

**PENGGABUNGAN ARTIFICIAL NEURAL NETWORK DENGAN
GENETIC ALGORITHM UNTUK DEMAND FORECASTING**

TUGAS AKHIR

Disusun Oleh:

**STENDY ILIANDRE
1117013**



**INSTITUT
TEKNOLOGI
HARAPAN
BANGSA**

Veritas vos liberabit

**PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2021**

PENGGABUNGAN ARTIFICIAL NEURAL NETWORK DENGAN GENETIC ALGORITHM UNTUK DEMAND FORECASTING

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh
gelar sarjana dalam bidang Informatika

Disusun Oleh:

**STENDY ILIANDRE
1117013**



**INSTITUT
TEKNOLOGI
HARAPAN
BANGSA**

Veritas vos liberabit

**PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2021**

PERNYATAAN ORISINALITAS

Saya menyatakan bahwa Tugas Akhir yang saya susun ini adalah hasil karya saya sendiri.

Semua sumber yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Saya bersedia menerima sanksi pencabutan gelar akademik apabila di kemudian hari

Tugas Akhir ini terbukti plagiat.

Bandung, Juni 2021



Stendy Iliandre

1117013

PENGESAHAN TUGAS AKHIR

Tugas Akhir dengan judul :

**PENGGABUNGAN ARTIFICIAL NEURAL NETWORK DENGAN GENETIC ALGORITHM
UNTUK DEMAND FORECASTING**

yang disusun oleh:

STENDY ILIANDRE

1117013

telah berhasil dipertahankan di hadapan Dewan Penguji Sidang Tugas Akhir yang dilaksanakan pada:

Hari / tanggal : Rabu / 30 Juni 2021

Waktu : 16.00

Menyetujui,

Pembimbing:



VENTJE JEREMIAS LEWI ENGEL, M.T., CEH.

NIK. 116019

PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Institut Teknologi Harapan Bangsa, saya yang bertanda tangan di bawah ini:

Nama : Stendy Iliandre

NIM : 1117013

Program Studi : Informatika

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Harapan Bangsa **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Rights)** atas karya ilmiah saya yang berjudul:

**PENGGABUNGAN ARTIFICIAL NEURAL NETWORK DENGAN GENETIC ALGORITHM
UNTUK DEMAND FORECASTING**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Harapan Bangsa berhak menyimpan, mengalihmediakan, mengelola dalam pangkalan data, dan memublikasikan karya ilmiah saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Bandung, Juni 2021

Yang menyatakan



STENDY ILIANDRE

KATA PENGANTAR

Terima kasih kepada Tuhan yang Maha Esa karena dengan bimbingan-Nya dan karunia-Nya penulis dapat melaksanakan Tugas Akhir yang berjudul "PENGGABUNGAN ARTIFICIAL NEURAL NETWORK DENGAN GENETIC ALGORITHM UNTUK DEMAND FORECASTING ". Laporan ini disusun sebagai salah satu syarat kelulusan di Institut Teknologi Harapan Bangsa. Pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yang Maha Esa, karena oleh bimbingan-Nya penulis selalu mendapat pengharapan untuk menyelesaikan tugas akhir ini.
2. Bapak Ventje Jeremias Lewi Engel, M.T., CEH., selaku pembimbing Tugas Akhir yang senantiasa memberi dukungan, semangat, ilmu-ilmu, saran dan dukungan kepada penulis selama Tugas Akhir berlangsung dan selama pembuatan Laporan Tugas Akhir ini.
3. Ibu Ir. Inge Martina, M.T., selaku penguji I Tugas Akhir. Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
4. Ibu Leo Rama Kristian, S.T., M.T., selaku penguji II Tugas Akhir. Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
5. Seluruh dosen dan staff Program Studi Informatika ITHB yang telah membantu dalam menyelesaikan Laporan Tugas Akhir ini.
6. Segenap jajaran staf dan karyawan ITHB yang turut membantu kelancaran dalam menyelesaikan Laporan Tugas Akhir ini.
7. Kedua orang tua dan kakak tercinta yang selalu menyediakan waktu untuk memberikan doa, semangat dan dukungan yang tak habis-habisnya kepada penulis untuk menyelesaikan Laporan Tugas Akhir ini. Terima kasih untuk nasihat, masukan, perhatian, teguran dan kasih sayang yang diberikan hingga saat ini.
8. Jason, Andre, Albert, Valen yang selalu menyediakan waktu untuk cerita, semangat dan dukungan yang tak habis-habisnya kepada penulis untuk menyelesaikan Laporan Tugas Akhir ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna karena keterbatasan waktu dan pengetahuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran untuk membangun kesempurnaan tugas akhir ini sangat diharapkan. Semoga Tugas Akhir ini dapat membantu pihak-pihak yang membutuhkannya.

Bandung, Juni 2021
Hormat penulis,



STENDY ILIANDRE

DAFTAR ISI

LEMBAR PERNYATAAN HASIL KARYA PRIBADI	i
LEMBAR PENGESAHAN	ii
ABSTRAK	iii
ABSTRACT	iii
PUBLIKASI TUGAS AKHIR	iii
KATA PENGANTAR	iv
DAFTAR ISI	vii
DAFTAR TABEL	ix
DAFTAR GAMBAR	xi
BAB 1 PENDAHULUAN	1-1
1.1 Latar Belakang	1-1
1.2 Rumusan Masalah	1-2
1.3 Batasan Masalah	1-2
1.4 Tujuan Penelitian	1-3
1.5 Kontribusi Penelitian	1-3
1.6 Metode Penelitian	1-3
1.7 Sistematika Penulisan	1-4
BAB 2 LANDASAN TEORI	2-1
2.1 Tinjauan Pustaka	2-1
2.1.1 <i>Artificial Neural Network</i>	2-1
2.1.2 <i>Back-Propagation Learning Algorithm</i>	2-5
2.1.3 <i>Genetic Algorithm</i>	2-7
2.2 Pustaka	2-10
2.2.1 NumPy	2-10
2.2.2 Pandas	2-10
2.2.3 Scikit-learn	2-11
2.2.4 Matplotlib	2-11

2.2.5	Keras2-11
2.3	Tinjauan Studi2-12
2.4	Tinjauan Objek2-13
2.4.1	<i>Demand Forecasting</i>2-13
2.4.2	<i>Dataset</i>2-16
BAB 3	ANALISIS DAN PERANCANGAN	3-1
3.1	Analisis	3-1
3.2	Kerangka Pemikiran	3-1
3.3	Analisis Urutan Proses Global	3-3
3.4	Analisis Kasus	3-5
3.4.1	<i>Data Sampling</i>	3-5
3.4.2	<i>Normalization</i>	3-7
3.4.3	<i>Genetic Algorithm</i>	3-7
3.4.4	<i>Feed Forward Neural Network</i>	3-11
3.4.5	<i>Back Propagation</i>	3-12
BAB 4	IMPLEMENTASI DAN PENGUJIAN	4-1
4.1	Lingkungan Implementasi	4-1
4.1.1	Spesifikasi Perangkat Keras	4-1
4.1.2	Lingkungan Perangkat Lunak	4-1
4.2	Implementasi Perangkat Lunak	4-1
4.2.1	<i>Class Preprocessing</i>	4-1
4.2.2	<i>Class GeneticAlgorithm</i>	4-2
4.2.3	<i>Class ArtificialNeuralNetwork</i>	4-3
4.2.4	<i>Class Measurement</i>	4-4
4.3	Pengujian	4-5
4.3.1	Pengujian <i>Artificial Neural Network</i> (ANN)	4-5
4.3.2	Pengujian <i>Artificial Neural Network</i> dengan <i>Genetic Algorithm</i> (ANN-GA)	4-15
4.3.3	Pengujian Data <i>Seasonal</i>	4-23
4.3.4	Pengujian <i>Auto Regressive Integrated Moving Average</i> (ARIMA)	4-24
4.4	Pembahasan Hasil Pengujian	4-26
BAB 5	PENUTUP	5-1
5.1	Kesimpulan	5-1
5.2	Saran	5-1

DAFTAR REFERENSI

xii

DAFTAR TABEL

2.1	Daftar <i>method</i> yang digunakan dari <i>library NumPy</i>	2-10
2.2	Daftar <i>method</i> yang digunakan dari <i>library Pandas</i>	2-10
2.3	Daftar <i>method</i> yang digunakan dari <i>library Sklearn</i>	2-11
2.4	Daftar <i>method</i> yang digunakan dari <i>library Matplotlib</i>	2-11
2.5	Daftar <i>method</i> yang digunakan dari <i>library Keras</i>	2-12
2.6	<i>State-of-the-art method</i>	2-13
2.7	Contoh data	2-16
3.1	Contoh data kosong yang terdapat pada <i>dataset</i>	3-6
3.2	<i>Dataset</i> tahun 2013 berdasarkan Product_1359	3-7
3.3	<i>Dataset</i> dinormalisasi menjadi <i>range</i> -1 sampai 1	3-7
3.4	<i>Dataset</i> setelah normalisasi	3-8
3.5	Data masukan	3-8
3.6	Matriks populasi	3-9
3.1	Perhitungan nilai <i>fitness</i>	3-9
3.7	Matriks fitness value	3-9
3.8	Matriks <i>selected parent</i>	3-10
3.9	Matriks <i>crossover</i>	3-10
3.10	Matriks <i>mutation</i>	3-10
3.11	Matriks solusi terbaik	3-10
3.2	Perhitungan nilai <i>input</i> dengan bobot yang disebut <i>transfer function</i>	3-11
3.3	Perhitungan nilai <i>activation function</i>	3-11
3.4	Perhitungan nilai <i>error</i> pada <i>output layer</i>	3-12
3.5	Perhitungan turunan nilai <i>error</i> pada <i>activation function</i>	3-12
3.6	Perhitungan turunan nilai <i>activation function</i> pada <i>transfer function</i>	3-13
3.7	Perhitungan turunan nilai <i>transfer function</i> pada bobot.	3-13
3.8	Perhitungan untuk menentukan nilai turunan pada <i>error</i> terhadap bobot.	3-13
3.9	Mengubah nilai bobot lama dengan yang baru.	3-14
3.10	Perhitungan nilai <i>error</i> setelah menggunakan bobot yang baru.	3-14
4.1	<i>Attribute</i> pada <i>class GeneticAlgorithm</i>	4-2
4.2	Daftar <i>Method</i> pada <i>class GeneticAlgorithm</i>	4-3
4.3	<i>Attribute</i> pada <i>class ArtificialNeuralNetwork</i>	4-4
4.4	Daftar <i>Method</i> pada <i>class ArtificialNeuralNetwork</i>	4-4
4.5	Tabel ARIMA dengan p 1; d 2; q 1 untuk <i>dataset random</i>	4-24
4.6	Tabel ARIMA dengan p 1; d 2; q 1 untuk <i>dataset seasonal</i>	4-25

4.7	Tabel ANN tanpa GA <i>learning rate</i> 0.001; <i>epoch</i> 1000; <i>batch size</i> 1	4-26
4.8	Tabel ANN tanpa GA <i>learning rate</i> 0.001; <i>epoch</i> 2000; <i>batch size</i> 1	4-26
4.9	Tabel ANN dengan GA <i>learning rate</i> 0.001; <i>epoch</i> 2000; <i>batch size</i> 1; <i>pop</i> 5	4-26
4.10	Tabel ANN dengan GA <i>learning rate</i> 0.001; <i>epoch</i> 2000; <i>batch size</i> 1; <i>pop</i> 10; <i>gen</i> 5	4-27
4.11	Tabel ANN dengan GA <i>learning rate</i> 0.001; <i>epoch</i> 2000; <i>batch size</i> 1; <i>pop</i> 5; <i>gen</i> 10	4-27
4.12	Tabel ANN dengan GA <i>learning rate</i> 0.001; <i>epoch</i> 2000; <i>batch size</i> 1; <i>pop</i> 10; <i>gen</i> 10	4-27
4.13	Tabel konfigurasi skenario perbandingan RMSE dan <i>time</i>	4-27

DAFTAR GAMBAR

2.1	<i>Neural Network (Multi Layer Perceptron)</i>	2-1
2.2	<i>Neural Network (Single Layer Perceptron)</i>	2-2
2.3	<i>Neuron [1]</i>	2-3
2.4	Sigmoid Biner	2-4
2.5	Sigmoid Bipolar	2-5
2.6	Implementasi <i>Learning Rate</i>	2-6
2.7	Ilustrasi <i>Genetic Algorithm</i>	2-8
2.8	Proses utama dari <i>Genetic Algorithm</i> [6]	2-8
2.9	Contoh <i>Demand</i> ; x = waktu/tanggal, y = produk yang terjual	2-14
2.10	Contoh <i>Demand Forecasting</i>	2-15
3.1	Kerangka Pemikiran	3-2
3.2	<i>Flowchart Proses Global</i> pada <i>historical sales</i>	3-4
3.3	<i>Demand</i> dari setiap gudang berdasarkan tahun.	3-5
4.1	Bagan Skenario Pengujian	4-5
4.2	Grafik ANN menggunakan <i>learning rate</i> = 0.0001; <i>batch size</i> = 1	4-6
4.3	Grafik ANN menggunakan <i>learning rate</i> = 0.0001; <i>batch size</i> = 2	4-7
4.4	Grafik ANN menggunakan <i>learning rate</i> = 0.0001; <i>batch size</i> = 4	4-8
4.5	Grafik ANN menggunakan <i>learning rate</i> = 0.001; <i>batch size</i> = 1	4-9
4.6	Grafik ANN menggunakan <i>learning rate</i> = 0.001; <i>batch size</i> = 2	4-10
4.7	Grafik ANN menggunakan <i>learning rate</i> = 0.001; <i>batch size</i> = 4	4-11
4.8	Grafik ANN menggunakan <i>learning rate</i> = 0.01; <i>batch size</i> = 1	4-12
4.9	Grafik ANN menggunakan <i>learning rate</i> = 0.01; <i>batch size</i> = 2	4-13
4.10	Grafik ANN menggunakan <i>learning rate</i> = 0.01; <i>batch size</i> = 4	4-14
4.11	Grafik ANN menggunakan <i>learning rate</i> = 0.0001; <i>batch size</i> = 1	4-15
4.12	Grafik ANN menggunakan <i>learning rate</i> = 0.0001; <i>batch size</i> = 1	4-16
4.13	Grafik ANN-GA menggunakan <i>learning rate</i> = 0.001; <i>batch size</i> = 1	4-17
4.14	Grafik ANN-GA menggunakan <i>learning rate</i> = 0.001; <i>population</i> = 5; <i>generation</i> = 5	4-18
4.15	Grafik ANN-GA menggunakan <i>learning rate</i> = 0.001; <i>generation</i> = 5	4-19
4.16	Grafik ANN-GA menggunakan <i>learning rate</i> = 0.001; <i>population</i> = 5; <i>generation</i> = 10	4-20
4.17	Grafik ANN-GA menggunakan <i>learning rate</i> = 0.001; <i>population</i> = 10; <i>generation</i> = 10	4-21
4.18	Grafik ANN-GA menguji dengan jumlah <i>training</i> yang sama dan <i>testing</i> beberapa hari	4-22

4.19	Grafik perbandingan ANN dan ANN-GA menguji <i>dataset seasonal</i>	4-23
4.20	Grafik perbandingan ARIMA, ANN, dan ANN-GA menguji <i>dataset random</i>	4-24
4.21	Grafik perbandingan ARIMA dan ANN-GA menguji <i>dataset seasonal</i>	4-25
4.22	Grafik perbandingan RMSE dan <i>time</i> dengan skala 1:1000 <i>second</i>	4-28

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Setiap pemasok, produsen, atau pengecer tidak akan bisa lepas dari berbagai perencanaan dan pengendalian persediaan. Karena bagaimanapun juga hal tersebut memerlukan perhatian khusus agar bisa memberikan keuntungan bagi usaha tersebut. Pelaku usaha bisa saja memboroskan budget karena pengeluaran yang tidak terstruktur dengan baik apabila tidak memiliki manajemen persediaan yang baik. Sedangkan tujuan dari inventory adalah untuk menjaga berbagai investasi *inventory* dan menjaga kerugian dari *inventory* sekecil mungkin. Dari manajemen *inventory* ada istilah yang disebut *Demand Forecasting*. *Demand Forecasting* atau perkiraan permintaan ini bertujuan untuk menentukan jumlah yang harus dibeli, diproduksi, atau dikirim. Dengan peramalan ini diharapkan bisa menekan biaya penyimpanan dan stok yang tidak diperlukan.

Demand Forecasting bisa dilakukan dengan metode tradisional seperti ARIMA, *Exponential Smoothing*, *Moving Average* dan juga metode masa kini seperti *machine learning*. Metode *machine learning* yang digunakan untuk *Demand Forecasting* ada beberapa jenis seperti, *Artificial Neural Network* (ANN), *Support Vector Machines* (SVM), dan *Genetic Algorithm* (GA). *Neural Network* (NN) memiliki kelebihan pada sistem pembelajarannya yang bisa terus berkembang seperti otak manusia. NN bisa menjalankan *multitask* secara paralel tanpa mengurangi kinerja dari performa sistem, jika ada *neuron* yang tidak bekerja atau ada informasi yang hilang, NN bisa mendeteksi kesalahan itu dan tetap mengeluarkan nilai keluaran (tolerir kesalahan)[1]. *Artificial Neural Network* bisa memprediksi data non-linear dan bisa mengatasi data yang terus berubah seiring berjalannya waktu. ANN memang lebih efektif untuk *demand forecasting* dibandingkan SVM karena SVM akan kewalahan jika menangani *dataset* yang besar. Namun, NN memiliki kelemahan, yaitu harus ada parameter yang ditentukan sebelum *training* berjalan, membutuhkan data *training* yang besar, proses konvergensi lambat, dan sifatnya yang sering terjebak dalam *local minima*[2].

Adapun metode *Support Vector Machine* (SVM) menggunakan garis *support hyperplane* untuk memisahkan dua kelas data poin. *Hyperplane* akan dicari yang paling optimal (margin paling besar) agar data poin bisa diklasifikasikan dengan jelas. SVM bisa digunakan untuk analisis klasifikasi maupun regresi[1]. SVM sama dengan NN, yaitu *supervised learning* artinya menggunakan data label atau pengawasan manusia. SVM bekerja efektif jika jumlah dimensi lebih besar daripada jumlah sampelnya.

Di sisi lain ada metode *Genetic Algorithm* (GA) yang merupakan algoritme pembelajaran mesin yang terinspirasi dari susunan gen pada manusia. GA adalah algoritme yang mencontoh proses evolusi alami dengan konsep utamanya adalah individu-individu yang paling unggul akan

bertahan hidup, sedangkan individu-individu yang lemah akan punah[3]. GA memiliki konsep yang mudah dimengerti, mendukung *optimalisasi multi-objective*, menggunakan *probabilistic rules* bukan *deterministic rules*, *robust* terhadap *local minima*. Namun, perhitungan GA cukup memakan waktu.

Ada beberapa penelitian yang menggabungkan GA dengan NN. Hal ini disebabkan NN yang sering terjebak dalam *local minima* bisa diatasi dengan GA yang tahan terhadap *local minima*. Penggabungan dua metode ini juga telah digunakan untuk kasus lain, yaitu *Predicting Stock Price* dan *Credit Card Fraud Detection*[2][4]. Penelitian yang dilakukan oleh Montri Inthachot et al., membuktikan *hybrid ANN-GA* meningkatkan akurasi dengan *range* sebelumnya 52.57% - 59.86% menjadi 60.00% - 68.87% untuk memprediksi trend harga saham di Thailand[2]. Sedangkan untuk penelitian *Credit Card Fraud Detection* akurasinya meningkat dari 89.42% menjadi 95.58%[4].

Penelitian yang dilakukan oleh Prasanna Kumar et al., sudah menerapkan NN untuk *demand forecasting* yang menghasilkan *Mean Absolute Error* (MAE) sebesar 4.91% yang berarti errornya cukup kecil untuk penelitian *demand forecasting*. Penelitian tersebut menggunakan *Levenberg Marquardt* (trainlm) untuk tahap *training* yang melampaui performa algoritme *Back Propagation* (BP) lainnya seperti, traingd, traingda dan traincfg[5]. Namun, dalam *demand forecasting* belum ada penelitian yang hanya menggunakan *Genetic Algorithm*.

Berdasarkan permasalahan dan potensi di atas, penelitian ini akan melakukan pemodelan forecasting dengan cara menggabungkan ANN dengan GA untuk menutupi kekurangan ANN, yaitu sering terjebak dalam *local minima* dan kurang optimal jika belum ada data awalan. Penelitian ini akan menggunakan *dataset* historis penjualan untuk menghasilkan data ramalan.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada penelitian ini antara lain:

1. Bagaimana konfigurasi parameter ANN dan GA untuk *demand forecasting* yang meminimasi *error*?
2. Berapa waktu komputasi program untuk mendapatkan hasil *error* yang rendah?
3. Bagaimana hasil perbandingan ANN-GA dengan pengujian yang hanya menggunakan ANN saja?

1.3 Batasan Masalah

Batasan masalah yang dibahas pada penelitian ini adalah:

1. Penelitian ini hanya menggunakan 2 tipe data yaitu, stasioner dan *seasonal*.

2. Program hanya bisa menjalankan pengujian untuk satu barang dari *dataset*.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, tujuan penelitian ini adalah untuk mengetahui seberapa tinggi *error* metode *Neural Network* dengan *Genetic Algorithm* yang diterapkan.

1.5 Kontribusi Penelitian

Memprediksi *demand* atau *sales* dari suatu perusahaan menggunakan *Artificial Neural Network* yang sudah digabung dengan *Genetic algorithm*.

1.6 Metode Penelitian

1. Pengumpulan dan Pengolahan Data

Pada tahap ini dilakukan pengumpulan data dan kemudian dilakukan pengolahan normalisasi data.

2. Analisis Masalah

Pada tahap ini dilakukan analisis masalah, batasan, dan perancangan implementasi.

3. Implementasi algoritme

Tahap ini dilakukan proses training menggunakan *Back Propagation* dan *Genetic Algorithm* lalu dilanjutkan dengan proses *forecasting* menggunakan *Neural Network*.

4. Pengujian

Pengujian dilakukan pada data yang telah diolah menggunakan metode ANN-GA.

5. Evaluasi

Tahap ini dilakukan dengan menghitung *error* dari hasil pengujian. Hasil evaluasi ini kemudian dianalisis untuk mengetahui apakah metode yang digunakan lebih akurat dari penelitian yang sudah ada.

1.7 Sistematika Penulisan

BAB I : PENDAHULUAN

Bab ini berisi penjelasan mengenai masalah dan penyelesaian yang akan dikerjakan dalam Penelitian. Bab Pendahuluan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, kontribusi penelitian, metodologi penelitian, dan sistematika pembahasan.

BAB II : LANDASAN TEORI

Bab II berisi hasil studi literatur yang dilakukan dalam penggerjaan Penelitian. Bab ini menjelaskan hal-hal terkait teori dalam Penelitian, seperti tinjauan studi, tinjauan pustaka yang berisi obyek penelitian, landasan teori tentang metode, tahapan algoritme dan contoh penerapannya.

BAB III : ANALISIS DAN PERANCANGAN

Bab III secara umum membahas tahapan pada Penelitian. Tahapan penelitian terdiri dari pengumpulan data, pengolahan data, metode yang dipakai, pengujian metode, evaluasi dan validasi hasil pengujian.

BAB IV : IMPLEMENTASI DAN PENGUJIAN

Bab IV berisi penjelasan mengenai implementasi dan pengujian metode. Pembahasan implementasi mulai dari pengolahan *dataset*, pengujian *dataset*, evaluasi dan validasi hasil pengujian *dataset*.

BAB V : PENUTUP

Bab V merupakan penutup dari Laporan Penelitian. Bab ini berisi kesimpulan yang menjelaskan hal-hal yang diperoleh selama Penelitian dan menunjukan ketercapaian tujuan dari Penelitian dan berisi saran untuk penelitian selanjutnya.

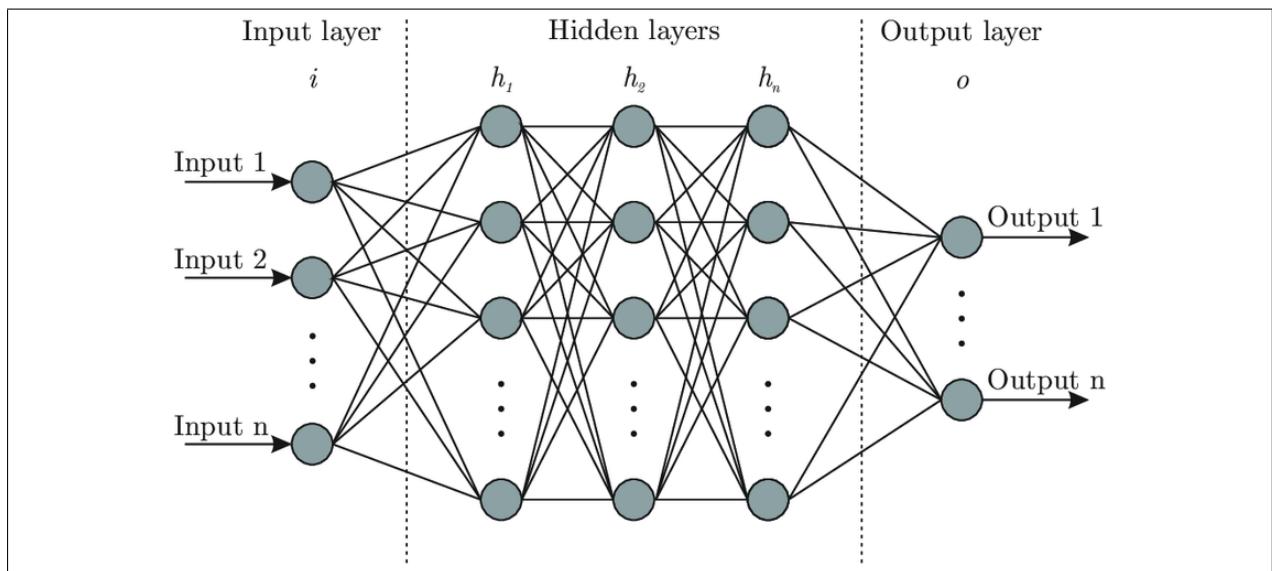
BAB 2 LANDASAN TEORI

2.1 Tinjauan Pustaka

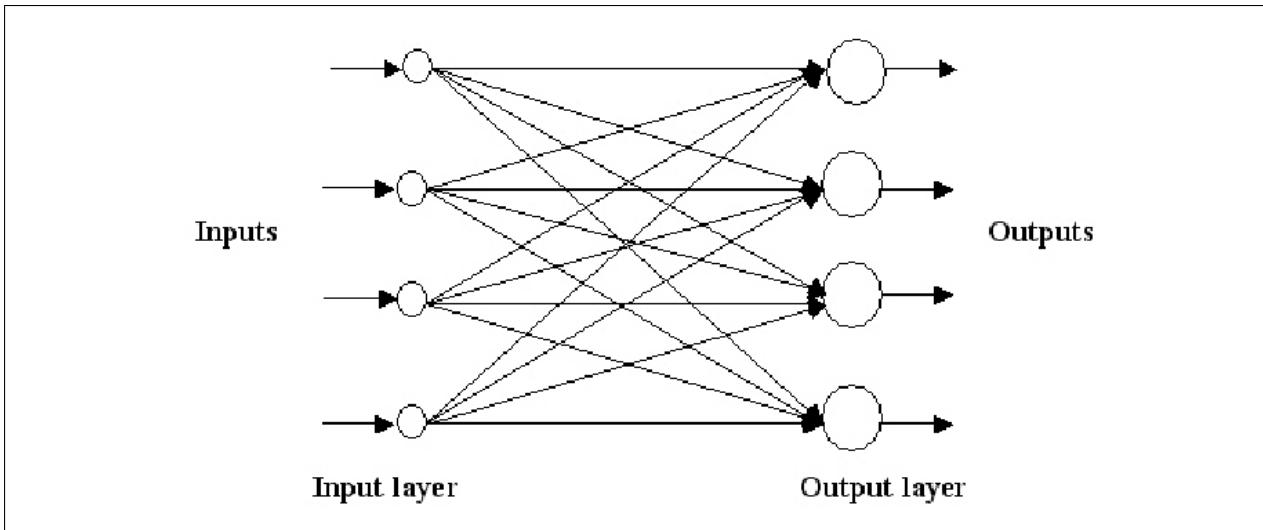
Bab ini menjelaskan teori-teori yang berkaitan mengenai teori penunjang dan jurnal terkait yang digunakan dalam proses penelitian ini seperti tinjauan studi, tinjauan pustaka yang berisi objek penelitian, landasan teori tentang metode, tahapan algoritme dan contoh penerapannya.

2.1.1 Artificial Neural Network

Artificial Neural Network adalah kecerdasan buatan yang mengadopsi kinerja otak manusia seperti memberi stimulus atau rangsangan, melakukan proses, dan memberikan keluaran. *Neural Network* juga mempunyai tiga lapisan layer yang disebut *input*, *hidden*, dan *output layer*. *Hidden layer* bisa terdapat satu sampai banyak lapisan yang disebut *Multilayer Perceptron*. Adapun yang tidak memiliki *hidden layer* (hanya *input* dan *output layer*) disebut *Single Layer Neural Network*[1].



Gambar 2.1 Neural Network (Multi Layer Perceptron).

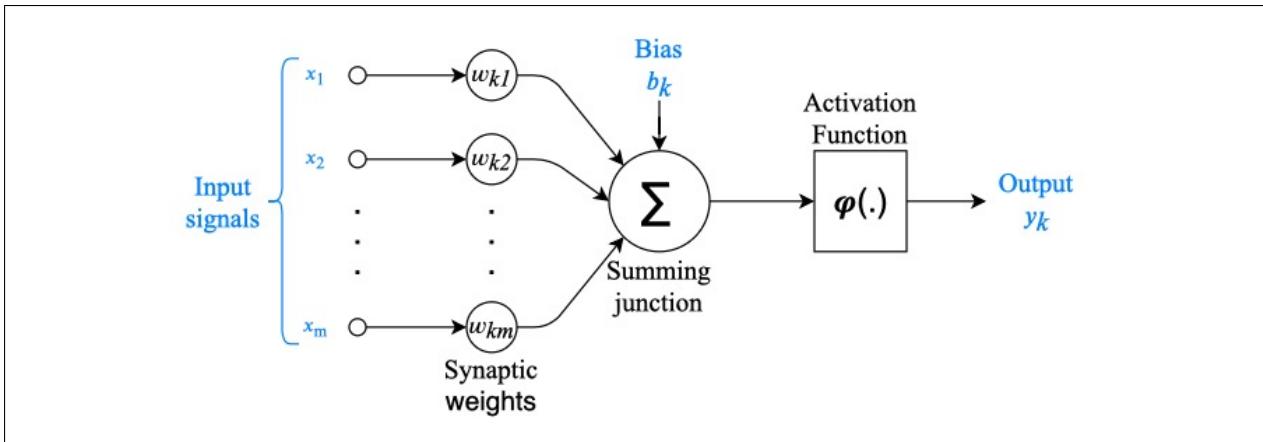


Gambar 2.2 Neural Network (Single Layer Perceptron).

Jumlah *neuron* di setiap bagian memengaruhi kemampuan generalisasi sistem, sementara jumlah *neuron* dan *hidden layer* memengaruhi efisiensi sistem. Dengan jumlah yang lebih besar, ada kemungkinan *over-fitting* pada proses latih dan generalisasi yang lemah pada data baru. Oleh karena itu, beberapa metode dapat digunakan untuk memilih jumlah *hidden layer* dan *neuron* yang tepat seperti *Genetic Algorithm*. Lebih dari satu *output layer* dapat digunakan, tergantung pada klasifikasi kesalahan yang diperlukan. Setiap *hidden layer* memiliki sejumlah *neuron*; peran masing-masing adalah untuk menghitung *weighted sum* dari nilai masukannya dan menerapkan jumlah tersebut sebagai masukan dari *activation function* yang biasanya merupakan fungsi sigmoid[6].

Neuron adalah unit pengolah informasi paling dasar untuk mengoperasikan *neural network*. Secara umum, terdapat tiga elemen dasar dalam *neuron*:^[1]

1. Sekumpulan sinapsis yang masing-masing memiliki bobot (*weight*). Sinyal masukan x_j pada sinapsis j yang terhubung dengan *neuron* k akan dikalikan dengan bobot sinapsis w_{kj} .
2. Suatu proses untuk menjumlahkan sinyal masukan yang sudah diproses dengan bobotnya pada setiap sinapsis *neuron* disebut *linear combiner*.
3. Fungsi aktivasi berfungsi untuk membatasi rentang dari *output neuron*.



Gambar 2.3 Neuron [1]

Dari Gambar 2.3 di atas, dapat dilengkapi dengan persamaan 2.1, 2.2, 2.3 berikut[1]

$$u_k = \sum_{j=1}^n w_{kj}x_j \quad (2.1)$$

$$v_k = b + \sum_{j=1}^n w_{kj}x_j \quad (2.2)$$

$$y_k = \varphi(v_k) \quad (2.3)$$

Keterangan

- u_k : keluaran *linear combiner* dari sinyal masukan
 - x : sinyal masukan
 - w_k : bobot sinapsis pada *neuron k*
 - n : jumlah masukan
 - b_k : nilai bias
 - v_k : keluaran *transfer function*
 - y_k : sinyal keluaran dari *neuron*
 - φ : fungsi aktivasi
-

Sedangkan fungsi aktivasi sigmoid yang biasa digunakan dapat dituliskan dengan persamaan berikut[7].

$$\varphi = f(x) = \frac{1}{(1 + e^{-x})} \quad (2.4)$$

$$\varphi = f(x) = \frac{1 - e^{-x}}{(1 + e^{-x})} \quad (2.5)$$

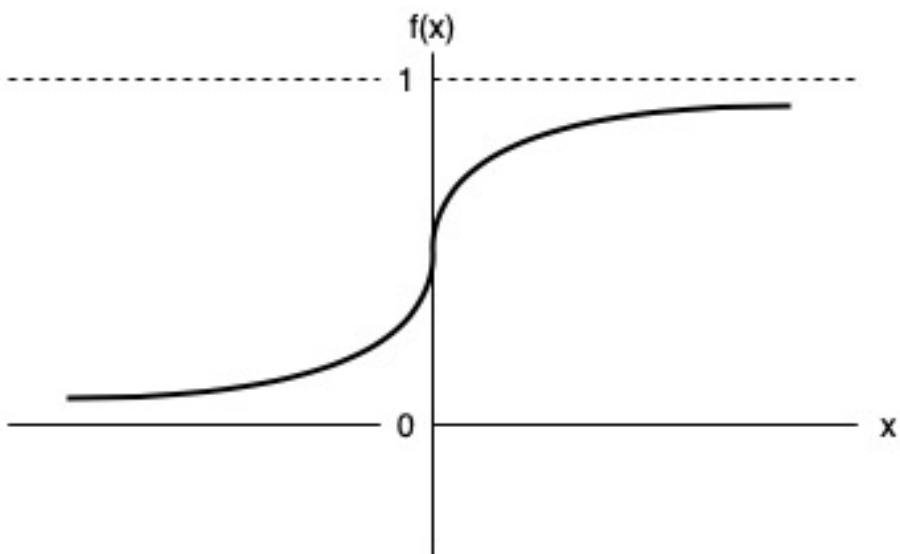
Keterangan

φ : keluaran dari *activation function*

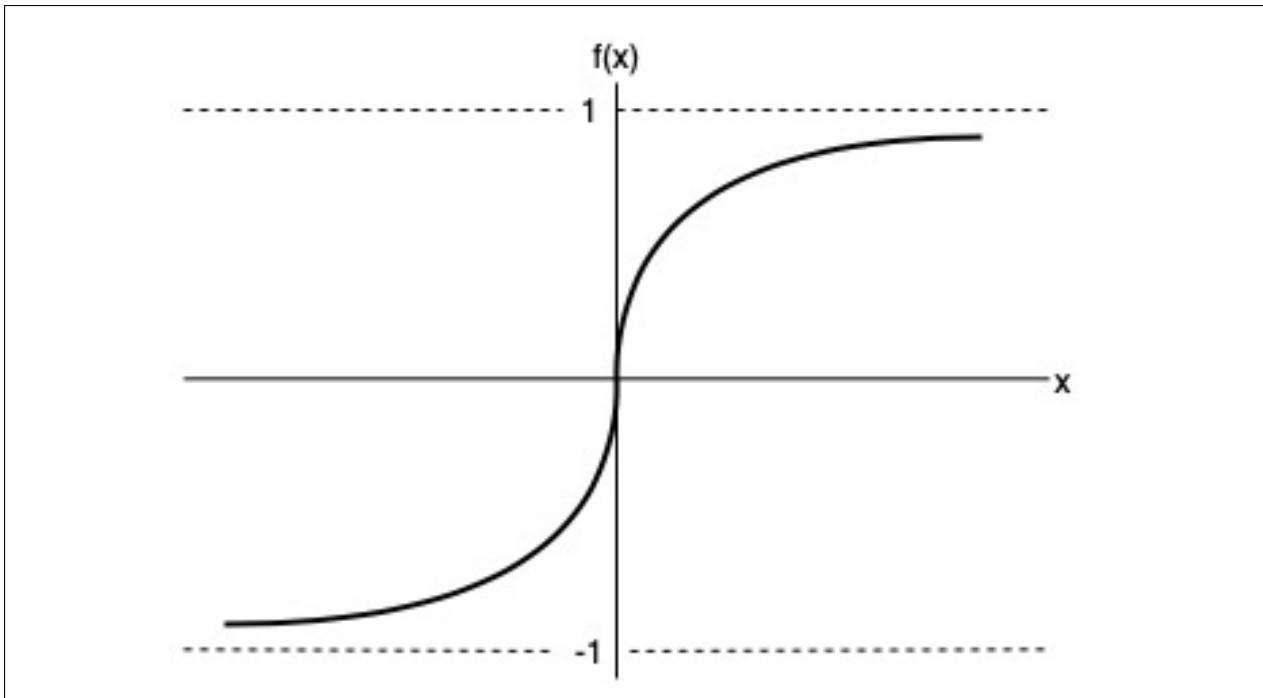
e : Euler's number

x : hasil dari *transfer function*

Rumus pada persamaan 2.4 adalah sigmoid biner yang mempunyai *range* 0 sampai 1 dan persamaan 2.5 adalah sigmoid bipolar dengan *range* -1 sampai 1 yang digambarkan sebagai berikut[7].



Gambar 2.4 Sigmoid Biner



Gambar 2.5 Sigmoid Bipolar

2.1.2 Back-Propagation Learning Algorithm

Back Propagation adalah algoritme *training* yang populer untuk metode latih MLP (*Multi Layer Perceptron*). BP bertujuan untuk meminimalkan *Mean Square Error* (MSE) proses latihnya dengan cara mengalikan vektor nilai masukan dengan bobot, yang mana bobot itu akan dijumlahkan dengan bias untuk mendapatkan nilai keluaran aktual. Nilai keluaran yang diinginkan akan dibandingkan dengan keluaran aktual, lalu dilanjutkan proses evaluasi sampai mendekati MSE yang diinginkan. Tingkat kesalahan dihitung dengan persamaan 2.6 berikut[6].

$$E = \frac{1}{N} \sum_{i=1}^N (y_i - \bar{y}_i)^2 \quad (2.6)$$

Keterangan

- E : tingkat *error* yang dihasilkan
 N : jumlah seluruh data poin
 y_i : nilai keluaran aktual pada iterasi i
 \bar{y}_i : nilai keluaran prediksi pada iterasi i
-

Jika BP sudah menghitung *error* dan *error* yang dihasilkan belum memenuhi target akan dilakukan *update weight* untuk memperkecil *error* yang dihasilkan. *Update weight* ditunjukkan oleh persamaan 2.7 berikut.

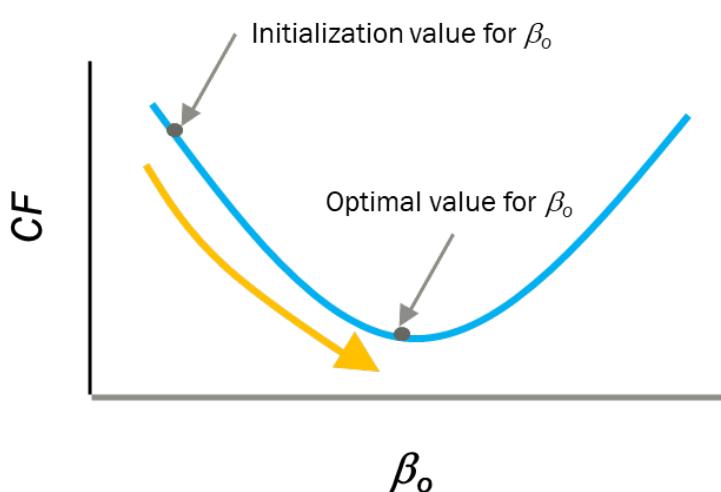
$$w_{xnew} = w_{xold} - \alpha \frac{\partial Error}{\partial w_x} \quad (2.7)$$

Keterangan

- w_x : bobot yang akan di-*update*
 - α : *learning rate*
 - $\partial Error$: *gradient descent* dari nilai *error*
 - ∂W_x : *gradient descent* dari nilai prediksi
-

Dibalik popularitas BP untuk algoritme *training* MLP, BP mempunyai kekurangan yaitu bisa terjebak dalam *local minima* saat *training* dan prosesnya bisa lambat jika jumlah iterasi nya banyak[6].

Learning Rate adalah *hyperparameter* yang penting dari sebuah *Artificial Neural Network*. *Learning rate* mengontrol seberapa besar perubahan model untuk estimasi *error* setiap kali bobot model diubah.



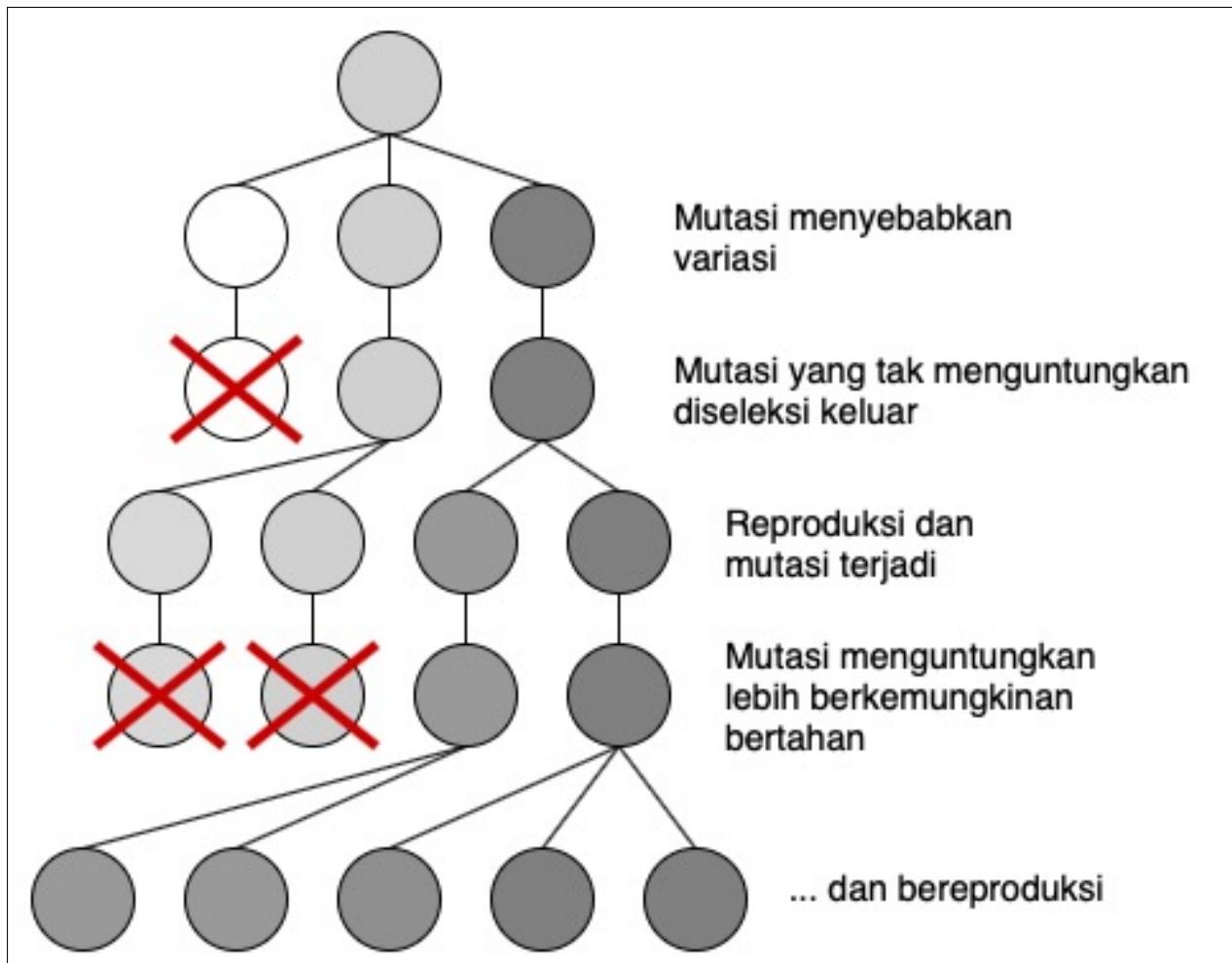
Gambar 2.6 Implementasi *Learning Rate*

Merujuk pada Gambar 2.6 di atas, memilih *learning rate* terlalu kecil bisa menyebabkan proses training yang lambat bahkan terjebak (*stuck*), sedangkan jika nilai *learning rate* terlalu besar bisa menyebabkan hasil *training* bobot yang kurang optimal karena terlalu cepat ataupun proses *training* yang tidak stabil[8].

Selain *learning rate*, *epoch* juga merupakan *hyperparameter* dari ANN yang berfungsi untuk menentukan berapa kali algoritme akan bekerja di seluruh *dataset* pelatihan. Nilai *epoch* sangat penting untuk pelatihan *dataset* karena *epoch* menentukan berapa banyak *training* yang dijalankan. Dalam satu *epoch* bisa terdapat satu atau lebih *batch* pengujian. Nilai *epoch* biasanya besar, bisa ratusan dan ribuan *epoch* untuk mendapatkan *error* yang minim[7]. Dalam satu *epoch* bisa dibagi beberapa bagian karena data yang terlalu banyak, pembagian ini disebut *batch size*. Implementasi *batch size* ini akan memengaruhi waktu yang dibutuhkan program untuk selesai. Jika jumlah sampel yang dibagi *batch size* tidak genap, pada *batch* terakhir akan mempunyai sampel yang lebih sedikit dibanding *batch* lainnya[1]. Dalam penelitian Dominic Masters., et al, dapat disimpulkan stabilitas dan performa yang paling baik untuk *training* jika menggunakan *mini batch*[9].

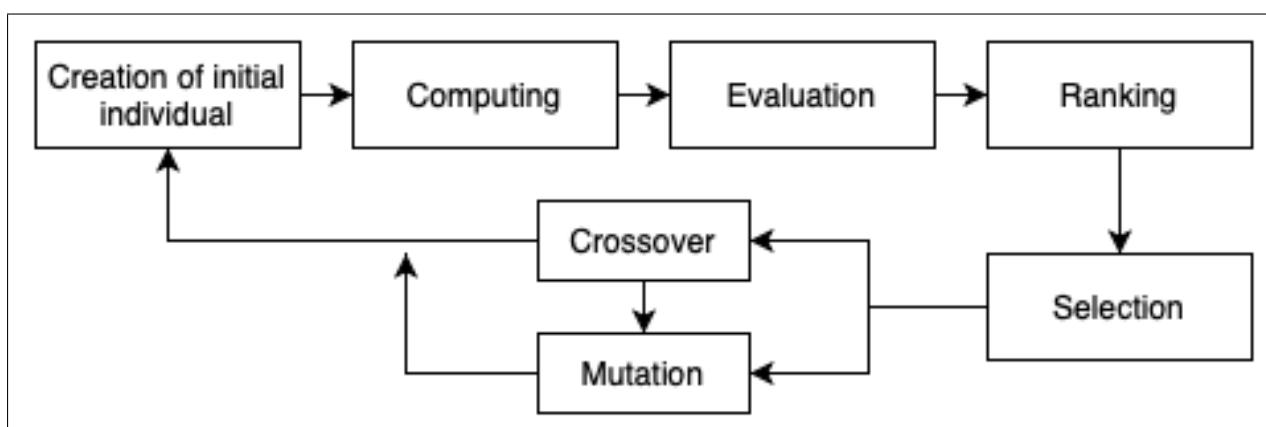
2.1.3 *Genetic Algorithm*

Algoritme genetika pada dasarnya adalah bentuk optimal yang dapat diterapkan pada fungsi yang kompleks. Tujuan dari algoritme genetika adalah mencari *fitness value* dari individu di suatu populasi. Algoritme ini didasari proses genetik yang ada pada makhluk hidup, analogi nya seperti seleksi alam. Individu dilambangkan dengan *fitness value* untuk mendapatkan solusi terbaik. Dari satu individu terdapat beberapa kromosom dan dalam kromosom terdapat beberapa gen. Setiap individu diseleksi dan individu yang bertahan bisa melakukan reproduksi silang (*crossover*) dan yang tidak lolos akan mati dengan sendirinya. Konsep dasar dari algoritme genetika ini diilustrasikan dalam gambar berikut[6].



Gambar 2.7 Ilustrasi *Genetic Algorithm*

Dan untuk alur dasar dari *Genetic Algorithm* digambarkan sebagai berikut



Gambar 2.8 Proses utama dari *Genetic Algorithm*[6]

Suatu individu dievaluasi berdasarkan fungsi tertentu sebagai ukuran performanya atau yang disebut nilai *fitness*. Penentuan nilai *fitness* sangat berpengaruh pada performa GA. Dikarenakan

seleksi dilakukan secara acak, maka tidak ada jaminan bahwa suatu individu bernilai *fitness* tertinggi akan selalu dipilih. Untuk menjaga agar individu yang mempunyai nilai *fitness* tertinggi tersebut tidak hilang selama evolusi, maka perlu dibuat satu atau beberapa salinan. Prosedur ini dikenal dengan istilah *elitism*. Untuk menghindari kecenderungan konvergen pada solusi optimal lokal dengan diperoleh nilai *fitness* baru yang memiliki variansi yang lebih besar maka perlu dilakukan penskalaan nilai *fitness* dengan menggunakan persamaan[3]

$$f(i) = f_{max} - (f_{max} - f_{min}) \left(\frac{R(i)-1}{N-1} \right) \quad (2.8)$$

Keterangan

- f_{min} : *fitness* minimum
 f_{max} : *fitness* maksimum
 N : jumlah individu
-

Seleksi dilakukan dengan memilih dua kromosom atau individu untuk dijadikan pasangan/*parent*. Metode seleksi yang digunakan adalah *tournament selection*. Sederhananya, metode ini mengambil dua kromosom secara acak dan kemudian menyeleksi salah satu yang mempunyai nilai *fitness* tertinggi untuk menjadi *parent* pertama. Cara tersebut dilakukan lagi untuk mendapatkan *parent* kedua. Seleksi dilakukan untuk mendapatkan *parent* yang nantinya akan digunakan untuk kawin silang dan mutasi. Proses ini dapat terjadi jika suatu bilangan acak yang muncul kurang dari probabilitas kawin silang dan probabilitas mutasi yang ditentukan. Generasi baru didapatkan dengan cara mengganti anggota populasi awal (hasil inisialisasi) dengan populasi baru yang terdiri dari kromosom hasil *elitism*, kawin silang dan mutasi[3].

Population adalah *hyperparameter* dari *Genetic Algorithm* yang menentukan berapa populasi baru yang akan dibuat dari data masukan. *Hyperparameter* ini menghasilkan nilai-nilai *random* yang akan diproses oleh GA sampai menemukan hasil yang terbaik untuk bobot pada *Neural Network*. Parameter populasi ini tidak perlu terlalu besar karena berpengaruh di pengukuran performa yang bisa membebani untuk evaluasi populasi yang besar. Maka dari itu, penelitian ini hanya menggunakan 5 dan 10 populasi. Sedangkan *generation* adalah *hyperparameter* yang menentukan berapa banyak *training* yang terjadi dalam GA. Satu *generation* artinya sudah melewati semua tahap yaitu menentukan *fitness*, memilih *parent*, *crossover*, dan *mutation*. Dari setiap generasi akan dihasilkan hasil *fitness* terbaik untuk dipakai lagi di generasi berikutnya[3].

2.2 Pustaka

Berikut adalah penjelasan dari *library* yang digunakan di dalam penelitian.

2.2.1 NumPy

NumPy (*Numerical Python*) adalah *library* Python yang fokus pada *scientific computing* yang bersifat *open-source*. Dikembangkan sejak tahun 2005. NumPy memiliki kemampuan untuk membentuk objek *N-dimensional array*, yang mirip dengan *list* pada Python. Keunggulan NumPy *array* dibandingkan dengan *list* pada Python adalah konsumsi *memory* yang lebih kecil serta *runtime* yang lebih cepat. NumPy juga memudahkan kita pada Aljabar Linear, terutama operasi pada Vector (1-d array) dan Matrix (2-d array). *Function* yang digunakan dari *library* ini yakni:

Tabel 2.1 Daftar *method* yang digunakan dari *library NumPy*

No	Method	Masukan	Luaran	Keterangan
1	array	dataX : object, data Y : object	-	Membuat <i>array</i> dari <i>object</i> yang ada.
2	random.uniform	low : float, high : float, size : float	rand	Mengacak bilangan dengan interval yang sudah ditentukan.

2.2.2 Pandas

Pandas (*Python for Data Analysis*) adalah *library* Python yang fokus untuk proses analisis data seperti manipulasi data, persiapan data, dan pembersihan data. Pandas menyediakan struktur data dan fungsi high-level untuk membuat pekerjaan dengan data terstruktur/tabular lebih cepat, mudah, dan ekspresif. *Function* yang digunakan dari *library* ini yakni:

Tabel 2.2 Daftar *method* yang digunakan dari *library Pandas*

No	Method	Masukan	Luaran	Keterangan
1	read_csv	dataset.csv : filepath	-	Membaca <i>file</i> berekstensi .csv.
2	DataFrame	-	-	Membentuk tipe data menjadi DataFrame.
3	loc	series2 : datetime	data[]	Memisahkan data berdasarkan kondisi yang ditentukan.
4	isnull	-	-	Untuk mengetahui data bernilai null atau tidak.
5	dropna	-	-	Membersihkan data dan menghilangkan data yang kosong.

2.2.3 Scikit-learn

Scikit-learn (secara umum dikenal dengan nama Sklearn) adalah *library* gratis untuk *software machine learning* untuk bahasa pemrograman python. Sklearn mempunyai banyak klasifikasi, regresi dan *clustering algorithms* termasuk *support vector machines* (SVM), *random forest*, *gradient boosting*, *k-means*, dan DBSCAN. *Library* ini juga didesain untuk beroperasi dengan *library numerical* dan *scientific* dari python yaitu NumPy dan SciPy. *Function* yang digunakan dari *library* ini yakni:

Tabel 2.3 Daftar *method* yang digunakan dari *library Sklearn*

No	Method	Masukan	Luaran	Keterangan
1	StandardScaler	<i>dataset</i> : float, <i>range_min</i> : int, <i>range_max</i> : int	<i>dataset</i> : float	Membuat suatu nilai untuk berada di <i>range</i> tertentu.

2.2.4 Matplotlib

Matplotlib adalah *library* Python yang fokus pada visualisasi data seperti membuat plot grafik. Matplotlib pertama kali diciptakan oleh John D. Hunter dan sekarang telah dikelola oleh tim developer yang besar. Awalnya matplotlib dirancang untuk menghasilkan plot grafik yang sesuai pada publikasi jurnal atau artikel ilmiah. Matplotlib dapat digunakan dalam skrip Python, Python dan IPython shell, server aplikasi web, dan beberapa *toolkit graphical user interface* (GUI) lainnya. *Function* yang digunakan dari *library* ini yakni:

Tabel 2.4 Daftar *method* yang digunakan dari *library Matplotlib*

No	Method	Masukan	Luaran	Keterangan
1	plot	-	-	Membuat plot grafik dari data.
2	pyplot.show	-	<i>graph</i>	Menampilkan plot grafik yang sudah dibuat.
3	pyplot.legend	-	-	Memberi legenda pada grafik sesuai dengan keterangannya.

2.2.5 Keras

Keras merupakan *library* jaringan saraf tiruan tingkat tinggi yang ditulis dengan bahasa *python*. *Library* ini menyediakan fitur yang digunakan dengan fokus mempermudah pengembangan lebih dalam tentang *Deep Learning*. *Function* yang digunakan dari *library* ini yakni:

Tabel 2.5 Daftar *method* yang digunakan dari *library Keras*

No	Method	Masukan	Luaran	Keterangan
1	model.add	hidden dim : int, input dim : int, activation function : string	-	Inisialisasi konfigurasi dimensi layer dan fungsi aktivasi.
2	model.compile	loss : string, learning rate : float, metrics : string	-	Inisialisasi konfigurasi indikator <i>learning rate</i> dan pengukuran yang dipakai (MSE dan RMSE).
3	model.fit	trainX : float[], trainY : float[], <i>epoch</i> : int, <i>batch size</i> : int	-	Menjalankan proses ANN dengan konfigurasi model <i>dataset</i> dan indikator yang memengaruhi efektifitas pengujian.

2.3 Tinjauan Studi

Pada bagian ini akan dijelaskan mengenai perbandingan dari berbagai penelitian terkait metode *Artificial Neural Network* dan kombinasinya.

Terdapat beberapa metode lain yang memiliki ruang lingkup yang mirip dengan penelitian mengenai *Demand Forecasting* menggunakan *machine learning* (ANN). Tabel 2.6 *State of the Art* akan menjelaskan perbedaan-perbedaan metode yang telah dipelajari oleh penulis dari jurnal. Dari banyak jurnal yang dibaca dan diteliti, belum ada penelitian tentang penggabungan ANN-GA ini untuk *demand forecasting*. Jurnal dalam *state of the art* dipilih karena menggunakan algoritme yang sama dengan penelitian ini yaitu, penggabungan ANN dan GA. Walaupun bidangnya berbeda, secara konsep dan teori penelitiannya masih serupa, maka dari itu peneliti menggunakan jurnal-jurnal ini untuk referensi. Hasil penelitian dari jurnal tersebut juga berhasil karena dapat meningkatkan akurasi walaupun objek yang dibahas berbeda dengan penelitian ini, namun konsep dan teori yang dipakai sangat mendukung untuk penelitian ini.

Tabel 2.6 State-of-the-art method

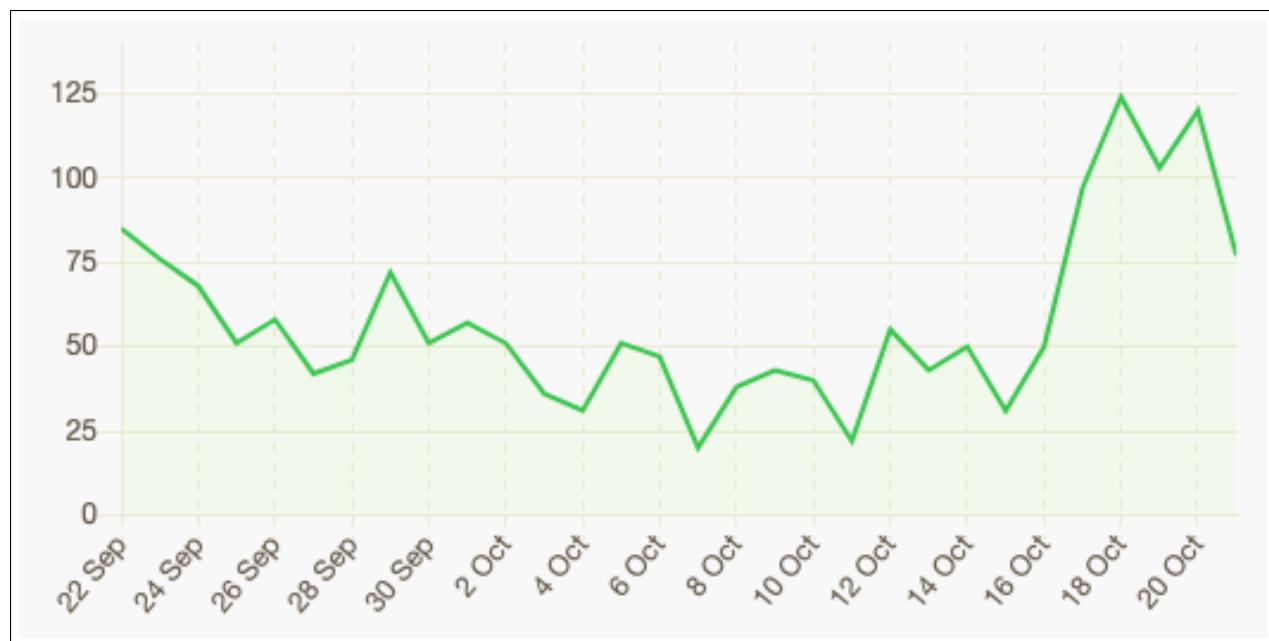
Penelitian	Metode	Hasil Penelitian
Purnawansyah. “Backpropagation and Genetic Algorithm in Network Traffic Activities Forecasting”, <i>International Journal of Computing and Informatics (IJCANDI)</i> (2017) [10]	1. <i>Back-Propagation</i> 2. <i>Genetic Algorithm</i>	Untuk data <i>Human Development Index</i> yang digunakan, <i>Back Propagation</i> dengan konfigurasi 3-48-1 mampu menghasilkan MSE sebesar 0.0006387%.
Montri Inthachot., et al. ”Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend”, <i>Computational Intelligence and Neuroscience</i> (2016) [2]	1. <i>Feed-Forward NN</i> 2. <i>Genetic Algorithm</i>	<i>Hybrid ANN-GA</i> menghasilkan penambahan akurasi 12.4011% menjadi 63.60% dibanding metode ANN saja. Tapi hasil ini masih harus ditingkatkan untuk prediksi <i>trend</i> yang lebih akurat.
Ali Saud Al Tobi., et al. “A Review on Applications of Genetic Algorithm for Artificial Neural Network” <i>International Journal of Advance Computational Engineering and Networking</i> (2016) [11]	1. <i>Genetic Algorithm</i> 2. <i>Back-Propagation</i> 3. <i>Hybrid GABP-ANN</i>	Akurasi meningkat dari 93% menjadi 95% dengan menggabungkan metode <i>Genetic Algorithm</i> dan <i>Back Propagation Algorithm</i> untuk proses latih (<i>hybrid</i>). GA juga berguna untuk menentukan jumlah <i>neuron</i> yang terbaik untuk seleksi fitur.

2.4 Tinjauan Objek

Pada bagian ini akan dijelaskan tentang objek penelitian yang akan dilakukan yaitu *Demand* dan terkait juga dengan *dataset* yang akan digunakan secara singkat.

2.4.1 *Demand Forecasting*

Berdasarkan pengertian dari Kamus Besar Bahasa Indonesia, *demand* diartikan sebagai permintaan/tuntutan terhadap suatu barang atau jasa. *Demand* adalah prinsip ekonomi untuk memenuhi keinginan konsumen dengan membeli barang atau jasa dan bersedia membayar dengan harga yang sudah ditetapkan. Dengan anggapan semua faktor lain konstan, kenaikan harga barang atau jasa akan menurunkan kuantitas *demand*, begitu juga sebaliknya. *Demand* digambarkan dengan statistik berikut



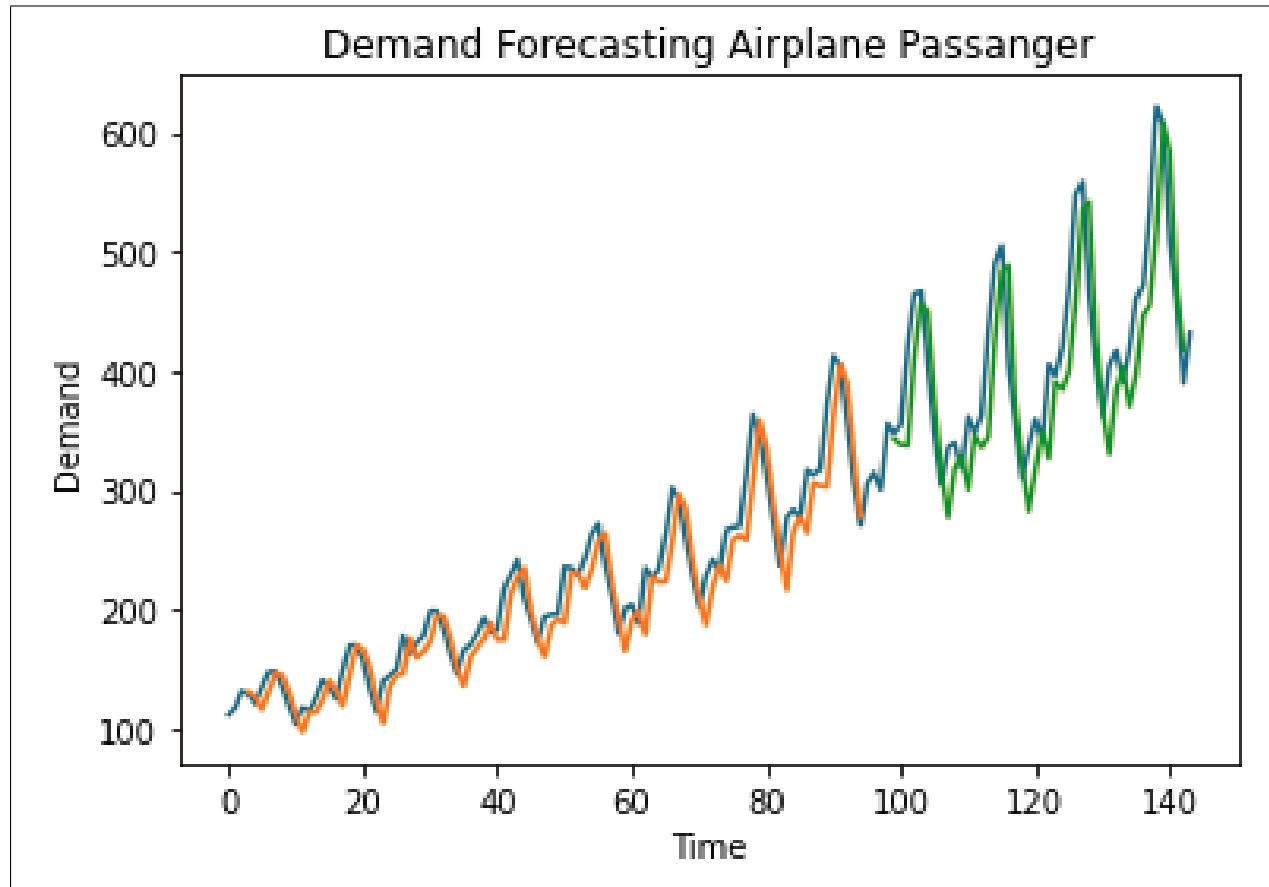
Gambar 2.9 Contoh Demand; x = waktu/tanggal, y = produk yang terjual

Demand berhubungan dekat dengan *supply*. *Supply* bergantung pada stok dan harga barang dan *supply* harus mengimbangi *demand*, karena saat harga barang terlalu tinggi *demand* akan berkurang dan begitu juga sebaliknya. Setiap pelaku usaha mempunyai stok yang akan disimpan sampai stok itu diperlukan. Pembeli mengurangi stok barang sesuai dengan *demand* yang ada, dan pemasok/*supplier* menambahkan stok untuk mengimbangi *demand* dan *supply* barang tersebut. Mengetahui *demand* akan memengaruhi pelayanan kepada pembeli, memberi kepuasan, dan meningkatkan nilai dari produk tersebut. Selain itu, mengetahui *demand* akan memengaruhi operasional yang lebih produktif, efisien, dan menekan biaya penyimpanan. *Rule of thumb* mengatakan biaya untuk menyimpan stok barang adalah sekitar 20% dari nilai barang tersebut dalam setahun. Dalam penyimpanan tersebut kualitas barang tertentu bisa berkurang apabila stok yang dimiliki jauh melampaui *demand* yang ada[12].

Forecasting arti umumnya adalah untuk memprediksi sesuatu di masa yang akan datang. Banyak bidang yang memakai *forecasting* ini seperti bidang teknologi, ekonomi dan bisnis. Rencana bisnis bisa berjalan efektif di masa depan saat keadaan yang berlaku sesuai dengan perkiraan/peramalan di masa mendatang. Hal ini tidak bisa menjadi acuan pasti, tetapi *forecasting* bisa memberi gambaran agar stok yang dipersiapkan tidak terlalu jauh dari kenyataan di masa depan. *Demand forecasting* bisa berfungsi untuk penentuan harga dan keputusan yang akan diambil[12].

Forecasting sering kali digolongkan menjadi *short-term*, *medium-term (mid-term)*, dan *long-term*. *Short-term forecasting* meramalkan kejadian hanya beberapa periode waktu ke depan (harian, mingguan, bulanan). *Mid-term forecasting* meramalkan satu sampai dua tahun ke depan, dan

long-term forecasting bisa meramalkan beberapa tahun ke depan. *Short* dan *mid-term forecasting* diperlukan untuk aktivitas yang mencakup manajemen operasional, pengelolaan stok, *budgeting*. *Long-term forecasting* diperlukan untuk *strategic planning* dari pelaku usaha misalnya *new product planning*, *facility location planning*, dan *research and development* (R&D). Dilihat dari keperluan masing-masing golongan, *short-term forecast* diharapkan memiliki tingkat akurasi paling tinggi karena menyangkut penjualan dan pengelolaan stok. Dalam penelitian ini *forecast* yang dimaksud adalah untuk *short-term forecast* yang berarti memerlukan tingkat akurasi yang tinggi[13].



Gambar 2.10 Contoh *Demand Forecasting*

Demand Forecasting menggunakan data *time-series* dengan sumbu x nya adalah waktu seperti pada Gambar 2.10. *Demand Forecasting* melibatkan metode kuantitatif seperti penggunaan data penjualan historis atau *demand* bulanan. Secara tradisional, metode untuk analisis *time series* seperti *simple average*, *moving average*, *exponential smoothing*, ARIMA, dan lainnya digunakan untuk peramalan kuantitatif (*quantitative forecasting*). Masalah umum pada pendekatan *time series* adalah ketidaktepatan prediksi dan ketidakstabilan angka. Sebagian besar metode *time series* tradisional lebih sulit untuk dikembangkan. Akhir-akhir ini, aplikasi *neural network* telah meningkat dalam bisnis. Salah satu aplikasi penting dari ANN adalah di bidang *forecasting*

penjualan. Hal yang menarik dari ANN untuk *forecasting* karena ANN adalah metode *self-adaptive* yang bisa beradaptasi sendiri[12].

2.4.2 Dataset

Dataset yang dipakai dalam penelitian ini adalah data historis penjualan. *Dataset* didapatkan dari situs *Kaggle* dengan *link* berikut:

<https://www.kaggle.com/felixzhao/productdemandforecasting>. *Kaggle* mempunyai banyak *database* yang bisa didapatkan secara gratis untuk umum. *Dataset* ini berisi data historis penjualan dari perusahaan manufaktur dengan *footprint* global, yaitu Walmart. Kolom data terdiri dari kode produk, kode gudang, kategori produk, tanggal pembelian, dan banyaknya kuantiti order. Perusahaan ini menyediakan 2.160 produk dalam 33 produk kategori dan 4 gudang penyimpanan yang berbeda dengan 1.048.575 data yang tercakup dari tahun 2011 sampai tahun 2017. Pembuatan produk-produk tersebut tersebar di lokasi berbeda di seluruh dunia. Data ini memiliki format ekstensi *Comma-Separated Values* (.csv). Tabel 2.7 berikut adalah contoh data yang diperoleh dari *dataset* yang mewakili data penjualan.

Tabel 2.7 Contoh data

No	Product Code	Warehouse	Product Category	Date	Order Demand
1	Product_0993	Whse_J	Category_028	2012/7/27	100
2	Product_0979	Whse_J	Category_028	2012/1/19	500
3	Product_0979	Whse_J	Category_028	2012/2/3	500
4	Product_0979	Whse_J	Category_028	2012/2/9	500
5	Product_0979	Whse_J	Category_028	2012/3/2	500
6	Product_0979	Whse_J	Category_028	2012/4/19	500
7	Product_1159	Whse_J	Category_006	2012/1/6	50000
8	Product_1159	Whse_J	Category_006	2012/1/18	100000
9	Product_1159	Whse_J	Category_006	2012/2/2	50000
10	Product_1159	Whse_J	Category_006	2012/2/22	50000
11	Product_1938	Whse_J	Category_001	2012/6/6	4
12	Product_1938	Whse_J	Category_001	2012/7/13	4
13	Product_1938	Whse_J	Category_001	2012/12/13	4
14	Product_1157	Whse_J	Category_006	2012/3/8	150000
15	Product_1157	Whse_J	Category_006	2012/4/23	150000

BAB 3 ANALISIS DAN PERANCANGAN

Bab ini menjelaskan analisis masalah yang diatasi berserta pendekatan dan alur kerja dari algoritme yang dikembangkan, mengimplementasikan metode yang digunakan, dan hasil yang akan ditampilkan.

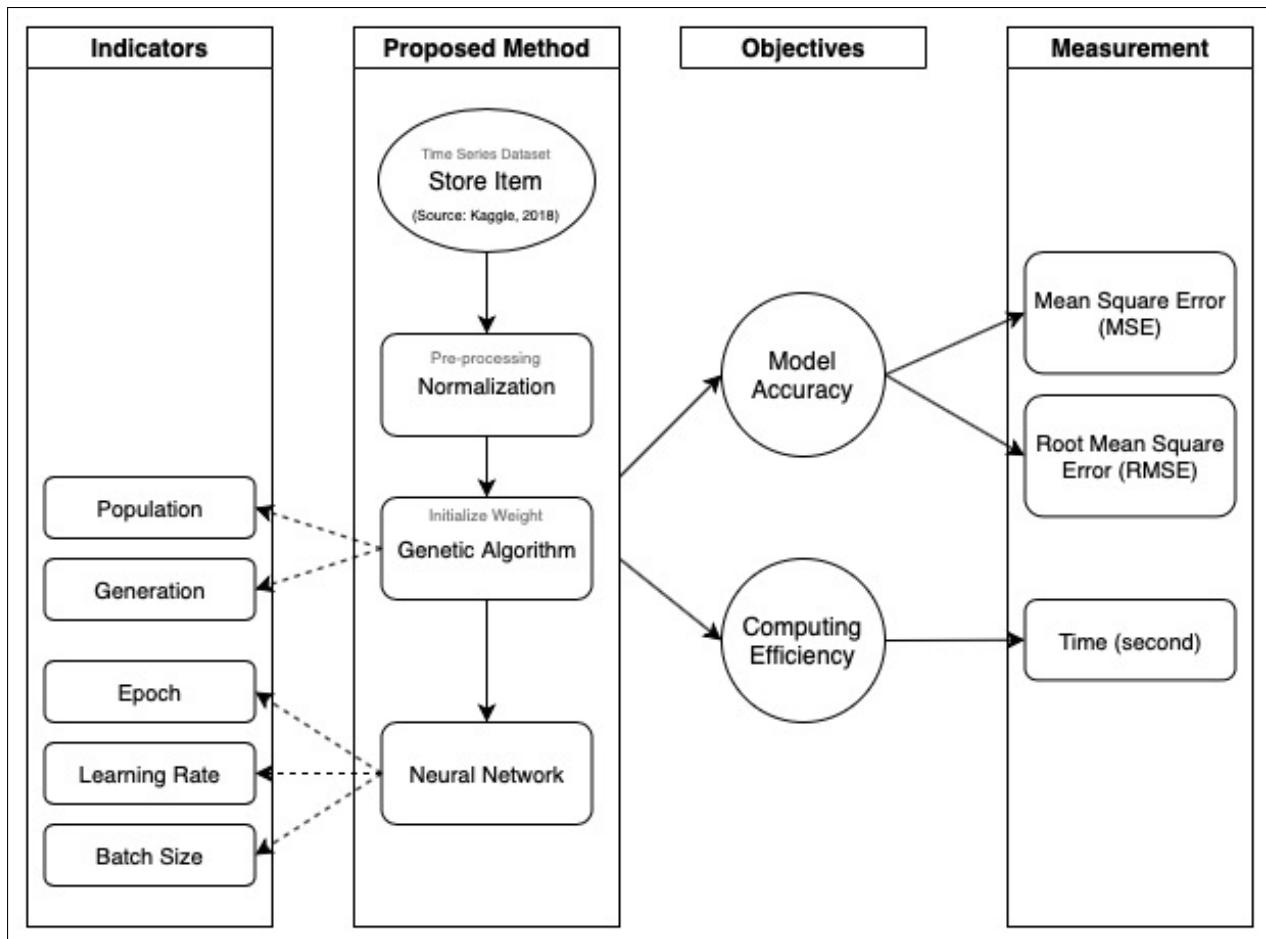
3.1 Analisis

Pada Bab I telah dijelaskan bahwa *Demand Forecasting* menjadi kebutuhan yang cukup penting bagi pelaku usaha (toko, agen, perusahaan, dan lainnya). Pada penelitian ini, penulis menggunakan *dataset* penjualan dengan parameter yang umum agar lebih terbayang pada transaksi yang sederhana. *Dataset* yang digunakan menggunakan data yang cukup banyak, dengan seperti itu diharapkan tahap *training* akan menghasilkan tingkat *error* yang rendah untuk mendukung tahap *testing* agar lebih baik.

Penelitian ini akan menggunakan penggabungan algoritme *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA). ANN akan berjalan normal tanpa ada perubahan algoritme utama, namun saat inisialisasi bobot pada ANN, bobot tersebut akan diisi oleh GA setelah melalui seluruh proses GA. Jika tanpa penggabungan ini, bobot ANN akan diinisialisasi dengan angka acak. Inisialisasi GA pun memakai angka acak, namun akan disempurnakan melalui proses *selection*, *crossover*, *mutation* dan perhitungan nilai *fitness*. Dengan penggabungan algoritme tersebut, diharapkan pengujian akan mengeluarkan hasil yang lebih baik lagi daripada ANN tanpa GA.

3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran dari metode yang diusulkan untuk melakukan *Demand Forecasting*. Seperti pada Gambar 3.1 berikut, data akan dinormalisasi di tahap *pre-processing* menggunakan *min-max normalization*, lalu data akan dilatih menggunakan *genetic algorithm* dan *back propagation algorithm* seperti yang dikatakan Ali Saud Al Tobi., et al. pada penelitian nya bahwa algoritme *hybrid* dari ANN dan GA mampu meningkatkan akurasi dari algoritme ANN saja[11].

**Gambar 3.1** Kerangka Pemikiran

Pada penelitian ini, indikator yang dipilih dari algoritme genetik adalah *population* dan *generation*. Merujuk pada sub-bab *Genetic Algorithm* di Bab 2, *population* adalah *hyperparameter* yang menentukan jumlah populasi baru yang akan dibuat dari data masukan. Semakin besar jumlah *population* berarti semakin banyak pilihan populasi untuk diseleksi dalam tahap *ranking* dan *selection* seperti pada Gambar 2.8. Sedangkan *generation* adalah *hyperparameter* untuk mengatur pengulangan dari proses GA untuk mendapatkan nilai *fitness* yang lebih baik lagi.

Indikator yang dipilih dari *Neural Network* ada tiga dan akan dijelaskan sebagai berikut. *Learning rate* dipakai karena menjadi *hyperparameter* yang sangat penting dalam ANN. Seperti pada sub-bab *Back-Propagation Learning Algorithm* di Bab 2, *learning rate* adalah parameter yang berfungsi untuk mengatur besar perubahan model saat bobot diperbaharui. *Learning rate* yang terlalu kecil akan lambat dan *stuck* sedangkan jika terlalu besar, hasilnya tidak akan maksimal karena perubahan bobot tidak stabil. Nilai umum yang digunakan pada LR adalah 0.1 atau 0.01[8]. Nilai umum (*default*) dari *library Keras* yang dipakai adalah 0.001. Maka dari itu, nilai yang akan digunakan pada penelitian ini adalah 0.01, 0.001 dan 0.0001 sebagai rujukan ke

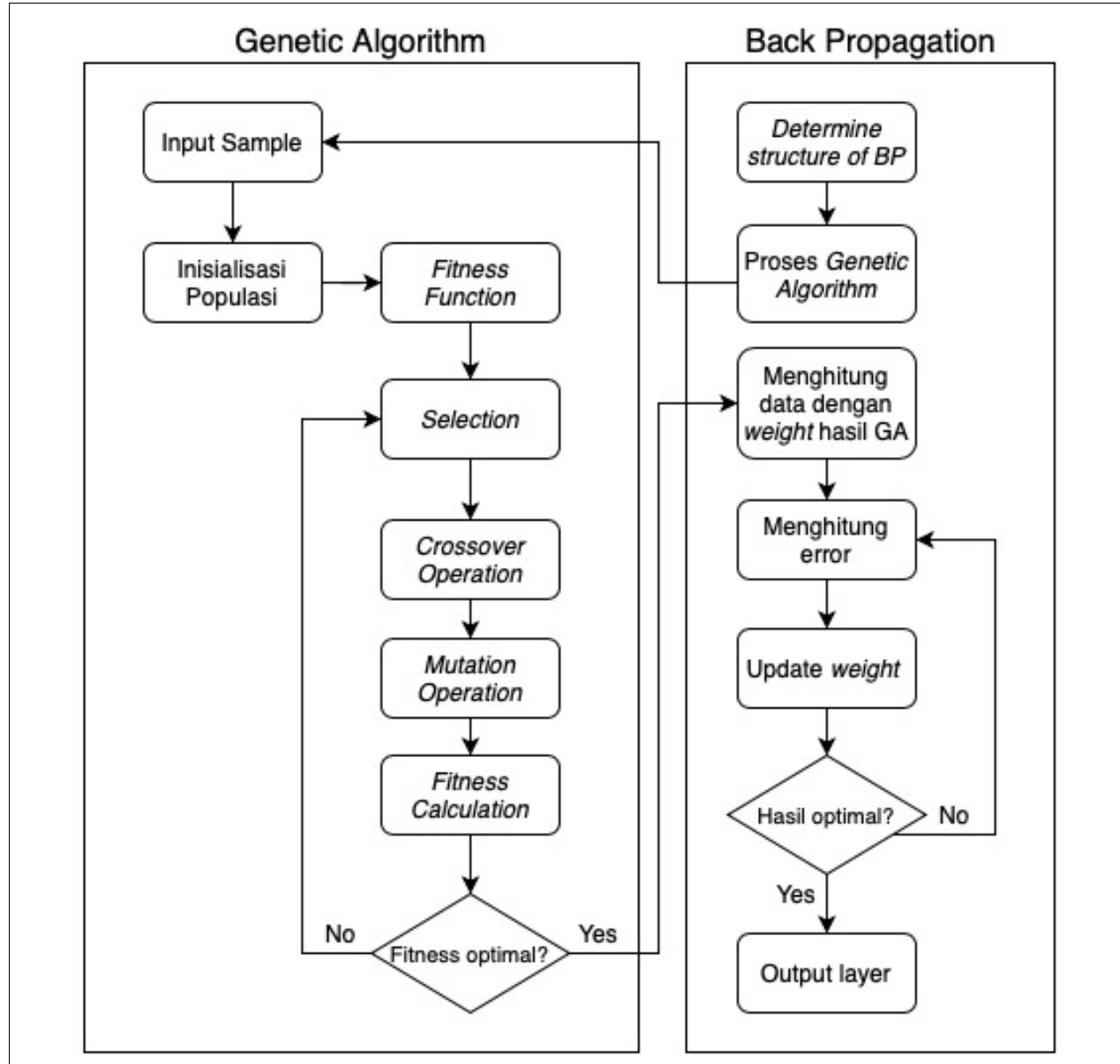
nilai umum (*default*) dan 0.0001 untuk melihat perbedaan yang dinilai kecil untuk *learning rate*.

Selanjutnya indikator *epoch*, adalah *hyperparameter* yang menentukan berapa kali algoritme akan bekerja di seluruh *dataset* pelatihan. Nilai *epoch* sangat penting untuk pelatihan *dataset* karena *epoch* menentukan berapa banyak *training* yang dijalankan. Nilai *epoch* biasanya besar, bisa ratusan dan ribuan *epoch* untuk mendapatkan *error* yang minim. Nilai *epoch* yang optimal akan berbeda pada setiap kasus, namun pada penelitian ini nilai yang akan digunakan adalah 1000, 1500 dan 2000[7]. Dalam satu *epoch* bisa dibagi beberapa bagian karena data yang terlalu banyak, pembagian ini disebut *batch size*. Implementasi *batch size* ini akan memengaruhi waktu yang dibutuhkan program untuk selesai. Pada banyak kasus, hasil yang terbaik adalah 32 atau lebih kecil, bahkan sekecil 2 atau 4 *batch*[9]. Pada penelitian ini nilai *batch size* yang akan digunakan adalah 1, 2, dan 4.

Dalam penelitian ini pengukuran akurasi dilihat dari nilai *error* yang didapat dari pengujian yaitu, *Mean Square Error* (MSE) dan *Root Mean Square Error* (RMSE). MSE dan RMSE menjadi pengukuran yang baik untuk *dataset* yang memiliki data pencilan (*outliers*). Dalam pengukuran ini, nilai RMSE lebih baik karena nilai *error* yang didapat selisihnya lebih akurat karena efek kuadrat dari MSE sudah hilang. Pada penelitian ini juga diperlukan pengukuran waktu untuk membandingkan pengujian, karena dalam suatu pengujian mungkin menghasilkan *error* sedikit lebih kecil tapi memerlukan waktu yang jauh lebih lama. Hal ini perlu untuk evaluasi efisiensi waktu prediksi. Maka dari itu, penelitian ini juga menggunakan indikator *batch size* untuk mengoptimalkan waktu yang diperlukan.

3.3 Analisis Urutan Proses Global

Dalam penelitian ini dibagi menjadi dua tahap, yaitu tahap *training* dan tahap *testing*. Tahap *training* dilakukan untuk mendapatkan model peramalan yang optimal. Tahap *testing* dilakukan untuk menghitung tingkat *error* dan menjadi hasil acuan untuk data yang diramal.



Gambar 3.2 Flowchart Proses Global pada historical sales

Merujuk pada Gambar 3.2 di atas, uraian *flowchart* dimaksudkan sebagai berikut:

Genetic Algorithm dan Back Propagation Algorithm

1. Tentukan struktur ANN dengan inisialisasi *layer* dan *neuron*.
2. Masuk ke proses GA untuk memroses nilai bobot.
3. Masukkan dalam *fitness function* agar individu yang punya nilai *fitness* tinggi tidak hilang (*ranking*).
4. Seleksi kromosom/individu secara acak untuk menjadi *parent* pertama dan begitu juga untuk *parent* kedua.

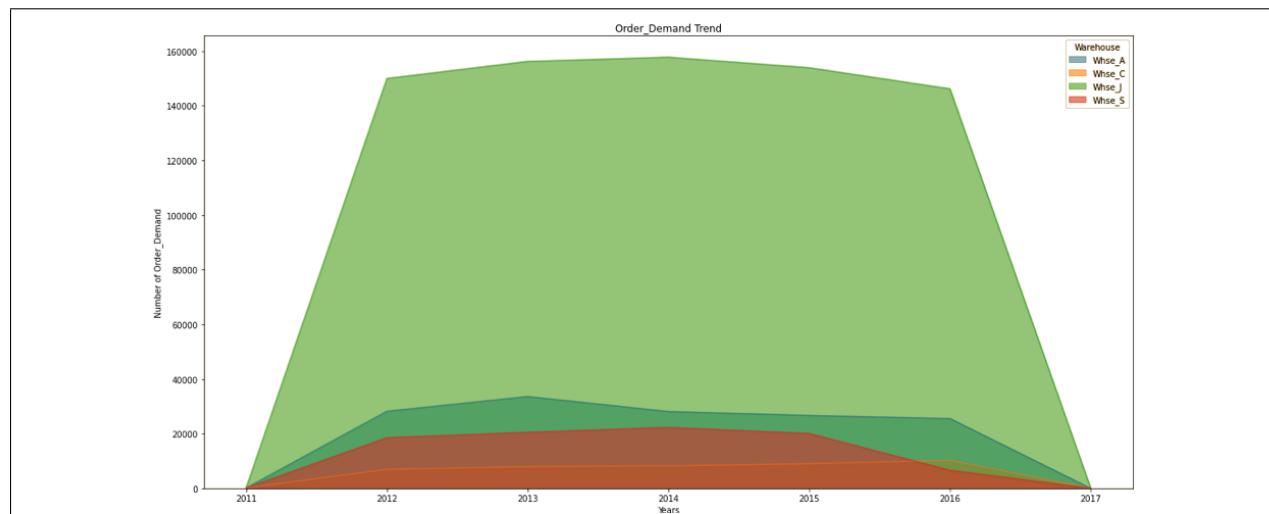
5. Menjalankan operasi kawin silang dan mutasi untuk mendapatkan generasi yang optimal.
 6. Menghitung nilai *fitness*, jika sudah cukup dilanjutkan dengan proses *Back Propagation*, jika belum kembali lagi pada proses seleksi di atas.
 7. Pada proses *Back Propagation* inisialisasi parameter bobot mengambil dari hasil GA yang sudah optimal dan jalankan perhitungan ANN.
 8. Hitung error pada *neuron output* tersebut lalu perbaharui *weight* pada setiap lapisan *neural network*.
 9. Jika hasil belum optimal, lakukan lagi langkah *back propagation* memperbaharui *weight* sampai hasil yang diinginkan tercapai.
-

3.4 Analisis Kasus

Pada bagian ini dilakukan analisis tahapan proses dengan melakukan perhitungan manual.

3.4.1 Data Sampling

Peneliti mencoba mengeluarkan gambaran jumlah *dataset* berdasarkan kolom *Warehouse*. Gambar 3.3 didapat dengan mengolah *dataset* menjadi pertahun dan menjumlahkan *demand* berdasarkan lokasi gudang. Grafik ini adalah *plotting* dari *coding Python* untuk melihat sebaran data yang ada.



Gambar 3.3 *Demand* dari setiap gudang berdasarkan tahun.

Setelah melihat grafik pada Gambar 3.3, terlihat data pada rentang tahun 2011 sampai 2012 dan 2016 sampai 2017 ada penurunan dan kenaikan yang tidak wajar dan tidak stabil seperti tahun lainnya. Dari fenomena tersebut, *dataset* tersebut dijabarkan untuk melihat penyebab dari ketidakseimbangan data. Hasil penjabaran data dapat dilihat pada Tabel 3.1, ditemukan *order demand* pada tahun 2011 sampai 2012 banyak data dengan kuantiti nol. Selain itu, ditemukan juga banyak data dengan tanggal yang kosong atau Nan (*not a number*). Maka dari itu, sebelum

melakukan *data sampling*, *dataset* ini perlu dilakukan pengolahan agar data yang dipakai bersih tanpa data kosong.

Tabel 3.1 Contoh data kosong yang terdapat pada *dataset*

No	Product Code	Warehouse	Product Category	Date	Order Demand
1	Product_1455	Whse_J	Category_019	2012/3/9	0
2	Product_1381	Whse_J	Category_019	2012/4/20	0
3	Product_1455	Whse_J	Category_019	2012/4/20	0
4	Product_1377	Whse_J	Category_019	2012/5/1	0
5	Product_1381	Whse_J	Category_019	2012/5/1	0
6	Product_1452	Whse_J	Category_019	Nan	110
7	Product_1455	Whse_J	Category_019	Nan	350
8	Product_1294	Whse_J	Category_019	Nan	1000
9	Product_1452	Whse_J	Category_019	Nan	99
10	Product_1381	Whse_J	Category_019	Nan	1250

Data yang akan digunakan terbagi menjadi 60% data *training*, 40% data *testing*. Data yang dipakai untuk *training* dan *testing* diambil secara terurut tidak acak (misalnya ada 10 data, diambil data ke 1-6 untuk *training* dan 7-10 untuk *testing*). Data tersebut berisi *record* penjualan perhari yang akan menjadi acuan untuk peramalan berikutnya. Dari sekian banyak data yang ada, penelitian ini hanya akan mengambil data dalam kurun waktu bulan Januari 2013 sampai bulan Desember 2016 sebanyak 833.008 data karena diluar kurun waktu tersebut, banyak data kosong yang tidak bisa dijadikan parameter seperti pada Gambar 3.3 dan Tabel 3.1 di atas.

Dari data historis penjualan itu diseleksi bisa menjadi data bulanan, data *demand* per produk, data *demand* per kategori dan lain-lain. Dari lima fitur tersebut, *Warehouse* tidak dipakai, *Product Code* dan *Product Category* hanya dipakai untuk menyaring data mana yang akan dipakai, dan dua fitur lainnya yaitu *Date* dan *Order Demand* akan dipakai karena sesuai dengan karakteristik *time series* yang memerlukan data waktu untuk sumbu x dan data *demand* untuk sumbu y. Pada penelitian ini, data disaring berdasarkan akan digunakan data harian berdasarkan kode Product_1359 karena menjadi produk dengan kuantitas penjualan terbanyak dalam kategori produk paling banyak, dan mempunyai nilai ekstrim yang menjadi tantangan dalam pengujian ini. Namun pada bagian perhitungan manual ini, akan dicontohkan pemilihan data bulanan berdasarkan produk dengan kode Product_1359 pada tahun 2013 saja.

Tabel 3.2 Dataset tahun 2013 berdasarkan Product_1359

No	Date	Demand
1	2013-01	9460000
2	2013-02	8216000
3	2013-03	8025000
4	2013-04	7902000
5	2013-05	8041000
6	2013-06	8324000
7	2013-07	7578000
8	2013-08	8732000
9	2013-09	7590000
10	2013-10	9626000
11	2013-11	10450000
12	2013-12	8748000

3.4.2 Normalization

Pada tahap awal, seluruh data akan dinormalisasi, normalisasi yang dimaksud berbeda dengan normalisasi pada *database*. Tahap ini merupakan normalisasi pada *Data Mining*, proses penskalaan nilai atribut dari data agar nilai keseluruhan data bisa ada di *range* tertentu. Dalam normalisasi ini data akan berada di *range* -1.0 sampai 1.0 menggunakan *Standard Scaler*.

Tabel 3.3 Dataset dinormalisasi menjadi *range* -1 sampai 1

x	Date	Demand	Normalisasi
0	2013-01	9460000	0.655292
1	2013-02	8216000	-0.422145
2	2013-03	8025000	-0.555641
3	2013-04	7902000	0.112813
4	2013-05	8041000	-0.661212
5	2013-06	8324000	-0.359749
6	2013-07	7578000	-1.000000
7	2013-08	8732000	0.021811
8	2013-09	7590000	-0.924178
9	2013-10	9626000	0.413092
10	2013-11	10450000	1.000000
11	2013-12	8748000	0.070382

3.4.3 Genetic Algorithm

Tahap *genetic algorithm* diawali dengan memasukkan data lalu mencari *weight* untuk setiap masukannya. *Weight* akan dihasilkan dari angka acak untuk setiap gen. Selanjutnya akan ditentukan jumlah kromosom yang akan membentuk suatu populasi. *Dataset* yang dipakai

sejumlah 12 data dengan periode *demand* Januari 2013 sampai Desember 2013.

Tabel 3.4 Dataset setelah normalisasi

x	Date	Demand
0	2013-01	0.655292
1	2013-02	-0.422145
2	2013-03	-0.555641
3	2013-04	0.112813
4	2013-05	-0.661212
5	2013-06	-0.359749
6	2013-07	-1.000000
7	2013-08	0.021811
8	2013-09	-0.924178
9	2013-10	0.413092
10	2013-11	1.000000
11	2013-12	0.070382

Data masukan yang dipakai akan merujuk pada *dataset* dengan data t-1, t-2, dan t-3 untuk prediksi data t.

Tabel 3.5 Data masukan

x1	x2	x3	y
0.655292	0.222145	0.155641	0.112813
0.222145	0.155641	0.112813	0.161212
0.155641	0.112813	0.161212	0.259749
0.112813	0.161212	0.259749	0.000000
0.161212	0.259749	0.000000	0.401811
0.259749	0.000000	0.401811	0.004178
0.000000	0.401811	0.004178	0.713092
0.401811	0.004178	0.713092	1.000000
0.004178	0.713092	1.000000	0.407382

Proses *genetic algorithm* dijalankan dengan menentukan jumlah populasi dan membuat nilai acak untuk populasi tersebut pada setiap gen (nilai bobot/*weight* pada setiap gen). Proses ini menggunakan *random generator* dari *library numpy*, yaitu *numpy.random* untuk 3 populasi dan setiap populasi memiliki 3 gen. Parameter yang dipakai dalam sub-bab ini hanya digunakan dalam perhitungan manual sebagai contoh perhitungan.

Tabel 3.6 Matriks populasi

p	0	1	2
0	0.94457713	0.67586074	0.66515298
1	0.26406061	0.10357071	0.52602485
2	0.79624431	0.78838770	0.61736967

Matriks masukan di atas dikalikan dengan nilai matriks populasi pada Tabel 3.6 seperti perhitungan Algoritme 3.1 di bawah ini.

Algoritme 3.1 Perhitungan nilai *fitness*.

$$\begin{aligned}
 f_0 &= p_{0,0}x_0 + p_{0,1}x_1 + p_{0,2}x_2 \\
 &= 0.94457713 * 0.655292 + 0.67586074 * 0.222145 + 0.66515298 * 0.155641 \\
 &= 0.872638
 \end{aligned}$$

$$\begin{aligned}
 f_1 &= p_{1,0}x_0 + p_{1,1}x_1 + p_{1,2}x_2 \\
 &= 0.26406061 * 0.655292 + 0.10357071 * 0.222145 + 0.52602485 * 0.155641 \\
 &= 0.27791555
 \end{aligned}$$

$$\begin{aligned}
 f_2 &= p_{2,0}x_0 + p_{2,1}x_1 + p_{2,2}x_2 \\
 &= 0.79624431 * 0.655292 + 0.78838770 * 0.222145 + 0.61736967 * 0.155641 \\
 &= 0.79299694
 \end{aligned}$$

Perhitungan matriks di atas menghasilkan nilai *fitness* seperti pada Tabel 3.7

Tabel 3.7 Matriks fitness value

f0	f1	f2
0.87263800	0.27791555	0.79299694

Setelah mendapatkan nilai *fitness*, akan dijalankan proses seleksi untuk memilih *parent* dengan memilih nilai *fitness* tertinggi dari satu generasi yang sudah ada. Jumlah *parent* yang dipakai disini adalah 2. Dari Tabel 3.7 akan diambil 2 lokasi *array* terbaik karena jumlah *parent* yang ditetapkan pada kasus ini 2, yaitu *array* ke-0 dan ke-2. Nilai *parent* disimpan menurut hasil populasi terakhir dengan nilai *fitness* terbaik, yaitu $p_{0,x}$ dan $p_{2,x}$. Hasil seleksi *parent* digambarkan pada Tabel 3.8 berikut.

Tabel 3.8 Matriks *selected parent*

p0	p1	p2
0.94457713	0.67586074	0.66515298
0.79624431	0.78838770	0.61736967

Dari nilai *parent* yang sudah ada, dilanjutkan proses *crossover* untuk menghasilkan keturunan dari suatu generasi. Dari nilai *parent* di Tabel 3.8 disilangkan memakai metode yang umum, yaitu dibagi dua dan disilangkan keduanya. Tabel 3.9 adalah hasil keluaran dari nilai *crossover*.

Tabel 3.9 Matriks *crossover*

c0	c1	c2
0.94457713	0.78838770	0.61736967

Proses selanjutnya untuk menghasilkan keturunan adalah mutasi. Proses mutasi menggunakan hasil dari proses *crossover* mengambil acak satu gen yang akan diubah dengan menambahkan angka acak. Tabel 3.10 adalah hasil proses mutasi dari nilai *crossover* sebelumnya.

Tabel 3.10 Matriks *mutation*

m0	m1	m2
0.94457713	0.78838770	0.28648684

Dari hasil mutasi di atas, akan dihasilkan nilai *fitness* terbaik dalam generasi tersebut. Nilai *fitness* tersebut akan dipakai untuk memroses nilai pada generasi selanjutnya untuk menghasilkan nilai *fitness* yang lebih baik lagi. Namun, karena *genetic algorithm* ini memakai bilangan acak (*random*) tidak ada jaminan hasilnya akan selalu lebih baik. Maka dari itu, nilai *parent* dari solusi terbaik sebelumnya akan disimpan dan digunakan kembali jika hasil populasi baru tidak lebih baik dari sebelumnya. Pada kasus ini dilakukan iterasi 3 generasi dan akan terbentuk matriks *weight* seperti pada Tabel 3.11 untuk digunakan pada tahap selanjutnya.

Tabel 3.11 Matriks solusi terbaik

w0	w1	w2

0.94457713	0.78838770	1.17053244
------------	------------	------------

Tahap *genetic algorithm* ini berguna untuk menentukan *weight* terbaik yang akan dipakai untuk perhitungan nilai keluaran. Perhitungan di atas mencakup perhitungan satu generasi, dan iterasi akan berlanjut sesuai dengan jumlah generasi yang diinginkan.

3.4.4 Feed Forward Neural Network

Tahap *feed forward neural network* ini diawali dengan inisialisasi *network* berupa jumlah *neuron* pada *input layer*, *hidden layer* dan *output layer*. Pada perhitungan ini *network* diinisialisasikan 1 layer masukan dengan 3 neuron. Setelah inisialisasi *network*, setiap *neuron* akan dimasukkan kedalam *transfer function* menggunakan persamaan 2.2 dengan asumsi bias adalah 0.

Algoritme 3.2 Perhitungan nilai *input* dengan bobot yang disebut *transfer function*.

$$\begin{aligned}v_k &= b + (w_{0,0}x_0 + w_{0,1}x_1 + w_{0,2}x_2) \\&= 0 + (0.94457713 * 0.655292 + 0.78838770 * 0.222145 \\&\quad + 1.17053244 * 0.155641) \\&= 0.9762930618\end{aligned}$$

Weight pada Algoritme 3.2 ini tidak lagi menggunakan angka acak karena sudah menggunakan *weight* dari *genetic algorithm*. Setelah *neuron* dimasukkan ke dalam *transfer function*, *neuron* diaktifkan menggunakan sigmoid biner seperti pada persamaan 2.4 yang diimplementasikan pada Algoritme 3.3.

Algoritme 3.3 Perhitungan nilai *activation function*.

$$\begin{aligned}y &= \varphi(0.9762930618) \\&= \frac{1}{1 + e^{-x}} \\&= \frac{1}{1 + e^{-0.9762930618}} \\&= \frac{1}{1 + 0.376705} \\&= 0.726372026\end{aligned}$$

Hasil keluaran dari *input layer* di atas akan menjadi nilai masukan pada *hidden layer* dan terus berulang sampai *output layer*.

3.4.5 Back Propagation

Tahap *back propagation* akan mengulas balik hasil dari masing-masing *neuron* agar mencapai tingkat *error* yang diinginkan. Nilai *error* dihitung menggunakan persamaan 2.6 pada Algoritme 3.4 berikut.

Algoritme 3.4 Perhitungan nilai *error* pada *output layer*.

$$\begin{aligned} E &= \frac{1}{2}(0.112813 - 0.726372026)^2 \\ &= \frac{0.3764546784}{2} \\ &= 0.1882273392 \end{aligned}$$

Nilai *error* dihitung dan dimasukkan kedalam setiap lapisan dengan urutan terbalik (dari *output layer* ke *hidden layer* dan seterusnya). Proses selanjutnya adalah memperbaharui nilai *weight* dengan persamaan 2.7. Untuk mendapatkan nilai *gradient descent* $\frac{\partial \text{Error}}{\partial w_x}$, *back propagation* mulai dari belakang yaitu $\frac{\partial \text{Error}}{\partial y}$ dengan Algoritme 3.5.

Algoritme 3.5 Perhitungan turunan nilai *error* pada *activation function*.

$$\begin{aligned} E &= \frac{1}{2}(t - y)^2 \\ \frac{\partial \text{Error}}{\partial y_0} &= 2 * \frac{1}{2}(t - y)^{2-1} \\ &= -(t - y) \end{aligned}$$

Variabel w_0 tidak langsung ditemukan dalam perhitungan variabel y_0 , melainkan perlu mundur lagi ke variabel v_0 dengan Algoritme 3.6.

Algoritme 3.6 Perhitungan turunan nilai *activation function* pada *transfer function*.

$$\begin{aligned}y_0 &= \Phi(v_0) \\ \frac{\partial y_0}{\partial v_0} &= y_0(1 - y_0)\end{aligned}$$

Setelah mendapatkan perhitungan $\frac{\partial y}{\partial v}$, mundur sekali lagi untuk mendapatkan $\frac{\partial v}{\partial w}$ sesuai dengan Algoritme 3.7.

Algoritme 3.7 Perhitungan turunan nilai *transfer function* pada bobot.

$$\begin{aligned}v_0 &= b + (w_{0,0}x_0 + w_{0,1}x_1 + w_{0,2}x_2) \\ \frac{\partial v_0}{\partial w_{0,0}} &= x_0 + 0 + 0 \\ &= x_0\end{aligned}$$

Gabung ketiga perhitungan menjadi $\frac{\partial Error}{\partial w}$ seperti pada perhitungan Algoritme 3.8 di bawah ini.

Algoritme 3.8 Perhitungan untuk menentukan nilai turunan pada *error* terhadap bobot.

$$\begin{aligned}\frac{\partial Error}{\partial w} &= \left(\frac{\partial Error}{\partial y_0}\right) \cdot \left(\frac{\partial y_0}{\partial v_0}\right) \cdot \left(\frac{\partial v_0}{\partial w_{0,0}}\right) \\ &= -(t - y_0) \cdot y_0(1 - y_0) \cdot x_0 \\ &= -(0.112813 - 0.726372026) \cdot 0.726372026(1 - 0.726372026) \\ &\quad \cdot 0.655292 \\ &= 0.07991178295\end{aligned}$$

Setelah itu, hitung bobot baru dengan *learning rate* diasumsikan adalah 0,5 dalam persamaan berikut.

Algoritme 3.9 Mengubah nilai bobot lama dengan yang baru.

$$\begin{aligned}w_{0,0new} &= w_{0,0old} - \alpha \frac{\partial Error}{\partial w_{0,0}} \\&= 0.94457713 - (0.5)(0.07991178295) \\&= 0.9046212385\end{aligned}$$

Setelah mendapatkan nilai *weight* baru, nilai pada setiap *neuron* dihitung kembali seperti proses yang sudah dijelaskan. Proses keseluruhan ini akan selesai jika tingkat *error* yang dihasilkan sudah sesuai dengan target atau nilai *epoch* sudah mencapai batas. Hasil perhitungan *error* menggunakan bobot baru yang sudah dihitung pada Algoritme 3.10 di bawah ini.

Algoritme 3.10 Perhitungan nilai *error* setelah menggunakan bobot yang baru.

$$\begin{aligned}E &= \frac{1}{2}(0.112813 - 0.72023433)^2 \\&= \frac{0.36896068}{2} \\&= 0.18448034\end{aligned}$$

Terlihat pada perhitungan di atas nilai *error* semakin kecil karena bobot sudah diperbarui oleh *back propagation*. Semua perhitungan di atas termasuk dalam satu iterasi (*epoch*), untuk *epoch* berikutnya akan menggunakan perhitungan yang sama, dengan data yang berbeda.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan menjelaskan mengenai proses implementasi dan pengujian terhadap sistem yang telah dibangun berdasarkan penjelasan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Pada lingkungan implementasi, akan dijelaskan mengenai perangkat yang digunakan dalam proses pembangunan sistem baik dari perangkat keras maupun perangkat lunak yang digunakan.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi dari perangkat keras yang digunakan dalam pembangunan aplikasi adalah sebagai berikut:

1. *Laptop* Macbook Pro.
2. *Processor* Intel Core i5 CPU @ 2.5GHz.
3. *Storage* kapasitas 1TB (500GB SSD + 500GB HDD).
4. *Memory* 10GB RAM.

4.1.2 Lingkungan Perangkat Lunak

Spesifikasi dari perangkat lunak yang digunakan dalam pembangunan aplikasi adalah sebagai berikut:

1. Sistem Operasi macOS Catalina 10.15.7.
2. Safari 14.1.
3. Python 3.8.5 (Jupyter Notebook).
4. Google Colab.

4.2 Implementasi Perangkat Lunak

Pada implementasi perangkat lunak dilakukan menurut analisis yang sudah disusun pada BAB III. Bagian ini akan menjelaskan mengenai kelas dan metode yang akan digunakan pada penelitian.

4.2.1 Class Preprocessing

Class Preprocessing berisikan *method* yang digunakan pada saat tahap *preprocessing* data *time-series* agar bisa digunakan untuk tahap selanjutnya. Pada *class preprocessing* ini *method-method* yaitu *read_csv*, *DataFrame*, *loc*, *isnull*, *dropna*, dan *normalization*. *read_csv*,

DataFrame, *loc*, *isnull*, *dropna* ini berasal dari *library Pandas* yang berfungsi untuk membaca *dataset*, menggolongkannya berdasarkan periode waktu yang dipilih, dan menghilangkan baris yang tidak memiliki nilai atau kosong agar data bersih untuk dilakukan proses selanjutnya. Sedangkan *method normalization* berisi *StandardScaler* yang berasal dari *library Sklearn* untuk mengubah nilai menjadi dalam *range* tertentu. Pada penelitian ini normalisasi dilakukan dalam *range* -1 sampai 1.

4.2.2 Class GeneticAlgorithm

Class GeneticAlgorithm berisikan *method* yang digunakan untuk mendapatkan hasil bobot terbaik yang akan dipakai untuk *Artificial Neural Network*. *Method* yang ada pada *Class GeneticAlgorithm* dapat dilihat pada Tabel 4.2.

Algoritme 4.1 Attribute pada class GeneticAlgorithm

Attribute:					
Float	dataset	Int	arr_index	Float	arr_value
Int	num_weights	Float	fitness	Float	parents
Int	max_fitness_idx	Int	crossover_point	Float	offspring
Int	pop_size	Float	new_population	Float	best_solution
Int	num_generations	Float	random_value		

Algoritme 4.2 Daftar Method pada class *GeneticAlgorithm*

No	Method	Masukan	Luaran	Keterangan
1	get_dataset_value	dataset : float[]	float[]	Digunakan untuk menginisialisasi populasi dalam sebuah data.
2	parent	parents mating : int[]	-	Digunakan untuk menentukan jumlah <i>parent</i> yang akan diambil dari populasi.
3	cal_pop_fitness	input : float[], new population : float[]	float[]	Digunakan untuk mencari nilai <i>fitness</i> pada populasi tersebut.
4	select_mating_plot	population : float[], fitness : float[], num_parents : int	float[]	Digunakan untuk <i>parent</i> yang terbaik pada suatu populasi berdasarkan nilai <i>fitness</i> .
5	crossover	parents : float[], offspring_size : int[]	float[]	Digunakan untuk mencari turunan terbaik pada suatu populasi berdasarkan nilai <i>parent</i> yang dikawin silang.
6	mutation	offspring crossover : float[]	float[]	Digunakan untuk memotong / mutasi gen dengan acak.

Pada *class* ini semua *method* menggunakan perhitungan secara manual tanpa proses atau perhitungan dari *library* seperti yang ada pada *class* lain. Dalam *class* ini hanya menggunakan atribut *random* yang berasal dari *library NumPy* untuk inisialisasi nilai populasi dan menambahkan nilai acak pada *method mutation*.

4.2.3 Class *ArtificialNeuralNetwork*

Class ArtificialNeuralNetwork berisikan *method* yang digunakan dari proses *neural network* yang akan memroses data sampai pada tahap keluaran. *Method* yang ada pada *class ArtificialNeuralNetwork* dapat dilihat pada Tabel 4.4.

Algoritme 4.3 Attribute pada class ArtificialNeuralNetwork

Attribute:					
Float	dataX	Float	dataY	Float	weights
Int	train_size	Int	test_size	Float	trainX
Float	trainY	Float	testX	Float	testY
Float	learning_rate	Int	epoch	Int	batch_size

Algoritme 4.4 Daftar Method pada class ArtificialNeuralNetwork

No	Method	Masukan	Luaran	Keterangan
1	create_dataset	dataset : float[], look_back : int	float[], float[]	Digunakan untuk mengubah dataset menjadi data latih sesuai bentuk pada Tabel 3.5.
2	initialize_weight	best_solution : float[]	float[]	Digunakan untuk memasukan custom weight dari keluaran genetic algorithm ke dalam neural network.
3	neural_network	dataset : float[], best_solution : float[]	-	Digunakan untuk menjalankan proses utama ANN dan menghasilkan error.

Pada class ini method *create_dataset* digunakan untuk mengubah *dataset* menjadi bentuk data *input* seperti pada Tabel 3.5. Method *initialize_weight* digunakan untuk mengambil hasil dari GA yang akan menjadi bobot dalam ANN. Method yang terakhir adalah method *neural_network* yang menggunakan bantuan sepenuhnya dari library *Keras* untuk proses utama *neural network*. *Model.add* berfungsi untuk inisialisasi dimensi *input*, *hidden neuron*, *activation function*, dan jumlah *layer* yang akan dipakai. *Model.compile* berfungsi untuk inisialisasi *metrics* yang akan digunakan, besaran *learning rate*, dan proses *training* dari ANN. Sedangkan *model.fit* digunakan untuk inisialisasi *epoch*, *batch size* dan proses *testing* dari ANN.

4.2.4 Class Measurement

Class Measurement berisikan method yang digunakan untuk mengukur nilai *error* dan menunjukkan plot grafik dari nilai *error* dan perbandingan terhadap model *dataset*. *Class* ini menggunakan *method* dan *attribute* dari library *Pandas* yang disebutkan dalam sub-bab Matplotlib di Bab 2. *Class* ini menjadi visual untuk evaluasi hasil pada sub-bab Pengujian.

4.3 Pengujian

Pada penelitian ini, pengujian yang dilakukan adalah membandingkan algoritme *Neural Network* dan algoritme *Neural Network* (NN) yang digabungkan dengan algoritme *Genetic Algorithm* (GA) untuk meramal *demand* pada suatu produk. Hasil yang diharapkan pada penelitian ini adalah mendapatkan MSE (*error*) yang lebih rendah pada algoritme gabungan NN dan GA.

Pada pengujian dalam penelitian ini akan melibatkan banyak kombinasi *hyperparameter* yang dianggap bisa memengaruhi nilai *error* di hasil akhir. *Parameter* yang digunakan pada NN adalah *learning rate*, *batch size*, dan *epoch*. Sedangkan untuk GA akan menggunakan *population* dan *generation*.

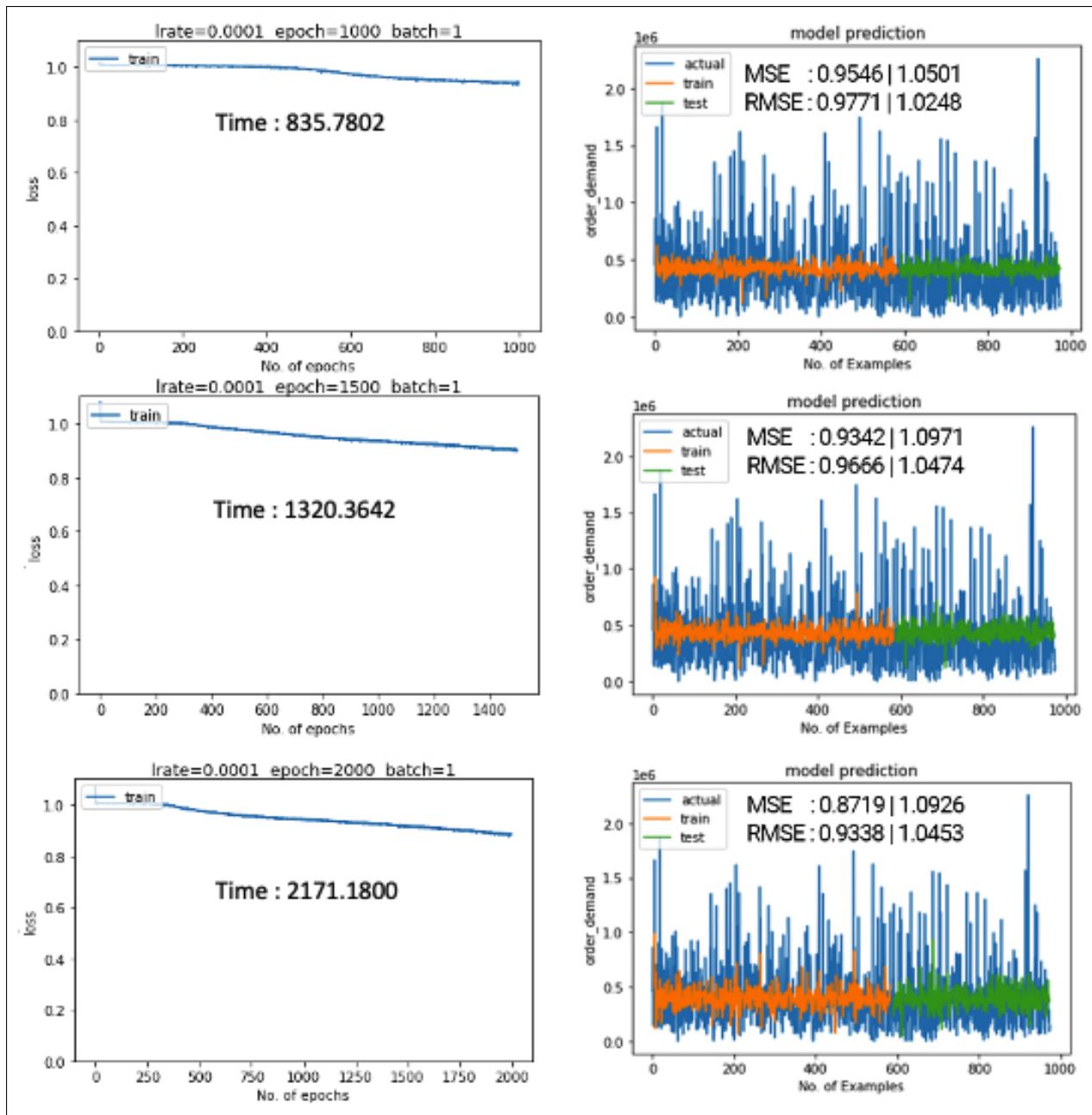
Pengujian pertama akan dilakukan hanya menggunakan algoritme NN dengan kombinasi 3 *parameter* dengan nilai *weight* acak. Pengujian selanjutnya algoritme GA akan menghitung nilai *weight* yang optimal untuk dipakai dalam algoritme NN menggunakan 5 kombinasi *parameter* tersebut.

ANN (27 kombinasi)			
LR	0.0001	0.001	0.01
Epoch	1000	1500	2000
Batch Size	1	2	4
ANN - GA (108 kombinasi)			
LR	0.0001	0.001	0.01
Epoch	1000	1500	2000
Batch Size	1	2	4
Population	5	10	
Generation	5	10	

Gambar 4.1 Bagan Skenario Pengujian

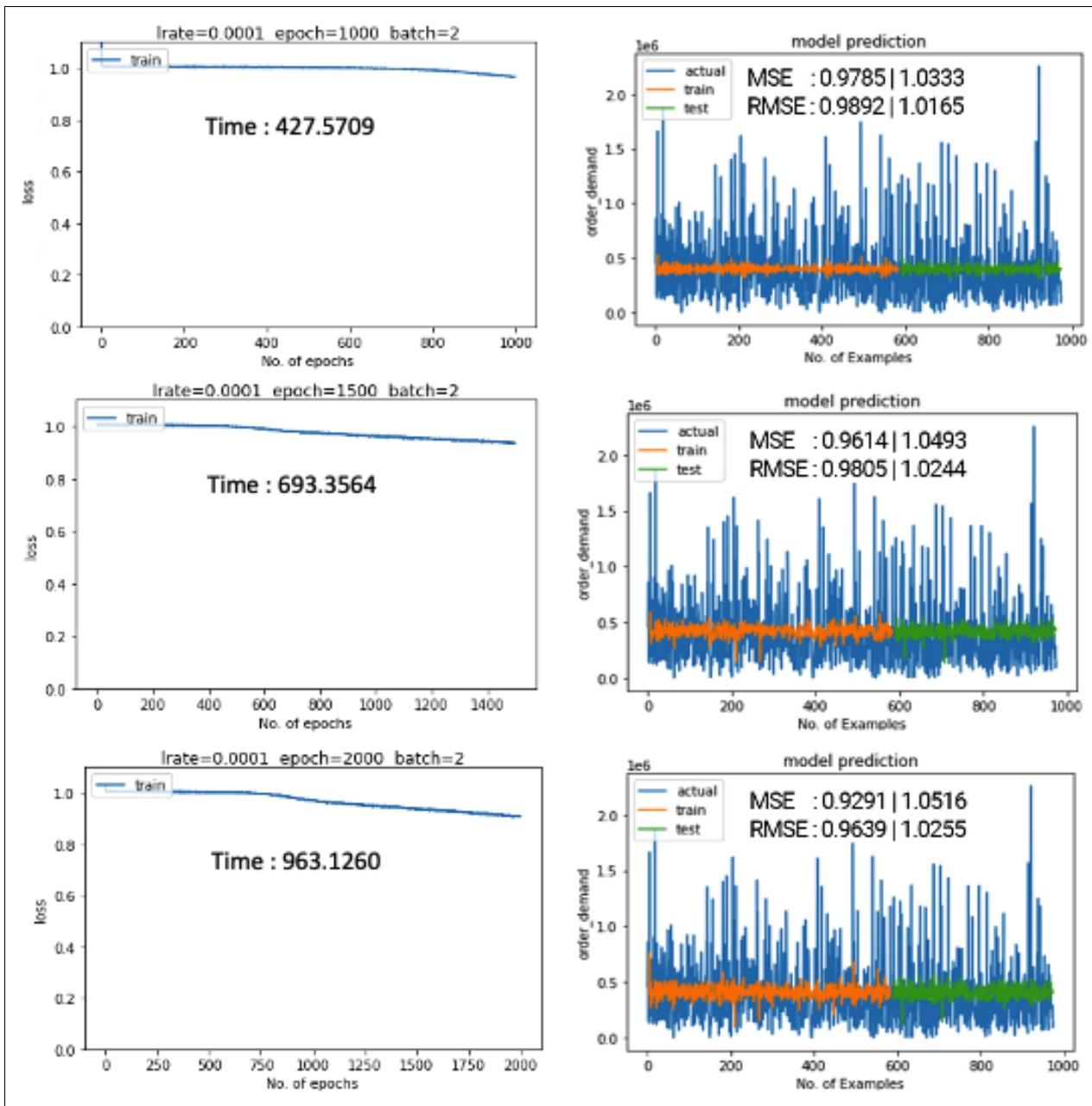
4.3.1 Pengujian *Artificial Neural Network* (ANN)

Pada bagian ini, akan dijelaskan perbandingan dari pembuatan model ANN terhadap data latih yang belum dikombinasikan dengan *Genetic Algorithm*. Di bawah ini merupakan grafik pergerakan *error* dan prediksi dari *dataset* yang sudah ditentukan.



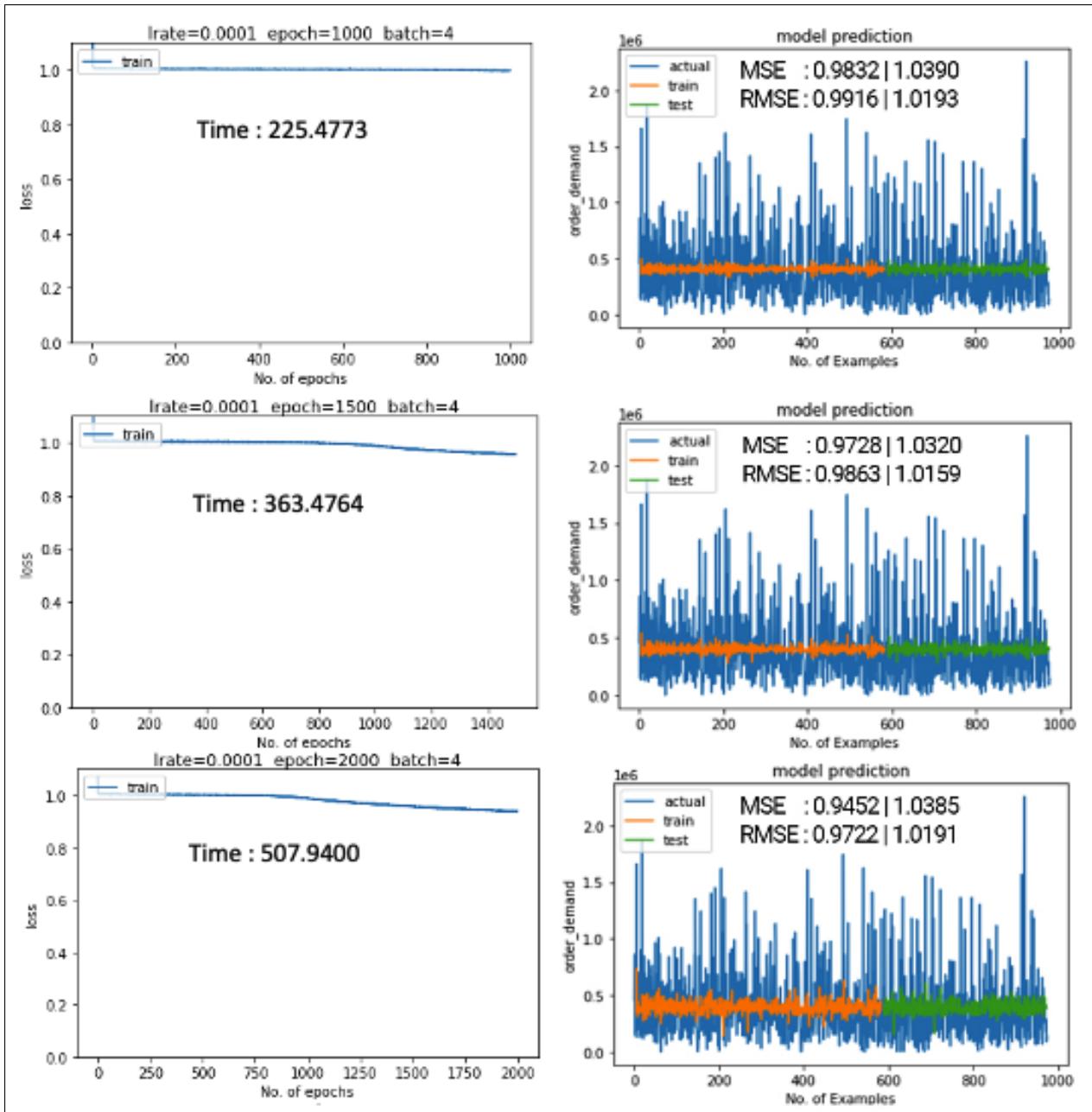
Gambar 4.2 Grafik ANN menggunakan $learning\ rate = 0.0001$; $batch\ size = 1$

Terlihat pada Gambar 4.2 $epoch$ 1000 belum ada penurunan yang signifikan sampai pengujian selesai di 835 detik. Saat menggunakan $epoch$ 1500 juga $error$ yang dihasilkan belum terlihat penurunan yang berarti (1.08%) dengan kenaikan waktu yang cukup jauh sekitar 58%. Hal yang sama terjadi pada $epoch$ 2000, nilai $error$ berkurang 4.6% dan penambahan waktu yang sangat besar (160%). Untuk pembacaannya, grafik kiri adalah proses penurunan nilai MSE dari $epoch$ pertama sampai terakhir. Lalu grafik kanan adalah *plotting* data aktual (biru), data *training* (oranye), dan juga data *testing* (hijau) dengan sumbu x adalah penandaan waktu ke sekian yang diuji dan sumbu y adalah jumlah *order demand* pada skala sebenarnya.



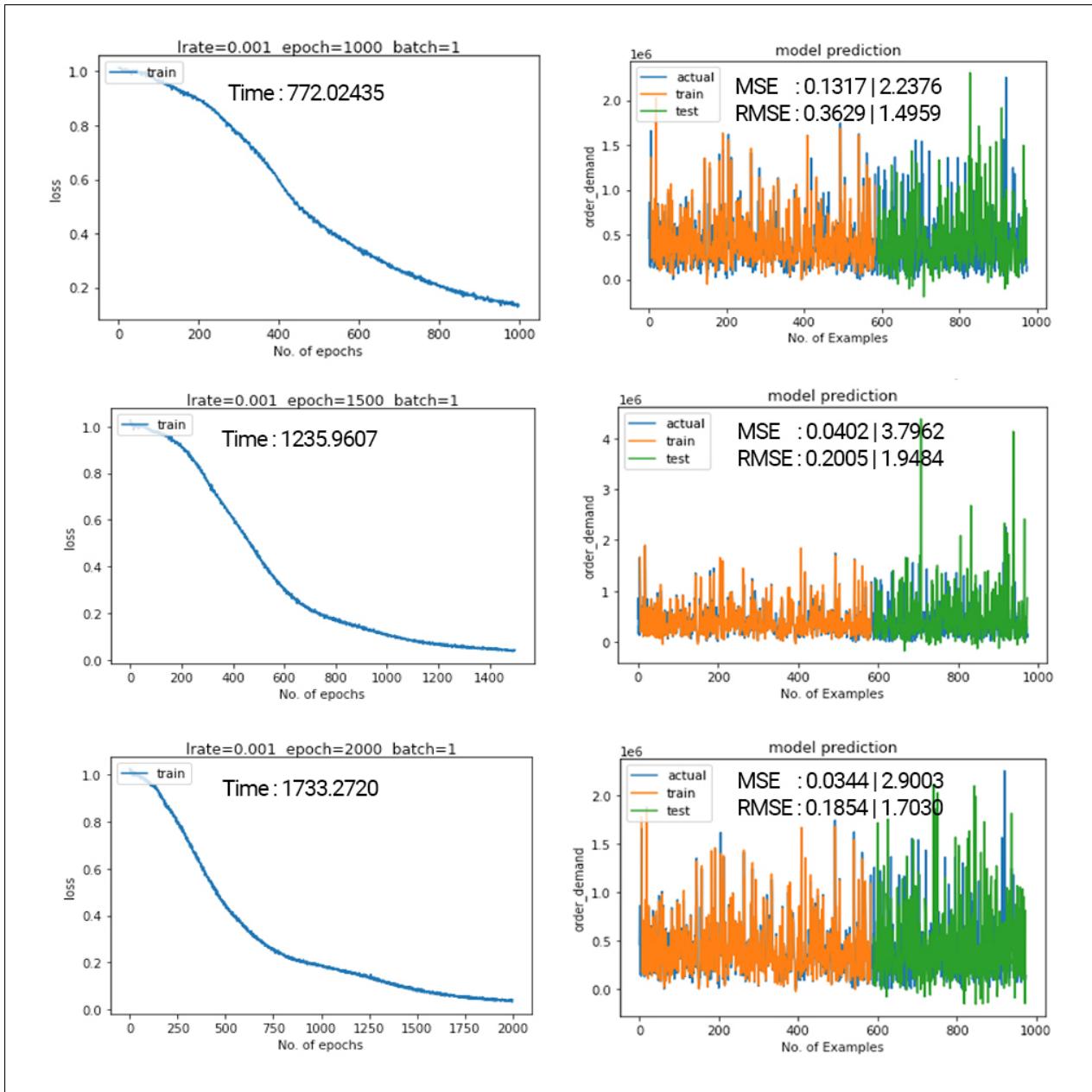
Gambar 4.3 Grafik ANN menggunakan *learning rate* = 0.0001; *batch size* = 2

Untuk pengujian pada Gambar 4.3, *epoch* 1000 menghasilkan waktu yang jauh lebih singkat dibandingkan dengan pengujian 4.2. Waktu yang dihasilkan menurun sampai 48% namun dengan penambahan *error* sebanyak 1.23% merujuk pada skenario yang sama dengan *batch size* 1. Pada *epoch* 1500 waktu pengujian kembali meningkat dan pengurangan *error* yang sangat sedikit yaitu 0.88% dari pengujian sebelumnya, Hal yang sama terjadi pada *epoch* 2000 dengan peningkatan waktu pengujian 125% dan pengurangan *error* 2.5%.



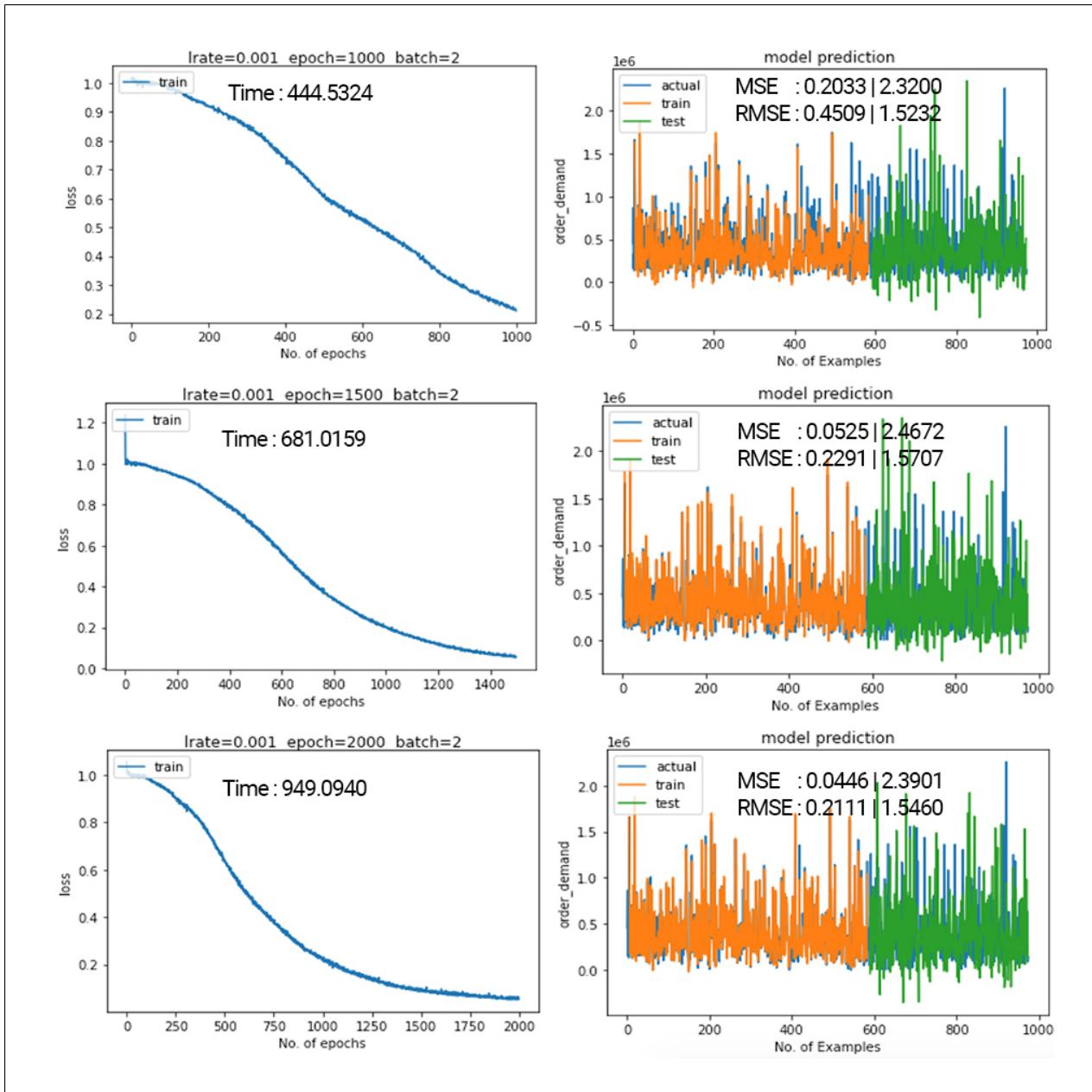
Gambar 4.4 Grafik ANN menggunakan $learning rate = 0.0001$; $batch size = 4$

Terlihat pada Gambar 4.4 pengujian $epoch$ 1000 nilai $error$ seperti tidak ada penurunan sama sekali karena hasil yang didapat masih sangat tinggi. Pada pengujian $epoch$ 1500 mulai terlihat ada penurunan nilai $error$ di kisaran $epoch$ 900 dan menurun sampai $epoch$ 1500. Pada pengujian dengan $epoch$ 2000 mulai terlihat penurunan nilai $error$ sejak $epoch$ 800 dengan catatan waktu 507 detik. Hal ini berarti mengeluarkan hasil yang serupa dengan skenario pengujian pertama dengan selisih $error$ 0.5% namun waktu pengujian berkurang hingga 39% dengan penambahan $epoch$ dan $batch size$.



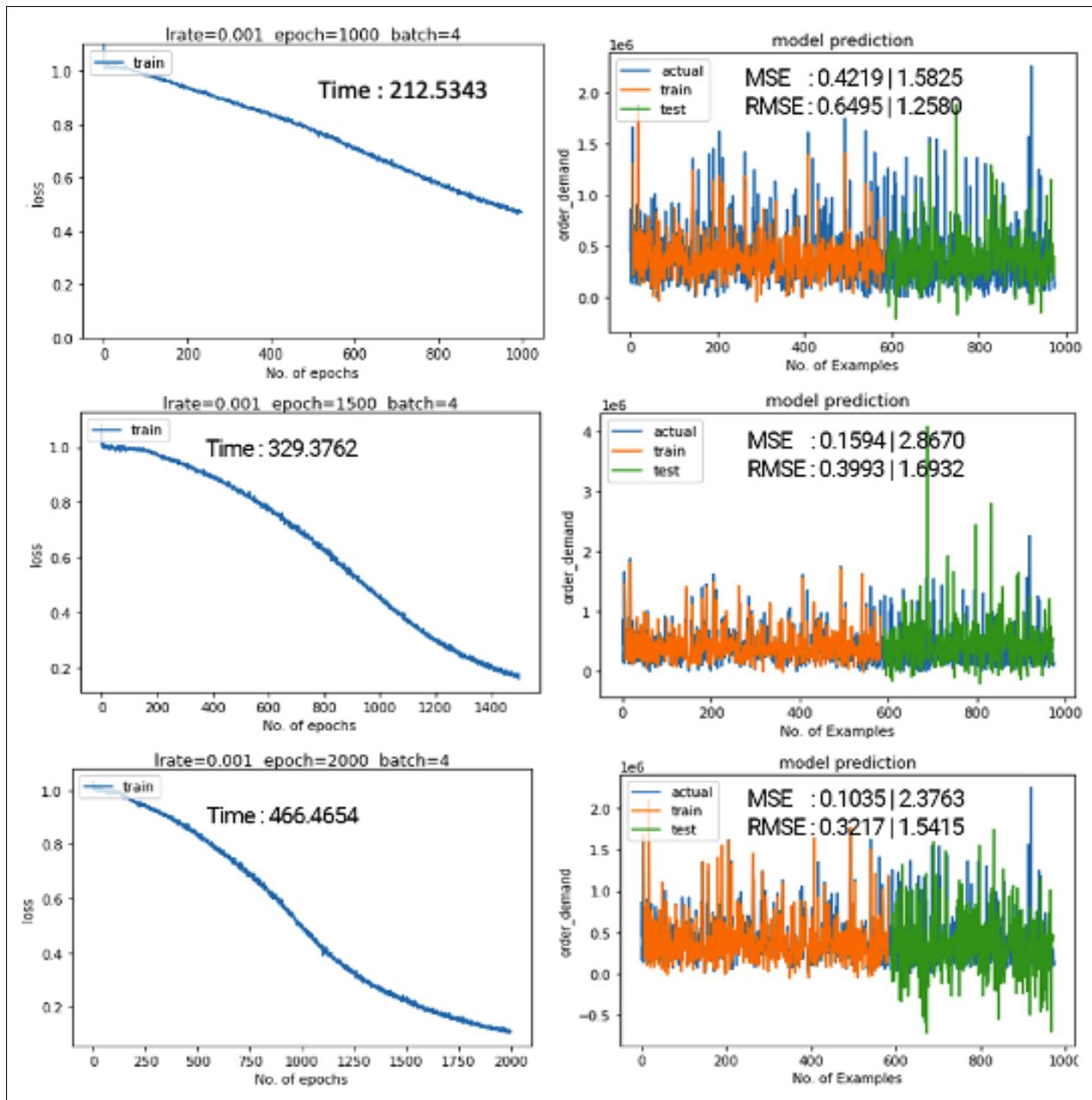
Gambar 4.5 Grafik ANN menggunakan *learning rate* = 0.001; *batch size* = 1

Pada Gambar 4.5 grafik *error* terlihat menunjukkan penurunan yang signifikan dibanding skenario sebelumnya dan nilai *error* sudah mendekati 1. Pada *epoch* 1500 dan 2000, MSE dan RMSE *training* sudah menyentuh angka yang cukup kecil dan terlihat pada grafik prediksinya sudah hampir sesuai dengan model yang didapatkan dari *dataset*. Hasil ini sangat jauh berbeda dengan skenario sebelumnya yang menggunakan *learning rate* 0.0001. Namun pada pengujian ini juga terlihat hasil MSE dan RMSE *testing* yang jauh lebih tinggi daripada *training*, hal ini disebut *overfit*.



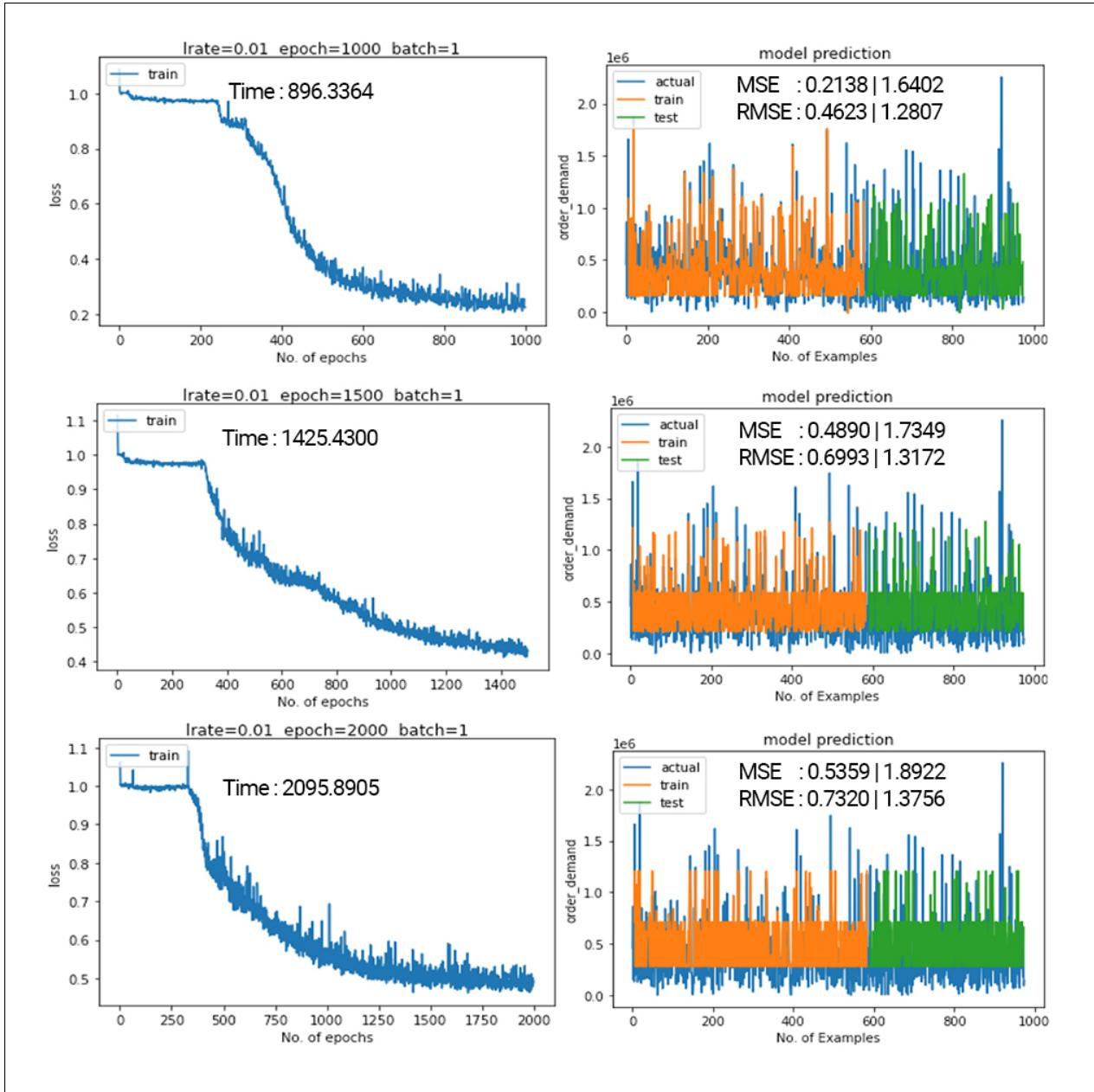
Gambar 4.6 Grafik ANN menggunakan *learning rate* = 0.001; *batch size* = 2

Pada Gambar 4.6 ini waktu yang diperlukan ANN berkurang 55% dari pengujian yang hanya menggunakan 1 *batch*. Namun dari hasil waktu yang lebih cepat itu memberikan hasil *error* yang sedikit lebih tinggi, sama seperti pada skenario sebelumnya dengan perbedaan *batch size* dalam pengujiannya.



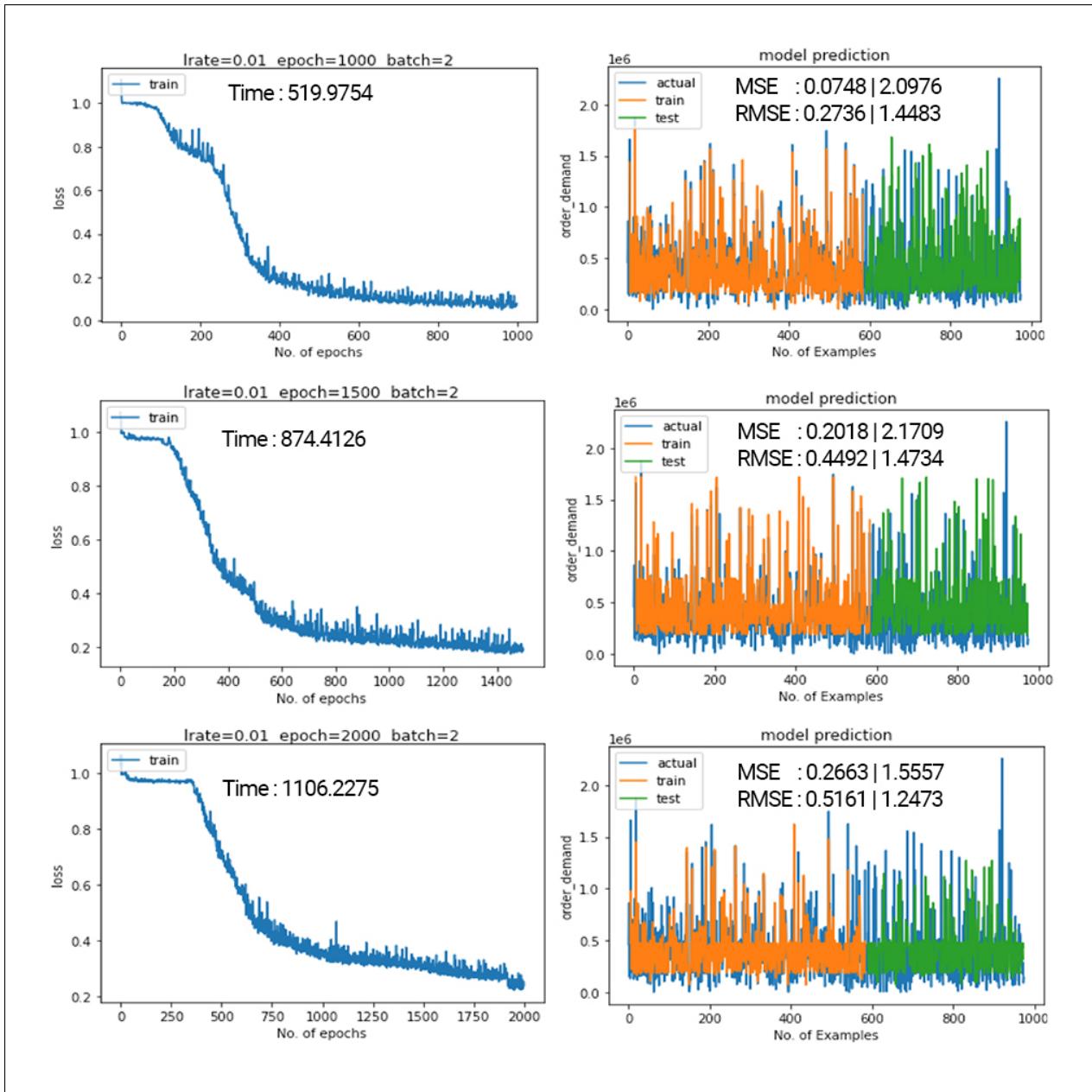
Gambar 4.7 Grafik ANN menggunakan *learning rate* = 0.001; *batch size* = 4

Terlihat pada Gambar 4.7 di *epoch* 1000 grafik *error* masih menunjukkan penurunan dan belum mendekati nol. Walaupun pengujian ini mendapat waktu yang paling cepat, tapi tingkat *error* nya masih cukup tinggi dan terlihat pada grafik prediksi masih belum sepenuhnya menutupi model dari *dataset* dengan tepat. Dan pada *epoch* 2000, tingkat *errornya* masih belum bisa mengalahkan pengujian sebelumnya. Dapat disimpulkan bahwa pengujian dengan parameter *learning rate* = 0.001 jauh lebih baik dari pengujian sebelumnya yang menggunakan *learning rate* = 0.0001 tapi semakin besar *batch size* belum tentu juga hasilnya lebih baik.



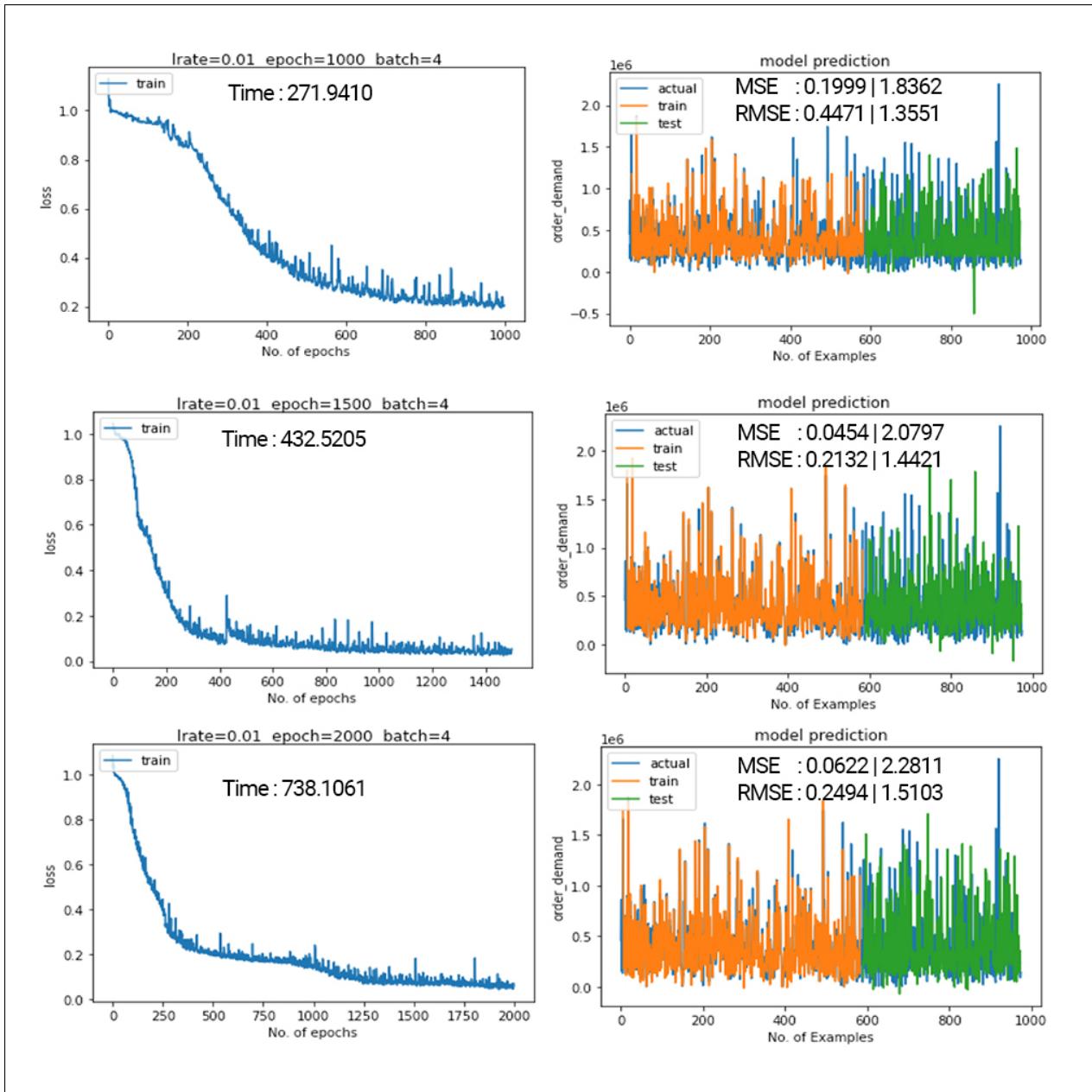
Gambar 4.8 Grafik ANN menggunakan $learning\ rate = 0.01$; $batch\ size = 1$

Pada Gambar 4.8 grafik *error* terlihat membentuk garis horizontal dan pada pengujian $epoch = 1000$ grafik *error* menurun di rentang $epoch$ 200 sampai 400. Hal ini disebabkan nilai *learning rate* yang terlalu besar jadi penurunan *error* di awal tidak berpengaruh sebelum menemukan pola yang sesuai dengan model *dataset*. Pada pengujian ini, semakin banyak $epoch$ nilai *error* yang didapat juga semakin besar karena parameter yang kurang optimal. Pada grafik *error* juga terlihat ketidakstabilan garis yang terbentuk akibat *learning rate* yang terlalu besar.



Gambar 4.9 Grafik ANN menggunakan $learning\ rate = 0.01$; $batch\ size = 2$

Pada Gambar 4.9 yang menggunakan $batch\ size = 2$ pengujian mendapatkan hasil yang lebih cepat dan grafik penurunan $error$ pada $epoch = 1000$ cukup berpengaruh di rentang $epoch$ 100 sampai 200. Penurunan ini lebih cepat dari pengujian sebelumnya yang menggunakan $batch\ size = 1$ pada parameter $epoch$ yang sama. Namun untuk pengujian kali ini perbandingan $epoch$ dan $error$ masih berbanding lurus karena semakin banyak $epoch$, semakin besar juga $error$ nya.



Gambar 4.10 Grafik ANN menggunakan $learning\ rate = 0.01$; $batch\ size = 4$

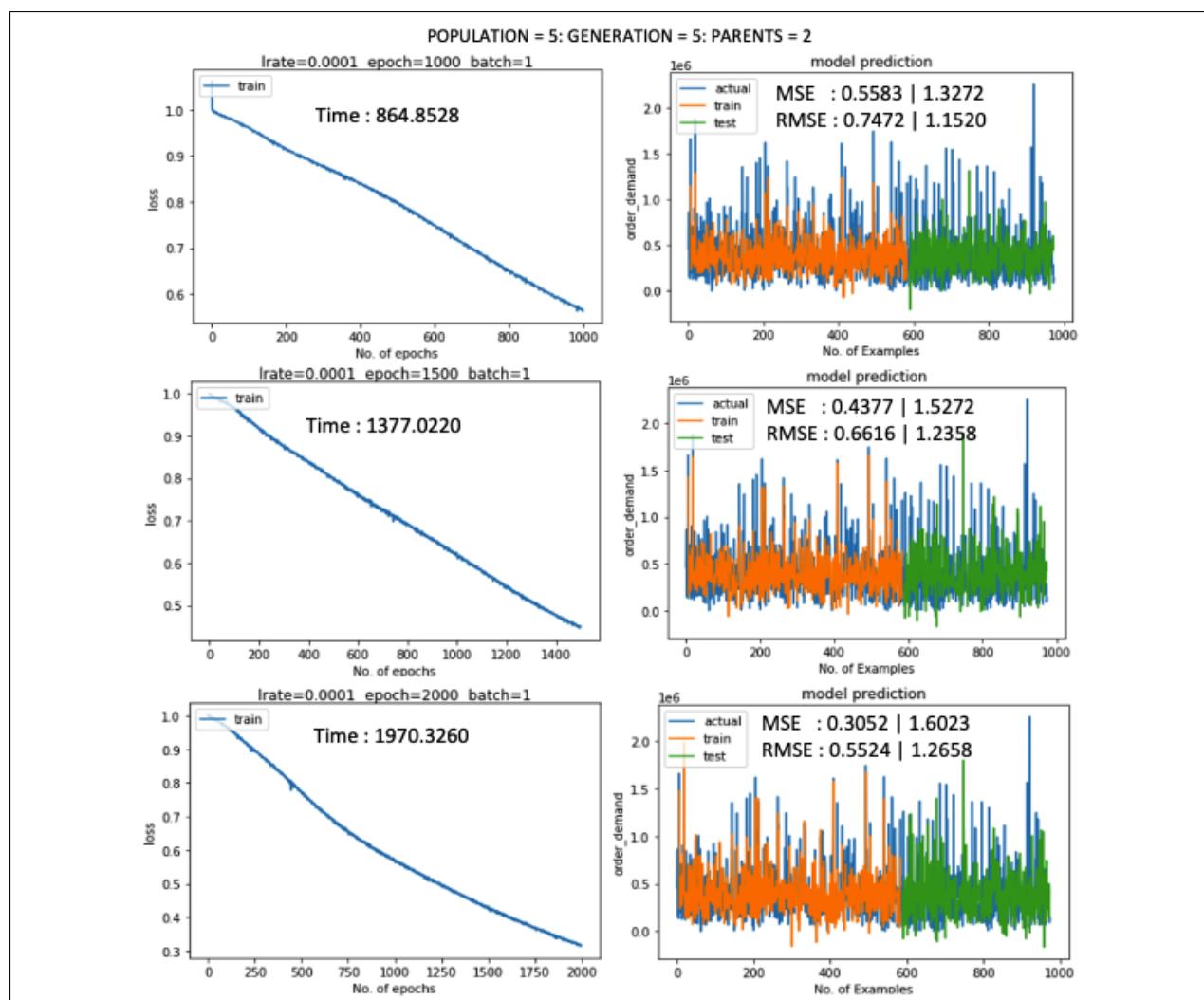
Pada Gambar 4.10 waktu yang diperlukan untuk menjalankan pengujian lebih singkat lagi karena menggunakan $batch\ size = 4$. Pada pengujian kali ini grafik penurunan $error$ terlihat lebih stabil walaupun masih ada beberapa lonjakan $error$. Nilai MSE dan RMSE sudah cukup baik dan $epoch$ 1500 menjadi parameter yang paling baik tingkat $errornya$ dalam pengujian $learning\ rate = 0.01$.

Dalam semua pengujian ANN yang sudah dibahas, hasil $error$ yang paling rendah yaitu menggunakan parameter $learning\ rate = 0.001$, $epoch = 2000$, dan $batch\ size = 1$ dengan nilai 0.00344 MSE dan 0.1854 RMSE. Namun hasil tersebut diraih dengan waktu yang sangat lama yaitu 1733.2720 detik (28.9 menit) untuk pengujinya. Dengan menggunakan $learning\ rate =$

0.001 juga menghasilkan grafik *error* yang stabil di setiap kombinasi pengujian. Selain hasil itu, pengujian terakhir yang menggunakan parameter *learning rate* = 0.01, *epoch* = 1500, *batch size* = 4 menghasilkan nilai *error* kedua terkecil dengan hasil 0.0454 MSE dan 0.2132 RMSE dan selesai dengan waktu yang cukup singkat juga yaitu 432.5205 detik (7.2 menit). Setiap parameter mempunyai pengaruh yang berbeda-beda dan tidak mutlak untuk setiap kasus mempunyai hasil yang serupa.

4.3.2 Pengujian Artificial Neural Network dengan *Genetic Algorithm* (ANN-GA)

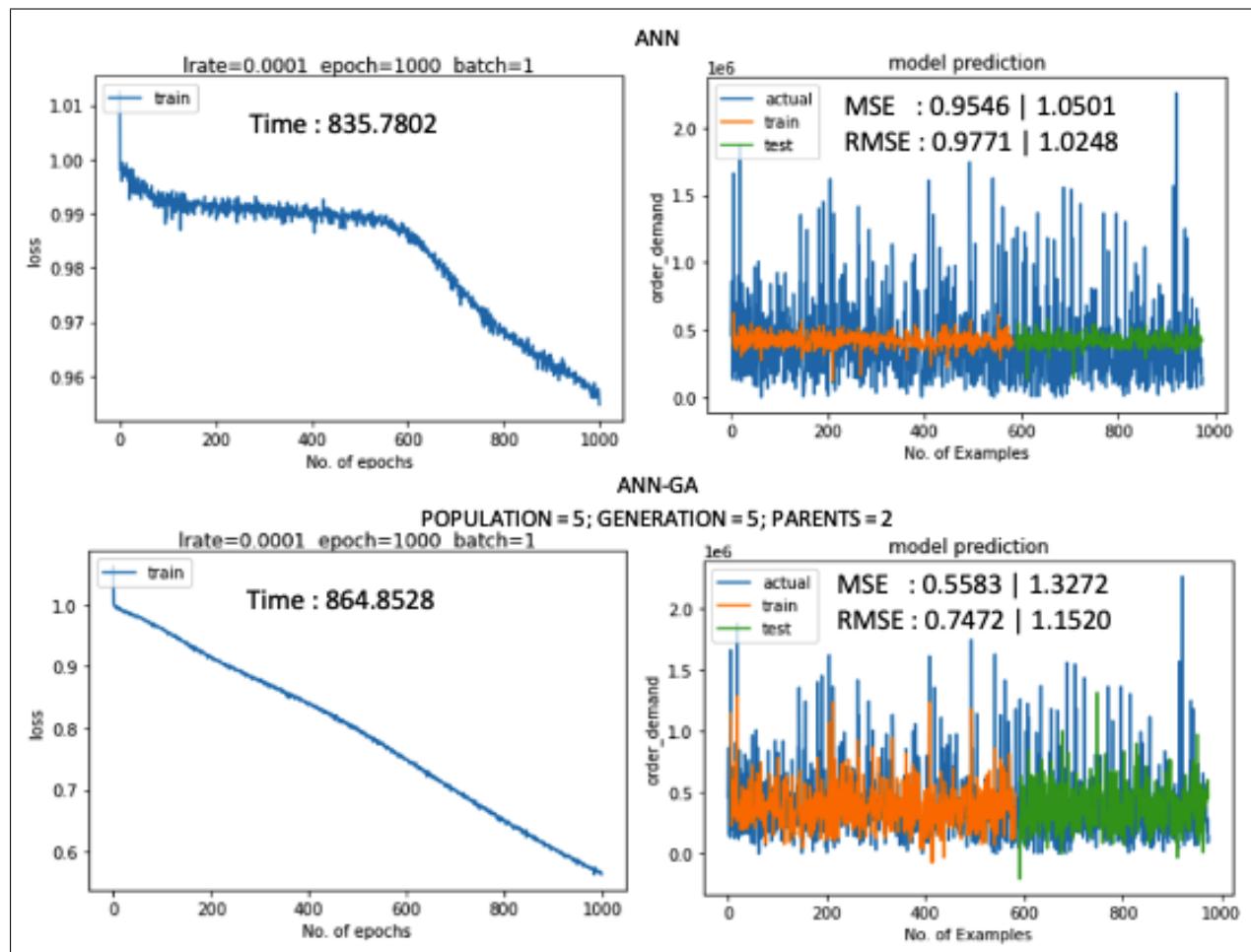
Pada bagian ini, akan dijelaskan perbandingan dari pembuatan model ANN terhadap data latih yang sudah dikombinasikan dengan *Genetic Algorithm*. Di bawah ini merupakan grafik pergerakan *error* dan prediksi dari *dataset* yang sudah ditentukan.



Gambar 4.11 Grafik ANN menggunakan *learning rate* = 0.0001; *batch size* = 1

Pengujian 4.11 di atas menggunakan parameter ANN yang sama pada pengujian ANN pertama, namun bobot yang dimasukkan ke dalam proses ANN sudah melalui algoritme genetika yang

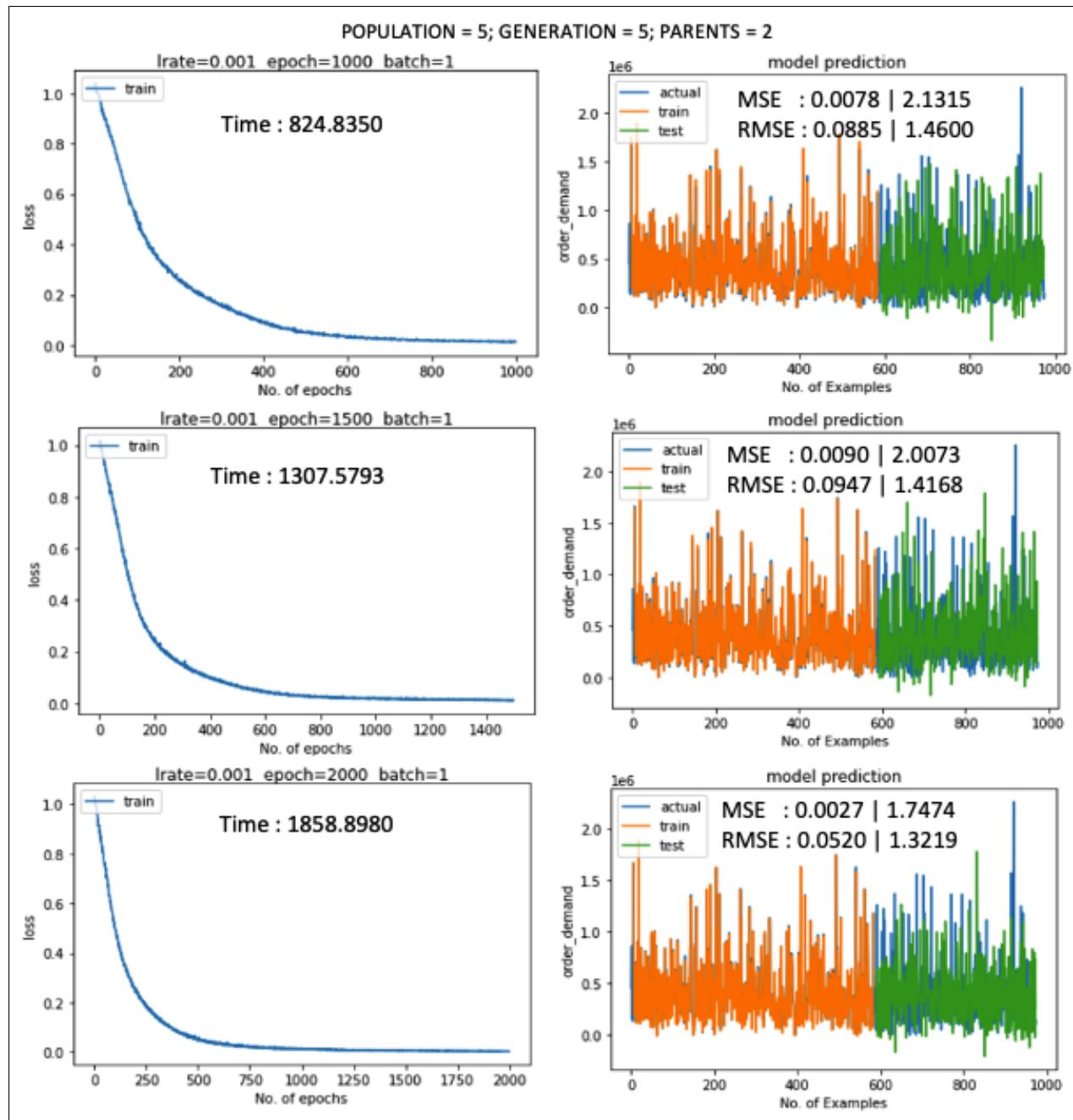
menghasilkan bobot optimal. Dari grafik penurunan *error*, MSE, RMSE, dan grafik prediksi terlihat jelas bahwa gabungan ANN dan GA ini mengeluarkan hasil yang lebih baik.



Gambar 4.12 Grafik ANN menggunakan *learning rate* = 0.0001; *batch size* = 1

Terlihat pada Gambar 4.12 grafik yang jauh berbeda antara algoritme ANN murni dan ANN yang sudah digabung dengan GA. Saat ANN memakai bobot dari GA prosesnya akan sedikit menambah waktu, tetapi ini bukan karena proses GA karena proses GA pada semua pengujian selesai dalam waktu yang kurang dari 1 detik. Grafik penurunan *error* menjadi lebih stabil dengan penurunan *error* yang cukup besar juga dengan selisih waktu yang tidak terlalu jauh.

Dari evaluasi pengujian ANN dapat disimpulkan *learning rate* terbaik ada di 0.001, maka dari itu pengujian ANN-GA pun akan dibahas dengan *learning rate* tersebut.

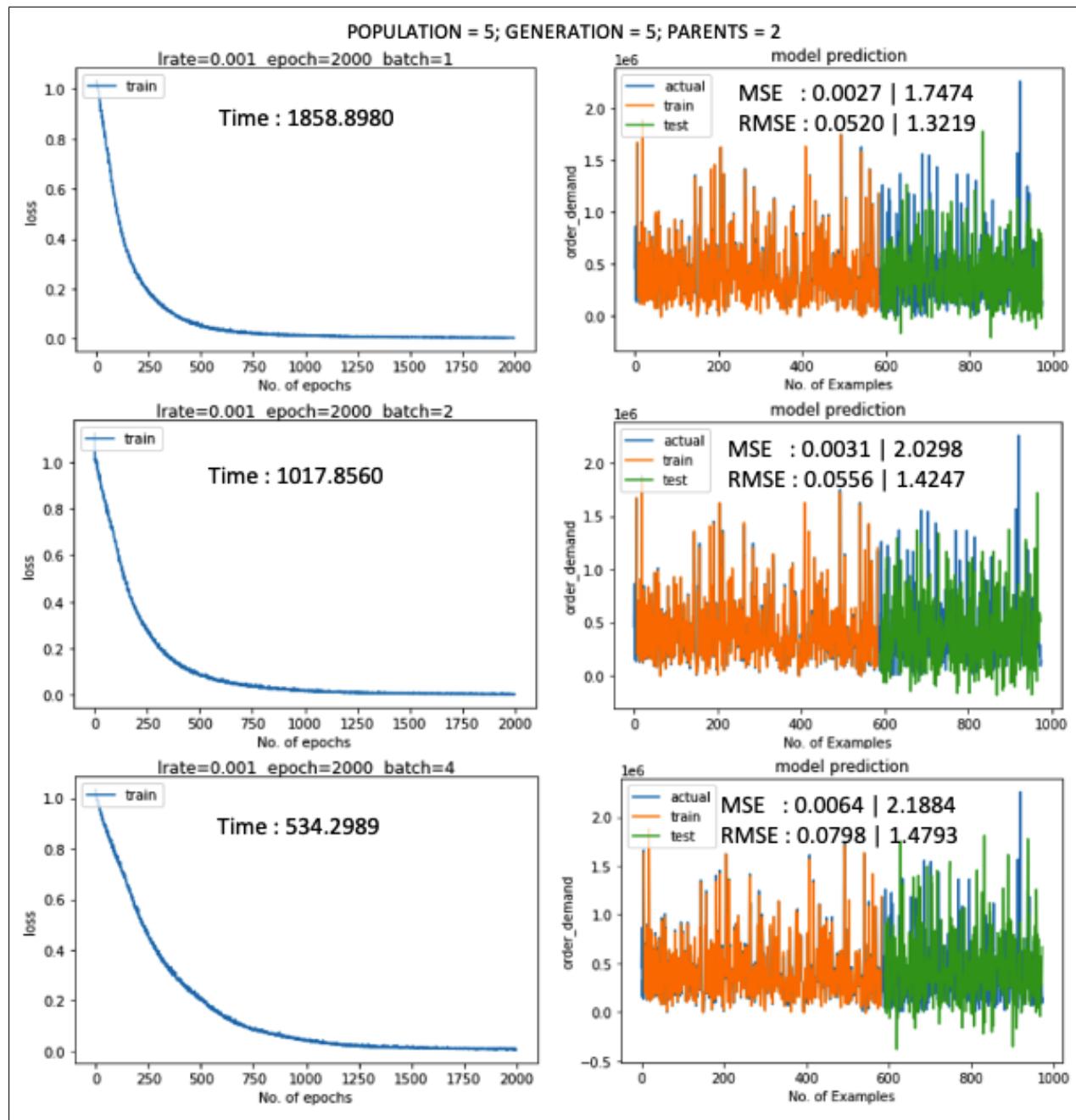


Gambar 4.13 Grafik ANN-GA menggunakan *learning rate* = 0.001; *batch size* = 1

Pada Gambar 4.13 didapatkan hasil yang jauh membaik, nilai MSE dan RMSE menyentuh dibawah 0.01 untuk semua pengujian, sedangkan pengujian ANN sebelumnya dengan parameter yang sama nilai MSE masih menyentuh 0.1. Hasil pengujian ini sudah terlihat bobot yang dipakai dari proses GA berjalan dengan baik walaupun proses ANN melambat sekitar 6-7% dari pengujian yang tidak menggunakan GA.

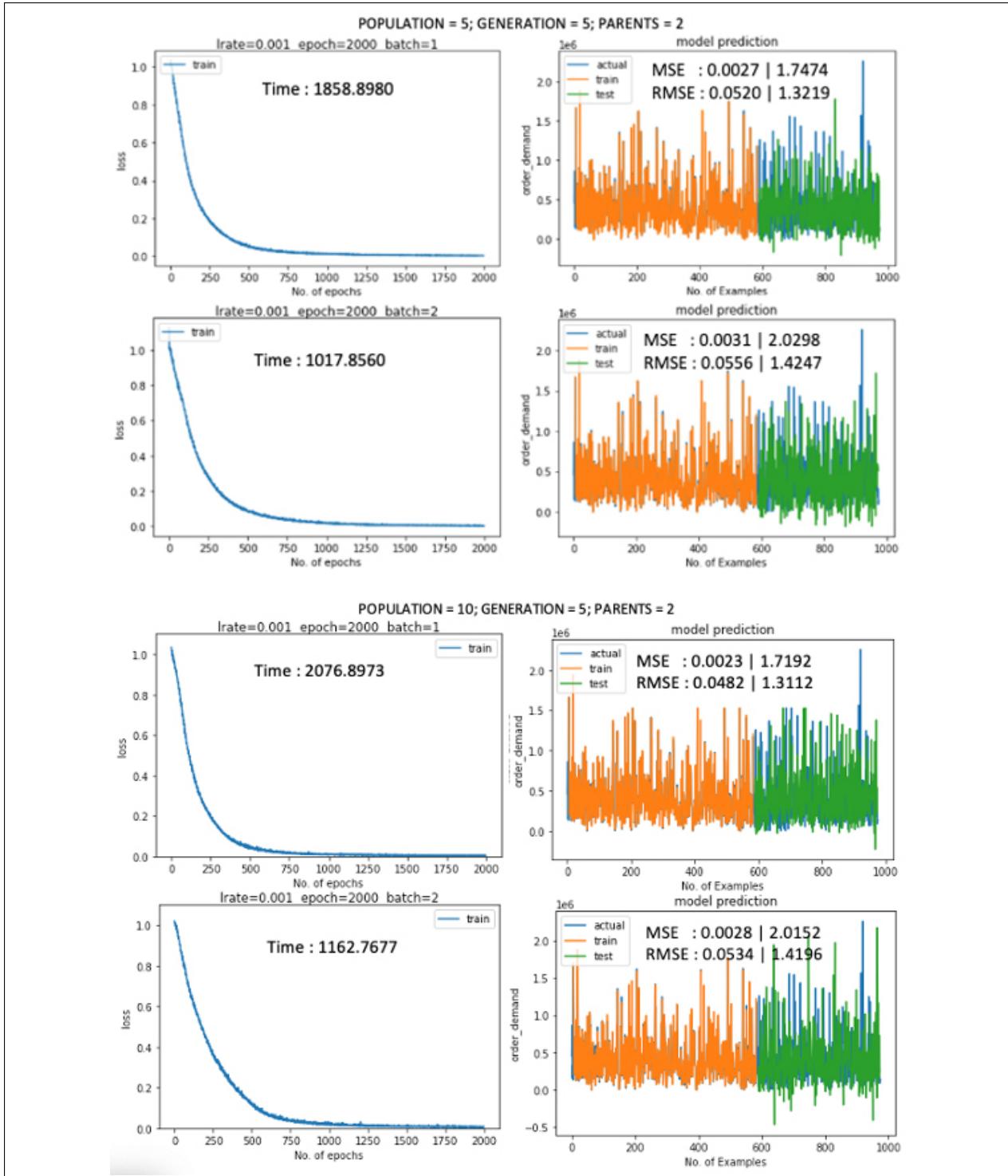
Dari hasil pengujian pada Gambar 4.13 nilai MSE dan RMSE terkecil didapat dalam *epoch* = 2000. Pada pengujian selanjutnya akan dibandingkan perbedaan hasil dengan beberapa *batch size* yang

berbeda dengan *epoch* yang sama. Hasil pengujinya ada pada Gambar 4.14 berikut.



Gambar 4.14 Grafik ANN-GA menggunakan *learning rate* = 0.001; *population* = 5; *generation* = 5

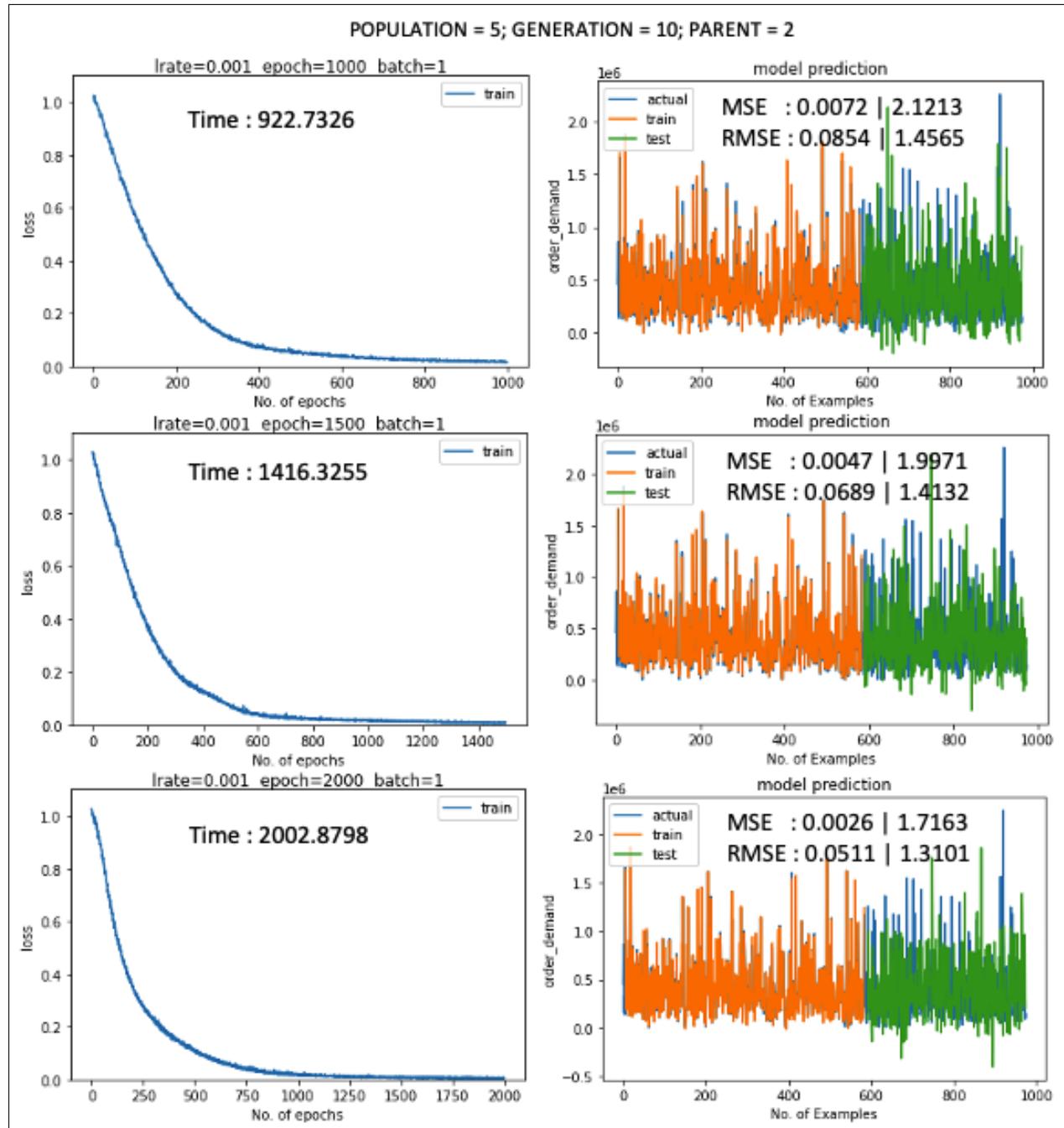
Pada Gambar 4.13 terlihat semakin besar *batch size* waktu pengujian akan semakin singkat namun nilai *error* yang dihasilkan juga semakin besar. Hal ini tidak berubah dari pengujian algoritme ANN tanpa GA yang sudah dilakukan sebelumnya.



Gambar 4.15 Grafik ANN-GA menggunakan *learning rate* = 0.001; *generation* = 5

Pada Gambar 4.15 dilakukan perbandingan pengubahan parameter pada GA yaitu *population* = 5 menjadi 10. Hasilnya pada *population* = 10 waktu yang diperlukan menjadi naik sekitar 12% dengan nilai *error* yang justru semakin besar juga. Pada *batch size* = 1 didapatkan nilai MSE yang sedikit lebih rendah, namun dengan melihat nilai RMSE kita bisa melihat selisih yang lebih rinci

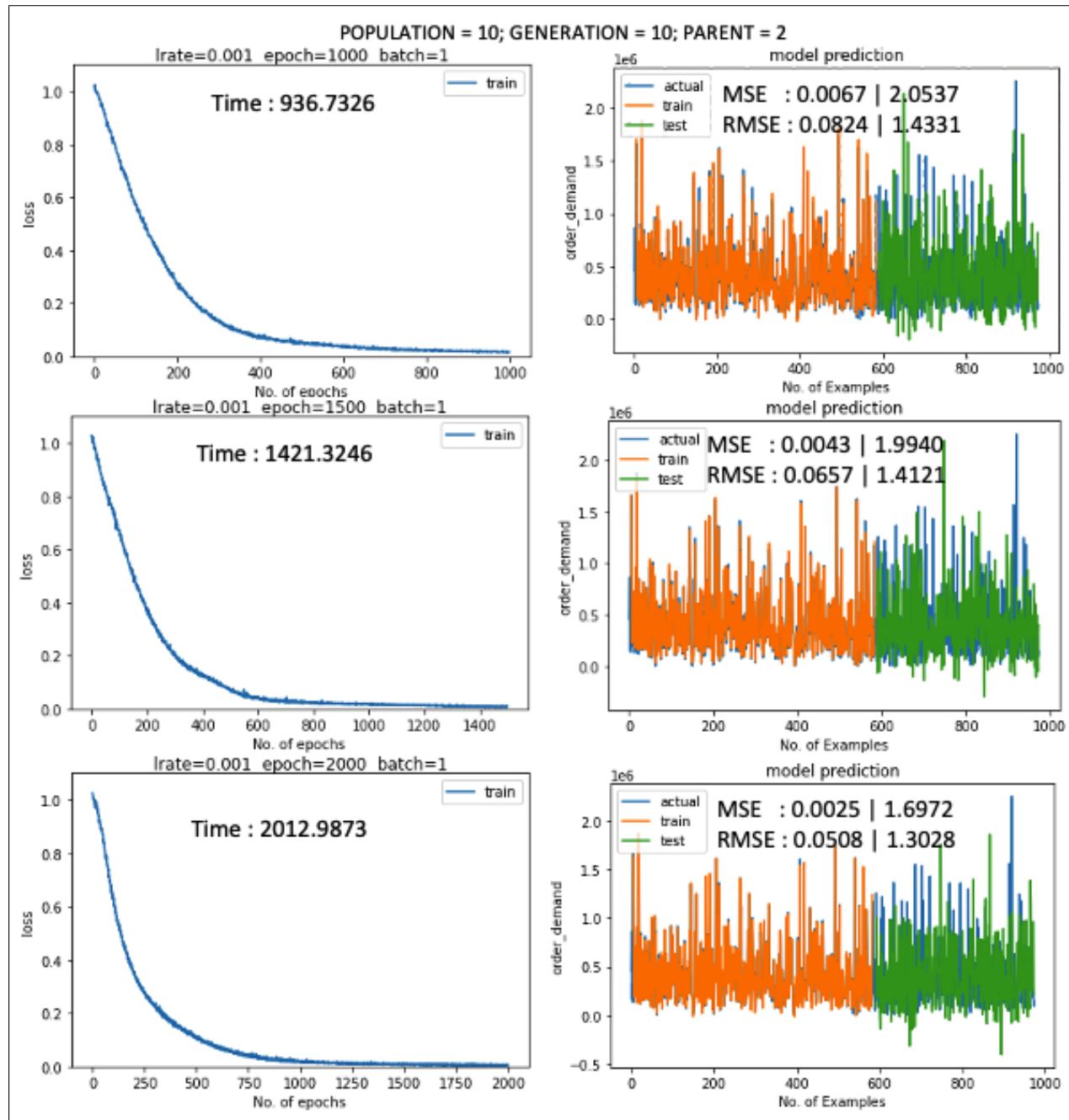
lagi bahwa *population* yang diubah menjadi 10 menurunkan nilai *error*. Hal ini disebabkan karena *population* pada GA berfungsi sebagai wadah untuk mencari nilai *fitness* terbaik. Semakin banyak *population*, semakin banyak pilihan (sampel) untuk memilih *fitness* tersebut.



Gambar 4.16 Grafik ANN-GA menggunakan *learning rate* = 0.001; *population* = 5; *generation* = 10

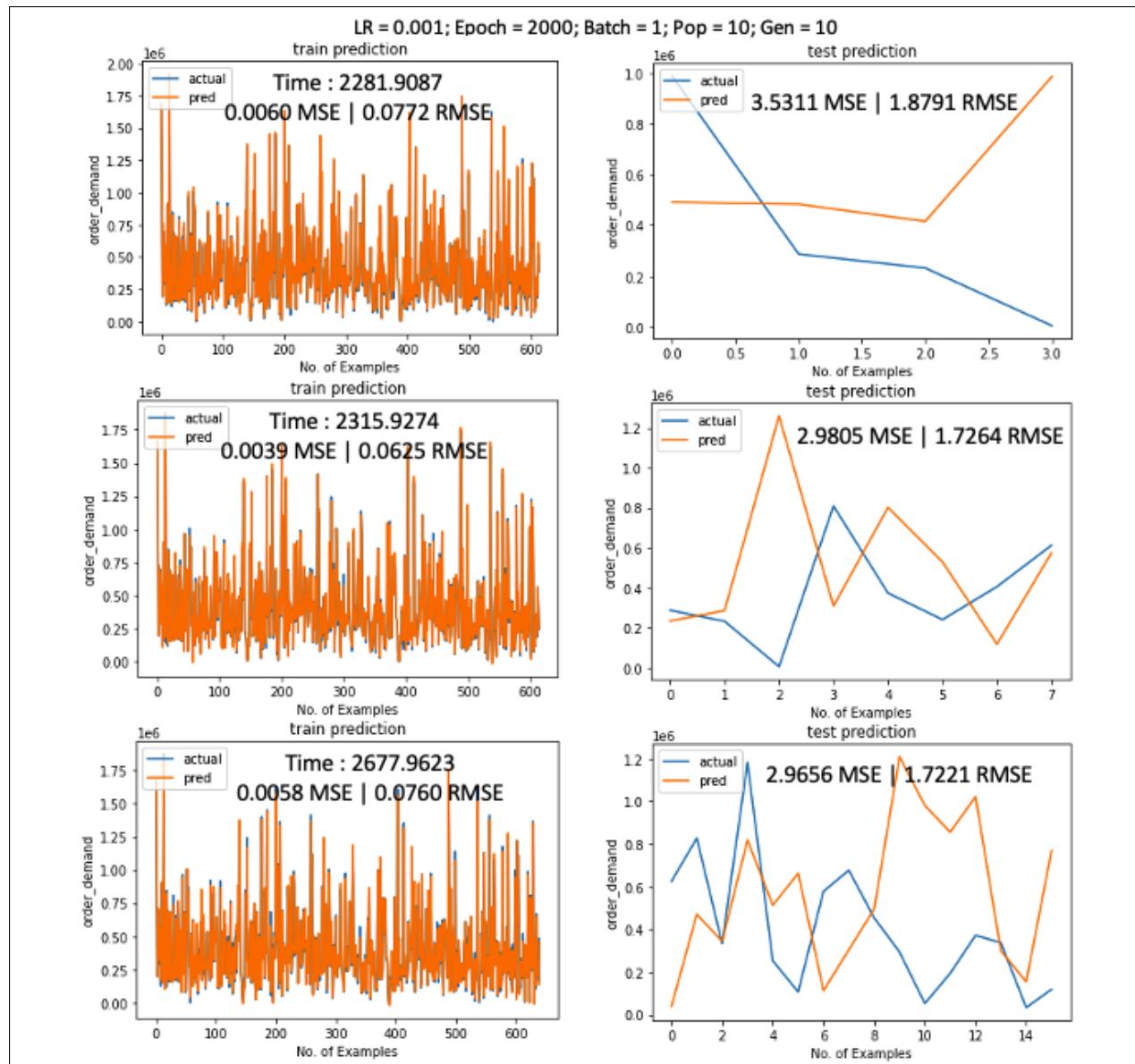
Pada Gambar 4.16 parameter GA yaitu *generation* diubah dari 5 menjadi 10 dengan tetap menggunakan *population* = 5. Hasilnya pada pengujian kali ini waktu yang didapat sekitar 8% lebih lambat daripada pengujian yang menggunakan *generation* = 5. Sedangkan *error* yang didapat lebih rendah sedikit dari pengujian sebelumnya. Hal ini terjadi karena *generation*

berfungsi mengulang proses GA sampai menemukan hasil terbaik, layaknya *epoch* dalam *neural network*.



Gambar 4.17 Grafik ANN-GA menggunakan *learning rate* = 0.001; *population* = 10; *generation* = 10

Pada Gambar 4.17 *population* dan *generation* diubah menjadi 10. Hasilnya pada pengujian kali ini waktu yang didapat sedikit lebih lambat sekitar 10-20 detik daripada pengujian yang menggunakan 5 *generation*. Sedangkan *error* yang didapat lebih rendah sedikit dari pengujian sebelumnya. Hal ini terjadi karena semakin besar *population* dan *generation*, GA akan semakin optimal walaupun waktunya akan bertambah.



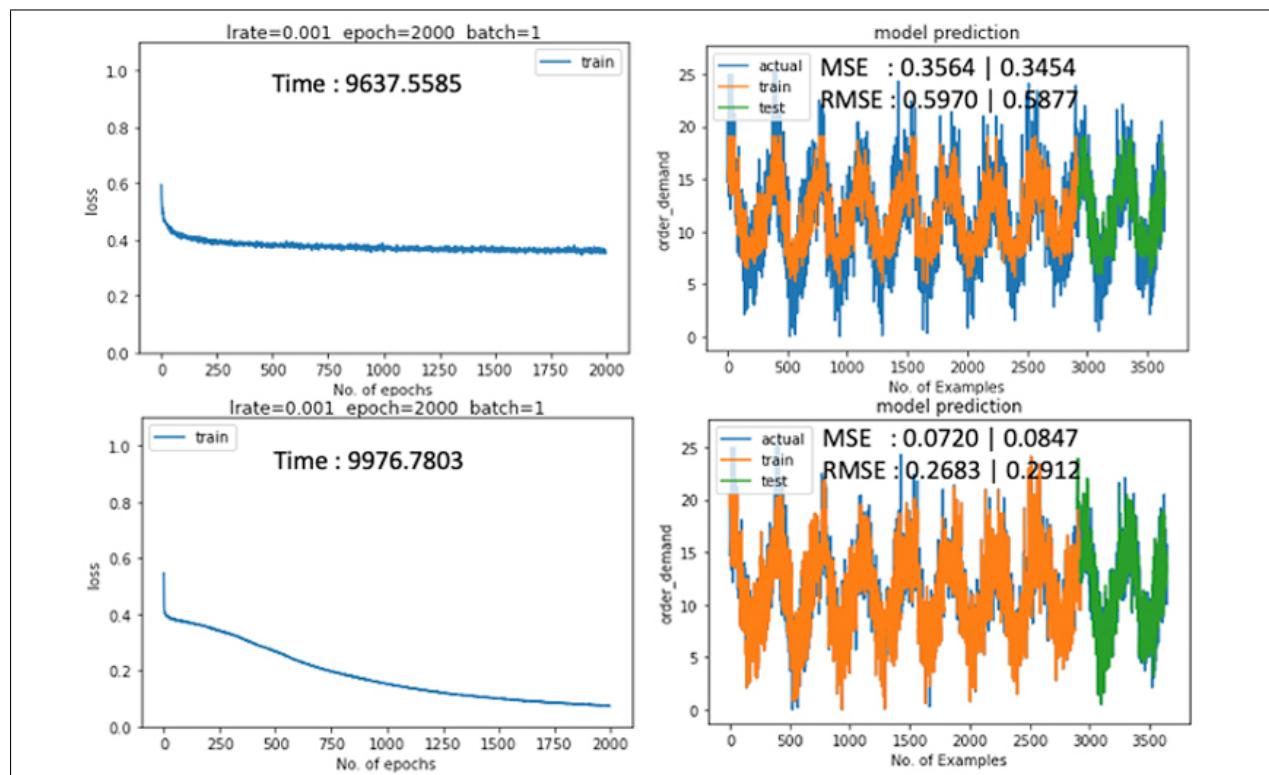
Gambar 4.18 Grafik ANN-GA menguji dengan jumlah *training* yang sama dan *testing* beberapa hari

Pada Gambar 4.18 pengujian dilakukan dengan model *training* 2.5 tahun dan *testing* beberapa periode (harian). Dari seluruh pengujian terlihat *training* yang dijalankan mengeluarkan hasil *error* yang serupa dengan pengujian sebelumnya karena memang jumlah *training* yang sama. *Neural Network* memang memerlukan data yang banyak untuk mengeluarkan hasil yang memuaskan. Pada *testing* 3, 7, 15 hari hasil *error* menunjukkan hasil yang buruk menandakan pengujian ini *overfitting*. Pengujian 3 hari memiliki hasil *testing* yang paling buruk karena data aktual menunjukkan hasil yang ekstrim untuk hari ke 2 sampai ke 3. Hal ini berbeda dengan pengujian lainnya, walaupun ada nilai ekstrim namun rata-rata nilai dalam jangkauan prediksi masih lebih bisa mengimbangi dibandingkan pengujian untuk 3 hari. Dengan pengujian tersebut dapat disimpulkan untuk *testing* dari program ini memerlukan beberapa waktu periode ke depan

agar nilai *error* bisa diminimasi dengan rata-rata yang semakin turun. Namun hal ini bukan berarti *testing* harus dilakukan sebanyak-banyaknya, karena jika terlalu banyak pun *error* bisa semakin meningkat.

4.3.3 Pengujian Data Seasonal

Pada bagian ini peneliti akan menggunakan *dataset* lain untuk menguji penggabungan algoritme terhadap data *seasonal*. Pengujian ini dilakukan untuk membandingkan apakah hasil pada *dataset* yang mempunyai pola teratur menjadi *overfitting*. Hasilnya dapat dilihat pada Gambar 4.19 berikut:



Gambar 4.19 Grafik perbandingan ANN dan ANN-GA menguji *dataset seasonal*

Pada Gambar 4.19 terlihat bahwa di gambar pertama dengan konfigurasi ANN tanpa GA sudah cukup bagus hasilnya tanpa ada *overfitting*. Namun dari pengujian ANN tanpa GA penurunan nilai *error* masih sangat minim dan cenderung datar grafik penurunannya. Lalu dilakukan pengujian menggunakan GA dengan 10 *population* dan 10 *generation* yang menjadi parameter terbaik untuk di pengujian *dataset* sebelumnya. Hasil ANN-GA terlihat sangat baik dengan penurunan *error* sekitar 50% dilihat dari nilai RMSE-nya. Hal ini disebabkan karena pengujian yang menggunakan GA membantu bobot dalam ANN agar lebih optimal menyesuaikan dengan model data-nya. Dapat disimpulkan *dataset* yang mempunyai karakteristik *seasonal* menunjukkan hasil yang lebih baik daripada *dataset random*.

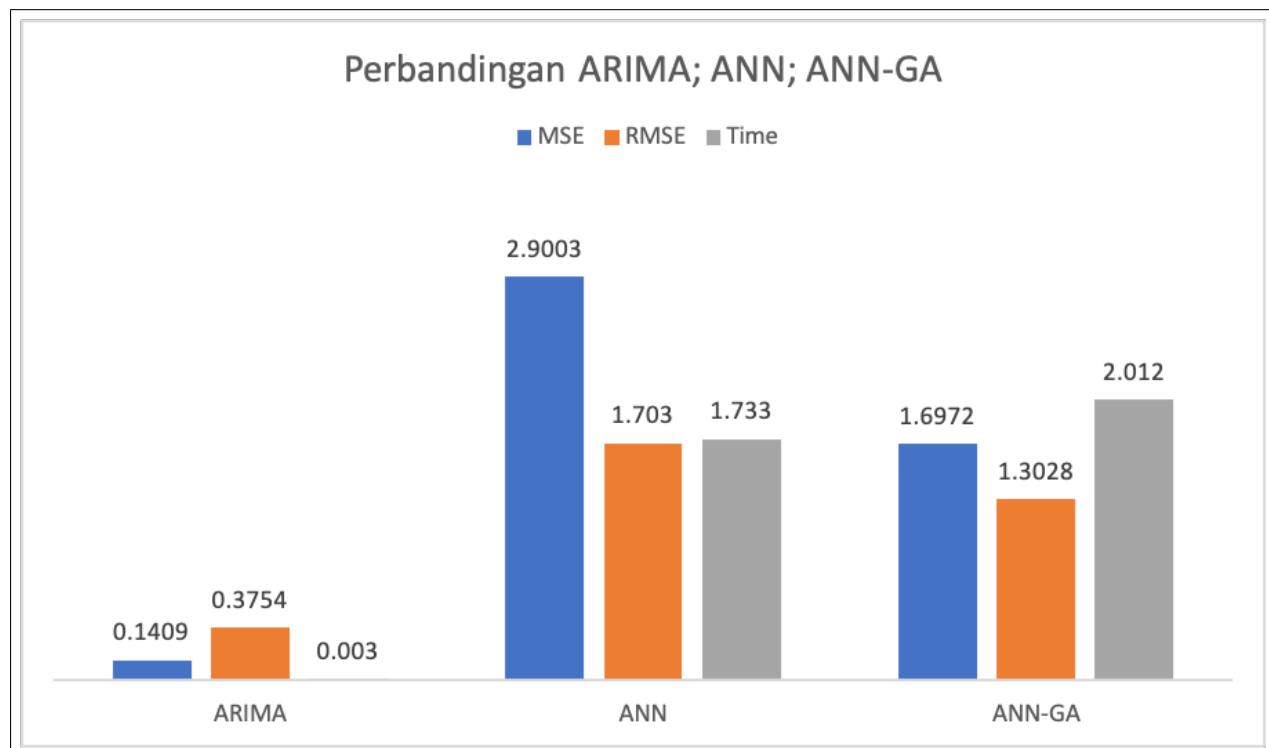
4.3.4 Pengujian Auto Regressive Integrated Moving Average (ARIMA)

Pada bagian ini peneliti mencoba metode lain di luar bidang *Machine Learning* yaitu *Statistical Method* yang dinilai metode terbaik untuk prediksi sampai saat ini, salah satunya ARIMA. Pada pengujian kali ini *dataset seasonal* juga dipakai untuk membandingkan performa dengan ANN.

Algoritme 4.5 Tabel ARIMA dengan p 1; d 2; q 1 untuk *dataset random*

	ARIMA
MSE	0.1409
RMSE	0.3754
Time	3

Pada Tabel 4.5 terlihat hasil prediksi ARIMA sudah cukup bagus dan hanya memerlukan 3 detik untuk mengeluarkan hasilnya. Hal ini sangat berbanding jauh dengan ANN ataupun ANN-GA yang sudah diuji. Untuk perbandingan *error* dan waktu yang diperlukan adalah sebagai berikut:



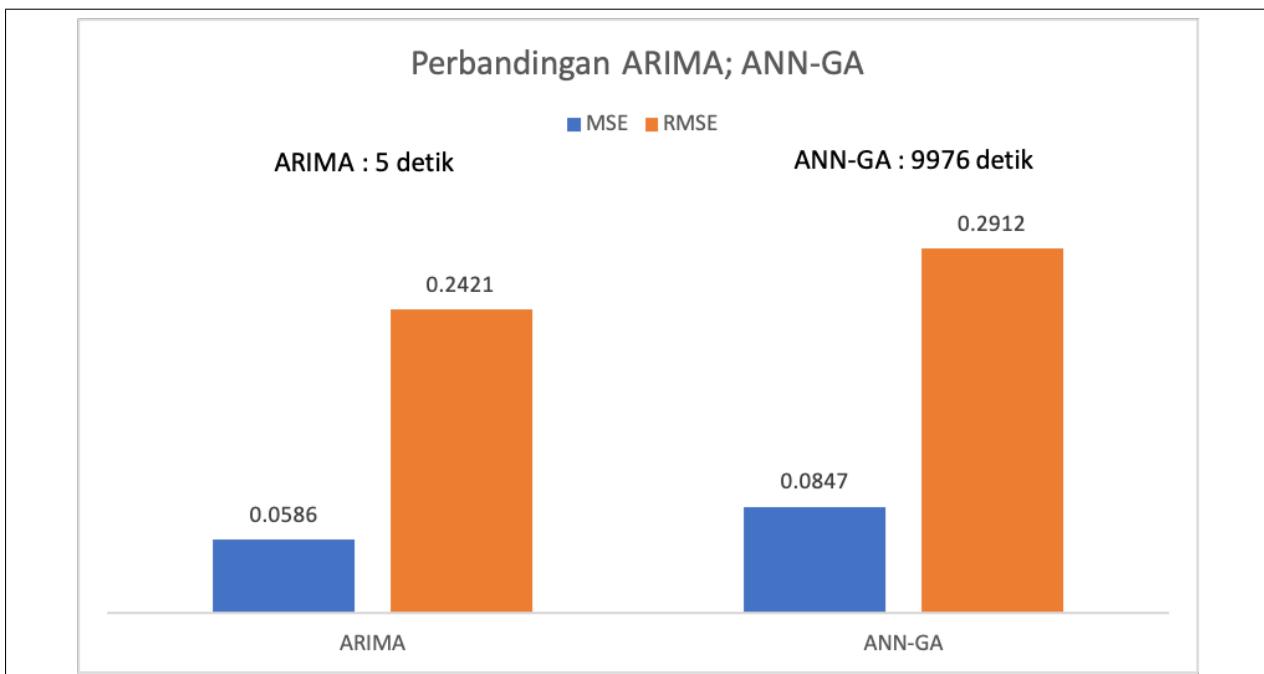
Gambar 4.20 Grafik perbandingan ARIMA, ANN, dan ANN-GA menguji *dataset random*

Pada Gambar 4.20 terlihat bahwa ARIMA masih sangat baik untuk prediksi *demand*. Namun dalam perbandingan ini metode *machine learning* terlihat sangat buruk karena hasil yang *overfitting*. Maka dari itu, peneliti mencoba membandingkan hasil ARIMA dan ANN-GA dengan *dataset seasonal* yang mempunyai hasil cukup baik tanpa ada *overfitting*. Hasil ARIMA pada *dataset* tersebut adalah sebagai berikut:

Algoritme 4.6 Tabel ARIMA dengan p 1; d 2; q 1 untuk *dataset seasonal*

	ARIMA
MSE	0.0586
RMSE	0.2421
Time	5

Pada Tabel 4.6 terlihat hasil yang lebih baik daripada hasil untuk *dataset random*. Untuk perbandingan dengan metode *machine learning* adalah sebagai berikut:



Gambar 4.21 Grafik perbandingan ARIMA dan ANN-GA menguji *dataset seasonal*

Pada Gambar 4.21 terlihat bahwa ARIMA menunjukkan hasil *error* yang sedikit lebih rendah daripada ANN-GA dan hanya memerlukan 5 detik untuk mengeluarkan hasil tersebut. Hal ini membuktikan metode ARIMA masih lebih baik dibandingkan ANN maupun ANN-GA dengan konfigurasi yang peneliti tentukan pada penelitian ini. ARIMA masih bisa beradaptasi dengan *dataset random* sekalipun hasil *error*-nya sedikit lebih tinggi.

Dengan melihat kedua Gambar 4.20 dan 4.21, terlihat pengujian ANN-GA pada *dataset stasioner* dan *dataset seasonal* berbeda jauh. Menggunakan konfigurasi parameter yang sama, *dataset stasioner* menempuh waktu 2012 detik (33.5 menit) dan pada *dataset seasonal* memerlukan waktu 9976 detik (166 menit). Waktu dalam pengujian ini berbeda jauh karena pada pengujian *dataset seasonal* jumlah data yang diuji adalah 3800 data, sementara *dataset stasioner* hanya menguji 900 data. Hal ini memberikan alasan bahwa waktu yang ditempuh *dataset seasonal* 4 kali lebih lambat karena data yang diuji pun 4 kali lebih banyak dibandingkan *dataset stasioner*.

4.4 Pembahasan Hasil Pengujian

Pada hasil pengujian untuk *dataset* menggunakan algoritme *Artificial Neural Network* (ANN) dan *Genetic Algorithm* (GA) ditemukan bahwa penggabungan kedua algoritme tersebut menghasilkan *error* yang jauh lebih kecil. Nilai *learning rate*, *epoch* dan *batch size* sangat memengaruhi hasil yang didapat pada algoritme ANN maupun gabungan ANN dan GA. Hasil terbaik ditemukan pada pengujian yang menggunakan parameter *population* = 5, dan *generation* = 5 pada GA dan *learning rate* = 0.001 dan *epoch* = 2000 pada ANN.

Ada beberapa skenario pengujian yang mengeluarkan hasil terbaik untuk menjadi perbandingan dengan nilai *error* dan *time* sebagai berikut:

Algoritme 4.7 Tabel ANN tanpa GA *learning rate* 0.001; *epoch* 1000; *batch size* 1

	Train	Test
MSE	0.1317	2.2376
RMSE	0.3629	1.4959
Time		772.0243

Pengujian 4.7 di atas mengeluarkan hasil yang cukup baik untuk *epoch* 1000 tanpa gabungan GA. Sementara itu, pada pengujian *epoch* 2000 mengeluarkan hasil yang lebih baik sebagai berikut:

Algoritme 4.8 Tabel ANN tanpa GA *learning rate* 0.001; *epoch* 2000; *batch size* 1

	Train	Test
MSE	0.0344	2.9003
RMSE	0.1854	1.7030
Time		1733.2720

Sementara itu, penggabungan ANN dan GA meningkatkan lagi hasil *error* yang semakin mengecil sebagai berikut:

Algoritme 4.9 Tabel ANN dengan GA *learning rate* 0.001; *epoch* 2000; *batch size* 1; *pop* 5; *gen* 5

	Train	Test
MSE	0.0027	1.7474
RMSE	0.0520	1.3219
Time		1858.8980

Pengujian selanjutnya menambahka jumlah populasi menjadi 10 dengan tetap mempertahankan 5 *generation* sebagai berikut:

Algoritme 4.10 Tabel ANN dengan GA *learning rate* 0.001; *epoch* 2000; *batch size* 1; *pop* 10; *gen* 5

	Train	Test
MSE	0.0023	1.7192
RMSE	0.0482	1.3112
Time	2076.8973	

Pengujian selanjutnya menambahkan jumlah populasi menjadi 10 dengan tetap mempertahankan 5 *generation* sebagai berikut:

Algoritme 4.11 Tabel ANN dengan GA *learning rate* 0.001; *epoch* 2000; *batch size* 1; *pop* 5; *gen* 10

	Train	Test
MSE	0.0026	1.7163
RMSE	0.0511	1.3101
Time	2002.8798	

Pengujian yang terakhir menggunakan 10 populasi, 10 *generation*, dan tetap menggunakan parameter ANN terbaik (LR 0.001; *epoch* 2000; *batch size* 1) sebagai berikut:

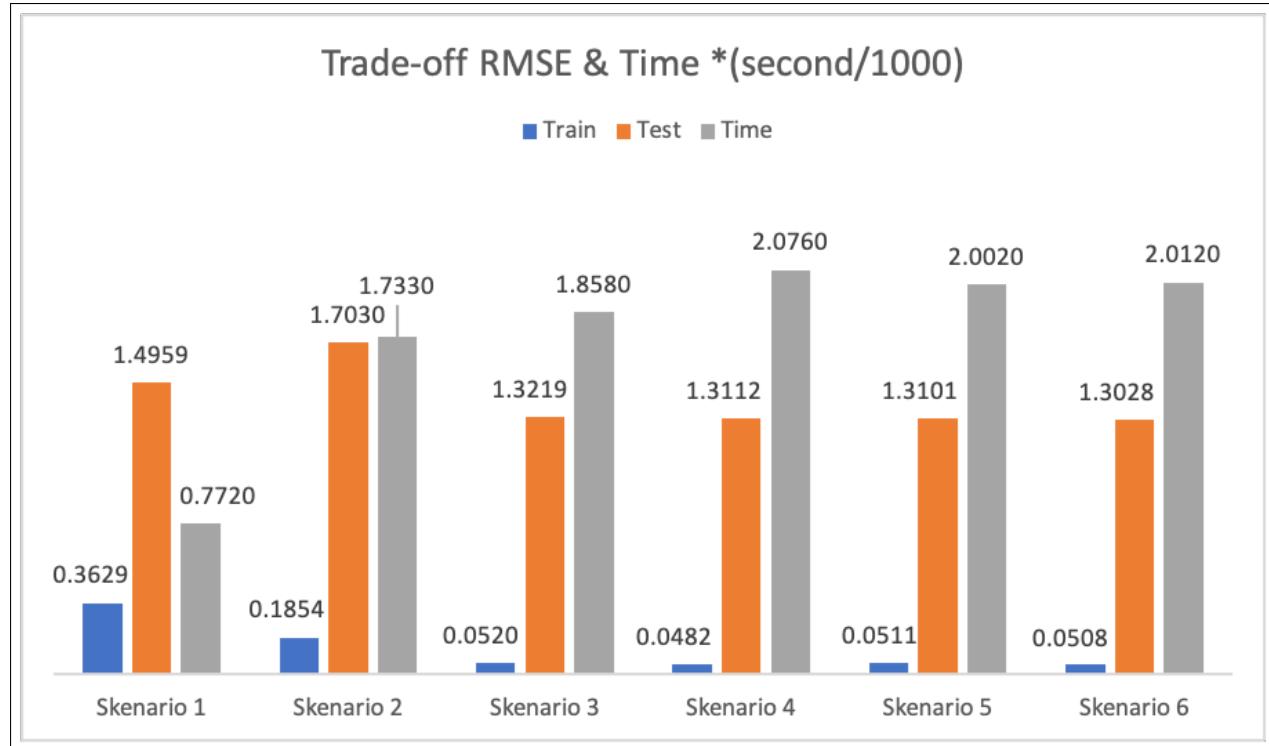
Algoritme 4.12 Tabel ANN dengan GA *learning rate* 0.001; *epoch* 2000; *batch size* 1; *pop* 10; *gen* 10

	Train	Test
MSE	0.0025	1.6972
RMSE	0.0508	1.3028
Time	2012.9873	

Hasil pengujian terbaik mempunyai konfigurasi sesuai pada Tabel 4.13 berikut:

Algoritme 4.13 Tabel konfigurasi skenario perbandingan RMSE dan *time*

	<i>L-Rate</i>	<i>Epoch</i>	<i>Batch Size</i>	<i>Population</i>	<i>Generation</i>
Skenario 1	0.001	1000	1	-	-
Skenario 2	0.001	2000	1	-	-
Skenario 3	0.001	2000	1	5	5
Skenario 4	0.001	2000	1	10	5
Skenario 5	0.001	2000	1	5	10
Skenario 6	0.001	2000	1	10	10



Gambar 4.22 Grafik perbandingan RMSE dan *time* dengan skala 1:1000 second

Terlihat pada Gambar 4.22 skenario 6 menjadi konfigurasi terbaik karena mendapat hasil RMSE terkecil walaupun terjadi *overfitting* pada *testing* dan waktu yang cukup tinggi. Namun waktu yang tinggi ini tidak menjadi masalah jika tujuan prediksi ini untuk prediksi jangka pendek atau menengah karena memerlukan ketepatan yang cukup tinggi sesuai dengan kegunaannya. Berbeda halnya dengan jangka panjang (*long-term forecast*) nilai *error* tidak perlu sekecil mungkin jika bisa memangkas waktu cukup signifikan, karena prediksi jangka panjang hanya perlu angka tersebut untuk gambaran proyeksi masa depan, bukan untuk menentukan stok dan perencanaan produksi.

Pada penelitian ini didapatkan hasil yang paling baik dengan menggunakan *learning rate* 0.001, *epoch* 2000, dan *batch size* 1 karena pada *learning rate* tersebut penurunan *error* terlihat sangat stabil dan menunjukkan hasil *error* yang minim. *Epoch* 2000 menjadi *epoch* yang optimal karena semakin banyak *epoch*, nilai *error* pada *training* semakin kecil. Nilai *batch size* juga disesuaikan dengan parameter lain untuk mencapai pengurangan waktu yang cukup banyak dengan tidak mengubah hasil *error* terlalu banyak. Hal ini terjadi dikarenakan nilai *learning rate* adalah salah satu *hyperparameter* terpenting dalam konfigurasi *neural network*. *Learning rate* dapat mengontrol seberapa besar model untuk beradaptasi. Selain itu, *epoch* adalah *hyperparameter* yang bisa diatur berdasarkan besaran *learning rate*. *Learning rate* yang terlalu besar akan menyebabkan model berpusat terlalu cepat dan kurang beradaptasi dengan detil. Sebaliknya jika *learning rate* terlalu kecil maka model akan terlalu lama untuk beradaptasi[8]. Dan juga sesuai dengan teori pada *genetic algorithm*, semakin besar *population* dan *generation* nilai *error* akan

semakin kecil namun waktu yang diperlukan akan bertambah. *Population* memengaruhi nilai sampel yang menjadi acuan untuk perhitungan *fitness*, dan *generation* memengaruhi keseluruhan proses karena semakin banyak parameter tersebut artinya semakin banyak pengulangan proses GA dari masukan sampai keluaran.

Pembahasan di atas merujuk pada hasil *training* yang menghasilkan nilai *error* sangat kecil. Namun untuk hasil *testing* pada setiap skenario yang dilakukan terdapat *gap* angka yang cukup jauh dibandingkan dengan *training*. Hasil *training* terbaik menghasilkan 0.0025 MSE dan 0.0508 RMSE sedangkan untuk *testing* menghasilkan 1.6972 MSE dan 1.3028 RMSE. Hal ini disebabkan *overfitting* yang terjadi jika model terlalu kompleks atau banyak *noise* pada data *training*[14]. Model data yang diuji pada penelitian ini memiliki data yang kompleks dengan *outlier* yang cukup banyak dan *range* data yang besar menjadikan hasil *testing overfitting*. Berbeda dengan model yang mempunyai pola lebih terstruktur seperti pengujian dengan data *seasonal*, hasil *testing* tidak menunjukkan *gap* yang jauh dengan data *training*. Hal ini bisa disimpulkan data *seasonal* yang diuji tidak *overfitting* maupun *underfitting* walaupun konfigurasi yang dipakai masih mengikuti konfigurasi untuk model sebelumnya yang memiliki karakteristik data *random*.

Lebih jauh lagi penelitian ini menunjukkan bahwa penggabungan ANN dan GA mempunyai hasil *error* yg lebih kecil. Hal ini dikarenakan *Artificial Neural Network* mempunyai kemampuan untuk mengenali pola dengan baik (*pattern recognition*). Namun ANN menggunakan nilai acak untuk inisialisasi bobotnya diawal, setelah itu algoritme baru mempelajari pola yang ada untuk diperbarui selama pengujian berjalan. Inisialisasi dan pembaharuan bobot yang tidak tepat dapat menyebabkan ANN terjebak di *local minima*. Hal ini dibantu dengan *Genetic Algorithm* yang dapat mencari solusi optimal untuk bobot yang dipakai ANN. Walaupun inisialisasi angka awal di GA tetap menggunakan angka acak, namun GA melakukan proses *selection*, *crossover* dan *mutation* untuk menghasilkan nilai *fitness* terbaik. Proses GA terus melakukan evolusi untuk mencari solusi optimal. Namun untuk saat ini penelitian yang dilakukan belum bisa mengalahkan metode ARIMA dari sisi nilai *error* ataupun waktu yang diperlukan program. Metode ANN-GA ini akan berguna dikemudian hari jika waktu yang diperlukan cukup beberapa detik saja dan nilai MSE *testing* bisa sekecil tahap *training* pada pengujian stasioner tanpa *overfitting*.

Sebagai perbandingan, pada penelitian yang dilakukan oleh Mohammad Badrul dengan kasus predksi hasil pemilukada menunjukkan hasil yang serupa. Pengujian dengan algoritme *neural network* menghasilkan nilai yang cukup baik, yaitu akurasi 91.64%, presisi 91.20% dan nilai AUC sebesar 0.942. Sedangkan pengujian dengan penggabungan ANN-GA menghasilkan nilai yang lebih baik lagi, yaitu akurasi 93.03%, presisi 91.28% dan nilai AUC 0.971. Maka dapat disimpulkan pengujian model pemilu legislatif DKI Jakarta dengan menggunakan penggabungan ANN-GA lebih baik daripada ANN saja[15].

BAB 5 PENUTUP

Bab ini berisi kesimpulan yang dilandasi oleh penelitian dan pengujian yang sudah dilakukan serta dilengkapi saran untuk perkembangan berikutnya.

5.1 Kesimpulan

Kesimpulan dari penggabungan ANN dengan GA pada *demand forecasting* adalah:

1. Konfigurasi parameter terbaik untuk penelitian ini adalah 0.001 LR, 2000 *epoch*, 1 *batch size*, 10 *population* dan 10 *generation* dengan menggunakan 60% *training* dan 40% *testing*.
2. Waktu komputasi yang dibutuhkan untuk memberikan solusi terbaik dengan kriteria *error* terkecil adalah 2012.9873 detik (33.5 menit) menggunakan algoritme ANN-GA.
3. Pada pengujian ANN ditemukan hasil 0.0344 MSE / 0.1854 RMSE untuk *training*, 2.9003 MSE / 1.7030 RMSE untuk *testing* dan waktu 1733.2720 detik (29 menit) dengan konfigurasi parameter yang sama. Hasil ini lebih buruk dari ANN-GA dengan peningkatan nilai sebesar 256% RMSE untuk *training* dan 28.8% RMSE untuk *testing* (mengacu pada hasil *training*).
4. Hasil simulasi data yang dilakukan dalam penelitian ini menunjukkan penggabungan ANN dan GA untuk kasus *demand forecasting* memberikan hasil yang lebih baik karena nilai *error* bisa tereduksi sampai 70% walaupun waktu komputasi meningkat sekitar 20%.
5. Penggabungan ANN dan GA pada untuk melakukan *demand forecasting* belum dapat menghasilkan solusi optimal yang mengungguli solusi metode ARIMA. Namun pendekatan ANN dan GA dapat memberikan solusi yang mendekati solusi ARIMA ketika digunakan untuk *forecast* pada pola data *seasonal*.
6. Kualitas solusi pengembangan ANN-GA sangat sensitif terhadap pola data. ANN-GA belum bisa memberikan solusi optimal untuk data yang bersifat stasioner.

5.2 Saran

Saran untuk penggabungan ANN-GA dalam *demand forecasting* adalah:

1. Menentukan nilai *learning rate*, *epoch*, dan *batch size* yang tepat untuk menghasilkan ANN yang optimal.
2. Menggunakan algoritme lain selain *Genetic Algorithm* untuk pemilihan bobot.
3. Menggunakan model data *seasonal* atau *cyclical* untuk meminimalisir *overfitting* pada hasil *testing*.

DAFTAR REFERENSI

- [1] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Pearson, 2008.
- [2] Montri Inthachot, Veera Boonjing and Sarun Intakosum, “Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend”, *Hindawi Publishing Corporation Computational Intelligence and Neuroscience*, Bangkok, Thailand, October 2016.
- [3] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1998.
- [4] Amusan, D.G, Olabode, A.O, Ojo, O.S, Folowosele, A.O and Oyediran, M.O, “Hybrid Design using Counter Propagation Neural Network-Genetic Algorithm Model for the Anomaly Detection in Online Transaction”, *International Journal of Advances in Scientific Research and Engineering (IJASRE)*, Ogbomoso, Nigeria, vol. 5, issue 9, September 2019.
- [5] Prasanna Kumar, Dr. Mervin Herbert, Dr. Srikanth Rao, “Demand Forecasting Using Artificial Neural Network Based on Different Learning Methods: Comparative Analysis”, *International Journal For Research in Applied Science and Engineering Technology (IJRASET)*, Mangalore, India, vol. 2, issue 4, April 2014.
- [6] Lakhmi C. Jain, R.P. Johnson, A.J.F. Van Rooij, *Neural Network Training Using Genetic Algorithm*, World Scientific, 1996.
- [7] Fausett, Laurene V., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1994.
- [8] David Kriesel, *A Brief Introduction to Neural Networks*, Dkriesel , 2005.
- [9] Dominic Masters; Carlo Luschi, ”Revisiting Small Batch Training for Deep Neural Networks”, *Graphcore Research*, Bristol, UK, 2018.
- [10] Purnawansyah, “Backpropagation and Genetic Algorithm in Network Traffic Activities Forecasting”, *International Journal of Computing and Informatics (IJCANDI)*, Makassar, Indonesia, February 2017.
- [11] Ali Saud Al Tobi, Maamer Bevan, Geraint Wallace, Peter Harrison, David Ramachandran, K. P, ”A Review on Applications of Genetic Algorithm For Artificial Neural Network”, *International Journal of Advance Computational Engineering and Networking*, Scotland, UK, 2016.
- [12] Donald Waters, *Inventory Control and Management*, John Wiley & Sons, 2003.

DAFTAR REFERENSI

- [13] Douglas C. Montgomery, Cheryl L. Jennings, Murat Kulahci, *Introduction to Time Series Analysis and Forecasting*, John Wiley & Sons, 2008.
- [14] Aurelien Geron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow*, John Wiley & Sons, 2003.
- [15] Mohammad Badrul, “Optimasi Neural Network Dengan Algoritma Genetika Untuk Prediksi Hasil Pemilukada”, *Bina Insani ICT Journal*, Jakarta, Indonesia, Juny 2016.