORK DOUBLE LOOP RESERVOIR* DAN *HARMONY SEARCH OPTIMIZATION*

MUHAMMAD MUHARROM AL HAROMAINY
NRP. 05111850010044

Dosen Pembimbing

Dr. Eng. Chastine Fatichah, M.Kom.
Dr. Ahmad Saikhu, S.Si., M.T.

DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2021

TESIS

PREDIKSI DATA TIME SERIES MULTIVARIAT MENGGUNAKAN ECHO STATE NETWORK DOUBLE LOOP RESERVOIR DAN HARMONY SEARCH OPTIMIZATION

Muhammad Muharrom Al Haromainy
NRP. 05111850010044

DOSEN PEMBIMBING

Dr. Eng. Chastine Fatichah, M.Kom
NIP: 19751220 20011220 02
Dr. Ahmad Saikhu, S.Si., M.T.
NIP. 197107182006041001

PROGRAM MAGISTER
DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2021

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya
oleh:
MUHAMMAD MUHARROM AL HAROMAINY
NRP: 05111850010044

Tanggal Ujian : 15 Februari 2021
Periode Wisuda :

Disetujui oleh:
Pembimbing

1. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. ……..
   NIP. 197512202001122002
2. Dr. Ahmad Saikhu, S.Si., M.T. ……..
   NIP. 197107182006041001

Penguji

1. Dr. Eng. Nanik Suciati, S.Kom., M.Kom. ……..
   NIP. 197104281994122001
2. Dr. Diana Purwitasari, S.Kom., M.Sc. ……..
   NIP. 197804102003122001
3. Dr. Bilqis Amaliah, S.Kom., M.Kom. ……..
   NIP. 197509142001122002

Kepala Departemen Teknik Informatika,

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
NIP. 197512202001122002

[*Halaman ini sengaja dikosongkan*]

PREDIKSI DATA *TIME SERIES* MULTIVARIAT MENGGUNAKAN *ECHO STATE NETWORK DOUBLE LOOP RESERVOIR* DAN *HARMONY SEARCH OPTIMIZATION*

Nama Mahasiswa : Muhammad Muharrom Al Haromainy
NRP : 05111850010044
Pembimbing I : Dr. Eng. Chastine Fatichah, S.Kom., M.Kom
Pembimbing II : Dr. Ahmad Saikhu, S.Si., M.T.

ABSTRAK

Prediksi data deret waktu multivariat banyak diterapkan di berbagai bidang. Beberapa metode digunakan untuk penelitian prediksi data deret waktu, seperti metode *Artificial Neural Network* (ANN) dan *Recurrent Neural Network* (RNN). RNN menghasilkan nilai akurasi lebih baik dibandingkan ANN, karena dapat menyimpan memori (*feedback loop*) dengan melakukan perulangan dalam arsitekturnya kemudian menggunakannya untuk prediksi sehingga tidak membuang informasi dari masa lalu. Namun, salah satu kelemahan utama RNN adalah kesulitan dalam mengadaptasi bobot. Metode pengembangan dari RNN adalah *Echo State Network* (ESN) model prediksi deret waktu nonlinear multivariat yang kuat dan adaptif. Terlepas dari keunggulan yang disebutkan, pengaturan parameter ESN, seperti inisialisasi parameter *reservoir* harus dilakukan beberapa kali percobaan hingga mendapatkan hasil yang sesuai. Kemudian, struktur *reservoir* yang acak menyebabkan ESN membutuhkan waktu yang lama untuk menghasilkan *reservoir*.

Maka dari itu, penelitian ini mengusulkan *Echo State Network* dengan *Double Loop Reservoir* untuk proses prediksi data deret waktu multivariat dan dioptimasi menggunakan *Harmony Search* (HS), sehingga diharapkan dapat meningkatkan performa hasil prediksi. Metode evaluasi menggunakan *Root Mean Square Error* (MSE) dan *Mean Absolute Percent Error* (MAPE). Hasil prediksi menggunakan metode ESN-DLR lebih baik dari pada metode RNN dan ESN dengan nilai kesalahan 0.0001 dan 0.0082 untuk dataset 1, 5.88e-6 dan 0.0049 untuk dataset 2. Penentuan nilai parameter metode ESN yang sangat berpengaruh terhadap hasil prediksi, dibantu dengan optimasi metode HS sehingga mendapatkan nilai kesalahan lebih baik sebesar 3.36e-5 dan 0.0048 untuk dataset 1, pada dataset 2 memeroleh nilai kesalahan 1.46e-6 dan 0.0007.

Kata kunci : *double loop reservoir, echo state network, reservoir computing, forecasting, harmony search*

[*Halaman ini sengaja dikosongkan*]

PREDICTION OF MULTIVARIATE TIME SERIES DATA USING ECHO STATE NETWORK DOUBLE LOOP RESERVOIR AND HARMONY SEARCH OPTIMIZATION

By	: Muhammad Muharrom Al Haromainy
Student Identity Number	: 05111850010044
Supervisor I	: Dr. Eng. Chastine Fatichah, S.Kom, M.Kom
Supervisor II	: Dr. Ahmad Saikhu, S.Si., M.T.

ABSTRACT

Multivariate time series data prediction is widely applied in various fields. Several methods are used to research time series data predictions, such as the Artificial Neural Network (ANN) and Recurrent Neural Network (RNN) methods. RNN produces better accuracy than ANN, because it can save memory (feedback loop) by looping in its architecture then using it for predictions so as not to throw information from the past. One of the main drawbacks of the RNN, however, is the difficulty in adapting weights. The development method of the RNN is the Echo State Network (ESN), a multivariate and adaptive nonlinear time series prediction model. Apart from the stated advantages, setting ESN parameters, such as reservoir parameter initialization, must be carried out several times to get the right results. Then, the random reservoir structure causes the ESN to take a long time to generate a reservoir.

Therefore, this study proposes Echo State Network with Double Loop Reservoir for the process of predicting multivariate time series data and is optimized using Harmony Search (HS), so that it is expected to improve the performance of the prediction results. The evaluation method uses the Root Mean Square Error (MSE) and Mean Absolute Percent Error (MAPE). The prediction results using the ESN-DLR method are better than the RNN and ESN methods with an error value of 0.0001 and 0.0082 for dataset 1, 5.88e-6 and 0.0049 for dataset 2. Determination of parameter values for the ESN method which greatly affects the prediction results, is assisted by optimization HS method so that it gets better error values of 3.36e-5 and 0.0048 for dataset 1, for dataset 2 it gets error values 1.46e-6 and 0.0007.

Keywords: double loop reservoir, echo state network, reservoir computing, forecasting, harmony search

[*Halaman ini sengaja dikosongkan*]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

BAB 1
PENDAHULUAN

Bab ini menjelaskan mengenai beberapa hal dasar dalam pembuatan penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1.1. Latar Belakang

Prediksi data deret waktu multivariat merupakan salah satu bidang penelitian penting. Prediksi data deret waktu banyak diterapkan di berbagai macam bidang. Selama bertahun-tahun, peneliti mengusulkan banyak teknik untuk memodelkan dan memprediksi data deret waktu multivariat (Bianchi, dkk., 2015; Chen, dkk., 2013), penelitian (Begam, dkk., 2019) memprediksi kecepatan angin karena pengaruhnya tinggi dalam menghasilkan tenaga angin dan didasari peningkatan permintaan akan energi dari hari ke hari. Kemudian (Ma, dkk., 2018) melakukan penelitian tentang prediksi penyimpangan trek yang sangat penting untuk pemeliharaan dan manajemen kereta api.

Beberapa metode digunakan untuk penelitian prediksi data deret waktu, seperti metode *Artificial Neural Network* (ANN) untuk mengetahui keterlambatan troposfer basah guna memprediksi cuaca secara *realtime* (Selbesoglu, 2019). Penelitian lain, metode ANN digunakan untuk memprediksi kecepatan angin (Navas, dkk., 2019), memprediksi dari data eksresi gen (Abiodun, dkk., 2018)(Daoud, dkk., 2019), dan masih banyak lainnya.

Metode *Recurrent Neural Network* (RNN) menghasilkan nilai akurasi lebih baik dibandingkan ANN (Ivanedra, dkk., 2019) karena memiliki sistematika perhitungan bobot secara berulang. RNN dapat menyimpan memori (*feedback loop*) dengan melakukan perulangan dalam arsitekturnya kemudian menggunakannya untuk prediksi sehingga tidak membuang informasi dari masa lalu. RNN dapat secara otomatis mempelajari semantik tingkat tinggi (Usama, dkk., 2019). Penelitian (Choi, dkk., 2019) menggunakan RNN untuk prediksi lintasan kendaraan menggunakan data lalu lintas perkotaan. Rusia juga menggunakan penelitian hasil ilmiah untuk membantu mengurangi biaya dan meningkatkan

efisiensi energi menggunakan jaringan saraf berulang (Zagrebina, dkk., 2019) karena membantu kesalahan prediksi yang menyebabkan biaya lebih mahal.

Model yang paling banyak digunakan adalah jaringan saraf berulang (RNN) (Cai, dkk., 2007). Namun, salah satu kelemahan utama RNN adalah kesulitan dalam mengadaptasi bobot (Xu, dkk., 2018). Berbagai algoritma telah digunakan untuk melatih RNN, namun algoritma ini tetap mengalami kompleksitas komputasi tinggi, proses pelatihan yang lambat, dan potensi ketidakstabilan (Schimdhuber, 2015). Metode pengembangan dari RNN adalah *Echo State Network* (ESN) model prediksi deret waktu nonlinear multivariat yang kuat dan adaptif (Shen, dkk., 2018). Penelitian (Shi, dkk., 2016) menerapkan ESN untuk memprediksi konsumsi energi gedung kantor dan menadapatkan hasil yang baik. Konsep utama algoritma ini adalah melatih bobot keluaran sendiri dengan metode regresi linear sederhana. Bobot yang dilatih hanya pada lapisan ketiga (Løkse, dkk., 2017). Berbeda dengan RNN, di mana semua bobot (tidak hanya bobot pada *output*) harus dilatih (Schiller, dkk., 2005).

Terlepas dari keunggulan yang disebutkan sebelumnya, pengaturan parameter ESN, seperti inisialisasi parameter *reservoir* masih dilakukan dengan percobaan beberapa kali hingga mendapatkan nilai parameter yang sesuai (Chouikhi, dkk., 2017). Di mana, *reservoir* adalah lapisan kedua dari ESN yang digunakan untuk ekspansi data *input*. Selanjutnya, struktur *reservoir* yang acak menyebabkan ESN membutuhkan waktu yang lama untuk menghasilkan *reservoir* dan kemampuan untuk menangani model nonlinier juga terbatas (Zhou, dkk., 2018).

Oleh karena itu, pada tesis ini diusulkan perbaikan struktur *reservoir* untuk memperpendek waktu *training* dan meningkatkan efisiensi prediksi metode *Echo State Network* menggunakan *Double Loop Reservoir* (DLR). Penelitian (Zhou, 2018) menjelaskan bahwa perbaikan struktur *reservoir* menggunakan DLR mendapatkan hasil lebih baik dari pada menggunakan *Multiple Loop Reservoir*. Selanjutnya, untuk mengatasi pencarian nilai parameter yang optimal, menggunakan metode *Haromny Search* (HS).

Terdapat enam *global* parameter pada ESN, namun menurut (Lukoševičius, 2012) salah satu parameter yang dapat dioptimasi adalah *leaking*

rate. Maka dari itu, *leaking rate* menjadi salah satu parameter yang akan dioptimasi. Parameter lain yang dioptimasi yaitu jumlah reservoir, karena penelitian ini mengusulkan perbaikan pada arsitektur lapisan reservoir. Pada tesis ini menggunkaan HS karena dibandingkan algoritma optimasi yang lain, HS memiliki keuntungan dapat menghasilkan vektor baru dengan mempertimbangkan vektor-vektor lainnya, dan hanya membutuhkan beberapa parameter untuk menjalankan algoritma HS. Pada penelitian (Saadat, dkk., 2017) juga melakukan optimasi menggunakan metode HS dan metode ESN dasar, tetapi tidak menyebutkan parameter atau hal apa yang dioptimasi. Apalagi semua parameter dioptimasi, berdampak pada proses komputasi semakin lama. Algoritma lain seperti *Genetic Algorithm* (GA) juga dapat digunakan, namun hasil optimisasi masih lebih baik menggunakan HS (Kim, dkk., 2019).

Penelitian ini mengusulkan *Echo State Network* dengan *Double Loop Reservoir* untuk proses prediksi data deret waktu multivariat dan dioptimasi menggunakan *Harmony Search*, sehingga diharapkan dapat meningkatkan performa hasil prediksi.

1.2. Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah sebagai berikut.

1. Bagaimana meningkatkan performa *Echo State Network* menggunakan *Double Loop Reservoir* untuk memprediksi data deret waktu multivariat?
2. Bagaimana merancang metode optimasi *Harmony Search* pada metode *Echo State Network* untuk memprediksi data deret waktu multivariat?
3. Bagaimana evaluasi kinerja metode yang diusulkan?

1.3. Tujuan

Tujuan yang akan dicapai dalam penelitian ini adalah memprediksi data deret waktu multivariat menggunakan metode *Echo State Network Double Loop Reservoir* yang dioptimasi menggunakan *Harmony Search*, sehingga diharapkan dapat meningkatkan performa hasil prediksi.

1.4. Manfaat

Manfaat dari penelitian ini adalah menghasilkan model prediksi yang lebih baik dan menghindari proses komputasi yang lama, serta mendapatkan nilai parameter yang sesuai agar mendapatkan hasil prediksi dengan nilai *error* lebih kecil.

1.5. Kontribusi Penelitian

Kontribusi pada penelitian ini adalah:

1. Menambahkan *double loop reservoir* pada lapisan kedua dari *Echo State Network*.
2. Optimasi parameter ESN menggunakan dengan *Harmony Search*.

1.6. Batasan Masalah

Batasan masalah pada penelitian ini adalah data menggunakan data deret waktu multivariat dengan kondisi atribut yang akan diprediksi dan atribut prediktor mempunyai relasi. Dibuktikan dengan melakukan uji coba *pearson correlation test*.

[*Halaman ini sengaja dikosongkan*]

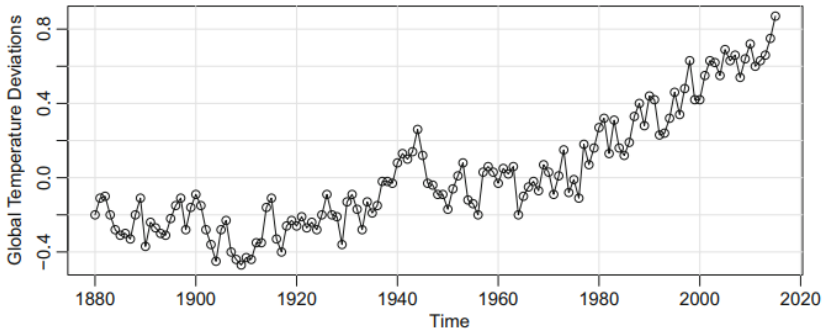
BAB 2
KAJIAN PUSTAKA

Bab ini akan menjelaskan tentang kajian pustaka yang terkait dengan landasan penelitian. Pustaka yang terkait adalah *echo state network*, *harmony search*, dan metode evaluasi.

2.1 *Time Series Data*

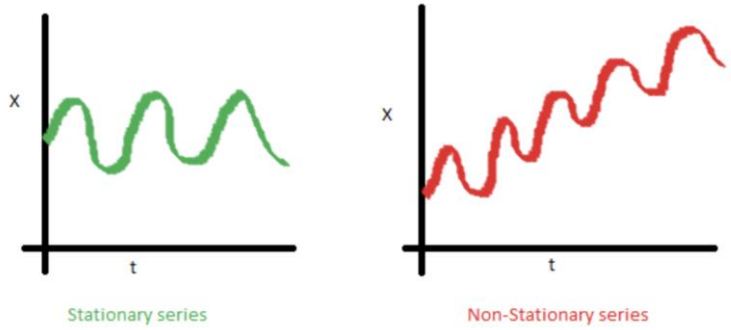
Time series data atau data deret waktu adalah kumpulan data hasil pengamatan secara historis dan berkala dalam jangka waktu tertentu seperti hari, minggu, bulan, atau tahun-an yang menggambarkan kronologis karakter sebuah populasi. Pendekatan sistematis yang digunakan untuk menjawab pertanyaan matematis dan statistik yang berkaitan dengan korelasi waktu biasanya disebut analisis data deret waktu(Shumway, 1999).

Beberapa masalah dan pertanyaan untuk analisis data deret waktu dengan mempertimbangkan data yang ingin digunakan dapat dari berbagai subjek yang berbeda. Contoh studi kasus dari permasalahan data deret waktu dan contoh pertanyaan statistik yang mungkin ada tentang data tersebut seperti, catatan suhu global yang ditunjukkan pada Gambar 2.1 adalah indeks suhu udara daratan samudra mulai tahun 1880-2015, terlihat bahwa kenaikan tren terjadi. Jadi, dapat dianalisis faktor utama penyebab tren naik suhu global, apakah disebabkan faktor manusia atau karena sebab sedimen glasial.

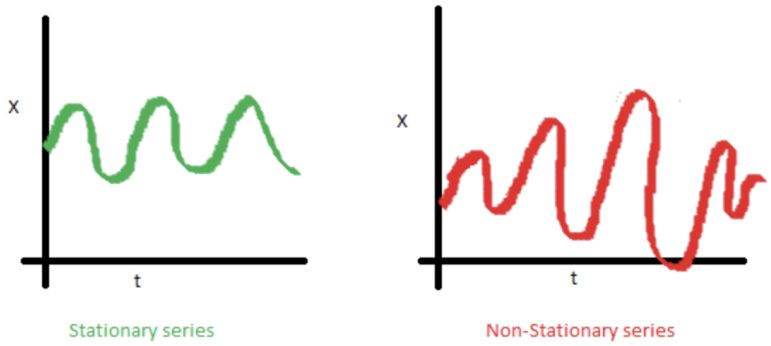


Gambar 2.1 Data deret waktu suhu global(Shumway, 1999)

Salah satu properti paling penting yang dimiliki data deret waktu adalah stasioner. Data deret waktu dikatakan stasioner jika sifat statistiknya, seperti *mean* dan *varian* tetap konstan dari waktu ke waktu. Secara umum, dapat dikatakan jika data deret waktu memiliki perilaku tertentu dari waktu ke watu, ada kemungkinan besar bahwa pola deret waktu akan mengikuti di masa yang akan datang. Data bersifat stasioner jika nilai *mean* konstan seperti ditunjukkan pada Gambar 2.2, kemudian mempunya nilai *varian* konstan seperti Gambar 2.3.



Gambar 2.2 *Mean* stasioner dan nonstasioner(Stewart, 2019)



Gambar 2.3 *Varian* stasioner dan nonstasioner(Stewart, 2019)

Apabila data belum stasioner, dapat dilakukan metode *detrending* atau *differencing*. Kemudian, agar tahu data tersebut sudah stasioner atau belum dapat dilakukan beberapa metode, seperti melakukan *plotting* data atau tes *dickey fuller*.

Data deret waktu ada yang berjenis *univariate* dan *multivariate*. Data deret waktu *univariate* biasanya hanya memiliki satu atribut yang digunakan untuk proses prediksi, misalnya data suhu udara setiap jam. Sedangkan *multivariate* memiliki lebih dari satu *dependent variabel*. Setiap variabel tidak hanya bergantung pada masa lalu, tetapi juga memiliki keterkaitan pada variabel lain.

Pengujian data deret waktu non-linier dapat dilakukan dengan beberapa metode, seperti *white test* atau *Reset* (*Regression specification error test*). Pengujian terserbut dapat dilakukan menggunakan pemrograman R. Berdasarkan dua metode tersebut, jika diperoleh *p-value* kurang dari *alpha* sebesar 0.05, maka menunjukkan data tersebut non-linier.

2.2 *Reservoir Computing*

Reservoir Computing (RC) adalah salah satu teknik dalam bidang *Machine Learning* (ML). Komputasi *reservoir* memetakan data *input* menjadi data yang berdimensi tinggi. RC sangat efisien jika diterapkan pada data deret waktu atau yang berkaitan data satu dengan data setelahnya(Schrauwen, dkk., 2007).

Terdapat tiga lapisan yang diterapkan pada metode komputasi *reservoir*. Pertama adalah lapisan *input*, berupa vektor yang memiliki panjang yang sesuai dengan atribut data yang digunakan. Lapisan ini terdiri dari beberapa *neuron* yang saling terhubung secara acak. Lapisan pertama dan lapisan kedua dihubungkan oleh vektor bobot yang terbentuk secara acak sebagai faktor pengali nilai *input* menuju lapisan *reservoir*. *Neuron* pada *reservoir* juga terhubung dengan *neuron* lain yang ada di dalam *reservoir* dan terhubung secara acak. Hubungan antarneuron berlangsung secara berulang menggunakan konsep RNN membuat *reservoir* memiliki memori untuk menyimpan *input* secara sekuen(Klibisz, 2016).

Lapisan ketiga adalah *readout*, terbentuk vektor dengan ukuran yang sesuai dengan jumlah *output* yang diinginkan. Lapisan kedua dan ketiga terhubung oleh nilai bobot, namun yang membedakan dengan bobot *input*, lapisan ini mengalami proses pembelajaran yang dilakukan dengan perhitungan mundur seperti *backpropagation*. Perbaikan bobot hanya terjadi pada lapisan *readout*. Beberapa metode yang menggunakan pengembangan dari *reservoir computing* adalah *Echo State Network* (ESN) dan *Liquid State Machine* (LSM)(Klibisz, 2016).

2.3 *Echo State Network*

Komputasi *reservoir* telah muncul dalam dekade terakhir sebagai alternatif untuk pelatihan jaringan saraf berulang. *Echo State Network* (ESN) merupakan

salah satu kunci komputasi *reservoir*. Walapun praktis, secara konsep sederhana, dan mudah diimplementasikan, ESN memerlukan beberapa pengalaman dan wawasan untuk mencapai kinerja yang baik.

Echo State Network (ESN) menyediakan arsitektur dan prinsip pembelajaran yang didasari oleh Recurrent Neural Networks (RNNs). Gagasan utama adalah untuk menggerakkan jaringan saraf berulang secara acak, besar, memperbaiki RNN dengan sinyal input sehingga menginduksi di setiap neuron dalam jaringan "reservoir". Mereka adalah *Machine Learning* (ML) yang paling mirip dengan otak biologis, subtrat kecerdasan alami (Lukoševičius, 2012).

Error Backpropagation (BP) sampai saat ini adalah pencapaian terpenting dalam pelatihan jarinagn saraf tiruan. Metode BP juga diperluan ke RNNs, tetapi hanya dengan keberhasilan parsial. Keterbatasan metode BP untuk RNN adalah bahwa bifurkasi dapat membuat pelatihan menjadi non-konvergen. Bahkan ketika mereka bertemu, konvergensi melambat, mahal secara komputasi, dan menyebabkan minimum lokal yang buruk.

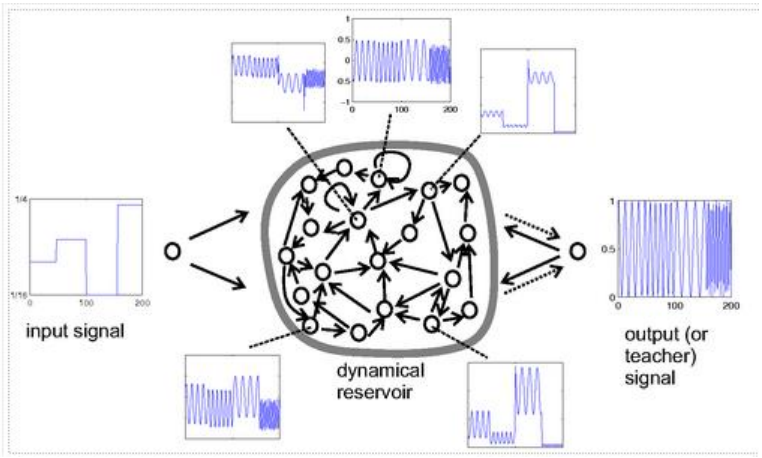
Alternatif penggunaan RNNs telah diusulkan dengan *Echo State Network* dalam ML, dan *Liquid State Machine* (LSM) dalam ilmu *neuroscience*. Tren yang dimulai oleh ESN dan LSM dikenal dengan *Reservoir Computing* (RC) (Verstraeten, dkk., 2007). RC saat ini merupakan area penelitian yang produktif. Pada *benchmark* yang dirancang untuk menantang akuisisi memori jangka pendek, ESN masih mengungguli RNNs (Jaeger, 2012). Kesederhanaan yang tampak dari ESN memerlukan beberapa pengalaman. Ada sejumlah hal yang bisa jadi salah. Secara khusus, generasi awal jaringan *reservoir* dipengaruhi oleh beberapa parameter *global*, dan ini harus ditetapkan secara tepat.

Untuk ilustrasi, melatih RNNs sebagai generator frekuensi. Sinyal *input* $u(n)$ adalah pengaturan frekuensi yang perlahan berubah, *output* yang diinginkan $y(n)$ adalah gelombang frekuensi yang ditunjukkan oleh input saat ini. Rangkaian *input-output* pelatihan $D = \big(u(1)y(1)\big), \dots, (u(nmax)y(nmax))$. *Input*-nya adalah fungsi langkah acak lambat yang menunjukkan frekuensi mulai dari 1/16

hingga ¼ Hz. Tugasnya adalah untuk melatih RNNs dari data pelatihan ini sehingga pada uji lambat sinyal input, output kembali menjadi gelombang frekuensi yang ditentukan input. Dalam pendekatan ESN, tugas ini dapat diselesaikan dengan langkah-langkah berikut.

Langkah pertama, berikan RNN acak. Membuat *reservoir* dinamik acak RNN, menggunakan model neuron apa saja. Ukuran reservoir N tergantung pada tugas. Dalam tugas demo generator frekuensi, N = 200 digunakan. Kemudian, pasang unit input ke *reservoir* dengan membuat koneksi acak *all-to-all*. Selanjutnya, membuat unit *output*. Jika tugas memerlukan *output feedback*, buat koneksi *output* ke *reservoir* yang dihasilkan secara acak. Jika tidak memerlukan *output feedback*, jangan membuat koneksi ke/dari unit *output*.

Langkah kedua, mendorong *reservoir* dinamis dengan data pelatihan D untuk $N = 1, \dots, nmax$. Dalam contoh demo, di mana ada koneksi *feedback output* ke *reservoir*. Dalam tugas tanpa *output feedback*, *reservoir* hanya didorong oleh input $u(n)$. Ini menghasilkan urutan $x(n)$ dari kondisi *reservoir* berdimensi-N. Setiap sinyal komponen $x(n)$ adalah transformasi nonliniar dari *input* penggerak. Dalam demo, masing-masing $x(n)$ adalah campuran individu dari sinyal input lambat dan gelombang cepat *output sinewave* pada Gambar 2.4.



Gambar 2.4 Skema Dasar ESN(Jeager, 2007)

Langkah ketiga, menghitung bobot *output* sebagai bobot regresi linier *output* $y(n)$ pada kondisi *reservoir* $x(n)$. Gunakan bobot ini untuk membuat koneksi *reservoir* ke *output* (panah putus-putus Gambar 2.4).

2.3.1 Model Dasar *Echo State Network*

ESN diterapkan pada *machine learning* di mana untuk sinyal *input* pelatihan yang diberikan $u(n) \in \mathbb{R}^{\mathrm{N}_u}$, sinyal target *output* yang diinginkan $y^{target}(n) \in \mathbb{R}^{\mathrm{N}_y}$. Di sini $n = 1, \dots, T$ adalah waktu diskrit dan T adalah jumlah titik data dalam dataset pelatihan. Mempelajari model dengan *output* $y^{target}(n) \in \mathbb{R}^{\mathrm{N}_y}$ di mana $y(n)$ cocok dengan $y^{target}(n)$ sebaik mungkin, meminimalkan ukuran kesalahan $\mathrm{E}(y, y^{target})$, dan yang lebih penting, menggeneralisasikan untuk tidak melihat data. Ukuran kesalahan E adalah *Mean Square Error* (MSE), misalnya *Root Mean Square Error* (RMSE) yang juga dirata-rata atas N_y dimensi i dari *output*.

ESN menggunakan tipe RNN dengan unit nilai kontinu waktu diskrit terintegrasi waktu. Jenis pembaharuan persamaan adalah

$$\tilde{x}(n) = \tanh\left(W^{in}[1; u(n)] + Wx(n-1)\right) \quad (2.1)$$

$$x(n) = (1-\alpha) \times (n-1) + a\tilde{x}(n) \quad (2.2)$$

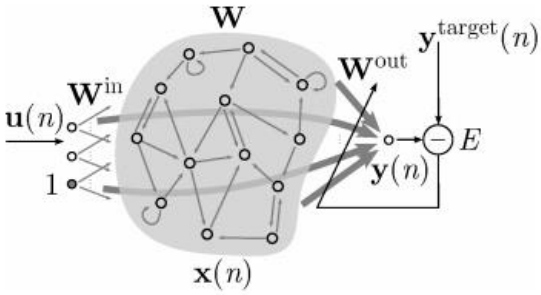
di mana $x(n) \in \mathbb{R}^{\mathrm{N}_x}$ adalah vektor aktivasi neuron *reservoir* dan $\tilde{x}(n) \in \mathbb{R}^{\mathrm{N}_x}$ adalah pembaharuannya, semua pada waktu n, $\tanh(.)$ diterapkan berdasarkan elemen, $[.;.]$ adalah singkatan dari gabungan vektor atau matriks vertikal, $W^{in} \in \mathbb{R}^{N_x \times 1 + \mathrm{N}_u}$ dan $W \in \mathbb{R}^{N_x \times N_x}$ masing-masing adalah input dan matriks bobot berulang dan $\alpha[0,1]$ adalah *leaking rate*. Pembungkus *sigmoid* dapat digunakan selain tanh. Model ini juga kadang-kadang digunakan tanpa *leak integration*, yang merupakan kasus khusus $\alpha = 1$ dan dengan demikian $\tilde{x}(n) \equiv x(n)$.

Lapisan fungsi linier didefinisikan sebagai

$$y(n) = W^{out}[1; u(n); x(n)] \quad (2.3)$$

Di mana $y(n) \in \mathbb{R}^{\mathrm{N}_y}$ adalah *output* jaringan, $W^{out} \in \mathbb{R}^{N_x \times (1 + N_u + N_x)}$ adalah matriks bobot, dan $[.;.;.]$ adalah singkatan dari vektor

vertikal atau matriks gabungan. Sebuah nonlinier tambahan dapat diterapkan ke $y(n)$ di persamaan 2.3, serta sebagai koneksi *feedback* W^{fb} dari $y(n-1)$ ke $\tilde{x}(n)$ di persamaan 2.1. Representasi grafis ESN yang menggambarkan notasi dan ide untuk pelatihan ditampilkan pada Gambar 2.5.



Gambar 2.5 *Echo State Network Model*(Lukoševičius, 2012)

Metode asli *Reservoir Computing* yang diperkenalkan dengan ESN sebagai beirkut:

1. Menghasilkan *reservoir* RNN yang besar dan acak (W^{in}, W, α);
2. Menjalankan dengan *input* pelatihan $u(n)$ dan mengumpulkan status aktivasi *reservoir* yang sesuai $x(n)$;
3. Menghitung bobot *readout* linier dari W^{out} dari *reservoir* menggunakan regresi linier, meminimalkan MSE antara $y(n)$ dan $y^{target}(n)$;
4. Menggunakan jaringan terlatih pada data *input* baru $u(n)$ menghitung $y(n)$ dengan menggunakan hasil *output* bobot W^{out} yang terlatih.

2.3.2 Memproduksi Sebuah *Reservoir*

Demi menghasilkan *reservoir* yang baik, penting untuk mengetahui dan memahami fungsi apa yangn tersedia.

2.3.2.1 Fungsi dari *Reservoir*

Dalam praktinya, penting untuk diingat bahwa *reservoir* bertindak sebagai ekspansi non linier dan sebagai memori *input* $u(n)$ secara bersamaan. Ada hubungan antara metode RC dan kernel dalam ML. *Reservoir* dapat dilihat sebagai ekspansi non linier berdimensi tinggi $x(n)$ dari input sinyal $u(n)$. Pada saat yang

sama, *reservoir* berfungsi sebagai memori, menyediakan konteks temporal. Lapisan kedua dari ESN disebut *reservoir*. ESN menggunakan arsitektur atau *framework* dari *resevoir computing*.

Ketika kedua aspek digabungkan, *reservoir* sebagai sistem dinamika *input* pendorong harus menyediakan ruang sinyal yang cukup dan relevan di $x(n)$, sehingga $y^{target}(n)$ dapat diperoleh sesuai yang diinginkan secara kombinasi linier. Namun ada beberapa masalah ketika mengatur parameter *reservoir* dan akan dijelaskan lebih rinci pada bagian 2.3.2.2.

2.3.2.2 Parameter Global

Mengingat model RNN persamaan 2.1, 2.3, *reservoir* didefinisikan oleh tuple (W^{in}, W, α). Matriks *input* dan koneksi berulang W^{in} dan W dihasilkan secara acak sesuai dengan beberapa parameter yang digunakan dan *leaking rate* α dipilih sebagai parameter bebas itu sendiri.

Dalam analogi dengan *machine learning* lain, dan terutama *Neural Networks*, pendekatan apa yang disebut "parameter", bisa disebut juga "*meta parameter*" atau "*hyper parameter*", karena mereka bukan bobot koneksi konkret tetapi parameter yang mengatur distribusinya.

Mendefinisikan parameter global dari *reservoir* adalah: ukuran N_x, *sparsity*, distribusi dari elemen bukan nol, dan jari-jari spektrum W; *scaling* (-s) dari W^{in}; dan *leaking rate* α.

1. Ukuran dari *reservoir*

   Satu parameter yang sangat jelas penting dari persamaan 2.1, 2.2 adalah N_x, jumlah unit-unit dalam *reservoir*. Semakin besar *reservoir*, semakin baik kinerja yang dapat diperoleh, asalkan langkah-langkah pengaturan yang tepat diambil terhadap *overfitting*. Karena pelatihan dan menjalankan ESN secara komputasi lebih ringan dibandingkan dengan pendekatan RNN. Semakin besar ruang sinyal *reservoir* $x(n)$, semakin mudah menemukan kombinasi linier dari sinyal untuk mendekati $y^{target}(n)$. *Reservoir* bisa menjadi terlalu besar ketika tugasnya ringan dan tidak cukup data yang tersedia $T < 1 + N_U + N_x$.

Pengaturan parameter global seringkali memerlukan beberapa percobaan, sehingga masing-masing tidak boleh menghabiskan terlalu banyak waktu. Parameter yang baik biasanya dapat ditransfer ke *reservoir* yang lebih besar.

Batas bawah untuk ukuran *reservoir* N_x secara kasar dapat diperkirakan dengan mempertimbangkan jumlah dari nilai riil independen. Jumlah maksimal nilai yang disimpan, yang disebut kapasitas memori dalam ESN tidak dapat melebihi N_x.

2. *Sparsity* dari *reservoir*

   Pada publikasi ESN yang asli, disarankan untuk membuat koneksi *reservoir* yang jarang, yaitu membuat sebagian besar elemen di W^{in} sama dengan nol. Koneksi yang jarang cenderung membuat kinerja lebih baik. *Sparsity* dari *reservoir* tidak memengaruhi kinerja terlalu banyak dan parameter ini jarang dioptimalkan. Namun, *sparsity* memungkinkan pembaruan *reservoir* cepat jika representasi matriks jarang yang digunakan.

3. Distribusi dari elemen bukan nol

   Matriks W biasanya dihasilkan jarang, dengan elemen bukan nol memiliki kesamaan., diskrit *bi-value*, atau distribusi normal yang berpusat di sekitar nol Distribusi diskrit *bi-value* cenderung memberikan sinyal yang kurang bagus, tetapi membuat analisis yang terjadi di *reservoir* lebih mudah.

   Matriks *input* W^{in} biasanya dihasilkan sesuai dengan jenis distribusi yang sama dengan W, tetapi biasnaya lebih padat.

4. *Spectral Radius*

   Salah satu parameter global paling sentral dari ESN adalah *spektra radiusl* dari koneksi matrik *reservoir* W. Misalnya, nilai eigen absolut maksimal dari matriks.

   Biasanya ketika *random sparse* W dihasilkan, *spectral radius* $\rho(W)$ dihitung. Kemudian W dibagi dengan $\rho(W)$ untuk menghasilkan matriks satuan *spectral radius*. Kemudian dengan mudah diskalakan

dengan *spectral radius* tertinggi yang akan ditentukan dalam prosedur penyetelan.

Agar ESN dapat bekerja, *reservoir* harus memenuhi apa yang disebut properti *echo state*. Keadaan *reservoir* $x(n)$ harus secara unik didefinisikan oleh *fading history* dari *input* $u(n)$. Dengan kata lain, *input* $u(n)$ yang cukup panjang, *reservoir* $x(n)$ tidak boleh tergantung pada kondisi awal yang ada sebelum *input*. Nilai $\rho(W)$ yang besar dapat menyebabkan *reservoir* menampung beberapa titik tetap, periodik, dan bahkan kacau, melanggar properti *echo state*.

Meskipun mungkin untuk melanggar properti *echo state* bahkan dengan $\rho(W) < 1$. Lebih penting lagi, properti *echo state* sering berlaku $\rho(W) \geq 1$ untuk *input* $u(n)$ bukan nol. Dalam praktiknya $\rho(W)$ harus dipilih untuk memaksimalkan kinerja, dengan nilai 1 berfungsi sebagai titik referensi awal.

5. *Input Scaling*

Penskalaan dari bobot *input* matriks W^{in} adalah parameter kunci lainnya untuk mengoptimalkan ESN. Agar W^{in} tersebar secara merata, biasanya ditentukan *input scaling* a sebagai rentang interval $[-a; a]$ dari sampel nilai W^{in}.

Untuk memiliki sejumlah kecil parameter yang dapat disesuaikan secara bebas. Seringkali semua kolom dari W^{in} diskalakan bersama menggunakan nilai penskalaan tunggal. Variasi jumlah parameter global yang bebas untuk ditetapkan ke W^{in} dari 1 hingga $N_u + 1$.

Disarankan dari publikasi asli ESN untuk menskalakan dan menggeser *input* data, mengoptimalkan besarnya keduanya. Dapat dicapai dengan menskalakan bobot *input* dari *input* aktif dan bias. Normalisasi data *input* disarankan untuk ESN seperti halnya pendekatan ML lainnya. Dianjurkan untuk menormalisasikan data dan dapat membantu menjaga *input* $u(n)$ untuk menghindari *outlier*.

Untuk model yang linier, W^{in} harus berkuran kecil, membiarkan unit beroperasi di sekitar titik 0 ketika aktivasi $\tanh(.)$. Untuk W^{in} besar, unit akan mudah jenuh mendekati nilai 1 dan -1. Sementara $\rho(W)$ juga memengaruhi model nonlinier, aktivasi *reservoir* menjadi tidak stabil ketika meningkatkan $\rho(W)$.

6. *Leaking rate*

Leaking rate α dari node *reservoir* dapat dianggap sebagai kecepatan dinamika pembaruan *reservoir*. Dinamika pembaruan *reservoir* dalam waktu kontinu dapat digambarkan sebagai *Ordinary Differential Equation* (ODE).

$$x = -x + \tanh(W_{in}[1; u] + W_x) \quad (2.4)$$

Jika menggunakan diskritisasi waktu *Euler* dari ODE ini, gunakan persamaan ini,

$$\frac{\Delta x}{\Delta t} = \frac{x(n+1) - x(n)}{\Delta t} \approx x, \quad (2.5)$$

Secara ekuivalen, α dapat dimasukkan sebagai waktu yang konstan dalam persamaan 2.4, jika menjaga $\Delta t = 1$. Kemudian $x(n)$ tidak pernah keluar dari interval (-1,1). Ini bisa sulit dan subyektif untuk ditentukan dalam beberapa kasus. Terutama, ketika skala waktu $u(n)$ dan $y^{target}(n)$ sangat berbeda. Ini adalah satu parameter global yang akan disetel coba-coba. Beberapa kontribusi bahkan menyarankan penerapan filter yang lebih kuat. Dalam beberapa kasus menetapkan α kecil, dengan demikian menginduksi dinamika lambat dari $x(n)$, dapat secara drastis meningkatkan durasi memori jangka pendek dalam ESN.

2.3.3 *Training Readouts*

Karena *readouts* dari ESN biasanya linier dan *feed-forward*, persamaan 2.3 dapat ditulis dalam notasi matriks sebagai,

$$y(n) = W^{out}X, \quad (2.6)$$

Di mana $Y \in \mathbb{R}^{N_y \times T}$ adalah semua $y(n)$ dan $X \in \mathbb{R}^{(1+N_u+N_x) \times T}$ adalah semua $[1; u(n); x(n)]$ diproduksi dengan menghadirkan *reservoir* dengan $u(n)$,

keduanya dikumpulkan ke masing-masing matriks dengan menggabungkan vektor-kolom secara horizontal selama pelatihan $n = 1, \dots, T$.

Menemukan bobot optimal W^{out} dengan meminimalkan *squared error* antara $y(n)$ dan $y^{target}(n)$ sama dengan memecahkan masalah sistem persamaan linier yang biasanya ditentukan secara berlebihan.

$$Y^{target} = W^{out} X, \tag{2.7}$$

Di mana $Y^{target} \epsilon \mathbb{R}^{N_y \times T}$ adalah semua $y(n)$, sehubungan dengan W^{out} kuadrat kecil, seperti kasus regresi linier. Dalam konteks ini X dapat disebut *design matrix*. Sistem ini terlalu ditentukan, karena biasanya $T \gg 1 + N_u + N_x$.

Ada beberapa cara yang dapat digunakan untuk menyelesaikan persamaan 2.7. Solusi yang paling *universal* dan stabil untuk persamaan 2.7 dalam konteks ini adalah *ridge regression*, juga dikenal sebagai regresi dengan *Tikhonov* regulasi:

$$W^{out} = Y^{target} X^t (X X^t + \beta I)^{-1}, \tag{2.8}$$

Di mana β adalah koefisien regularisasi dan I adalah matriks identitas..

2.4 *Harmony Search*

Harmony Search (HS) adalah algoritma pencarian *meta-heuristik* yang mencoba meniru proses improvisasi musisi dalam menemukan harmoni yang sesuai (Askarzadeh, dkk., 2017). HS mudah diterapkan, menyatu dengan cepat untuk menemukan solusi optimal dan cukup baik dalam waktu komputasi yang tidak terlalu lama. HS menunjukkan kinerja yang menjanjikan dalam masalah optimisasi. Algoritma *meta-heuristik* bisa menjadi alternatif yang efisien untuk menyelesaikan masalah optimasi yang kompleks. Sebuah harmoni didefinisikan oleh hubungan khusus antara beberapa gelombang suara yang mempunyai frekuensi berbeda. Kualitas improvisasi harmoni ditentukan oleh estimasi estetika. Untuk meningkatkan estimasi estetika dan menemukan harmoni terbaik, para musisi melakukan latihan demi latihan.

Ada kesamaan antara improvisasi musisi dan optimisasi. Optimisasi tujuan utamanya adalah untuk menemukan *global optimum* dari fungsi tujuan yang sedang dipertimbangkan dengan menyetel sejumlah variabel keputusan yang telah ditentukan. Dalam masalah optimisasi variabel keputusan membuat vektor solusi. Kemudian, nilai-nilai variabel keputusan dimasukkan ke dalam fungsi objektif dan

kualitas vektor solusi dihitung. Vektor solusi diperbarui selama iterasi terjadi hingga *global optimum* diperoleh.

Perbandingan prosesimprovisasi musik dan optimisasi terdapat beberapa kesamaan. Pertama, dalam proses musik, kualitas harmoni ditentukan oleh estimasi estetika. Dalam proses optimisasi kualitas vektor solusi ditentukan nilai fungsi objektif. Kedua, dalam proses musik, tujuan utamanya adalah untuk mendapatkan harmoni terbaik. Dalam proses optimisasi tujuan utamanya adalah untuk mendapatkan *global optimum*. Ketiga, dalam proses musik, musisi mengubah nada instrumen. Algoritma optimasi mengubah nilai-nilai varianel keputusan. Keempat, dalam proses musik, setiap upaya untuk memainkan harmoni disebut latihan. Dalam optimasi, setiap upaya untuk memperbarui vektor solusi disebut iterasi.

2.4.1 Implementasi HS untuk Optimasi

Optimasi merupakan usaha untuk mendapatkan solusi terbaik di antara banyak solusi. Algoritma HS, solusi yang layak disebut harmoni dan setiap variabel keputusan solusinya adalah sesuai dengan yang ditetapkan. HS termasuk *memory harmony* (HM) di mana sejumlah harmoni (N) yang telah ditentukan untuk disimpan. Misalkan tujuannya adalah untuk meminimalkan/memaksimalkan fungsi *fitness* (f). Masalah optimasi didefinisikan sebagai berikut:

$$\Delta f(x^*) = 0 \tag{2.9}$$

Di mana f adalah fungsi *fitness*, x^* adalah calon penyelesaian atau titik optimal. Bila $\Delta f(x^*) = 0$ dan $H(x^*)$ definit positif, maka x^* titik maksimum. Bila $\Delta f(x^*) = 0$ dan $H(x^*)$ definit negatif, maka x^* titik minimum.

2.4.2 Inisialisasi HM

HS pada awalnya, N harmoni diproduksi di ruang pencarian dan disimpan di HM. Tabel 2.1 menunjukkan struktur dari HM. Seperti yang ditunjukkan pada Tabel 2.1, harmoni i dapat ditentukan oleh sebuah vektor, harmoni $i = [x_{i,1}, x_{i,2}, \dots, x_{i,d}]$. Untuk menginisialisasi HM, persamaan 2.12 dapat digunakan. Kolom terakhir dari HM adalah nilai fungsi *fitness* yang sesuai dengan setiap

harmoni. Misalnya, $f1$ adalah nilai fungsi *fitness* untuk harmoni 1. Nilai ini dihitung dengan memasukkan variabel keputusan harmoni 1 ke dalam fungsi *fitness*.

Tabel 2.1 Struktur dari HM

	x_1	x_2	…	x_d	f
Harmony 1	$x_{1,1}$	$x_{1,2}$	…	$x_{1,d}$	f_1
Harmony 2	$x_{2,1}$	$x_{2,2}$	…	$x_{2,d}$	f_2
⋮	⋮	⋮		⋮	⋮
Harmony *N*	$x_{N,1}$	$x_{N,2}$	…	$x_{N,d}$	f_N

$$x_{i,j} = l_j + rand \times (u_i - l_j) \quad i = 1,2,\dots,N; \quad j = 1,2,\dots,d \qquad (2.10)$$

Di mana l_j dan u_j adalah batas bawah dan atas dari variabel keputusan j, dan $rand$ adalah angka acak dengan distribusi nilai dari [0 1]. Secara matematis, HM ditunjukkan dengan ekspresi berikut:

$$HM = \begin{bmatrix} Harmony\ 1 \\ Harmony\ 2 \\ \vdots \\ Harmony\ N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,d} \end{bmatrix} = \begin{bmatrix} f1 \\ f2 \\ \vdots \\ fN \end{bmatrix} \qquad (2.11)$$

2.4.3 Improvisasi Harmoni Baru

Langkah selanjutnya adalah mengimprovisasi harmoni baru, $x_{new} = [x_{new,1}, x_{new,2}, \dots, x_{new,d}]$. Fitur utama dari algoritma HS dibandingkan dengan *meta-heuristik* lain seperti *Genetic Algorithm* (GA) adalah harmoni baru dihasilkan dengan menggunakan semua harmoni yang ada. Untuk pembuatan variabel keputusan j, beberapa prosedur dilakukan hingga diperoleh harmoni baru.

Tahap pertama, angka acak dengan distribusi yang seragam dari [0 1] dihasilkan ($rand$). Jika $rand >$ HMCR, variabel keputusan dari harmoni baru dihasilkan secara acak oleh persamaan 2.14 *Harmony Memory Considering Rate* bervariasi antara 0 dan 1.

$$x_{new,j} = l_j + rand \times (u_i - l_j) \qquad (2.12)$$

Sebaliknya, jika $rand \leq$ HMCR, salah satu harmoni yang disimpan dalam HM dipilih secara acak, misalnya k di mana $1 \leq k \leq N$. Kemudian, $x_{new,j}$ dipilih oleh nilai harmoni k yang sesuai dari HM sebagai persamaan 2.15.

$$x_{new,j} = x_{k,j} \tag{2.13}$$

Tahap kedua, melepaskan dari *local optimum*, HS menggunakan mekanisme penyesuaian *pitch* di mana notasi improvisasi dapat digeser ke nilai tetangga dengan kisaran yang memungkinkan. Dalam HS, ada parameter bernama *Pitch Adjusting Rate* (PAR) yang bervariasi antara 0 dan 1. Nilai-nilai kecil PAR menyebabkan memiliki mekanisme penyesuaian *pitch* yang lemah dan nilai-nilai PAR yang besar menghasilkan mekanisme penyesuaian *pitch* yang kaya. Untuk mekanisme penyesuaian *pitch*, setelah tahap pertama, nomor acak dari [0 1] dengan distribusi yang seragam dihasilkan ($rand$). Jika $rand \leq$ PAR, notasi improvisasi harus digeser ke nilai tetangga dengan menggunakan persamaan 2.16. Jika $rand >$ PAR, notasi improvisasi tidak berubah.

$$x_{new,j} = x_{new,j} + bw \times (rand - 0.5) \times |u_i - l_j| \tag{2.14}$$

Di mana bw disebut *bandwith* generasi dan $rand$ adalah angka acak antara 0 dan 1 dengan distribusi seragam. Istilah $(rand - 0.5)$ menghasilkan angka acak dari $[-0.5\ 0.5]$. Karena pengguna tidak tahu bahwa lebih baik untuk meningkatkan nilai variabel keputusan atau menurunkannya, istilah $(rand - 0.5)$ digunakan secara acak untuk memilih gerakan. Dalam persamaan 2.14, istilah $|u_i - l_j|$ digunakan untuk mengontrol skala variabel keputusan karena dalam masalah optimasi, skala variabel keputusan dapat bervariasi secara signifikan seperti -10^4 hingga 10^4 dalam satu dimensi dan -10^{-5} ke 10^{-5} di yang lain.

2.4.4 *Replacement*

Setelah bagian 2.4.3 memiliki harmoni baru yang layak. Fungsi *fitness* dari harmoni baru (f_{new}) dihitung. Pada bagian ini dibandingkan harmoni baru dan harmoni terburuk yang disimpan di HM dalam hal nilai fungsi *fitness*. Jika f baru

lebih baik dari fit_h, harmoni h akan dihapus dari HM dan diganti harmoni baru. Selian itu, jika f_{new} lebih buruk daripada fit_h, harmoni baru ditinggalkan.

2.4.5 *Stopping Criterion*

Kebanyakan cara berhenti dari algoritma optimasi adalah untuk mencapai jumlah iterasi yang telah ditentukan (t_{max}). Ketika kriteria ini dipenuhi, algoritma diakhiri dan diteruskan ke bagian 2.4.6. Jika tidak, bagian 2.4.3 dan 2.4.4 diulangi sampai kriteria untuk berhenti terpenuhi.

2.4.6 Hasil Akhir

Harmoni terbaik yang disimpan dalam HM dikembalikan sebagai solusi optimal dari masalah yang sedang dipertimbangkan. HS memiliki tiga parameter, yaitu HMCR, PAR, dan bw. Peran parameter-parameter ini pada kinerja HS adalah sebagai berikut:

a) HMCR: HMCR memiliki nilai [0 1], menunjukkan kemungkinan menggunakan nilai historis yang disimpan dalam HM untuk memainkan catatan. Nilai HMCR yang kecil menghasilkan pencarian acak (dengan probabilitas 1-HMCR) dan sebaliknya. Misalnya, HMCR 0,9 dan akan berbunyi secara acak dengan probabilitas 0,1 dari kisaran yang mungkin.
b) PAR: Dengan nilai yang yang dimiliki [0 1] ini, setiap nilai yang diimprovisasi dari HM, memiliki perubahan untuk digantikan oleh nilai yang terletak di sekitar nilai yang dipilih dari HM. Dalam HS, peluang ini disediakan dan dikendalikan oleh probabilitas PAR. Akibatnya, nilai-nilai PAR yang besar meningkatkan kemungkinan penyesuaian *pitch* dan sebaliknya.
c) *bw*: Jika mekanisme penyesuaian *pitch* dipilih, nilai *bw* mengontrol ukuran langkah gerakan. Dengan menggunakan nilai besar *bw*, jarak antar niali baru dan nilai HM meningkat. Dapat menyesuaikan pencarian global dam lokal dengan nilai bw.

2.5 Metode Evaluasi

Evaluasi yang digunakan dalam penelitian ini menggunakan kesalahan nilai prediksi $\hat{y}$ terhadap y. Memungkinkan ada residual yang didapatkan dari

perbedaan $y - \hat{y}$ dan kesalah estimasi atau prediksi. Proses prediksi menggunakan ESN diterapkan benar sehingga kesalahan prediksi dapat diminimalkan. Untuk mengetahui kinerja ESN dalam memprediksi, digunakan rumus *Root Mean Square Error* (RMSE) dan *Mean Absolute Percentage Error* (MAPE).

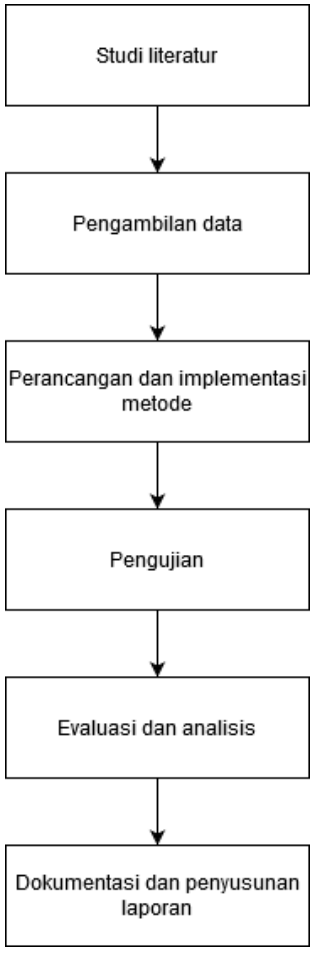
$$RMSE = E(y, y^{target}) = \frac{1}{N_y}\sum_{i=1}^{N_y}\sqrt{\frac{1}{T}\sum_{n=1}^{T}(y_i(n) - y_i^{target}(n))^2} \quad (2.15)$$

$$MAPE = E(y, y^{target}) = \frac{\sum_{i=1}^{n}\left|\left(\frac{y_i - y_i^{target}}{y_i}\right)100\right|}{n} \quad (2.16)$$

[*Halaman ini sengaja dikosongkan*]

BAB 3
METODOLOGI PENELITIAN

Bab ini akan memaparkan tentang metodologi penelitian yang digunakan pada penelitian ini, yang terdiri dari (1) studi literatur, (2) pengambilan data, (3) perancangan dan implementasi metode, (4) pengujian, (5) evaluasi dan analisis, dan (6) dokumentasi dan penyusunan laporan. Diagram alur metodologi penelitian dapat dilihat pada Gambar 3.1



Gambar 3.1 Alur metodologi penelitian

Penjelasan tahapan metode penelitian pada Gambar 3.1 akan diterangkan secara terperinci pada subbab berikut.

3.1 Studi Literatur

Tahap pertama dari penelitian dilakukan dengan mengumpulkan dan memahami berbagai publikasi hasil penelitian terbaru dan beberapa buku yang berkaitan dengan *echo state network* dan *harmony search*. Studi literatur berfokus pada *reservoir* di ESN. Sedangkan *harmony search* studi literatur difokuskan kepada pengembangannya. Solusi permasalahan dan landasan masalah didapatkan dari berbagai sumber referensi yang digunakan. Berdasarkan tahap studi literatur yang sudah dilakukan, beberapa informasi dapat diambil.

3.2 Pengambilan Data

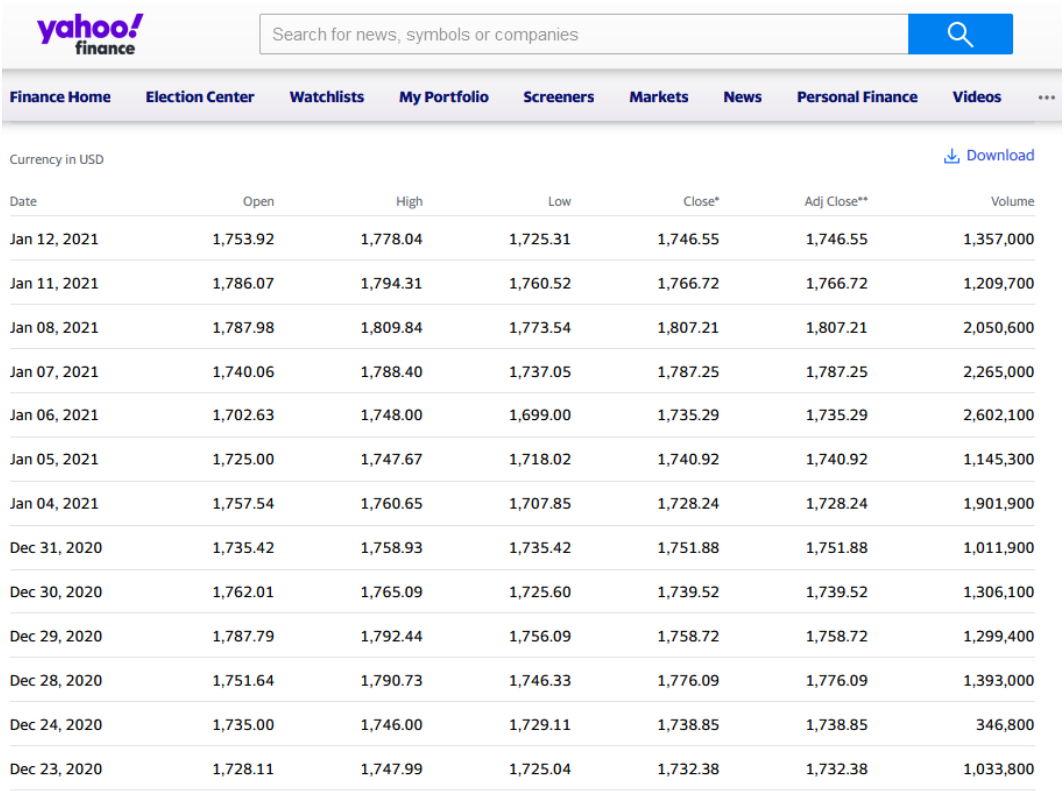
Data yang akan digunakan dalam penelitian ini, diambil dari situs *Yahoo Finance*. Jenis data *time series* yang digunakan pada penelitian ini dan data berbentuk multivariat, berarti mempunyai beberapa atribut. Contoh data yang akan dipakai ditunjukkan pada Tabel 3.1.

Tabel 3.1 Dataset *google stock price*

Date	**Open**	**High**	**Low**	**Close**	**Adj Close**	**Volume**
2005-06-15	136,986	138,132	133,215	136,886	136,886	419226
2005-06-16	136,617	138,630	136,025	138,201	138,201	250181
2005-06-17	138,979	139,626	137,434	139,626	139,626	209469
2005-06-18	137,529	143,297	135,357	142,814	142,814	422069
2005-06-19	143,497	144,607	141,952	143,382	143,382	303779
2005-06-20	126,986	132,132	143,215	196,886	136,886	429226
2005-06-21	136,617	131,630	146,025	188,201	148,201	240181
2005-06-22	128,979	134,626	197,434	179,626	159,626	259469
2005-06-23	117,529	149,297	195,357	192,814	172,814	482069
2005-06-24	143,497	142,607	181,952	193,382	193,382	393779

Contoh salah satu data yang digunakan untuk proses prediksi menggunakan deret waktu dengan perulangan harian. Terdapat 7 atribut, namun tidak digunakan semua, kemudian ditentukan atribut mana yang akan diprediksi dan atribut mana yang menjadi prediktor. Pada penelitian ini atribut *Open* akan diprediksi. Sedangkan atribut lainnya sebagai prediktor. Atribut predictor diseleksi

lagi, karena dipilih prediktor yang mempunyai korelasi dengan atribut *Open* menggunakan uji coba *pearson's correlation test*. Tampilan *yahoo finance* ditunjukkan pada Gambar 3.2. Contoh pada penelitian ini menggunakan data Alphabet Inc. atau *google stock price*. Bisa jadi penggunaan setiap jenis data berbeda, seperti tingkat korelasi antar atribut.



Date	Open	High	Low	Close*	Adj Close**	Volume
Jan 12, 2021	1,753.92	1,778.04	1,725.31	1,746.55	1,746.55	1,357,000
Jan 11, 2021	1,786.07	1,794.31	1,760.52	1,766.72	1,766.72	1,209,700
Jan 08, 2021	1,787.98	1,809.84	1,773.54	1,807.21	1,807.21	2,050,600
Jan 07, 2021	1,740.06	1,788.40	1,737.05	1,787.25	1,787.25	2,265,000
Jan 06, 2021	1,702.63	1,748.00	1,699.00	1,735.29	1,735.29	2,602,100
Jan 05, 2021	1,725.00	1,747.67	1,718.02	1,740.92	1,740.92	1,145,300
Jan 04, 2021	1,757.54	1,760.65	1,707.85	1,728.24	1,728.24	1,901,900
Dec 31, 2020	1,735.42	1,758.93	1,735.42	1,751.88	1,751.88	1,011,900
Dec 30, 2020	1,762.01	1,765.09	1,725.60	1,739.52	1,739.52	1,306,100
Dec 29, 2020	1,787.79	1,792.44	1,756.09	1,758.72	1,758.72	1,299,400
Dec 28, 2020	1,751.64	1,790.73	1,746.33	1,776.09	1,776.09	1,393,000
Dec 24, 2020	1,735.00	1,746.00	1,729.11	1,738.85	1,738.85	346,800
Dec 23, 2020	1,728.11	1,747.99	1,725.04	1,732.38	1,732.38	1,033,800

Gambar 3.2 Yahoo Finance

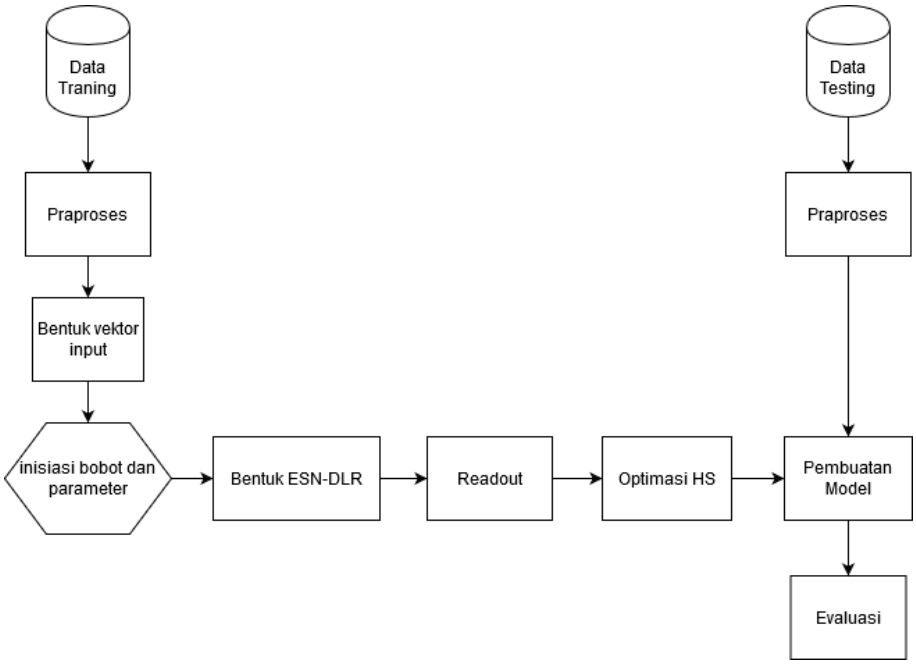
Data kedua diambil dari sumber yang sama yaitu *Yahoo Finance*, menggunakan *amazon stock price*. Perlakuan data sama seperti data pertama, data kedua ditunjukkan pada Tabel 3.2.

Tabel 3.2 Dataset *amazon stock price*

Date	Open	High	Low	Close	Adj Close	Volume
18/01/2006	43.189.999	44.590.000	43.099.998	44.320.000	44.320.000	8743300
19/01/2006	44.490.002	45.240.002	44.299.999	44.450.001	44.450.001	5156900
20/01/2006	44.230.000	44.360.001	43.200.001	43.919.998	43.919.998	8307400
23/01/2006	44.180.000	44.750.000	43.500.000	43.730.000	43.730.000	5752800
24/01/2006	43.660.000	44.430.000	43.419.998	44.020.000	44.020.000	4342400
25/01/2006	44.049.999	44.470.001	43.340.000	43.599.998	43.599.998	4287100
26/01/2006	43.950.001	44.779.999	43.790.001	44.680.000	44.680.000	4889500
27/01/2006	44.549.999	45.220.001	44.529.999	45.220.001	45.220.001	4174000
30/01/2006	45.410.000	45.970.001	44.740.002	44.959.999	44.959.999	4099800
31/01/2006	44.759.998	45.209.999	44.250.000	44.820.000	44.820.000	4056000

3.3 Perancangan dan Implementasi Metode

Diagram alir perancangan metode dalam penilaian hasil prediksi terdiri dari beberapa tahap seperti terlihat pada Gambar 3.3. Tahap pertama, mempersiapkan data yang digunakan sebagai data *training* dan *testing*. Data awal dilakukan praproses dahulu. Proses berikutnya adalah membentuk vektor *input* data. Setelah didapatkan nilai *input*, dilanjutkan membuat nilai acak terhadap bobot yang menghubungkan antara lapisan *input* dan *reservoir*. Hubungan membentuk ukuran yang sesuai antara jumlah *input* dan jumlah neuron dalam *reservoir*.



Gambar 3.3 Diagram alir metode usulan

Setelah itu, mengetahui nilai dari setiap neuron yang ada dalam *reservoir*. Dimulai dengan perulangan pertama, jika sudah diketahui nilainya, arah hubungan berbalik dan membentuk perulangan yang kedua. Setelah dari lapisan *reservoir*, pindah ke lapisan *readout* dan mendapatkan hasil akhir berupa bobot *output* yang selanjutnya masuk ke optimasi *harmony search* sampai mendapat nilai evaluasi terbaik. Akhir dari tahapan adalah evaluasi metode yang dirancang. Tahap-tahap usulan akan dijelaskan pada subbab berikut:

3.3.1 Praproses

Dataset yang digunakan terdiri dari beberapa atribut dan mempunyai satuan atau selisih angka yang jauh. Mulai satuan hingga ratusan, sehingga dapat menurunkan atau memperburuk performa mesin prediksi. Maka dari itu, perlu normalisasi data menggunakan metode *MaxMin*.

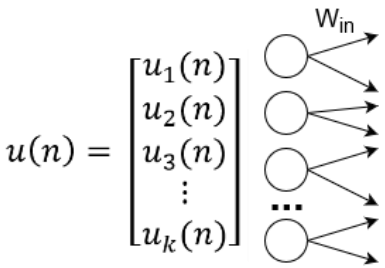
Tabel 3.3 Data setelah normalisasi

Open	**High**	**Low**	**Close**	**Adj Close**	**Volume**
0,450	0,553	0,123	0,723	0,723	1,000
0,420	0,653	0,113	0,613	0,613	0,991
0,423	0,153	0,213	0,443	0,443	0,899
0,412	0,353	0,103	0,411	0,411	0,900
0,881	0,189	0,001	0,900	0,900	0,923
0,450	0,553	0,123	0,723	0,723	1,000
0,420	0,653	0,113	0,613	0,613	0,991
0,423	0,153	0,213	0,443	0,443	0,899
0,412	0,353	0,103	0,411	0,411	0,900
0,881	0,189	0,001	0,900	0,900	0,923

3.3.2 Membuat vektor *input*

Setelah data melewati tahap normalisasi pada bagian 3.3.1, setiap baris data akan dijadikan *input* berupa vektor $u(n)$. Vektor input $u(n)$ merupakan elemen dari R^{N_u}. Vektor *input* $u(n)$ selanjutnya dihubungkan dengan *reservoir* menggunakan

bobot W_{in}. Bobot W_{in} menghubungkan setiap elemen $u(n)$ tepat satu neuron dari *reservoir*.



Gambar 3.4 Vektor *input* $u(n)$

3.3.3 Inisiasi bobot dan parameter global

Setelah membentuk vektor *input* $u(n)$, vektor tersebut dihubungkan dengan bobot W_{in} yang dibentuk dengan acak. Ukuran matriks W_{in} sama seperti dua matriks yang dihubungkan. Vektor $u(n)$ memiliki ukuran N_u, sedangkan vektor *reservoir* berukuran N_x. Maka dari itu diperoleh ukuran matriks W_{in} adalah $N_x \times N_u$ atau ditulis $W_{in} \in \mathbb{R}^{N_x \times N_u}$. Nilai awal parameter global mulai ukuran *reservoir* N, *spectral radius* dimulai dengan nilai awal 1, *leaking rate*, bobot *input* W_{in}, dan *input scaling* yang mengatur jarak atau selisih antar input bobot. Permisalan menggunakan empat atribut dan ukuran *reservoir* adalah 500, maka matriks W_{in} seperti Gambar 3.5.

$$W_{in} = \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} & \cdots & W_{1,500} \\ W_{2,1} & W_{2,2} & W_{2,3} & \cdots & W_{2,500} \\ W_{3,1} & W_{3,2} & W_{3,3} & \cdots & W_{3,500} \\ W_{4,1} & W_{4,2} & W_{4,3} & & W_{4,500} \end{bmatrix}$$

Gambar 3.5 Bobot *input* W_{in}

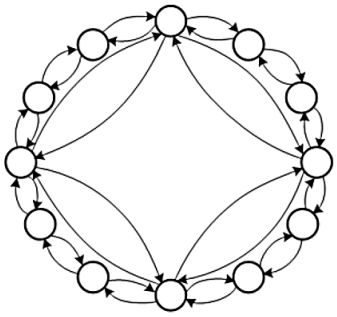
3.3.4 *Reservoir*

Hasil proses pada bagian 3.3.3 yaitu pemetaan dari vektor *input* $u(n)$ dengan W_{in} menjadi vektor *resevoir* $x(n)$. Nilai vektor $x(n)$ terpengaruh oleh setiap neuron yang ada, di mana setiap neuron dalam *reservoir* dihubungkan dengan bobot W_x. Karena setiap neuron terhubung satu sama lain sebanyak jumlah neuron atau N_x, maka W_x membentuk matrik bobot berukuran $N_x \times N_x$ atau dapat

dituliskan dengan notasi $W_x \in \mathbb{R}^{N_x \times N_x}$. Nilai setiap neuron didapatkan menggunakan persamaan 2.1 dan 2.2 pada bagian 2.3.1.

3.3.5 *Double loop reservoir*

Setelah didapatkan nilai neuron yang diperoleh pada bagian 3.3.5, dihitung kembali nilai neuron namun dengan arah sebaliknya seperti Gambar 3.6. Jadi setiap *neuron* terhubung mempunyai *forward connection* dan *feedback connection*. Nilai neuron sama seperti tahap sebelumnya, didapatkan menggunakan persamaan 2.1 dan 2.2.



Gambar 3.6 *Double loop reservoir*

Langkah *double loop reservoir* diperlihatkan pada gambar, berupa algoritma mulai terbentuknya *reservoir* koneksi pertama, kemudian dibuat perulangan sebaliknya pada neuron.

1. Tentukan jumlah neuron pada *reservoir* sebagai N. Tentukan interval neuron sebagai d.
2. Bentuk perulangan pertana dengan menghubungkan semua *internal* neuron dengan baik pada bentuk perulangan. Secara khusus $W_{i,j}$ adalah hubungan dari neuron i ke neuron j.
3. Bentuk perulangan kedua. Ambil neuron pertama sebagai titik mulai. Hubungkan dengan neuron pasangannya.

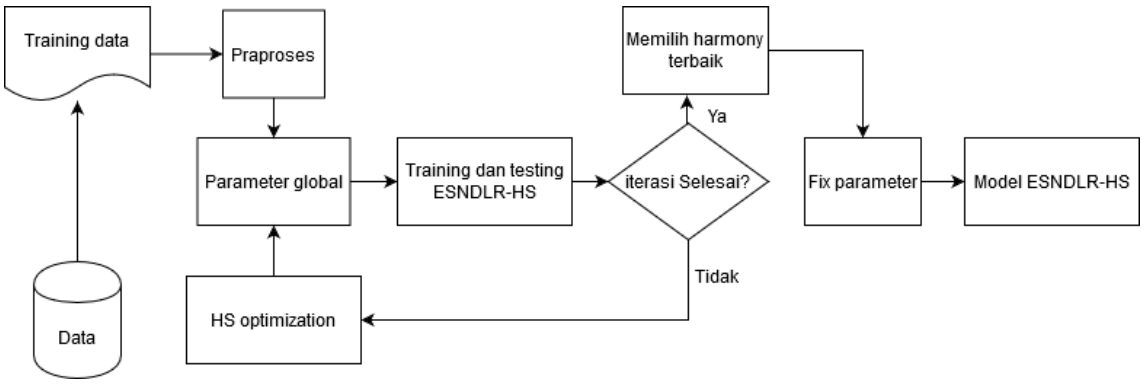
Gambar 3.7 Algoritma *double loop reservoir*

3.3.6 *Readout*

Setelah menyelesaikan tahapan pada lapisan *reservoir*, dilanjutkan menghitung *output* pada lapisan *redout*. Nilai *output* $y(t)$ adalah sebuah vektor yang berukuran N_y. Terdapat bobot W_{out} yang menghubungkan antara lapisan *reservoir* dan *readout*. Karena bobot W_{out} berada di antara lapisan *reservoir* dan *readout*, menghasilkan matriks bobot berukuran $N_y \times N_x$ atau dapat ditulisan dengan $W_{out} \in \mathbb{R}^{N_y \times N_x}$, nilai lapisan *readout* didapatkan menggunakan persamaan 2.3. Kemudian, W_{fb} atau bobot yang menghubungkan antara *readout* untuk kembali ke *reservoir* memiliki ukuran yang sama seperti W_{out}, hanya saja dilakukan *transpose*.

3.3.7 Proses *learning*

Proses terakhir setelah mendapatkan nilai dari lapisan *readout*, selanjutnya adalah proses pembelajaran pada matriks W_{out}. Hasil ini akan dimasukkan ke dalam proses algoritma *harmony search* untuk dihitung nilai *error* dari *fitness function* menggunakan MSE, kondisi akan berhenti ketika mendapatkan hasil *error* yang ditargetkan atau paling kecil, dapat dilihat pada Gambar 3.8. Inisialisasi parameter global HS seperti *Harmony Memory Size* sejumlah vektor solusi yang terbentuk, HMCR berkisar 0.7 sampai 0.99, PAR umumnya berkisar 0.1 sampai 0.5, dan *stop creation* sejumlah iterasi untuk melakukan improvisasi.



Gambar 3.8 Diagram alur ESNDLR-HS

3.4 Uji coba dan Analisa Hasil

Setelah tahapan perancangan dan implementasi metode maka pada tahap ini dilakukan uji coba untuk mengetahui kemampuan dalam memberikan penilaian hasil prediksi. Metode pengujian menggunakan perhitungan *Root Mean Square Error* (RMSE) dan *Mean Absolute Percent Error* (MAPE) . Hasil dari pengujian akan dibandingkan dengan metode ESN, RNN, dan ANN. Di mana, beberapa parameter awal didapatkan dari proses optimasi ESN. sehingga dapat diketahui perbedaan hasil pengujian dari beberapa metode sebelumnya dengan metode usulan. Dari perbandingan hasil pengujian akan dilakukan Analisa hasil dengan beberapa skenario dalam uji coba

1. Pengujian menggunakan *Echo State Network Double Loop Reservoir* dan dioptimasi menggunakan *Harmony Search*.
2. Pengujian menggunakan metode *Echo State Network*.
3. Pengujian menggunakan metode *Recurrent Neural Network*.

Output prediksinya *one step ahead* yaitu melakukan prediksi untuk mengetahui nilai persatu waktu ke depan, misalnya satu hari ke depan. Rencana implementasi akan diterapkan pada bahasa pemrograman *python* menggunakan *package* ESN sebagai dasar, yang tersedia pada *repository* untuk pemrograman *python* (pypi.org). Kemudian, dari *package* dasar yang tersedia, akan disesuaikan dengan metode yang diusulkan.

BAB 4
UJI COBA DAN PEMBAHASAN

Bab ini membahas tentang hasil uji coba dan evaluasi terhadap implementasi metode *echo state network double loop reservoir* yang dioptimasi menggunakan *harmony search* sebagaimana dijelaskan pada Bab 3. Bab ini meliputi lingkungan uji coba yang akan dijelaskan pada sub-bab 4.1. Selanjutnya, spesifikasi dataset dijelaskan pada sub-bab 4.2, pelaksanaan uji coba pada sub-bab 4.3 dan terakhir sub-bab 4.4 menjelaskan hasil dan evaluasi.

4.1 Lingkungan Uji Coba

Spesifikasi *hardware* yang digunakan untuk implementasi dan uji coba adalah Intel(R) Core™ i7-8550U CPU @1.80GHz (8 CPUs), RAM berkapasitas 8 GB, sistem operasi Windows 10 64*bit* serta kapasitas harddisk HDD 1 TB, sedangkan *software* yang digunakan untuk implementasi dari metode yang diusulkan adalah Anaconda (*Spyder* dan *Jupyter Notebook*).

4.2 Dataset

Seperti yang sudah dijelaskan pada bab 3.2, dataset yang digunakan berjenis multivariat. Berarti mempunyai beberapa atribut data. Atribut data yang menjadi prediktor harus dilakukan uji *pearson correlation*. Hasil uji tes korelasi ditunjukkan pada Tabel 4.1. Setelah atribut dipilih, dilanjutkan proses normalisasi data yang bertujuan untuk homogenitas karena dapat memengaruhi hasil prediksi.

Tabel 4.1 *Pearson's correlation*

	High	**Low**	**Close**	**Adj Close**	**Volume**
Nilai Kepercayaan	99.98%	99.98%	99.97%	99.97%	-54.76%
p-value	0.0	0.0	0.0	0.0	> 0.05

Hipotesis diterima jika p-value = 0.0 < α = 0.05, dari hasil Tabel 4.4 berarti atribut *volume* tidak dijadikan sebagai variabel independen. Tinggal 4 atribut di antaranya *high*, *low*, *close*, dan *adj close* digunakan sebagai prediktor. Sedangkan atribut yang akan diprediksi adalah *open*.

4.3 Uji Coba

Pada bab 3.4 dijelaskan bahwa pengujian akan dilakukan dalam beberapa skenario, ada 2 dataset.3 jenis metode prediksi seperti RNN, ESN, ESN-DLR. Kemudian ESN dan ESN-DLR akan dibandingkan tersendiri hingga dilakukan optimasi *harmony search*. Untuk memudahkan pengujian dan pembahasan, disusun 2 skenario besar berdasarkan metode prediksi antara lain:

- Skenario Uji Coba 1

  Uji coba dilakukan dengan membandingkan RNN, ESN, dan ESN-DLR bertujuan untuk mengetahui metode prediksi lebih baik.
- Skenario Uji Coba 2

  Uji coba dilakukan antara metode ESN dan ESN-DLR disesuai pembandingan beberapa konfigurasi parameter.
- Skenario Uji Coba 3

  Uji coba dilakukan dengan membandingkan ESN, dan ESN-DLR yang dioptimasi menggunakan *harmony search* bertujuan untuk apakah metode optimasi berhasil mendapatkan parameter yang sesuai untuk metode prediksi.

Skenario 1 fokus pada jenis model prediksi yang digunakan, nanti akan diamati bagaimana evaluasi hasilnya, metode mana yang lebih baik. Kemudian pada skenario 2, fokus pada metode ESN dan ESN-DLR. Pada skenario 2 dilakukan uji coba dengan beberapa nilai yang berbeda pada parameter. Skenario 3, dilakukan optimasi pada skenario 2, sehingga diketahui nilai parameter yang sesuai untuk model prediksi ESN dan ESN-DLR. Berikut beberapa spesifikasi dan besaran nilai yang digunakan dalam pengujian, baik untuk skenario 1, skenario 2, dan skenario 3 antara lain:

Tabel 4.2 Parameter RNN

Parameter	**Nilai**
Epoch	10, 50
Test size	33%

Tabel 4.3 Parameter ESN

Parameter	**Nilai**
Jumlah reservoir	30, 50, 100
Leaking rate	0.2
Input scaling	1.0
Spectral radius	1.0
Reservoir density	0.2
Input density	1.0
Test size	33%

Tabel 4.4 Parameter ESN-DLR

Parameter	**Nilai**
Jumlah reservoir	30, 50, 100
Leaking rate	0.2
Input scaling	1.0
Spectral radius	1.0
Reservoir density	0.2
Input density	1.0
Test size	33%

Tabel 4.5 Parameter HS

Parameter	**Nilai**
Iterasi	10
Batas bawah *leaking rate*	0.0
Batas atas *leaking rate*	1.0
Batas bawah jumlah reservoir	5
Batas bawah jumlah reservoir	100
Harmony memory	5

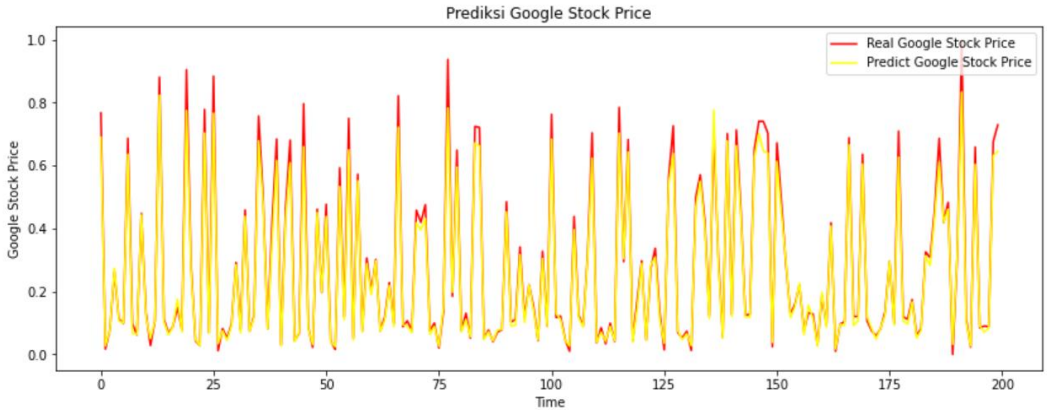
4.3.1 Skenario Uji Coba 1

Seperti sudah dijelaskan pada bab 3.2 dan 4.3, pengujian menggunakan 2 dataset dan 3 metode model prediksi (RNN, ESN, ESN-DLR). Hasil-hasil pelaksanaan pengujian skenario 1.

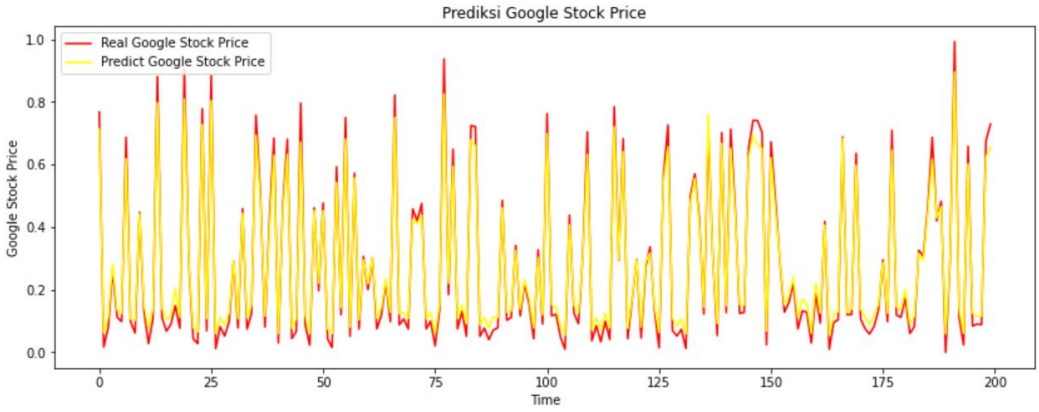
Tabel 4.6 Hasil pengujian RNN dataset 1

Uji coba	RMSE	MAPE
RNN, epoch 10	0.1237	0.2687
RNN, epoch 50	0.1158	0.2600

Dari hasil uji coba RNN pada dataset 1 *google stock price*. Penerapan epoch 50 mendapatkan hasil lebih baik senilai 0.1158 untuk RMSE dan 0.2600 untuk MAPE. Namun, selisih dengan percobaan pertama di Tabel 4.6 tidak terlalu jauh. Hasil grafik ditunjukkan pada Gambar 4.1 dan 4.2.



Gambar 4.1 RNN dataset 1, epoch 10

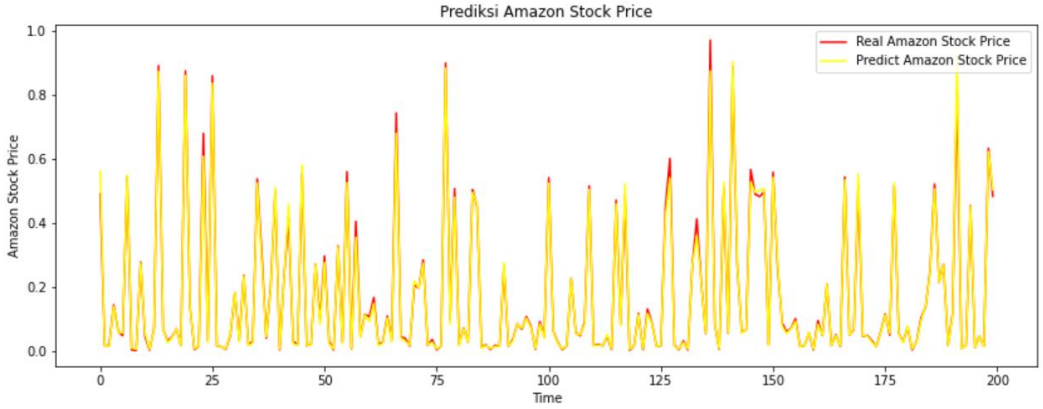


Gambar 4.2 RNN dataset 1, epoch 50

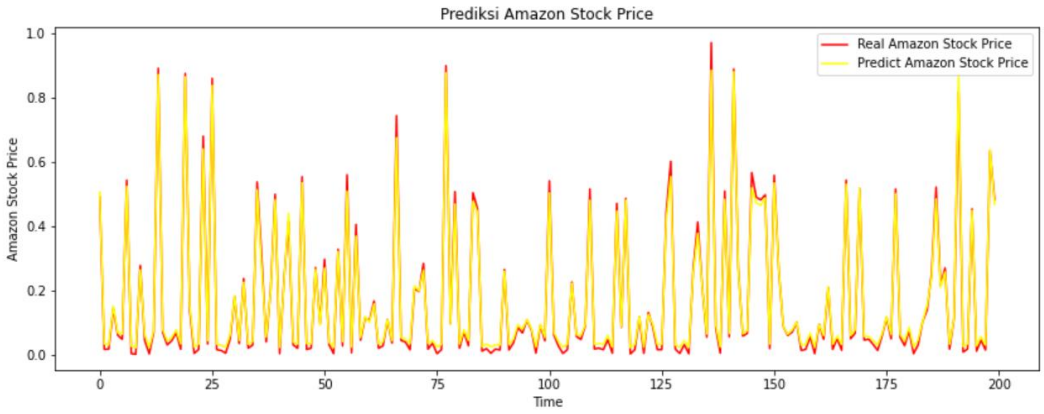
Selanjutnya untuk hasil uji coba RNN pada dataset 2 *amazon stock price*. Penerapan epoch 50 juga mendapatkan hasil lebih baik senilai 0.0963 untuk RMSE dan 0.2135 untuk MAPE. Namun, selisih dengan percobaan pertama di Tabel 4.7 tidak terlalu jauh. Hasil grafik ditunjukkan pada Gambar 4.3 dan Gambar 4.4.

Tabel 4.7 Hasil pengujian RNN dataset 2

Uji coba	RMSE	MAPE
RNN, epoch 10	0.0981	0.2152
RNN, epoch 50	0.0963	0.2135



Gambar 4.3 RNN dataset 2, epoch 10



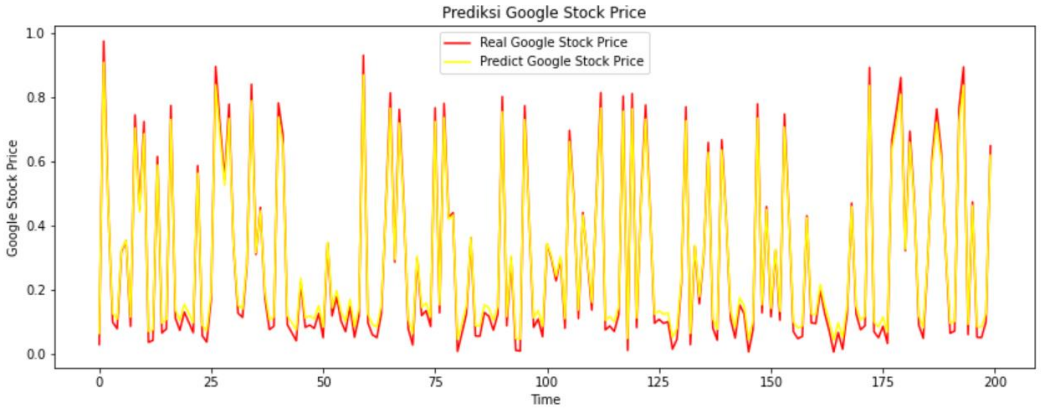
Gambar 4.4 RNN dataset 2, epoch 50

Hasil uji coba metode ESN pada dataset 1 *google stock price*. Penerapan jumlah reservoir 30 mendapatkan hasil lebih baik senilai 0.0005 untuk RMSE dan 0.0153 untuk MAPE. Jumlah reservoir yang semakin kecil mendapatkan nilai

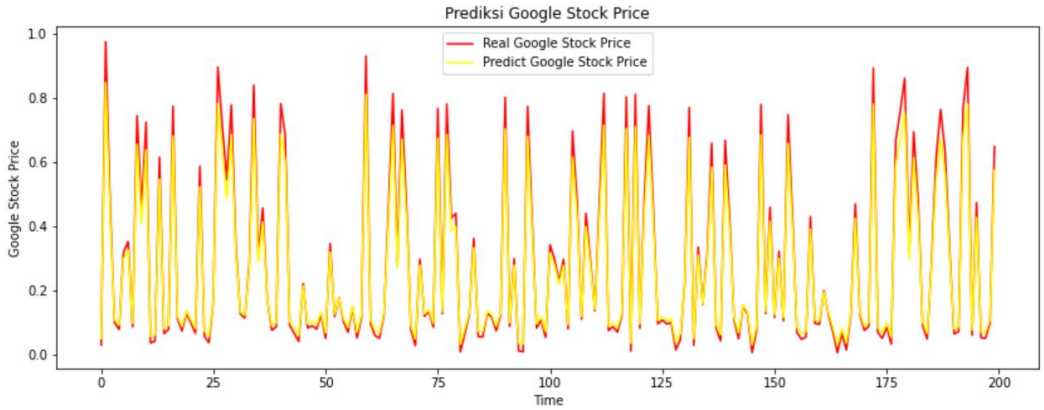
kesalahan semakin kecil pula. Hasil grafik ditunjukkan pada Gambar 4.5, Gambar 4.6 dan Gambar 4.7.

Tabel 4.8 Hasil pengujian ESN dataset 1

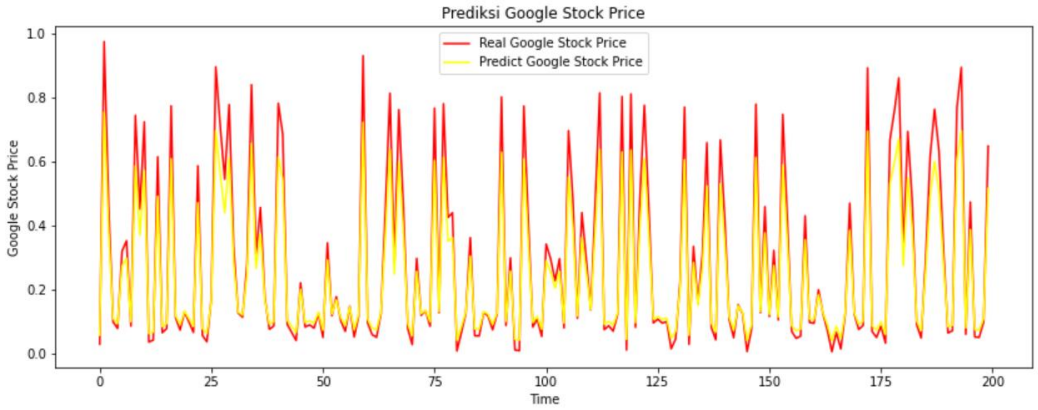
Uji coba	RMSE	MAPE
ESN, n_reservoir 30	0.0005	0.0153
ESN, n_reservoir 50	0.0023	0.3291
ESN, n_reservoir 100	0.0048	0.0565



Gambar 4.5 ESN dataset 1, reservoir 30



Gambar 4.6 ESN dataset 1, reservoir 50

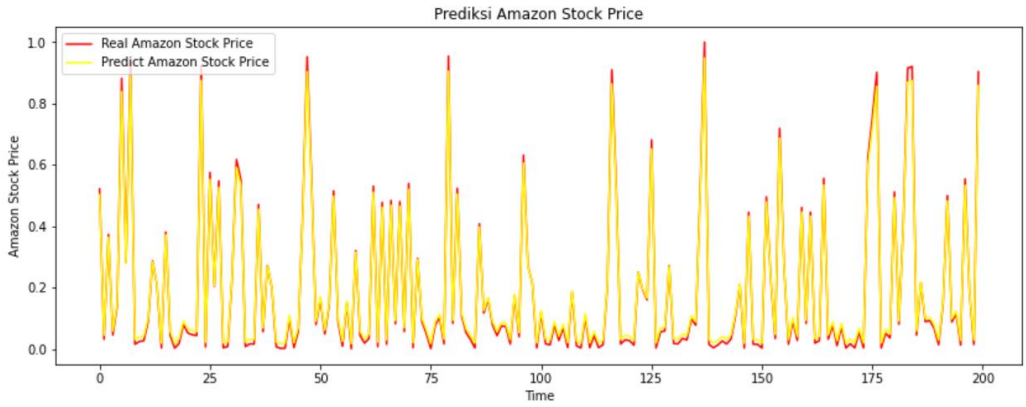


Gambar 4.7 ESN dataset 1, reservoir 100

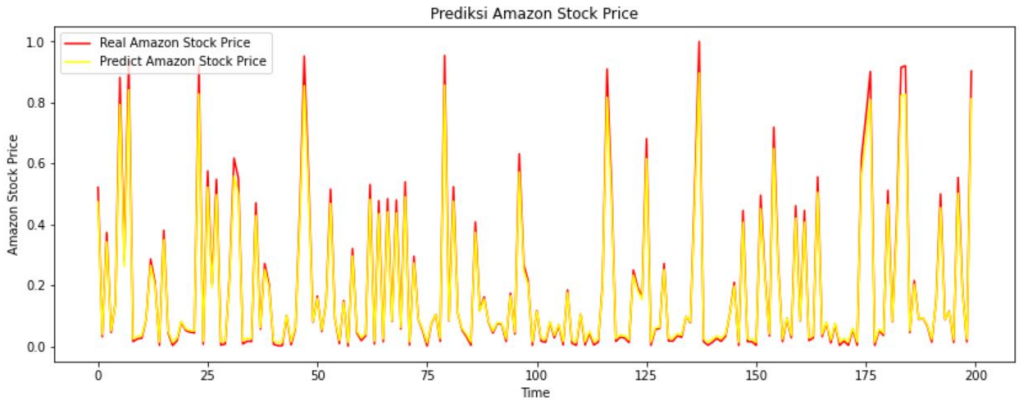
Hasil uji coba metode ESN pada dataset 2 *amazon stock price*. Penerapan jumlah reservoir 30 mendapatkan hasil lebih baik senilai 0.0004 untuk RMSE dan 0.0170 untuk MAPE. Jumlah reservoir yang semakin kecil mendapatkan nilai kesalahan semakin kecil pula, seperti halnya dataset 1. Hasil grafik ditunjukkan pada Gambar 4.8, Gambar 4.9 dan Gambar 4.10.

Tabel 4.9 Hasil pengujian ESN dataset 2

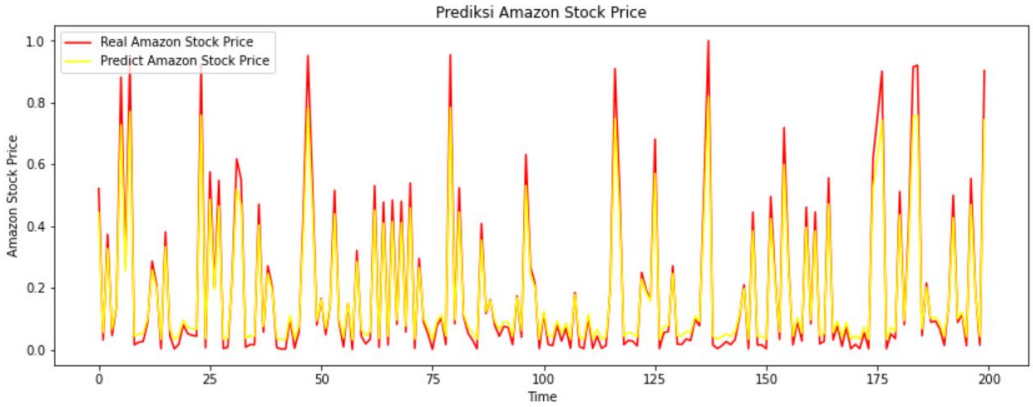
Uji coba	**RMSE**	**MAPE**
ESN, n_reservoir 30	0.0004	0.0170
ESN, n_reservoir 50	0.0009	0.0202
ESN, n_reservoir 100	0.0023	0.0332



Gambar 4.8 ESN dataset 2, reservoir 30



Gambar 4.9 ESN dataset 2, reservoir 50

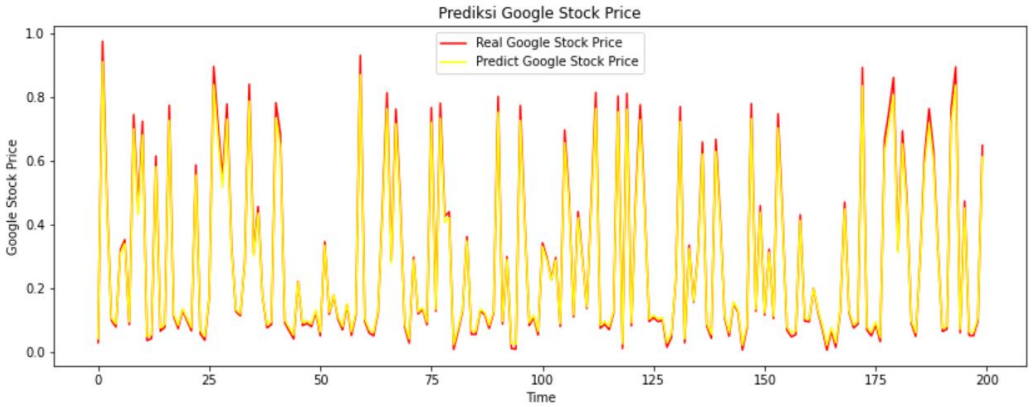


Gambar 4.10 ESN-DLR dataset 1, reservoir 100

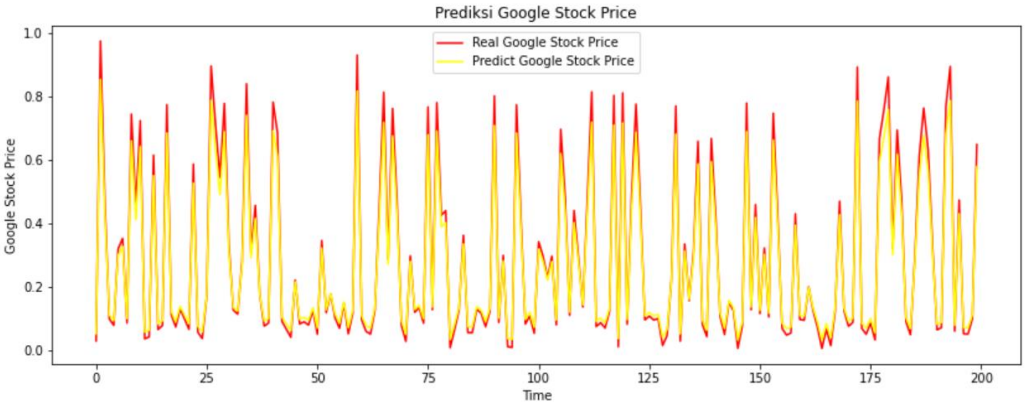
Hasil uji coba metode ESN-DLR pada dataset 1 *google stock price*. Penerapan jumlah reservoir 30 mendapatkan hasil lebih baik senilai 0.0003 untuk RMSE dan 0.0105 untuk MAPE. Jumlah reservoir yang semakin kecil mendapatkan nilai kesalahan semakin kecil pula. Hasil grafik ditunjukkan pada Gambar 4.11, Gambar 4.12 dan Gambar 4.13.

Tabel 4.10 ESN-DLR dataset 1

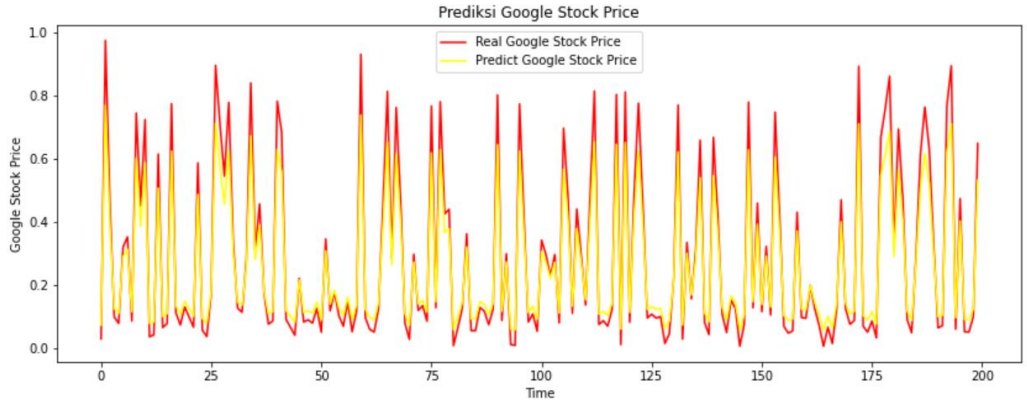
Uji coba	RMSE	MAPE
ESN-DLR, n_reservoir 30	0.0003	0.0105
ESN-DLR, n_reservoir 50	0.0015	0.0262
ESN-DLR, n_reservoir 100	0.0067	0.0546



Gambar 4.11 ESN-DLR reservoir 30 dataset 1



Gambar 4.12 ESN-DLR reservoir 50 dataset 1

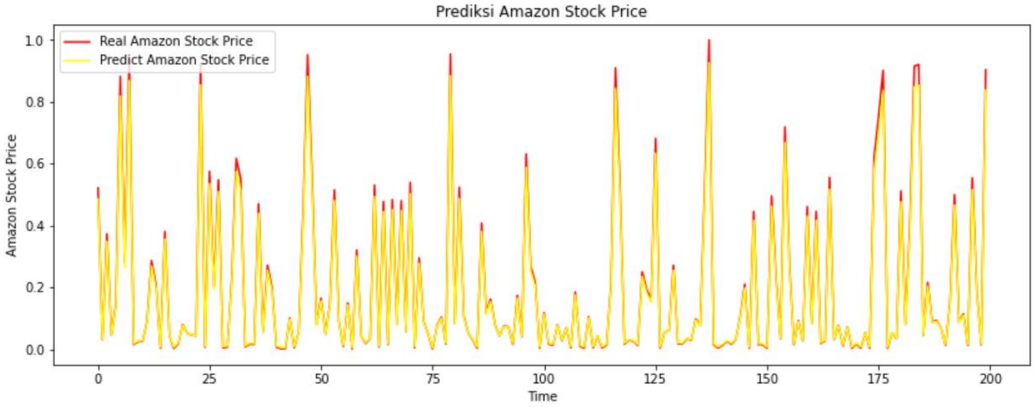


Gambar 4.13 ESN-DLR reservoir 100 dataset 1

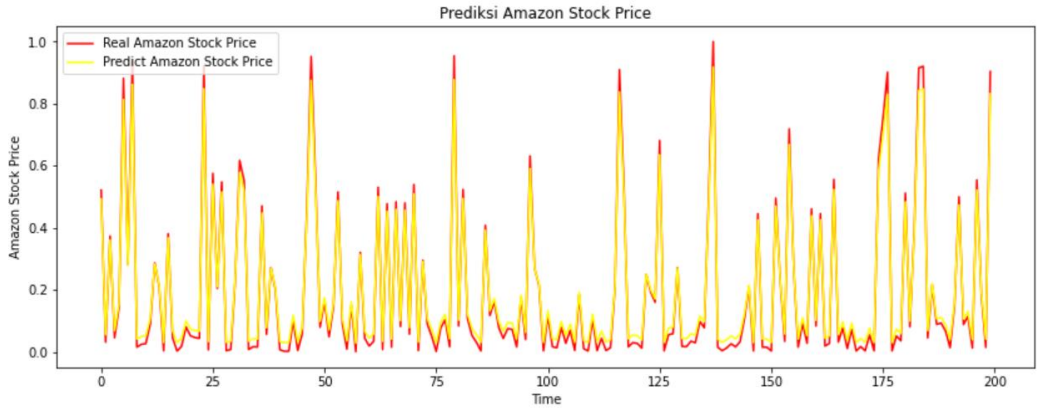
Hasil uji coba metode ESN-DLR pada dataset 2 *amazon stock price.* Penerapan jumlah reservoir 30 mendapatkan hasil lebih baik senilai 0.0004 untuk RMSE dan 0.0121 untuk MAPE. Jumlah reservoir yang semakin kecil mendapatkan nilai kesalahan semakin kecil pula. Hasil grafik ditunjukkan pada Gambar 4.14, Gambar 4.15 dan Gambar 4.16.

Tabel 4.11 ESN-DLR dataset 2

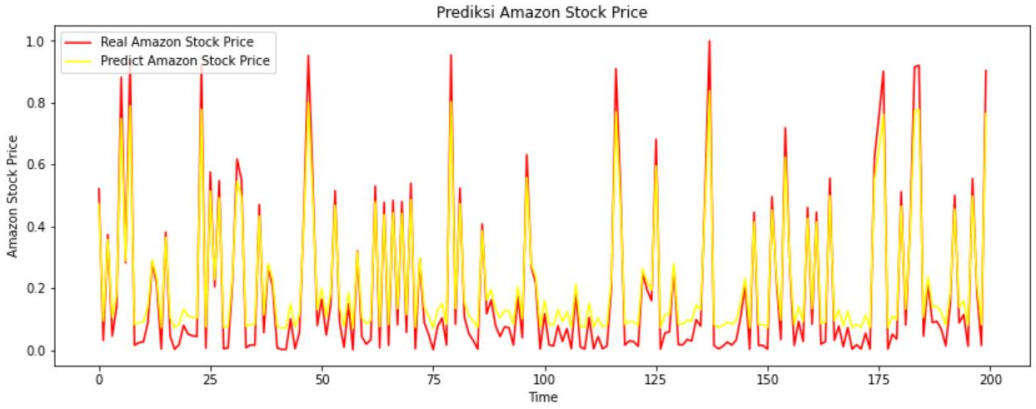
Uji coba	**RMSE**	**MAPE**
ESN-DLR, n_reservoir 30	0.0004	0.0121
ESN-DLR, n_reservoir 50	0.0007	0.0243
ESN-DLR, n_reservoir 100	0.0028	0.0399



Gambar 4.14 ESN-DLR reservoir 30 dataset 2



Gambar 4.15 ESN-DLR reservoir 50 dataset 2



Gambar 4.16 ESN-DLR reservoir 100 dataset 2

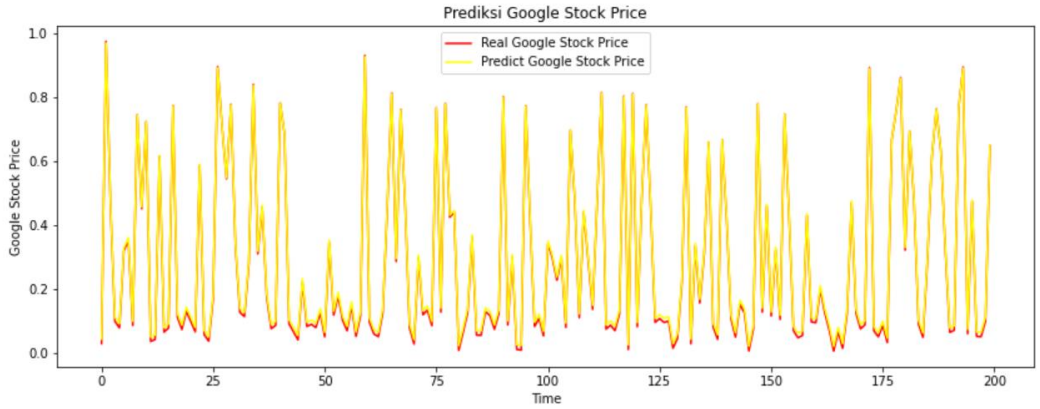
Menurut percobaan pada skenario 1, pada RNN ketika perbedaan epoch 10 dan 50 hasil nilai kesalahan tidak terlalu jauh. Pada dataset 1 yang diimplementasikan pada metode RNN mendapat nilai kesalahan terkecil 0.0158 dan untuk dataset 2 nilai kesalahan terkecil 0.0963. Pada metode ESN nilai RMSE dan MAPE untuk dataset 1 mempunyai nilai kesalahan terkecil 0.0005 dan 0.0153, untuk dataset 2 mendapat nilai 0.0004 dan 0.0170. Kemudian, metode ESN-DLR untuk dataset 1 mempunyai nilai kesalahan 0.0003 dan 0.0105, untuk dataset 2 0.0004 dan 0.0121

4.3.2 Skenario Uji Coba 2

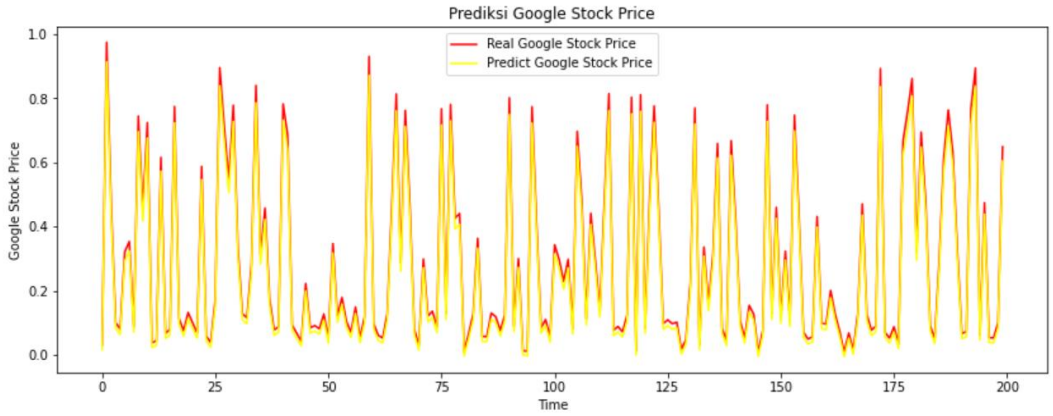
Hasil uji coba metode ESN pada dataset 1 *google stock price*. Penerapan jumlah reservoir 10 mendapatkan hasil lebih baik senilai 0.0002 untuk RMSE dan 0.0104 untuk MAPE. Hasil grafik ditunjukkan pada Gambar 4.17, Gambar 4.18 dan Gambar 4.19.

Tabel 4.12 Hasil pengujian ESN dataset 1

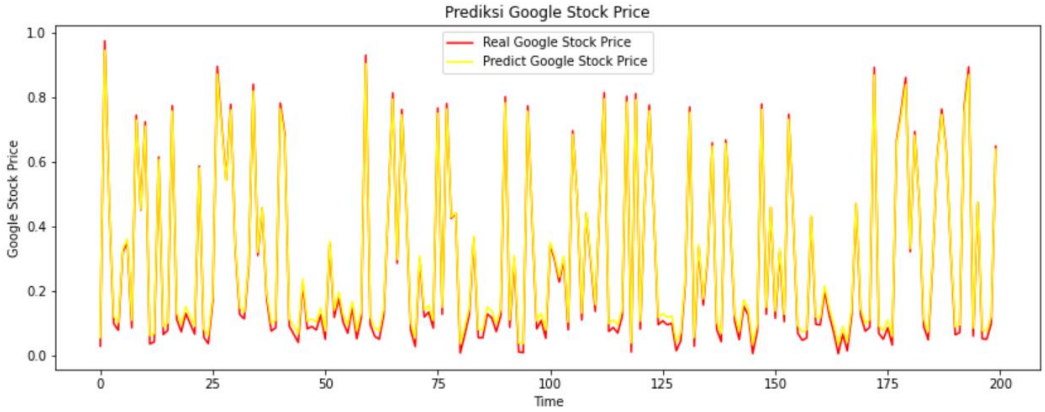
Uji coba	N-reservoir	*Leaking rate*	RMSE	MAPE
ESN	10	0.2	0.0002	0.0104
ESN	15	0.2	0.0007	0.0220
ESN	20	0.2	0.0003	0.0185



Gambar 4.17 ESN reservoir 10 dataset 1



Gambar 4.18 ESN reservoir 15 dataset 1

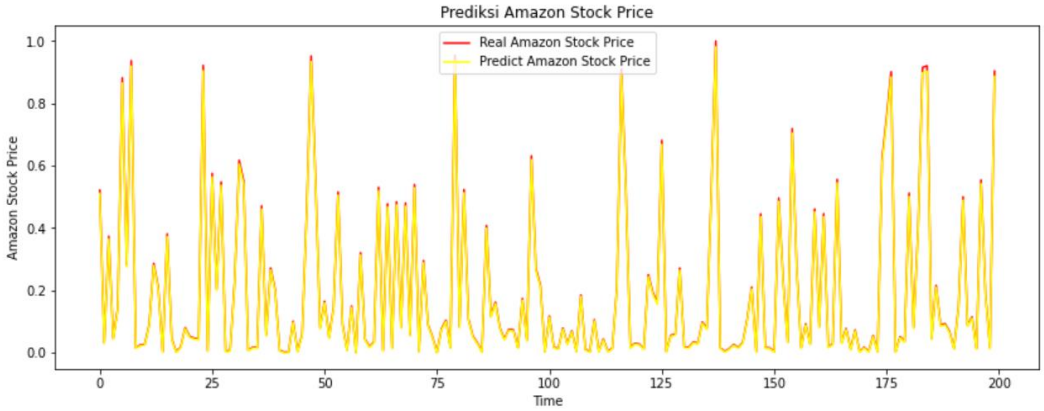


Gambar 4.19 ESN reservoir 20 dataset 1

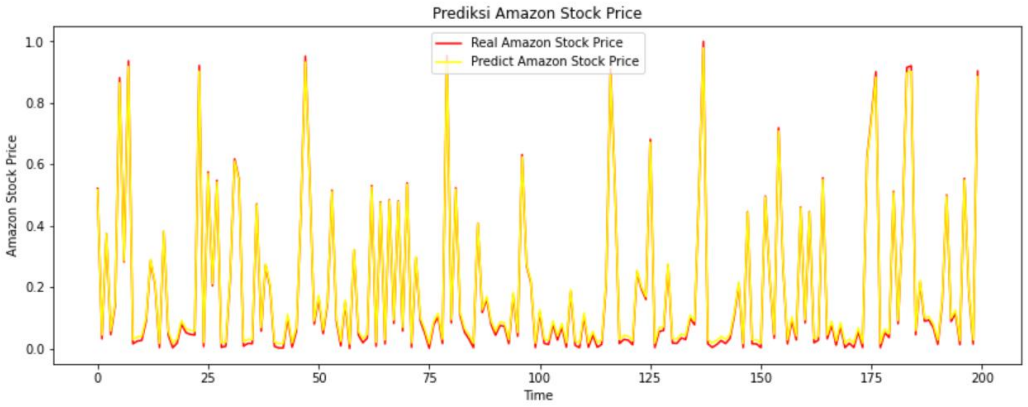
Hasil uji coba metode ESN pada dataset 2 *amazon stock price*. Penerapan jumlah reservoir 10 mendapatkan hasil lebih baik senilai 3.33e-5untuk RMSE dan 0.0036 untuk MAPE. Hasil grafik ditunjukkan pada Gambar 4.20, Gambar 4.21 dan Gambar 4.22.

Tabel 4.13 Hasil pengujian ESN dataset 2

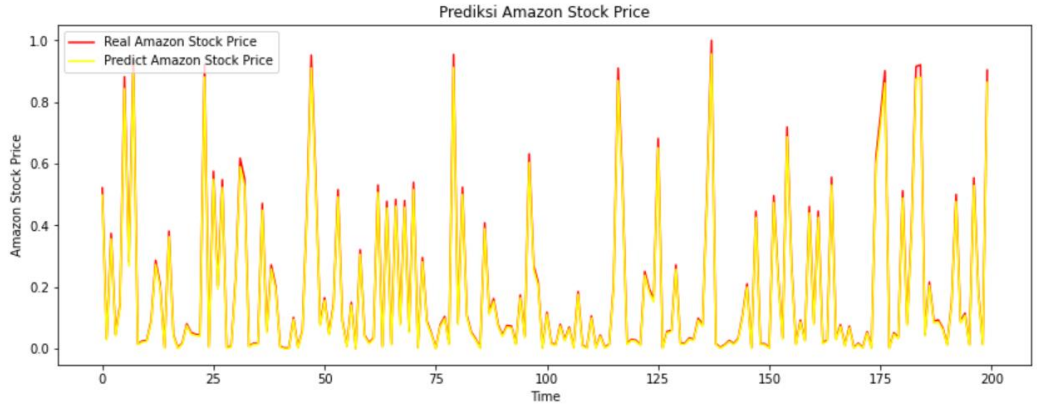
Uji coba	**N-reservoir**	***Leaking rate***	**RMSE**	**MAPE**
ESN	10	0.2	3.33e-5	0.0036
ESN	15	0.2	0.0001	0.0069
ESN	20	0.2	0.0004	0.0184



Gambar 4.20 ESN reservoir 10 dataset 2



Gambar 4.21 ESN reservoir 15 dataset 2

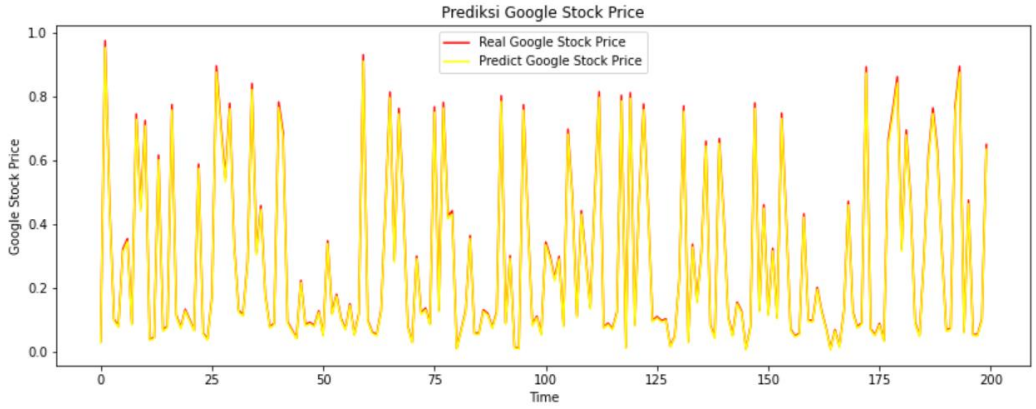


Gambar 4.22 ESN reservoir 20 dataset 2

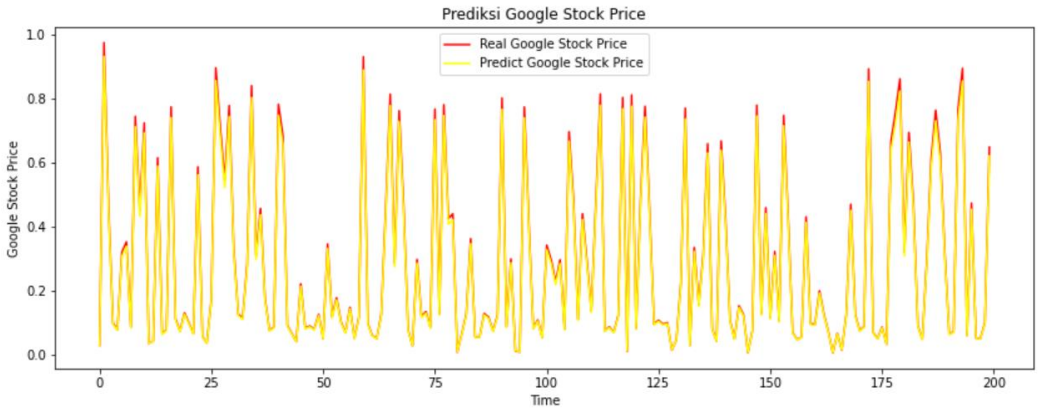
Hasil uji coba metode ESN-DLR pada dataset 1 *google stock price*. Penerapan jumlah reservoir 15 mendapatkan hasil lebih baik senilai 0.0001 untuk RMSE dan 0.0082 untuk MAPE. Hasil grafik ditunjukkan pada Gambar 4.23, Gambar 4.24 dan Gambar 4.25.

Tabel 4.14 Hasil pengujian ESN-DLR dataset 1

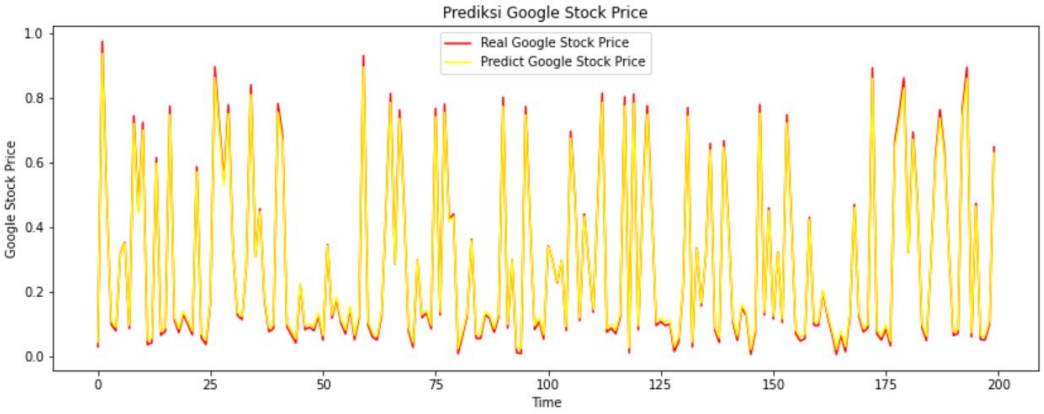
Uji coba	**N-reservoir**	***Leaking rate***	**RMSE**	**MAPE**
ESN-DLR	10	0.2	0.0002	0.0117
ESN-DLR	15	0.2	0.0001	0.0082
ESN-DLR	20	0.2	0.0009	0.0270



Gambar 4.23 ESN-DLR reservoir 10 dataset 1



Gambar 4.24 ESN-DLR reservoir 15 dataset 1

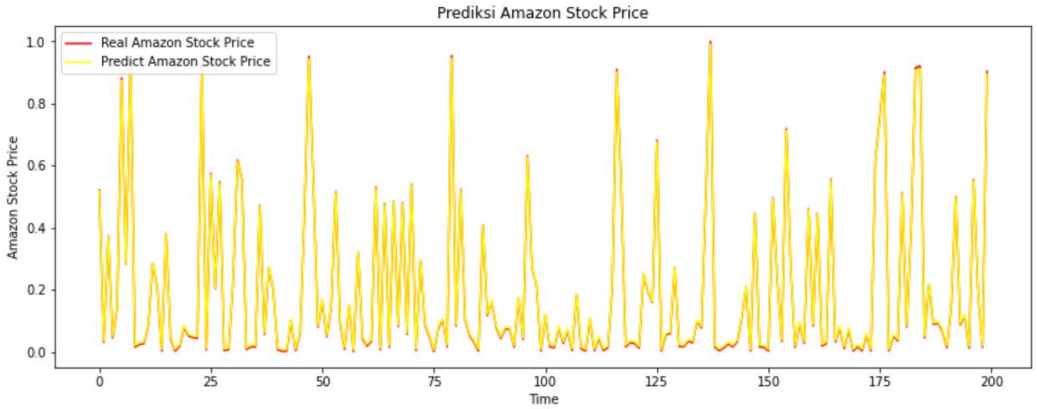


Gambar 4.25 ESN-DLR reservoir 20 dataset 1

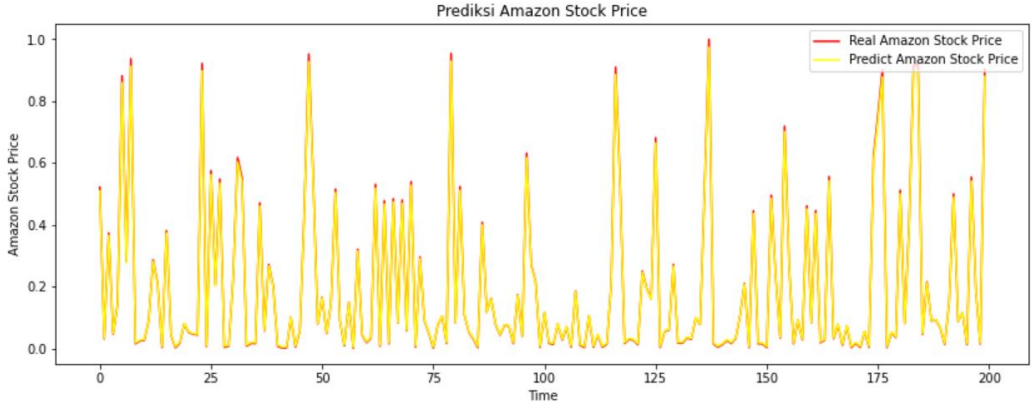
Hasil uji coba metode ESN-DLR pada dataset 2 *amazon stock price*. Penerapan jumlah reservoir 10 mendapatkan hasil lebih baik senilai 5.88e-6 untuk RMSE dan 0.0049 untuk MAPE. Hasil grafik ditunjukkan pada Gambar 4.26, Gambar 4.27 dan Gambar 4.28.

Tabel 4.15 Hasil pengujian ESN-DLR dataset 2

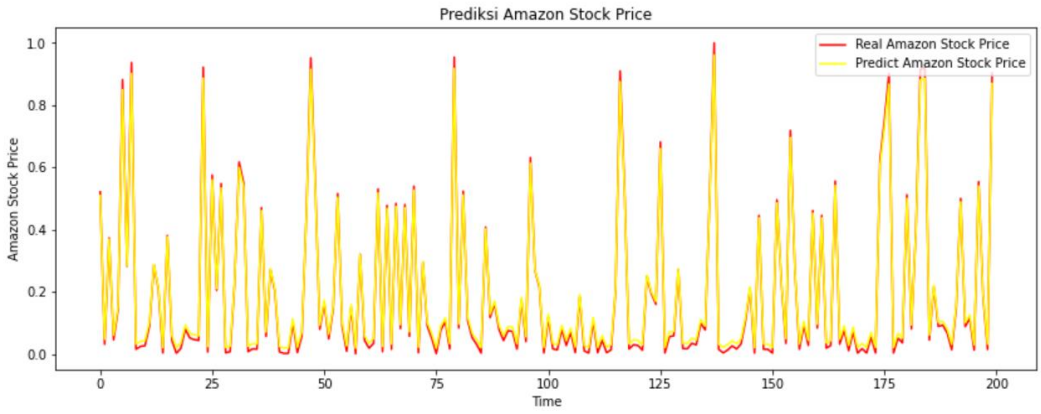
Uji coba	**N-reservoir**	***Leaking rate***	**RMSE**	**MAPE**
ESN-DLR	10	0.2	5.88e-6	0.0049
ESN-DLR	15	0.2	0.0001	0.0095
ESN-DLR	20	0.2	0.0001	0.0117



Gambar 4.26 ESN-DLR reservoir 10 dataset 2



Gambar 4.27 ESN-DLR reservoir 15 dataset 2



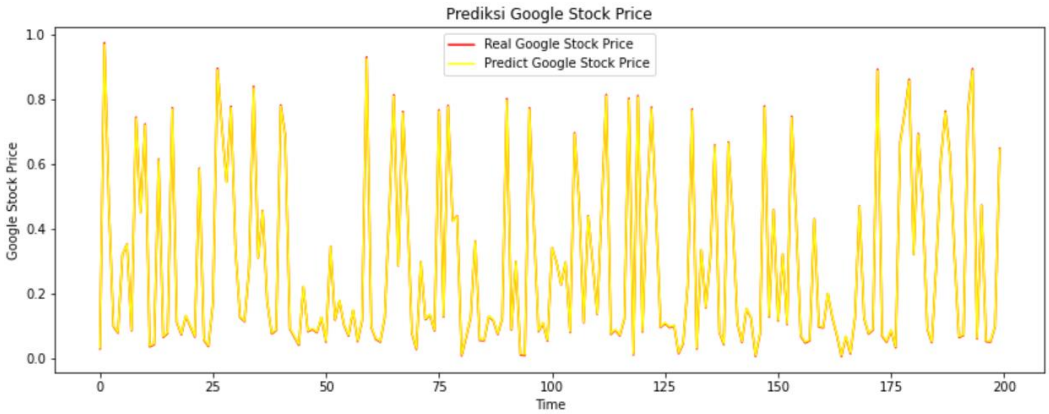
Gambar 4.28 ESN-DLR reservoir 20 dataset 2

4.3.3 Skenario Uji Coba 3

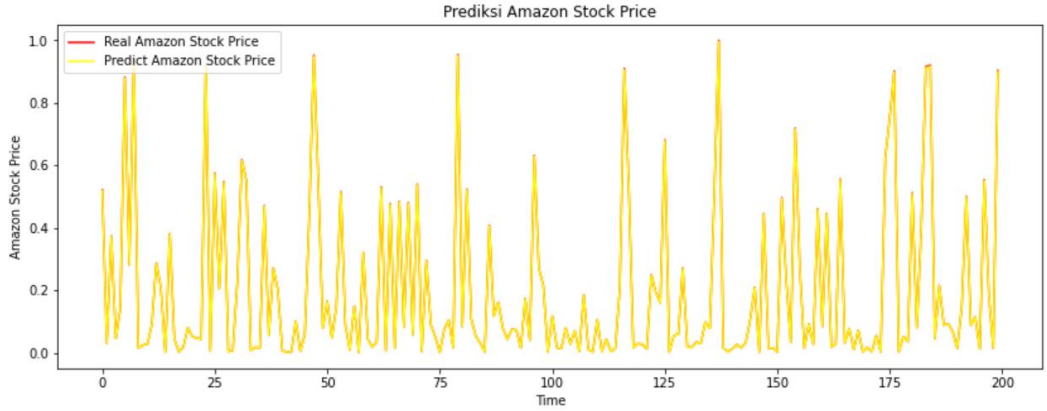
Hasil uji coba metode ESN yang dioptimasi menggunakan HS pada dataset 1 *google stock price* mendapatkan kombinasi nilai parameter ESN paling tepat untuk jumlah reservoir 5 dan nilai *leaking rate* 0.0948. Kemudian dataset 2 *amazon stock price* nilai parameter jumlah reservoir terbaik adalah 11 dan *leaking rate* 0.0010. Hasil grafik ditunjukkan pada Gambar 4.29 dan Gambar 4.30.

Tabel 4.16 Hasil pengujian ESN-HS

Uji coba	N-reservoir	*Leaking rate*	RMSE	MAPE
ESN-HS, Dataset 1	5	0.0948	3.37e-5	0.0047
ESN-HS, Dataset 2	11	0.0010	1.50e-6	0.0008



Gambar 4.29 ESN-HS dataset 1

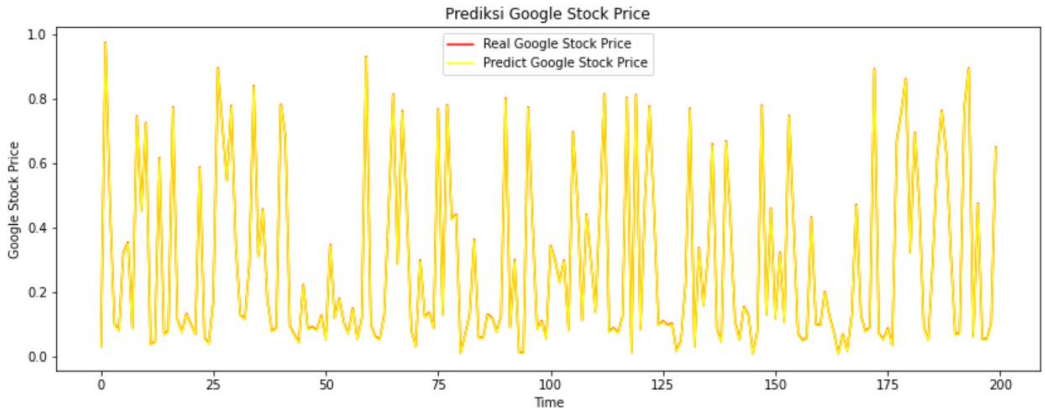


Gambar 4.30 ESN-HS dataset 2

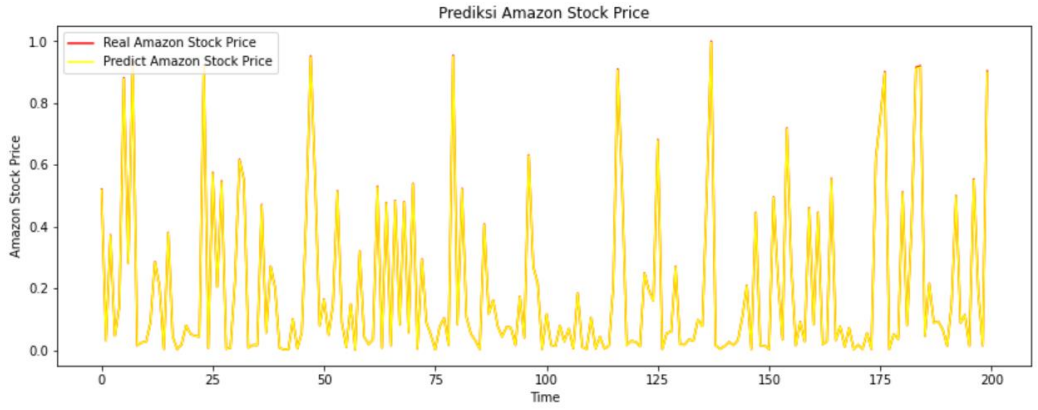
Hasil uji coba metode ESN-DLR yang dioptimasi menggunakan HS pada dataset 1 *google stock price* mendapatkan kombinasi nilai parameter ESN-DLR paling tepat untuk jumlah reservoir 11 dan nilai *leaking rate* 0.1874. Kemudian dataset 2 *amazon stock price* nilai parameter jumlah reservoir terbaik adalah 5 dan *leaking rate* 0.0018. Hasil grafik ditunjukkan pada Gambar 4.31 dan Gambar 4.32.

Tabel 4.17 Hasil pengujian ESN-DLR-HS

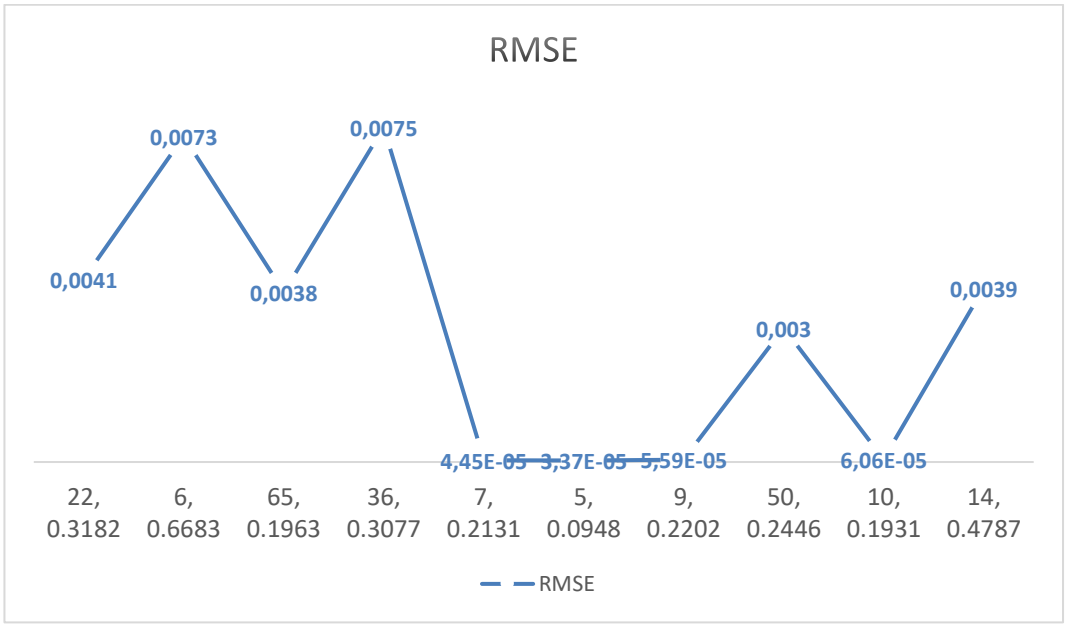
Uji coba	**N-reservoir**	***Leaking rate***	**RMSE**	**MAPE**
ESN-DLR-HS, Dataset 1	11	0.1874	3.36e-5	0.0048
ESN-DLR-HS, Dataset 2	5	0.0018	1.46e-6	0.0007



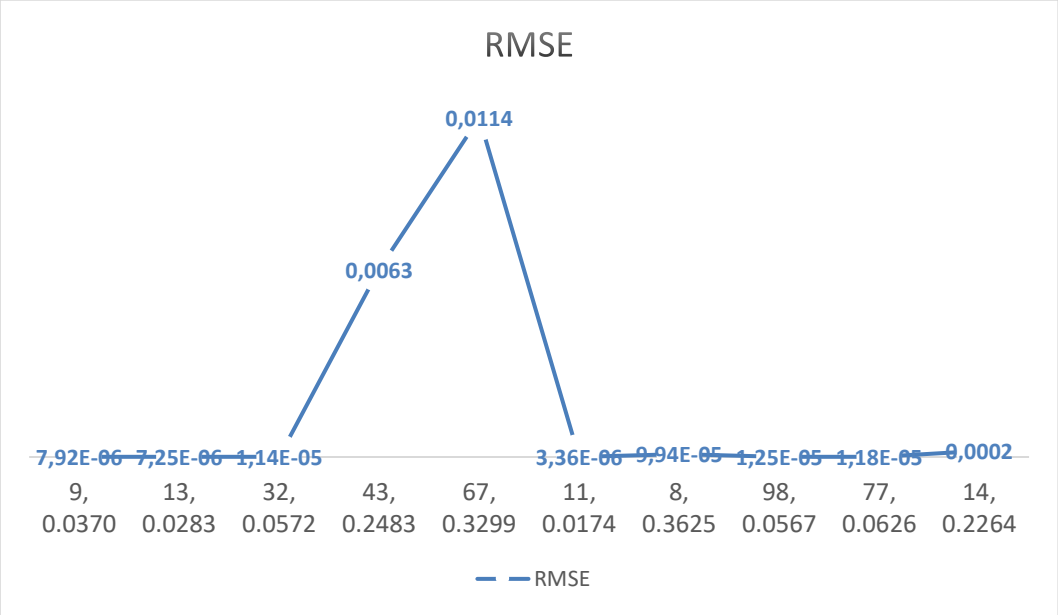
Gambar 4.31 ESN-DLR-HS dataset 1



Gambar 4.32 ESN-DLR-HS dataset 2



Gambar 4.33 Grafik optimasi parameter ESN dataset 1



Gambar 4.34 Grafik optimasi parameter ESN-DLR dataset 1

Keseluruhan uji coba dirangkum pada Tabel 4.18 untuk dataset 1 dan Tabel 4.19 untuk dataset 2. Dilihat dari nilai kesalahan, metode ESN lebih baik dari pada RNN, proses pelatihan model prediksi juga lebih cepat. Kemudian, ESN dikembangkan lagi dan berhasil mendapatkan nilai keslahan lebih kecil dari pada ESN model dasar. Namun, metode ESN sangat berpengaruh terhadap nilai parameter yang ditentukan. Makanya perlu metode optimasi *harmony search* sehingga dapat memeroleh hasil prediksi yang baik.

Tabel 4.18 Rangkuman uji coba pada dataset 1

Metode	**RMSE**	**MAPE**
RNN	0.1158	0.2600
ESN	0.0005	0.0153
ESN-DLR	0.0001	0.0082
ESN-HS	3.37e-5	0.0047
ESN-DLR-HS	3.36e-5	0.0048

Tabel 4.19 Rangkuman uji coba pada dataset 2

Metode	RMSE	MAPE
RNN	0.0963	0.2135
ESN	0.0004	0.0170
ESN-DLR	5.88e-6	0.0049
ESN-HS	1.50e-6	0.0008
ESN-DLR-HS	1.46e-6	0.0007

Tabel 4.20 Seluruh hasil uji coba pada dataset 1

Uji coba	RMSE	MAPE
RNN, epoch 10	0.1237	0.2687
RNN, epoch 50	0.1158	0.2600
ESN, n_reservoir 30	0.0005	0.0153
ESN, n_reservoir 50	0.0023	0.3291
ESN, n_reservoir 100	0.0048	0.0565
ESN-DLR, n_reservoir 30	**0.0003**	**0.0105**
ESN-DLR, n_reservoir 50	0.0015	0.0262
ESN-DLR, n_reservoir 100	0.0067	0.0546
ESN, n_reservoir 10	0.0002	0.0104
ESN, n_reservoir 15	0.0007	0.0220
ESN, n_reservoir 20	0.0003	0.0185
ESN-DLR, n_reservoir 10	0.0002	0.0117
ESN-DLR, n_reservoir 15	**0.0001**	**0.0082**
ESN-DLR, n_reservoir 20	0.0009	0.0270
ESN-HS, n_reservoir 5, *leaking rate* 0.0948	3.37e-5	0.0047
ESN-DLR-HS, n_reservoir 11, *leaking rate* 0.1874	**3.36e-5**	**0.0048**

Tabel 4.21 Seluruh hasil uji coba pada dataset 2

Uji coba	**RMSE**	**MAPE**
RNN, epoch 10	0.0981	0.2152
RNN, epoch 50	0.0963	0.2135
ESN, n_reservoir 30	0.0004	0.0170
ESN, n_reservoir 50	0.0009	0.0202
ESN, n_reservoir 100	0.0023	0.0332
ESN-DLR, n_reservoir 30	**0.0004**	**0.0121**
ESN-DLR, n_reservoir 50	0.0007	0.0243
ESN-DLR, n_reservoir 100	0.0028	0.0399
ESN, n_reservoir 10	3.33e-5	0.0036
ESN, n_reservoir 15	0.0001	0.0069
ESN, n_reservoir 20	0.0004	0.0184
ESN-DLR, n_reservoir 10	**5.88e-6**	**0.0049**
ESN-DLR, n_reservoir 15	0.0001	0.0095
ESN-DLR, n_reservoir 20	0.0001	0.0117
ESN-HS, n_reservoir 11, *leaking rate* 0.0010	1.50e-6	0.0008
ESN-DLR-HS, n_reservoir 5, *leaking rate* 0.0018	**1.46e-6**	**0.0007**

BAB 5
KESIMPULAN

Pada bab ini dijelaskan mengenai kesimpulan akhir yang didapatkan setelah melakukan serangkaian uji coba pada bab sebelumnya. Penjelasan pada bab ini juga akan menjawab permasalahan yang telah dijabarkan pada Bab 1 sebelumnya. Selain itu disajikan pulan saran-saran untuk pengembangan penelitian di akhir bab ini.

5.1 Kesimpulan

Kesimpulan yang diperoleh setelah melewati proses uji coba dan evaluasi adalah sebagai berikut :

1. Implementasi metode prediksi ESN mendapatkan nilai kesalahan lebih kecil dari pada RNN, dikarenakan proses pelatihan hanya dilakukan di lapisan ketiga. Selain itu membuat proses pelatihan lebih cepat. Pada dataset 1 nilai kesalahan ESN sebesar 0.0005 untuk RMSE dan 0.0153 untuk MAPE. Pada dataset 2 mendapatkan nilai sebesar 0.0004 untuk RMSE dan 0.0170 untuk MAPE. Nilai tersebut lebih baik dari pada RNN.
2. Setelah ESN terbukti lebih baik dari pada RNN, ESN dikembangkan menjadi ESN-DLR. Di mana awal pembuatan reservoir setiap node ditentukan acak pada model ESN dasar, diubah menjadi berurut dan diatur jumlah bobot 0 ketika pembuatan angka random di lapisan kedua. Sehingga mendapatkan nilai kesalahan lebih kecil untuk dataset 1 dan 2. Pada dataset 1 ESN-DLR sebesar 0.0001 untuk RMSE dan 0.0082 untuk MAPE. Kemudian di dataset 2 sebesar 5.88e-6 untuk RMSE dan 0.0049 untuk MAPE.
3. Penentuan parameter pada metode ESN dan ESN-DLR sangat berpengaruh terhadap model prediksi yang dihasilkan. Metode optimasi *harmony search* berhasil mengatasi masalah tersebut. Sehingga didapatkan nilai parameter yang sesuai untuk ESN dan nilai kesalahan diminimalkan
4. Metode *harmony search* mudah diimplementasikan dengan pengaturan parameter yang sedikit dan dengan proses yang cepat, berhasil menemukan solusi parameter optimal.

5.2 Saran

Saran yang dapat dipertimbangkan untuk pengembangan penelitian selanjutnya adalah sebagai berikut :

1. Dataset yang digunakan pada penelitian ini terbatas data *stock price*. Untuk studi lebih lanjut dapat menggunakan data-data yang lebih beragam.
2. Konfigurasi ESN jika tidak sesuai berdampak sangat buruk. Bisa dikembangkan dengan mencoba metode optimasi yang lain.
3. Proses pelatihan terjadi di lapisan ketiga, metode pelatihan dapat dikembangkan dengan metode lainnya.

DAFTAR PUSTAKA

Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A. and Arshad, H. (2018), "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, Vol. 4 No. 11, hal. e00938.

Askarzadeh, A. and Rashedi, E. (2017), *Harmony Search Algorithm*, available at:https://doi.org/10.4018/978-1-5225-2322-2.ch001.

Begam, K.M. and Deepa, S.N. (2019), "Optimized nonlinear neural network architectural models for multistep wind speed forecasting ☆," *Computers and Electrical Engineering*, Elsevier Ltd, Vol. 78, hal. 32–49.

Bianchi, F.M., Santis, E.D.E., Rizzi, A., Sadeghian, A. and Member, S. (2015), "Short-Term Electric Load Forecasting Using Echo State Networks and PCA Decomposition," *IEEE Access*, IEEE, Vol. 3, hal. 1931–1943.

Cai, X., Zhang, N., Venayagamoorthy, G.K. and Wunsch, D.C. (2007), "Time series prediction with recurrent neural networks trained by a hybrid PSO – EA algorithm," Vol. 70, hal. 2342–2353.

Chen, H., Zeng, Z. and Tang, H. (2013), "Lanslide Deformation Prediction Based on Recurrent Neural Network," *Neural Process Lett*, Vol. 41 No. 2, hal. 169–178.

Choi, S., Kim, J. and Yeo, H. (2019), "Attention-based Recurrent Recurrent Neural Network Network for Urban Vehicle Trajectory Prediction," *Procedia Computer Science*, Elsevier B.V., Vol. 151 No. 2018, hal. 327–334.

Chouikhi, N., Ammar, B., Rokbani, N. and Alimi, A.M. (2017), "PSO-based analysis of Echo State Network parameters for time series forecasting," *Applied Soft Computing Journal*, Elsevier B.V., Vol. 55, hal. 211–225.

Daoud, M., Mayo, M., Box, P.O. and Zealand, N. (2019), "A survey of neural network-based cancer prediction models from microarray data," *Artificial Intelligence In Medicine*, Elsevier, Vol. 97 No. October 2017, hal. 204–214.

Ivanedra, K., Mustikasari, M., Informatika, T., Gunadarma, U., Network, R.N. and Learning, D. (2019), "IMPLEMENTASI METODE RECURRENT NEURAL NETWORK PADA TEXT THE IMPLEMENTATION OF TEXT SUMMARIZATION WITH ABSTRACTIVE," Vol. 6 No. 4, available at:https://doi.org/10.25126/jtiik.201961067.

Jaeger, H. (2012), "Long Short-Term Memory in Echo State Networks : Details of a Simulation Study Long Short-Term Memory in Echo State Networks : Details of a Sim- ulation Study," No. 27.

Jeager, H. (2007), "Echo state network," *Scholarpedia*.

Kim, Y., Yoon, Y. and Woo, Z. (2019), "A comparison study of harmony search and genetic algorithm for the max-cut problem," *Swarm and Evolutionary Computation*, Elsevier B.V., Vol. 44 No. August 2017, hal. 130–135.

Klibisz, A. (2016), "A Primer on Reservoir Computing," hal. 1–14.

Løkse, S., Maria, F. and Jenssen, R. (2017), "Training Echo State Networks with Regularization Through Dimensionality Reduction," *Cognitive Computation*, Cognitive Computation, No. 1, available at:https://doi.org/10.1007/s12559-017-9450-z.

Lukoševičius, M. (2012), "A Practical Guide to Applying Echo State Networks," hal. 659–686.

Ma, Z., Dong, Y., Wang, C. and Shao, X.U.N. (2018), "Forecast of Non-Equal Interval Track Irregularity Based on Improved Grey Model and PSO-SVM," *IEEE Access*, IEEE, Vol. 6, hal. 34812–34818.

Navas, R.K.B., Prakash, S. and Sasipraba, T. (2019), "Artificial Neural Network based computing model for wind speed prediction: A case study of Coimbatore, Tamil Nadu, India," *Physica A*, Elsevier B.V., available at:https://doi.org/10.1016/j.physa.2019.123383.

Saadat, J., Moallem, P. and Koofigar, H. (2017), "Training Echo Estate Neural Network Using Harmony Search Algorithm," Vol. 15 No. 1, hal. 163–179.

Schiller, U.D. and Ã, J.J.S. (2005), "Analyzing the weight dynamics of recurrent learning algorithms," Vol. 63, hal. 5–23.

Schimdhuber, J. (2015), "Deep Learning in Neural Networks : An Overview," *Neural Networks*, Vol. 61, hal. 85–117.

Schrauwen, B., Verstraeten, D. and Campenhout, J. Van. (2007), "An overview of reservoir computing: theory, applications and implementations," No. April, hal. 25–27.

Selbesoglu, M.O. (2019), "Prediction of tropospheric wet delay by an artificial neural network model based on meteorological and GNSS data," *Engineering Science and Technology, an International Journal*, Karabuk University, available at:https://doi.org/10.1016/j.jestch.2019.11.006.

Shen, L., Chen, J., Zeng, Z., Yang, J. and Jin, J. (2018), "A novel echo state network for multivariate and nonlinear time series prediction," *Applied Soft Computing Journal*, Elsevier B.V., Vol. 62, hal. 524–535.

Shi, G., Liu, D. and Wei, Q. (2016), "Energy consumption prediction of office buildings based on echo state networks," *Neurocomputing*, Elsevier, hal. 1–11.

Shumway, R.H. (1999), *Time Series Analysis and Its Applications*, 4th ed., Springer International Publishing.

Stewart, M. (2019), "Predicting Stock Prices with Echo State Networks," *towardsdatascience*, available at: towardsdatascience.com (accessed 8 March 2020).

Usama, M., Ahmad, B., Xiao, W., Hossain, M.S. and Muhammad, G. (2019), "Self-attention based recurrent convolutional neural network for disease prediction using healthcare data," *Computer Methods and Programs in Biomedicine*, Elsevier B.V., hal. 105191.

Verstraeten, D., Schrauwen, B., Haene, M.D. and Stroobandt, D. (2007), "An experimental unification of reservoir computing methods," Vol. 20, hal. 391–403.

Xu, M., Han, M. and Lin, H. (2018), "Wavelet-denoising multiple echo state networks for multivariate time series prediction," *Information Sciences*, Elsevier Inc., available at:https://doi.org/10.1016/j.ins.2018.07.015.

Zagrebina, S.A., Mokhov, V.G. and Tsimbol, V.I. (2019), "Electrical Energy Consumption Prediction is based on the Recurrent Neural Network," *Procedia Computer Science*, Elsevier B.V., Vol. 150, hal. 340–346.

Zhou, J., Yang, X., Sun, L., Han, C. and Xiao, F.U. (2018), "Network Traffic Prediction Method Based on Improved Echo State Network," *IEEE Access*, IEEE, Vol. 6, hal. 70625–70632.