Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202296738
Vol. 9, No. 7, Desember 2022, hlm. 1349-1356 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

PENGEMBANGAN AUTO-AI MODEL GENERATIF ANALISIS KOMPLEKSITAS WAKTU ALGORITMA UNTUK DATA MULTI-SENSOR IOT PADA NODE-RED MENGGUNAKAN EXTREME LEARNING MACHINE

Imam Cholissodin*[1], Dahnial Syauqy[2], Dwi Ady Firmanda[3], Ibrahim Aji[4], Edy Rahman[5], Syazwandy Harahap[6], Fernando Septino[7]

[1,2,3,4,5,6]Universitas Brawijaya, Malang
Email: [1]imamcs@ub.ac.id, [2]dahnial87@ub.ac.id, [3]d2a.fman3da@student.ub.ac.id,
[4]ibrahimaji242@student.ub.ac.id, [5]edhyra@student.ub.ac.id, [6]hrpwandy@student.ub.ac.id,
[7]fernandoseptino@student.ub.ac.id
*Penulis Korespondensi

(Naskah masuk: 01 Desember 2022, diterima untuk diterbitkan: 26 Desember 2022)

Abstrak

Awal mulanya, algoritma hanya dipakai untuk solusi penyelesaian persamaan matematika sederhana, seperti aljabar, aritmatika, probabilitas, dan lainnya yang lebih banyak dikerjakan secara manual dan membutuhkan waktu dan upaya yang cukup tinggi seperti pada kasus penghitungan nilai kompleksitas waktu algoritma dengan model rumus T(n), baik untuk algoritma *non-rekursif* maupun *rekursif*. Namun dengan perkembangan teknologi komputer untuk AI, *Machine Learning* maupun *Deep Learning*, algoritma dengan basis AI tersebut, dalam penelitian ini dikembangkan untuk menemukan solusi general persamaan model T(n) secara otomatis dari desain algoritma sederhana atau kompleks. Langkah dalam penelitian digunakan pembuatan model generatif berbasis algoritma *Extreme Learning Machine* (ELM) berdasarkan pencatatan nilai waktu komputasi pada beberapa kali pengujian untuk mengotomasi penentuan model persamaan kompleksitas waktu algoritma secara general baik untuk pencarian *best case, worst case* maupun *average case* untuk *non-rekursif*, dan *base case* dan *recurrent case* untuk *rekursif*, maupun keduanya. Hasil komparasi nilai T(n) dari ELM, yang tercepat atau terkecil waktu komputasinya digunakan sebagai rekomendasi algoritma untuk pengolahan data multi-sensor pada *Internet of Things* (IoT) simulator maupun non-simulator menggunakan Node-RED dengan tambahan platform yaitu flespi dan Heroku, sebagai solusi general untuk semua jenis kasus dan analisis algoritmanya. Berdasarkan pengujian didapatkan selisih nilai antara data aktual dengan hasil prediksi dalam ukuran nilai rata-rata MAPE sebesar 11,90%, yang menunjukkan nilai kesalahan yang cukup kecil.

Kata kunci: *auto-ai, model generatif, kompleksitas waktu, analisis algoritma, multi-sensor iot, node-red, elm*

DEVELOPMENT OF AUTO-AI MODEL OF GENERATIVE TIME COMPLEXITY ANALYSIS ALGORITHM FOR MULTI-SENSOR IOT DATA ON NODE-RED USING EXTREME LEARNING MACHINE

Abstract

Initially, algorithms were only used for solving simple mathematical equations such as algebra, arithmetic, probability, and others that were mostly carried out manually and required quite a lot of time and effort as in the case of calculating the value of the time complexity of the algorithm with the formula model of T(n), both for non-recursive and recursive algorithms. However, with the development of computer technology for AI, both Machine Learning and Deep Learning, the AI-based algorithms in this study were developed to identify general solutions to the T(n) model equation automatically from simple or complex algorithm designs. The steps in the study are utilized to create a generative model based on the Extreme Learning Machine (ELM) algorithm according to the recording of computational time values on several tests to automate the determination of time complexity equation model of the algorithm in general including the search of best cases, worst cases, and average cases for non-recursive, and base cases and recurrent cases for recursive, as well as algorithms that contain both. The results of the comparison of T(n) values from ELM revealed that the fastest or smallest computational time is used as the algorithm recommendations for multi-sensor data processing in the Internet of Things (IoT) simulators and non-simulators by utilizing Node-RED with additional platforms i.e., flespi and Heroku, as a general solution for the entire types of cases and analysis of their algorithms. Based on the tests that have been carried out, the difference

in value between the actual data and the prediction results in the size of the MAPE average value of 11.90%, which shows a fairly small error value.

Keywords: auto-ai, *generative model, time complexity, analysis algorithm, iot multi-sensor, node-red, elm*

1. PENDAHULUAN

Permasalahan di era modern saat ini memiliki banyak peluang kemudahan solusi dengan bermunculan seiring perkembangan teknologi *Artificial Intelligence* (AI) serta kebutuhan zaman yang mengedepankan ketepatan, kecepatan, serta keamanan (Xu *et al.*, 2021). Permasalahan-permasalahan tersebut dapat terjadi diskala nasional maupun internasional, seperti permasalahan ekonomi tentang kebijakan penerapan bitcoin atau uang digital lainnya, politik seperti pemilihan umum (pemilu) secara online, sosial, dan geopolitik serta untuk meminimalkan peluang konflik antar sesama anak bangsa maupun antar negara maupun lainnya (Olsher, 2015) (Lv et al., 2022). Selain itu, terdapat pula permasalahan yang terjadi di dunia maya atau dunia komputerisasi, seperti pemecahan *puzzle* tercepat (Więckowski and Shekhovtsov, 2021), penghitung paling akurat (Latha and Jeeva, 2019), dan sebagainya khususnya pada permasalahan pemilihan algoritma dengan suatu persamaan model T(n) atau ditulis dengan C(n) yang biasanya dihitung secara manual dan membutuhkan waktu lama serta ketelitian yang ekstra untuk mendapatkan hasil analisis kompleksitas waktu yang tepat (Cholissodin & Riyandani, 2016), berupa nilai *best case* atau terbaik, *worst case* maupun *average case* untuk algoritma *non-rekursif*, serta *base case* dan *recurrent case* untuk algoritma *rekursif* sebagai pembanding untuk kecepatan pengolahan data sebagai rekomendasi pada perangkat IoT dengan pemanfaatan teknologi komputasi awan (*cloud computing*) secara mudah (Jin et al., 2022).

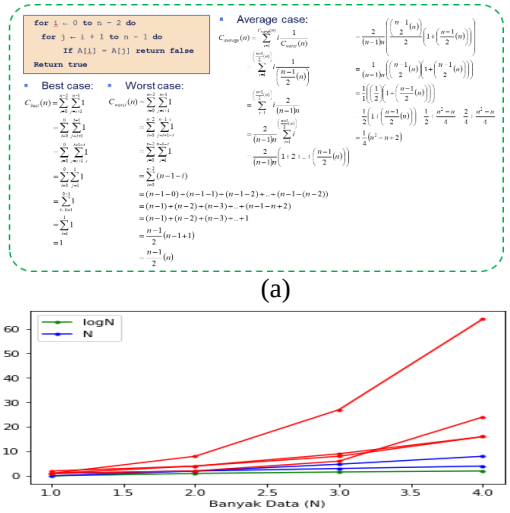
Secara umum, untuk mengatasi permasalahan komputerisasi khususnya, dibutuhkan suatu solusi atau langkah-langkah penyelesaian yang disebut algoritma berbasis AI yang di dalamnya termasuk keilmuan *Machine Learning* dan *Deep Learning* (Cholissodin *et al.*, 2019). Algoritma berasal dari seorang ilmuwan Persia pada abad ke-9 M yang bernama Muhammad ibn Musa al-Khawarizmi atau secara singkat, al-Khawarizmi (Crossley and Henry, 1990). Beliau dikenal sebagai ilmuwan yang banyak melakukan terobosan saintis di bidang matematika, astronomi, dan geografi kala itu. al-Khawarizmi mempopulerkan konsep sains matematika yang sangat umum di masa sekarang, yaitu aljabar yang kebetulan juga berasal dari nama beliau. Otomasi sebagai kemampuan utama algoritma yang berbasis AI memiliki peran utama yang tidak dapat dipisahkan baik dalam bidang sains maupun dalam praktik komputasi analisis efisiensi kompleksitas waktu algoritma (Levitin, 2012).

Berdasarkan pada beberapa permasalahan sulitnya analisis efisiensi waktu komputasi algoritma tersebut, maka dalam penelitian ini diusulkan pengembangan model generatif untuk menyelesaikan permasalahan penentuan model analisis untuk menemukan solusi general persamaan model $T(n)$ secara otomatis dari desain algoritma sederhana atau kompleks menggunakan algoritma *Extreme Learning Machine* (ELM) untuk rekomendasi algoritma yang nantinya digunakan pada perangkat IoT (Bandekar and Vijayalakshmi, 2020), berdasarkan pencatatan nilai waktu komputasi pada beberapa kali pengujian untuk mengotomasi penentuan model persamaan kompleksitas waktu algoritma tersebut pada teknologi *cloud*, dari hasil algoritma dengan cara melakukan *deploy* IoT app menggunakan beberapa pilihan alat pendukung aplikasi seperti Google Colaboratory, ngrok, Node-RED, dan Heroku (Zheng and Zhao, 2022). Penggunaan data multi sensor pada Node-RED ini karena dari banyak kasus yang ada lebih banyak melibatkan data yang bervariasi dengan menggunakan data yang diharapkan dapat melakukan pencatatan dari parameter yang lebih presisi atau detail untuk suatu pengolahan dengan tujuan tertentu (Misalkan pengamatan kondisi cuaca, pengamatan terkait dengan prediksi dari lama waktu irigasi dalam pertanian, dsb). Selain itu, penggunaan Node-RED karena menyediakan *grafik user interface* (GUI) yang mudah untuk digunakan dalam mengintegrasikan banyak sumber data baik dari *webservice* (API) maupun dari alat-alat perangkat IoT.

2. DASAR TEORI

2.1. Analisis Kompleksitas Waktu Algoritma

Analisis kompleksitas waktu suatu algoritma dapat menunjukkan informasi latar belakang yang memberikan gagasan umum seberapa cepat dan seberapa lama algoritma itu akan berjalan terhadap variabel permasalahan yang digunakan sebagai masukan. Sehingga, analisis tersebut dapat menentukan sebuah estimasi waktu seberapa lama algoritma akan berjalan jika diberikan set permasalahan sebanyak N data atau input nilai (McConnell, 2001; Agarwal et al., 2022). Dengan kata lain, analisis algoritma dapat menilai suatu kualitas untuk sebuah algoritma, yaitu seberapa besar nilai dari *time efficiency* (efisiensi waktu) komputasinya (Rutanen, 2018; Iwama and Teruyama, 2020).



Gambar 1. (a) Analisis kompleksitas waktu algoritma UniqueElement(), (b) Ilustrasi Waktu Komputasi $T(n) = C_{op}$ x $C(n)$, di mana biasanya dituliskan $C_{op} = 1$ satuan waktu

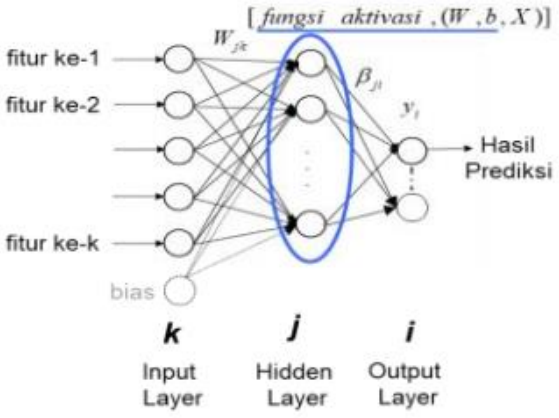
2.2. Machine Learning

Machine Learning (ML) merupakan bagian dari AI yang memiliki kumpulan beragam algoritma yang mampu membuat penentuan model persamaan atau fungsi keputusan untuk prediksi, klasifikasi, *clustering* atau lainnya secara otomatis oleh mesin atau sistem komputasi digital dari hasil pembelajaran data (Arumugaraja et al., 2022; Cassoli et al., 2021). Salah satu *Machine Learning* yang paling cepat proses komputasinya adalah algortima *Extreme Learning Machine (*ELM*)*. Proses dalam algoritma ELM tersebut tidak membutuhkan iterasi, karena semua proses di dalamnya menggunakan operasi matriks satu tahapan perulangan dimasing-masing proses, baik *training* maupun *testing*. Komputasi langkah perhitungan yang dilakukan pada proses pelatihan dan proses pengujian dapat dilihat pada Gambar 2 (Cholissodin et al., 2019; Lingitz et al., 2018) (…., 2020).

Algoritma ini dipilih karena kelebihan yang dimiliki dalam kecepatan proses pembelajaran (training) apabila dibandingkan dengan algoritma jaringan saraf tiruan yang lainnya seperti Back Propagation, Learning Vector Quantization (LVQ). Karena di dalam ELM hanya membutuhkan satu kali proses iterasi yaitu semua operasi pembelajarannya berbasis matrix.

2.3. Simulator IoT dan Robotika Node-RED pada Teknologi Cloud

Istilah "IoT" sering dikaitkan dengan Kevin Ashton yang pada tahun 1997 dan pekerjaannya di Procter and Gamble menggunakan tag RFID untuk pengelolaan rantai pasok (Azman, 2020) yang mana terdapat jaringan yang menghubungkan objek fisik dengan internet.



Langkah-langkah training algoritma ELM dengan bias:

1. Buat random W_{jk} sebagai bobot masukan dan nilai bias b jika ada. Ukuran matrik bias b adalah [1 x j]
2. Hitung matrik keluaran hidden layer H = 1/(1+exp(-(X.W^T + ones(N_{train},1)*b))). Dan N_{train} adalah banyaknya data training.
3. Hitung output weight $\hat{\beta} = H^+.Y$ dimana H$^+$=(H^T.H)$^{-1}$.H^T
4. Hitung $\hat{Y} = H.\hat{\beta}$

Langkah-langkah testing algoritma ELM dengan bias:

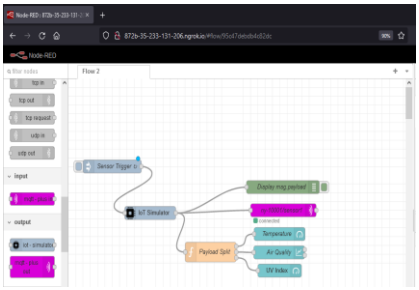
1. Diketahui W_{jk}, nilai bias b jika ada dan $\hat{\beta}$
2. Hitung H = 1/(1+exp(-(X_{test}.W^T + ones(N_{test},1)*b)))
3. Hitung Hitung $\hat{Y} = H.\hat{\beta}$
4. Hitung nilai evaluasi, misal dengan MAPE atau akurasi, atau lainnya

(b)

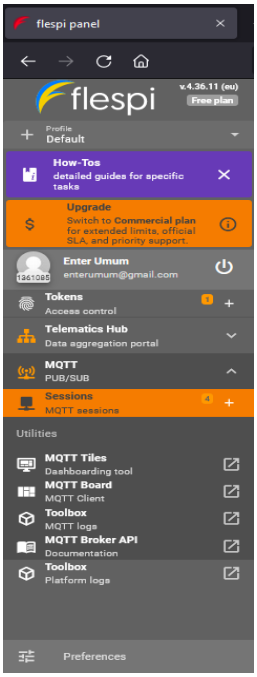
Gambar 2. (a) Arsitektur ELM, (b) Langkah proses training dan testing Algoritma ELM

Pada era modern sekarang ini dengan berkembangnya teknologi *Cloud* menyebabkan meningkatnya penggunaan dari IoT untuk berbagai sektor strategis dan menjadi bagian utama untuk memberikan kemudahan dan manfaat besar pada kegiatan kehidupan sehari-hari. Perkembangan IoT juga turut mempengaruhi perkembangan penggunaan komponen IoT dalam bidang robotika untuk meningkatkan efisiensi dan jangkauan perangkat dari perangkat yang dibuat (Aloraini et al., 2022; Semeraro et al., 2023). Sedangkan Node-RED seperti yang ditunjukkan pada Gambar 3 adalah *tools* berbasis web untuk pemrograman visual dengan fungsi javascript yang dikembangkan oleh IBM untuk menghubungkan perangkat keras, *Application Programming Interface* (API), dan *service* atau lainnnya dari IoT.

Sebelum dilakukan implementasi pada perangkat yang digunakan perlu melakukan percobaan awal pada sebuah simulator, dimana lingkungan pada simulator tersebut dimiripkan dengan kondisi yang ada lingkungan perangkat yang akan diimplementasikan sehingga dapat menjadi suatu landasan peneliti dan menghemat biaya atau operasional usaha yang dibutuhkan untuk mencapai hasil yang optimal dari penelitian.



(a)



(b)

Gambar 3. (a) Node-RED flow editor dengan Ngrok, (b) MqTT Board pada flespi untuk Subcriber dari Node-RED

3. PERANCANGAN DAN IMPLEMENTASI

Pada tahap perancangan penelitian dilakukan dengan membuat pemisahan implementasi kode program untuk mendapatkan nilai waktu komputasinya, misal dalam satuan detik, menit atau lainnya. Setelah mendapatkan data nilai komputasinya, maka data-data tersebut dimasukkan pada implementasi algoritma ELM seperti pada Kode Program 1, untuk mendapatkan model persamaan $T(n)$ secara otomatis, di mana n menyatakan banyak data sebagai nilai fitur atau variabel independen dan $T(n)$ sebagai nilai target atau variabel dependennya.

Pengelompokan analisis kompleksitas waktu algoritma, dalam penelitian ini dibagi menjadi dua bagian berikut:

a. Pendekatan konvensional (McConnell, 2001; Levitin, 2012), dengan cara manual dengan menghitung T(n) atau C(n) yang nantinya dapat dilihat perbandingan antara algoritmanya apakah salah satunya sebagai *Big O (upper bound)* atau *Big θ (Big Theta atau tight bound)*, atau *Big Ω (Big Omega).*

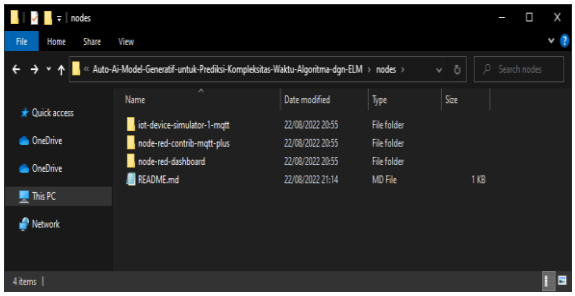
```
1   fungsi untuk elm_train():
2       ..
3       .
4       bobot = np.random.rand(..)
5       bias = np.random.rand(..)
6       h = 1 / \
7               (1 + np.exp(-
8       (np.dot(data_training[],
9       np.transpose(bobot)) + bias)))
10
11      h_plus = np.dot(np.linalg.inv(..),
12      np.transpose(h))
13      output_weight = np.dot(h_plus,
14      data_training[..])
15
16      ..
17      .
18      return redirect('/_daa_list')
19
20  fungsi untuk elm_test():
21      ..
22      .
23      h = 1 /(1 + np.exp(-
24      (np.dot(data_testing,
25      np.transpose(baca_bobot_input)) +
26      baca_bias)))
27      predict_T_n = np.dot(h,
28      baca_bobot_output)
29
30      response = jsonify({..})
31      ..
32      .
33      return response
```

Kode Program 1. *Snippet pseudocode* Algoritma ELM untuk memprediksi nilai $T(n)$

Pendekatan ini memang tidak perlu mengimplementasikan *psudocode* suatu algoritma menjadi bentuk kode program yang dapat di-*compile*, karena memang analisis waktunya dilakukan secara manual. Tetapi keterbatsan dari pendekatan ini adalah jika dihadapkan pada satu algoritma yang kompleks, yang didalamnya memuat *pseudocode non-rekursif* sekaligus *rekursif*, maka analisis secara manual akan sangat sulit dilakukan. Apalagi sekarang bentuk algoritma ketika diimplementasikan antara satu bahasa pemrograman misal python vs java vs c/c++ vs lainnya bisa jadi sangat berbeda sintaknya, belum lagi jika sintaksnya dibuat yang mendukung pemrograman terdistribusi seperti *map-reduce* dengan *Hadoop* dan *Spark* untuk komputasi *Big Data* maupun pemrograman *parallel* pada GPU.

b. Pendekatan dengan menggunakan *Machine Learning* sebagai *Auto Analysis* untuk penetuan model generatif analisis kompleksitas waktu suatu algoritma dengan pemberian kode *timer* pada kode algoritma tersebut, yaitu dengan algoritma ELM untuk memprediksi nilai T(n), ini sebagai kontribusi dalam penelitian ini.

Kode repositori terkait Node-RED disimpan pada komputer maupun pada github untuk persiapan proses *deploy* ke Heroku seperti pada Gambar 7.



(a)

```
{
    "name": "node-red-heroku",
    "version": "0.0.2",
    "dependencies": {
    "feedparser": "^2.2.10",
    "https-proxy-agent": "^5.0.1",
    "nano": "~10.x",
    "node-red": "^2.2.2",
    "node-red-contrib-iot-simulator": "file:nodes/iot-device-
     simulator-1-mqtt/node-red-contrib-iot-simulator",
    "node-red-contrib-mqtt-plus":
    "file:nodes/node-red-contrib-mqtt-plus",
    "node-red-dashboard": "^3.1.7",
    "pg": "^8.3.0",
    "redis": "~0.10.1",
    "when": "~3.x"
    }
}
```

(b)

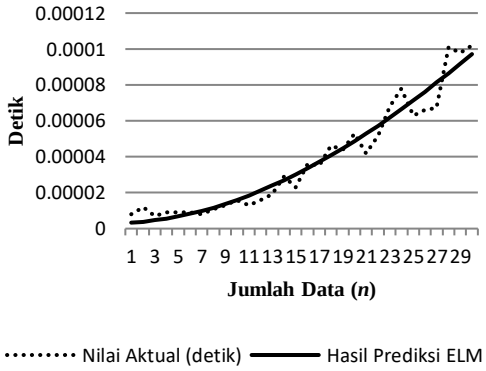
Gambar 7. (a) Menambahkan dependensi nodes IoT Simulator pada Node-RED untuk Deploy ke Heroku, (b) Konfigurasi pada file package.json

4. PENGUJIAN DAN ANALISIS

Hasil proses pendataan untuk mendapatkan nilai hitung $C(n)$ atau $T(n)$-nya dengan disisipkan *timer* yang diletakkan pada awal dan akhir koding, lalu dihitung nilai *delta*-nya. Pada Tabel 1 merupakan bagian percobaan yang dilakukan pada algoritma pengurutan, yaitu Timsort dengan pencatatan waktu dalam detik sebagai pendekatan nilai $T(n)$ sebagai nilai aktualnya dan dibandingkan dengan hasil prediksi $T(n)$ dari algoritma ELM pada data sampel yang telah disediakan

Tabel 1. Hasil T(n) dari Algoritma Timsort (O(n log (n))) dengan Pencatatan waktu dalam detik sebagai pendekatan nilai $T(n)$ dibandingkan $T(n)$ dari hasil Prediksi ELM

n	Waktu Timsort dengan T(*n*)	
	Nilai Aktual (detik)	**Hasil Prediksi ELM**
1	0,000008	0,00000333
2	0,000012	0,00000389
3	0,000007	0,00000465
4	0,000009	0,00000564
5	0,000009	0,00000684
6	0,000009	0,00000825
7	0,000008	0,00000988
8	0,000011	0,00001171
9	0,000013	0,00001374
10	0,000016	0,00001598
11	0,000013	0,00001842
12	0,000016	0,00002106
13	0,000019	0,00002389
14	0,000029	0,00002691
15	0,000023	0,00003011
16	0,000036	0,00003349
17	0,000035	0,00003705
18	0,000046	0,00004078
19	0,000044	0,00004468
20	0,000053	0,00004874
21	0,000042	0,00005296
22	0,000052	0,00005733
23	0,000068	0,00006185
24	0,000078	0,00006651
25	0,000063	0,00007131
26	0,000066	0,00007624
27	0,000067	0,00008129
28	0,000101	0,00008647
29	0,000098	0,00009176
30	0,000102	0,00009716



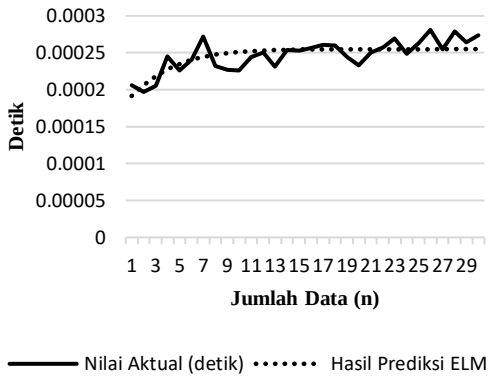
Gambar 8. Hasil Pengujian Timsort dengan pencatatan waktu dalam detik sebagai pendekatan nilai T(n) sebagai nilai aktualnya dan dibandingkan dengan hasil prediksi T(n) dari algoritma ELM

Pada uji tambahannya, yaitu dilakukan percobaan pada algoritma lainnya. Pada Tabel 2 merupakan percobaan menggunakan metode serupa dengan nama algoritma Counting sort.

Tabel 2. Hasil T(n) dari Algoritma Counting Sort (O(n)) dengan Pencatatan waktu dalam detik sebagai pendekatan nilai $T(n)$ dibandingkan $T(n)$ dari hasil Prediksi ELM

n	Waktu Counting Sort dengan T(*n*)	
	Nilai Aktual (detik)	**Hasil Prediksi ELM**
1	0,000206	0,0001916
2	0,000197	0,00020637
3	0,000205	0,00021837
4	0,000245	0,00022779
5	0,000226	0,00023502
6	0,000241	0,00024046
7	0,000272	0,00024449
8	0,000232	0,00024744
9	0,000227	0,00024959
10	0,000226	0,00025114
11	0,000244	0,00025225
12	0,00025	0,00025305
13	0,000231	0,00025362
14	0,000254	0,00025403
15	0,000253	0,00025432
16	0,000256	0,00025453
17	0,000261	0,00025468

n	Waktu Counting Sort dengan T(*n*)	
	Nilai Aktual (detik)	Hasil Prediksi ELM
18	0,00026	0,00025478
19	0,000244	0,00025486
20	0,000233	0,00025491
21	0,00025	0,00025495
22	0,000257	0,00025497
23	0,000269	0,00025499
24	0,000249	0,00025501
25	0,000263	0,00025502
26	0,000281	0,00025502
27	0,000255	0,00025503
28	0,000279	0,00025503
29	0,000264	0,00025503
30	0,000274	0,00025503



Gambar 9. Hasil Pengujian Counting Sort dengan pencatatan waktu dalam detik sebagai pendekatan nilai T(n) sebagai nilai aktualnya dan dibandingkan dengan hasil prediksi T(n) dari algoritma ELM

Jika semakin kecil waktu komputasi yang dibutuhkan, maka akan semakin baik untuk digunakan sebagai rekomendasi algoritma yang nantinya diimplementasikan pada IoT. Karena spesifikasinya umumnya sangat terbatas. Berdasarkan pengujian pada Gambar 8, didapatkan selisih nilai antara data aktual dengan hasil prediksi dalam ukuran nilai *Mean Absolute Percentage Error* (MAPE) sebesar 19,03 %, yang menunjukkan nilai kesalahan yang cukup kecil. Tentunya nilai MAPE tersebut dapat lebih diminimalkan lagi dengan menggunakan lebih banyak skenario pengujian yang dilibatkan, termasuk misal dengan penggunaan teknik optimasi sebagai pertimbangan untuk percobaan penelitian ke depannya.

Pada saat pengujian pada algoritma Counting sort yang ditunjukkan pada Gambar 9 didapatkan nilai MAPE 4,76 % yang menunjukkan nilai evaluasi yang cukup baik. Namun, hasil tersebut perlu ditingkatkan lagi tingkat konsistensi hasilnya melalui beberapa percobaan sehingga penelitian lebih lanjut masih diperlukan untuk meningkatkan tingkat kedekatan data aktual terhadap data prediksi dan memenuhi reliabilitas dari algoritma yang digunakan.

5. KESIMPULAN

Berdasarkan pengujian yang dilakuakn dengan menggunakan nilai *n* mulai dari 1 sampai 30 sebagai *variable independent* dan *T*(*n*) sebagai *variable independent* didapatkan bahwa nilai t(n) merupakan suatu nilai yang dapat diprediksi menggunakan algoritma *extreme learning machine*. Kemudian rata-rata hasil nilai waktu t(n) tersebut setelah dibandingkan selisihnya terkait dengan t(n) yang didapatkan dari analisis kompleksitas algoritma secara manual atau *native* atau konvensional diketahui memiliki korelasi yang baik. Hal ini menunjukkan bahwa pendekatan Auto-AI yang digunakan dalam menentukan estimasi nilai efisiensi waktu kompleksitas suatu algoritma dapat dinyatakan sebagai *generative model* yang dapat dianalogikan untuk bisa digunakan pada berbagai tipe algoritma termasuk algoritma *rekursif*, *non-rekursif*, gabungan dari *rekursif* dan *non-rekursif*, algoritma dengan penggunaan pemrograman parallel (GPU), algoritma dengan penggunaan pemrograman terdistribusi (menggunakan *single node* maupun *multi node* yang biasanya digunakan untuk mengolah data yang besa atau *big data*), serta algoritma yang merupakan gabungan dari semuanya

Pada penelitian selanjutnya akan dicoba untuk diterapkan penggunaan Auto-AI model generatif tersebut pada suatu algoritma yang yang disertakan juga solusi kasus yang diselesaikan. Contohnya misalkan membuat model generatif untuk menentukan waktu komputasi dari suatu algoritma yang bergantung pada suatu *device* tertentu (algoritma yang digunakan untuk perhitungan jumlah kendaraan yang masuk dan yang keluar, algoritma yang digunakan untuk memberikan prediksi mengenai perlakuan misalkan dari *multi sensor* pada bidang pertanian, perkebunan, transportasi, dan lainnya). Sehingga pendekatan Auto-AI tersebut dapat lebih dikembangkan lagi untuk bisa lebih cepat dalam meningkatkan tingkat kepresisian terkait estimasi waktu dari suatu algoritma baik yang sederhana maupun sangat kompleks, serta dilakukan pengujian juga terhadap spesifikasi perangkat keras yang digunakan untuk implementasi.

DAFTAR PUSTAKA

CHOLISSODIN, I. & RIYANDANI, E., 2016. Review: A State-of-the-Art of Time Complexity (Non-Recursive and Recursive Fibonacci Algorithm). *Journal of Information Technology and Computer Science*, 1(1), pp.14–27. https://doi.org/10.25126/jitecs.2016112.

CHOLISSODIN, I., SUTRISNO, S., SOEBROTO, A.A., Hasanah, U. and Febiola, Y.I., 2019. *AI, Machine Learning & Deep Learning*. Filkom UB.

CROSSLEY, J.N. & HENRY, A.S., 1990. Thus spake al-Khwārizmī: A translation of the text of Cambridge University Library Ms. Ii.vi.5. *Historia Mathematica*, 17(2), pp.103–131. https://doi.org/10.1016/0315-0860(90)90048-I.

IWAMA, K. & TERUYAMA, J., 2020. Improved average complexity for comparison-based sorting. *Theoretical Computer Science*, 807, pp.201–219. https://doi.org/10.1016/j.tcs.2019.06.032.

LATHA, C.B.C. & JEEVA, S.C., 2019. Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques. *Informatics in Medicine Unlocked*, 16, p.100203. https://doi.org/10.1016/j.imu.2019.100203.

LEVITIN, A., 2012. *Introduction to the Design and Analysis of Algorithms*. 3rd ed. [online] Available at: <https://www.booksfree.org/introduction-to-the-design-and-analysis-of-algorithms-by-anany-levitin-3rd-edition-pdf/> [Accessed 7 August 2022].

LV, Z., WANG, N., MA, X., SUN, Y., MENG, Y. AND TIAN, Y., 2022. Evaluation Standards of Intelligent Technology based on Financial Alternative Data. *Journal of Innovation & Knowledge*, 7(4), p.100229. https://doi.org/10.1016/j.jik.2022.100229.

MCCONNELL, J.J., 2001. *Analysis of algorithms: an active learning approach*. Boston: Jones and Bartlett Publishers.

OLSHER, D.J., 2015. New Artificial Intelligence Tools for Deep Conflict Resolution and Humanitarian Response. *Procedia Engineering*, 107, pp.282–292. https://doi.org/10.1016/j.proeng.2015.06.083.

RUTANEN, K., 2018. Minimal characterization of O-notation in algorithm analysis. *Theoretical Computer Science*, 713, pp.31–41. https://doi.org/10.1016/j.tcs.2017.12.026.

WIĘCKOWSKI, J. & SHEKHOVTSOV, A., 2021. Algorithms Effectiveness comparison in solving Nonogram boards. *Procedia Computer Science*, 192, pp.1885–1893. https://doi.org/10.1016/j.procs.2021.08.194.

AGARWAL, R., SINGH, J., & GUPTA, V. 2022. Prediction of temperature elevation in rotary ultrasonic bone drilling using machine learning models: An in-vitro experimental study. *Medical Engineering & Physics*, 103869. https://doi.org/10.1016/j.medengphy.2022.103869

ALORAINI, F., JAVED, A., RANA, O., & BURNAP, P. 2022. Adversarial machine learning in IoT from an insider point of view. *Journal of Information Security and Applications*, *70*, 103341. https://doi.org/10.1016/j.jisa.2022.103341

ARUMUGARAJA, M., PADMAPRIYA, B., & POORNACHANDRA, S. 2022. Design and development of foot worn piezoresistive sensor for knee pain analysis with supervised machine learning algorithms based on gait pattern. *Measurement*, *200*, 111603. https://doi.org/10.1016/j.measurement.2022.111603

BANDEKAR, S. R., & VIJAYALAKSHMI, C. (2020). Design and Analysis of Machine Learning Algorithms for the reduction of crime rates in India. *Procedia Computer Science*, *172*, 122–127. https://doi.org/10.1016/j.procs.2020.05.018

CASSOLI, B. B., ZIEGENBEIN, A., METTERNICH, J., ĐUKANOVIĆ, S., HACHENBERGER, J., & LAABS, M. 2021. Machine Learning use case in manufacturing – an evaluation of the model's reliability from an IT security perspective. *Procedia CIRP*, *104*, 1161–1166. https://doi.org/10.1016/j.procir.2021.11.195

JIN, S., LIU, N., & YU, Y. 2022. Time complexity analysis of quantum difference methods for linear high dimensional and multiscale partial differential equations. *Journal of Computational Physics*, *471*, 111641. https://doi.org/10.1016/j.jcp.2022.111641

LINGITZ, L., GALLINA, V., ANSARI, F., GYULAI, D., PFEIFFER, A., SIHN, W., & MONOSTORI, L. 2018. Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer. *Procedia CIRP*, *72*, 1051–1056. https://doi.org/10.1016/j.procir.2018.03.148

SEMERARO, F., GRIFFITHS, A., & CANGELOSI, A. 2023. Human–robot collaboration and machine learning: A systematic review of recent research. *Robotics and Computer-Integrated Manufacturing*, *79*, 102432. https://doi.org/10.1016/j.rcim.2022.102432

ZHENG, T., & ZHAO, L.-P. 2022. Dynamic optimization analyses and algorithm design for the parallel heat exchange system in spacecraft. *Applied Thermal Engineering*, *212*, 118519. https://doi.org/10.1016/j.applthermaleng.2022.118519

Halaman ini sengaja dikosongkan