

# Perbandingan Algoritma LSTM, GRU, dan ELM untuk Memprediksi Kadar CO di DKI Jakarta

**Marchel Yusuf Rumlawang Arpipi, Chairisni Lubis, Zyad Rusdi**

Jurusan Teknik Informatika, FTI Universitas Tarumanagara, DKI Jakarta

e-mail: [marchel.535210039@stu.untar.ac.id](mailto:marchel.535210039@stu.untar.ac.id)

## *Abstrak*

*Polusi udara adalah salah satu masalah lingkungan yang serius karena berpengaruh pada kesehatan manusia sehingga diperlukan tindakan khusus untuk memantau tingkat polusi udara secara berkala. Indeks standar pencemaran udara (ISPU) menjadi indikator pemantauan terhadap kualitas udara. Prediksi ISPU yang sesuai target dengan akurasi yang tinggi serta cepat menjadi tantangan oleh sebab perubahan pola cuaca dan emisi polutan yang bervariasi dari waktu ke waktu. Pada penelitian ini, tiga metode pembelajaran mesin digunakan yakni LSTM, GRU, dan ELM untuk memprediksi ISPU di DKI Jakarta dari tahun 2016-2020 dengan univariate data yakni karbon monoksida (CO). Algoritma Extreme Learning Machine (ELM) menunjukkan hasil yang lebih baik dan lebih cepat pada percobaan dibandingkan dengan LSTM atau GRU. Hasil yang diperoleh dapat diaplikasikan untuk memprediksi tingkat pencemaran udara karbon monoksida dengan akurat dan dapat membantu pengembangam sistem pemantauan polusi udara yang tidak hanya terbatas pada satu variabel namun dapat digunakan untuk lebih dari satu variabel agar hasil diberikan memiliki jangkauan yang luas.*

**Kata kunci**— LSTM, GRU, ELM, prediksi kualitas udara, ISPU

## *Abstract*

*Air pollution is a serious environmental issue because it affects human health, so special measures are needed to monitor air pollution levels periodically. The Air Pollution Standard Index (ISPU) is an indicator for monitoring air quality. Predicting ISPU according to the target with high accuracy and speed is a challenge due to changes in weather patterns and pollutant emissions that vary over time. In this study, three machine learning methods were used, namely LSTM, GRU, and ELM, to predict ISPU in DKI Jakarta from 2016–2020 with univariate data, namely carbon monoxide (CO). The Extreme Learning Machine (ELM) algorithm showed better and faster results in experiments compared to LSTM or GRU. The results obtained can be applied to accurately predict carbon monoxide air pollution levels and can help develop an air pollution monitoring system that is not limited to one variable but can be used for more than one variable so that the results provided have a wider range.*

**Keywords**— LSTM, GRU, ELM, air quality prediction, ISPU

---

## 1. PENDAHULUAN

Kualitas udara yang baik sangat penting untuk kesehatan manusia dan melindungi lingkungan sekitar. Pencemaran udara menjadi masalah yang mencakup skala global akibatnya kualitas hidup manusia dan makhluk lain di bumi terganggu. Indeks Standar Pencemaran Udara (ISPU) adalah salah satu metode pengukuran kualitas udara yang menggunakan variabel seperti PM10, PM2.5, O<sub>3</sub>, NO<sub>2</sub>, SO<sub>2</sub>, dan karbon monoksida (CO). Namun, penggunaan ISPU dibatasi oleh masalah lain yakni kurangnya stasiun pemantauan dan ketidakmampuan memprediksi perubahan polusi udara di masa mendatang. Dalam beberapa tahun terakhir, Machine Learning dan algoritma pembelajaran yang mendalam (Deep Learning) banyak dipakai untuk memprediksi kualitas udara dan mengidentifikasi polutan yang berbahaya bagi kehidupan di bumi.

Berdasarkan penelitian terdahulu menunjukkan bahwa penerapan model LSTM dan LSTM Bidirectional lebih bagus dibandingkan model Gated Recurrent Unit (GRU) untuk permasalahan data yang bersifat time series kualitas udara. Hal ini mengacu pada hasil performance model LSTM dan LSTM Bidirectional (LSTM-Bi) [1]. Namun dalam penelitian ini menyarankan penggunaan algoritma ELM karena lebih baik dipakai dalam penyelesaian masalah peramalan dengan waktu training yang cepat dan mudah untuk diaplikasikan pada masalah yang kompleks. Dibandingkan algoritma LSTM atau GRU, ELM memberikan nilai error yang kecil dihitung menggunakan Root Mean Square Error (RMSE) dan Mean Absolute Error (MAE) serta cepat dalam komputasinya sehingga penelitian ini menyarankan penggunaan algoritma ELM untuk prediksi polusi udara di Indonesia dengan harapan bahwa penerapannya mendapatkan hasil yang efektif terhadap satu atau lebih variabel yang diukur.

Akan tetapi disamping kelebihannya, ELM memiliki kelemahan yaitu pada ELM jumlah hidden neuron ditentukan dengan cara melakukan banyak percobaan untuk mendapatkan informasi baru, sehingga tidak dapat diketahui besaran jumlah hidden neuron yang tepat untuk menghasilkan akurasi yang baik, hal ini mengakibatkan beberapa sampel kemungkinan akan terdapat rendahnya hasil klasifikasi pada kondisi tertentu [2]. Hasil yang diperoleh dapat diaplikasikan untuk memprediksi tingkat pencemaran udara karbon monoksida dengan akurat dan dapat membantu pengembangan sistem pemantauan polusi udara yang tidak hanya terbatas pada satu variabel namun dapat digunakan untuk lebih dari satu variabel agar hasil diberikan memiliki jangkauan yang luas.

## 2. METODE PENELITIAN

Dataset yang digunakan dalam penelitian ini adalah dataset indeks pencemaran udara (ISPU) DKI Jakarta dari tahun 2016 sampai bulan agustus tahun 2020 sebagai sampel untuk percobaan dalam penelitian. Karena parameter/variabel pada penelitian ini merupakan univariate data dapat dilakukan dekomposisi musiman pada dataset sehingga menunjukkan tren yang terjadi, memberikan gambaran pola yang berulang dalam jangka waktu tertentu, dan memperlihatkan sisa yang tidak bisa dijelaskan oleh tren (residu dari deret waktu yang dianalisis). Sumber dataset diperoleh dari website Jakarta Open Data: <https://data.jakarta.go.id/group/lingkungan-hidup?q=Indeks+Standar+Pencemaran+Udara&sort=1>.

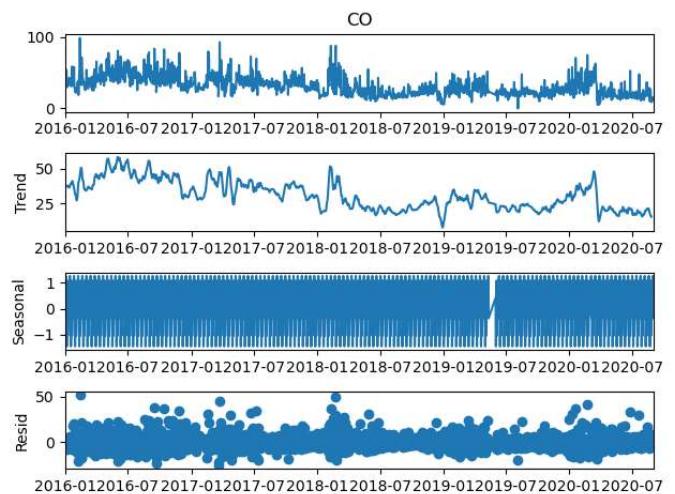
---

## 2.1 Pra-pemrosesan Data

Data yang semula diambil telah disatukan ke dalam file berformat *Comma Separated Value (CSV)* dengan jumlah sampel sebanyak 1686, memiliki 10 variabel tetapi dilakukan drop pada kolom dataset menyisakan variabel “tanggal” dan “co” yang akan diprediksi. Namun, dilakukan perubahan nama dari “tanggal” menjadi “Date” dan “co” menjadi “CO” untuk menyesuaikan penulisan kode pada jupyter notebook. Dilakukan juga pemeriksaan missing value (Gambar 1) dan melakukan handling missing value agar dapat melihat dekomposisi musiman dari dataset (Gambar 2).

Date,CO
2016-01-01,53
2016-01-02,29
2016-01-03,33
2016-01-04,36
2016-01-05,54
2020-02-11,      ←
Tidak ada nilai disini

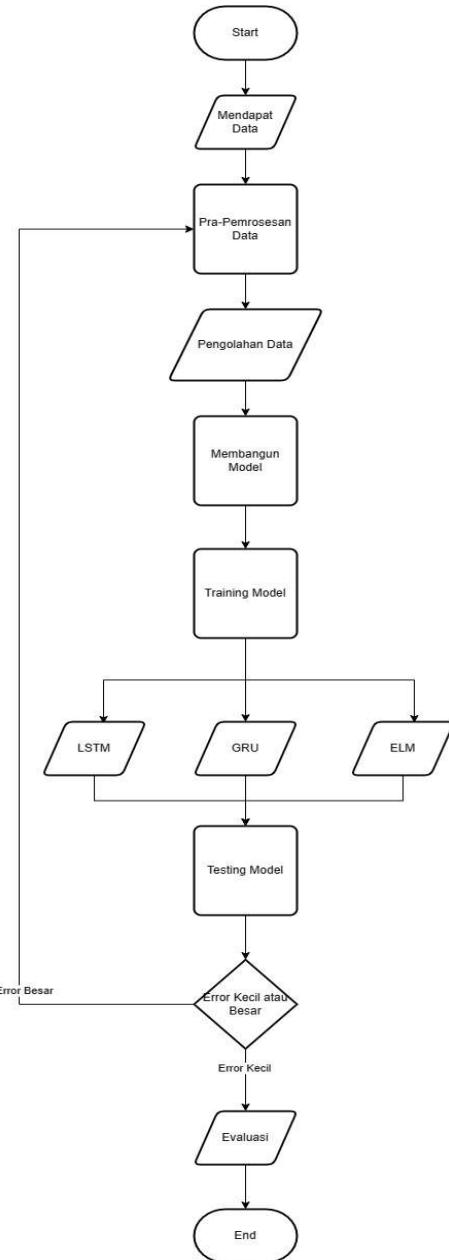
Gambar 1 Missing Value pada dataset



Gambar 2 Dekomposisi Seasonal

## 2.2 Pengolahan Data

Ditahap ini dataset dibagi menjadi training dataset dan testing dataset. Pembagiannya dimulai dari tahun 2016-01-01 sampai 2020-01-01 (1442 sampel) adalah data training dan dari tahun 2020-01-02 sampai 2020-08-31 (244 sampel). Pada bagian ini pula training dan test dataset akan diskalakan rentang nilainya dengan modul `MinMaxScaler()` agar generator time series dapat dibangun sebelum algoritma dipakai untuk melatih mesin dalam melakukan prediksi. Alur penelitian dapat dilihat pada Gambar 3 dibawah ini.



Gambar 3 Alur Penelitian

a. *Training Model*

Proses training dilakukan terlebih dahulu pada model LSTM dan GRU. Kedua model dibangun sama persis yaitu dengan membuat objek *Sequential()* yang kemudian diisi dengan 64 atau 128 neuron yang diuji secara bergantian dengan menggunakan *activation='relu'* dan *input\_shape=(n\_input, n\_features)* dimana *n\_input=6* (jumlah yang akan diprediksi dalam bulan) adalah dimensi input dan *n\_features=1* adalah dimensi fitur yang dimiliki yang sudah dibuat sebelumnya pada saat generator time series didefinisikan. Ditambahkan juga layer *Dense(1)* ke dalam model untuk memetakan input ke output yang terhubung sepenuhnya

---

antar layer dengan *optimizer*=’SGD’ sedangkan kinerja model dinilai menggunakan fungsi *loss*=’mse’. Menampilkan ringkasan dari arsitektur model yang sudah dibangun dengan *model.summary()*. Kemudian model dilatih dengan iterasi *epochs* sebanyak 50 kali.

Model ELM dilatih menggunakan modul *ELMRegressor* dari *pyrcn.extreme\_learning\_machine* dengan *regressor=skRidge()* sebagai parameter algoritma *Ridge Regression* dan mengubah dimensi data input menjadi matriks 2 dimensi 1 kolom melalui *X\_train.reshape(-1,1)* serta kinerjanya dinilai dengan *Root Mean Square Error (RMSE)* dan *Mean Absolute Error (MAE)* untuk melihat perbandingan error paling kecil.

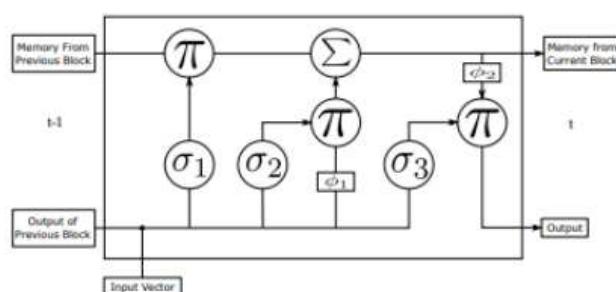
#### b. Testing Model

Ketiga model yang sudah ditraining kemudian ditest untuk melihat efektifitas model yang sudah dibangun.

### 2.3 Pengacuan Pustaka

#### a. Long Short-Term Memory

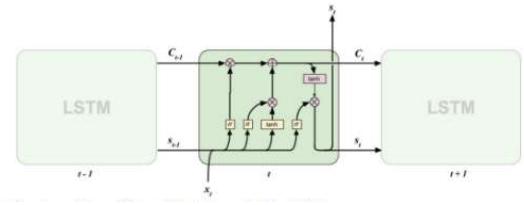
Algoritma *Long Short-Term Memory (LSTM)* merupakan sebuah evolusi dari arsitektur RNN yang memungkinkan jaringan mempertahankan ketergantungan jangka panjang antara data pada waktu tertentu dari banyak langkah waktu sebelumnya, dimana pertama kali diperkenalkan oleh Hochreiter & Schmidhuber (1997) [3]. LSTM memiliki bentuk rantai modul berulang dari jaringan saraf tiruan, dengan setiap modul mencakup tiga gerbang (gate) kontrol, yaitu gerbang forget (forget gate), gerbang masukan (input gate), dan gerbang keluaran (output gate). Setiap gerbang terdiri atas lapisan jaringan saraf sigmoid dan operasi perkalian pointwise. Hasil dari keluaran pada lapisan sigmoid merupakan angka dalam interval [0, 1], mewakili sebagian informasi masukan yang harus dilewati [4].



Gambar 4 Perulangan Pada LSTM

Simbol  $\pi$  dan  $\Sigma$  mewakili elemen perkalian bijak dan penjumlahan masing-masing. Operasi penggabungan diwakili oleh simbol ( $\bullet$ ) poin. Itu komponen dasar LSTM adalah status sel, sebuah baris yang berjalan dari memori dari blok sebelumnya ( $St - 1$ ) ke memori blok saat ini ( $St$ ). Ini memungkinkan informasi mengalir lurus ke bawah. Jaringan dapat menentukan

jumlah informasi sebelumnya mengalir. Itu dikendalikan melalui lapisan pertama ( $\sigma_1$ ) [5].. Gambar 5 menunjukkan proses komputasi pada model LSTM.



Sumber: Zhao, Chen, Wu, Chen, & Liu, 2017

Gambar 5 Proses Looping pada Algoritma LSTM

Proses komputasi pada LSTM dilakukan dengan tahapan berikut : Nilai dari suatu input hanya dapat disimpan ke dalam cell state hanya jika diijinkan oleh input gate. Hasil hitungan nilai input gate dan kandidat dari cell state diproses dengan **persamaan (1) dan (2)** .

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$

$i_t$  adalah nilai input gate,  $W_i$  adalah bobot nilai input waktu ke  $t$ ,  $x_t$  adalah input pada waktu ke  $t$ ,  $U_i$  adalah bobot nilai output waktu ke  $t-1$ ,  $h_{t-1}$  adalah output waktu ke  $t-1$ ,  $b_i$  adalah bias di input gate, dan  $\sigma$  adalah fungsi sigmoid.

$$C_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (2)$$

$C_t$  adalah value candidate cell state,  $W_c$  adalah bobot nilai input di cell ke  $c$ ,  $x_t$  adalah input pada waktu ke  $t$ ,  $U_c$  adalah bobot output dari cell ke  $t-1$ ,  $h_{t-1}$  adalah output di cell ke  $t-1$ ,  $b_c$  adalah bias di cell ke  $c$ ,  $\tanh$  adalah fungsi hyperbolic tangent.

Selanjutnya nilai forget gate dihitung dengan persamaan (3).

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (3)$$

$f_t$  adalah nilai dari forget gate,  $W_f$  adalah bobot nilai input di waktu ke  $t$ ,  $x_t$  adalah input pada waktu ke  $t$ ,  $U_f$  adalah bobot nilai output dari waktu ke  $t-1$ ,  $h_{t-1}$  adalah output dari waktu ke  $t-1$ ,  $b_f$  adalah bias pada forget gate, dan  $\sigma$  adalah fungsi sigmoid.

Selanjutnya perhitungan memory cell state menggunakan persamaan (4)

$$C_t = i_t \cdot C_t + f_t \cdot C_{t-1} \quad (4)$$

$C_t$  adalah nilai memory cell state,

$i_t$  adalah nilai input gate,

$f_t$  adalah forget gate, dan

$C_{t-1}$  adalah memory cell state pada cell sebelumnya.

Selanjutnya setelah memperoleh memory cell state baru, output gate dapat dihitung menggunakan persamaan (5).

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (5)$$

$o_t$  adalah nilai dari output gate,

$W_o$  adalah bobot nilai input pada waktu ke t,

$x_t$  adalah input pada waktu ke t,

$U_o$  adalah bobot untuk output dari waktu ke  $t-1$ ,

$h_{t-1}$  adalah output dari waktu ke  $t-1$ ,

$b_o$  adalah bias pada output gate, dan

$\sigma$  adalah fungsi sigmoid.

Nilai output final dihitung dengan persamaan (6).

$$h_t = o_t \cdot \tanh(C_t) \quad (6)$$

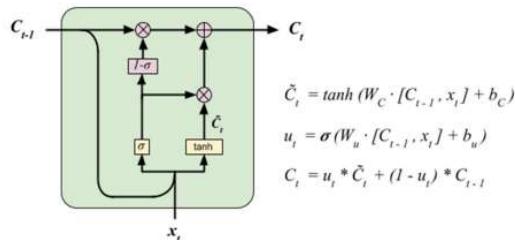
$h_t$  adalah output final,

$o_t$  adalah output gate, dan

$C_t$  adalah memory cell state yang baru

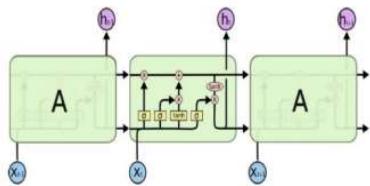
$\tanh$  adalah fungsi hyperbolic tangent [1].

### b. Gated Recurrent Unit



Gambar 6 Arsitektur GRU

Algoritma *Gated Recurrent Unit (GRU)* merupakan salah satu varian yang populer dari sekian banyaknya varian LSTM. Keutamaan GRU adalah komputasinya lebih sederhana dari LSTM, namun mempunyai akurasi yang setara dan masih cukup efektif untuk menghindari permasalahan gradien yang menghilang [6].

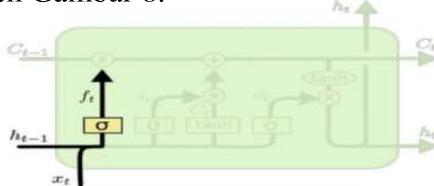


Gambar 7 Proses Looping pada Algoritma GRU

Persamaan metode GRU menurut diuraikan pada persamaan 1.

$$\begin{aligned}
 &= (\quad \cdot [h_{t-1}, \quad] + \quad [h_{t-1}, \quad] + \\
 &\quad (1) \\
 &= (\quad \cdot [h_{t-1}, \quad] + \quad) \\
 &= h(\quad \cdot [h_{t-1}, \quad] + \quad) \\
 &= \quad * \quad -1 + \quad * \quad \\
 &= (\quad \cdot [h_{t-1}, \quad] + \quad) h = \quad * \tanh(\quad) . \\
 \\
 &= (\quad \cdot [h_{t-1}, \quad] + \\
 &\quad (2) \\
 &= \text{forget gate} \\
 &= \text{fungsi sigmoid} \\
 &= \text{nilai input pada orde ket} \\
 &= \text{nilai bias pada forget gate} \\
 &W_f = \text{nilai weight untuk forget gate [7].}
 \end{aligned}$$

Ada 3 jenis gate pada algoritma GRU, yakni *forgot gate* (gate penentu data apa yang harus dihapus dari cell), *input gate* (gate penentu nilai input yang akan diupdate pada bagian state memori), dan *output gate* (gate yang menentukan hasil akhir berdasarkan nilai input dan memori cell). Langkah pertamanya adalah GRU menentukan data apa yang perlu dihapus dari state memori. Keputusan ini dirancang oleh lapisan *sigmoid* yang disebut “*forgot gate layer*”. Lapisan ini mengambil  $h_{t-1}$  dan  $x_t$  sebagai input dan mengubah hasilnya menjadi angka 0 atau 1 dalam state cell  $C_{t-1}$  seperti yang ditunjukkan oleh Gambar 8.



Gambar 8 First step GRU pada forgot gate layer

### c. Extreme Machine Learning

Algoritma *Extreme Learning Machine (ELM)* merupakan jaringan saraf tiruan feed-forward dengan satu hidden layer atau lebih dikenal dengan

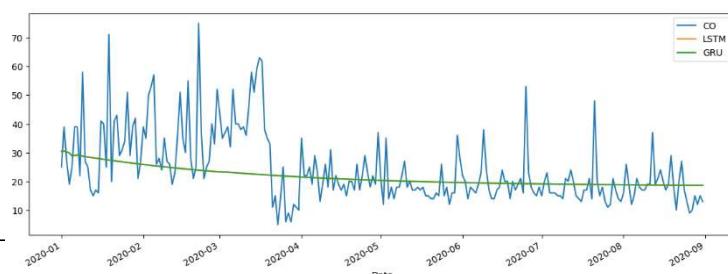
istilah single hidden layer *feed-forward neural network* [8] dan hanya memiliki satu hidden layer yang biasanya disebut dengan *Single-hidden Layer Feed-forward Neural Networks* (SLFNs). Dalam ELM, bobot awal yang berupa bobot input nodes ke hidden nodes dan bobot bias ke hidden nodes dipilih secara sembarang. Moore-Penrose Generalized Inverse kemudian digunakan untuk melakukan perhitungan analitik untuk menentukan bobot akhir (bobot hidden nodes ke output) [9].

Extreme learning machine mempunyai karakteristik yang memukau dan substansial, namun berbeda dengan algoritma pembelajaran yang berdasarkan atas pada gradien yang populer untuk jaringan saraf feed-forward. Karakteristik yang dimaksudkan sebagai berikut:

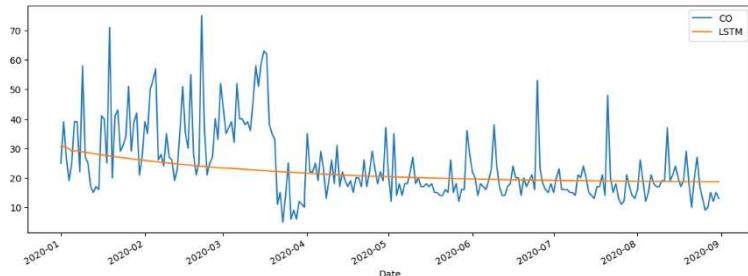
- a. Kecepatan mengkaji pada extreme learning machine tergolong sangat cepat. Dalam simulasi yang diperoleh dari hasil laporan dalam literatur, fase pembelajaran pada extreme learning machine dapat dituntaskan cukup hitungan detik untuk aplikasi yang jumlahnya tergolong banyak.
- b. Extreme Learning Machine mempunyai kinerja generalisasi yang jauh lebih baik dibandingkan dengan pembelajaran berbasiskan pada gradien, misalkan seperti dalam kebanyakan kasus yaitu backpropagation.
- c. Extreme Learning Machine memiliki kecenderungan dalam pencapaian penyelesaian sederhana tanpa memiliki efek masalah yang sifatnya sepele. Algoritma pembelajaran extreme learning machine terlihat jauh lebih sederhana dibandingkan dengan algoritma pembelajaran jaringan saraf feed-forward pada kebanyakannya. Namun berbeda terhadap algoritma pembelajaran berbasiskan pada gradien yang cuma bekerja untuk fungsi aktivasi terdiferensiasi, algoritma extreme learning machine ini mampu diterapkan kedalam SLFNs guna berfungsi melatih SLFNs, dengan jumlah fungsi aktivasi yang banyak dan tidak mempunyai diferensiasi. [10].

### 3. HASIL DAN PEMBAHASAN

Perbandingan performa prediksi terhadap karbon monoksida yang paling mendekati nilai sebenarnya (*ground truth*) dari antara ketiga algoritma yang diuji bisa terlihat pada grafik Gambar 9 dan 10.

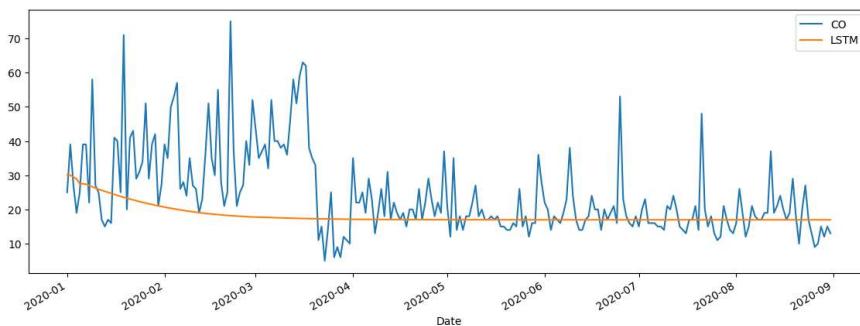


Gambar 9 Prediksi dengan algoritma LSTM (64 neuron).

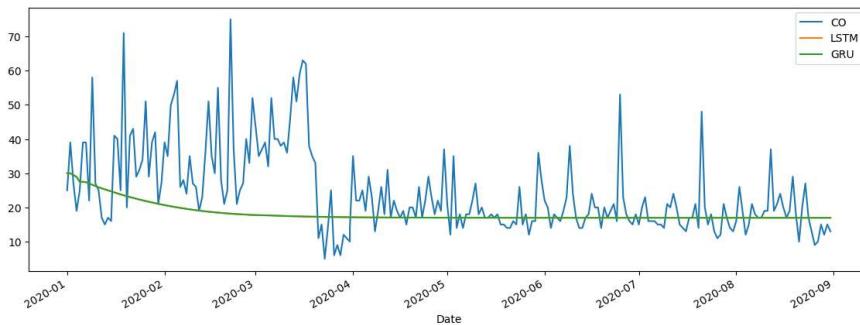


Gambar 10 Prediksi dengan algoritma GRU (64 neuron)

Dapat terlihat bahwa hasil prediksi antara algoritma LSTM dan GRU adalah sama persis. Hal ini mungkin terjadi karena parameter/atau variabel yang digunakan pada tiap algoritma adalah sama sehingga hasil prediksi terlihat mirip atau mungkin karena kesalahan dalam proses pelatihan model yang menyebabkan hasilnya menjadi sama persis akibat spesifikasi laptop yang mengalami kehilangan performa pada saat proses. Meskipun jarang terjadi, spesifikasi laptop yang mumpuni justru memungkinkan algoritma memberikan hasil yang maksimal dan sesuai yang diharapkan. Berikut tampilan hasil prediksi karbon monoksida pada algoritma LSTM dan GRU:



Gambar 11 Prediksi dengan algoritma LSTM (128 neuron).

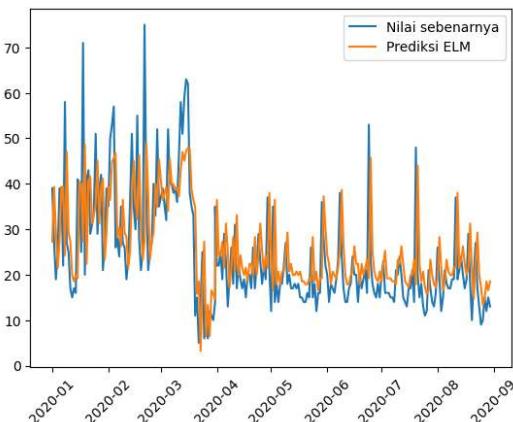


Gambar 12 Prediksi dengan algoritma GRU (128 neuron).

Gambar 11 dan 12 menunjukkan performa yang cukup baik walaupun masih menampilkan hasil yang sama persis antara kedua algoritma. Meskipun tidak berbeda

---

jauh dengan kedua algoritma lain yang diuji ELM (Gambar 13) memberikan keunggulan dengan evaluasi error yang relatif lebih kecil yang membuktikan bahwa algoritma ELM efektif dalam memprediksi kualitas udara sehingga dapat menjadi acuan apabila dalam melakukan pemantauan/pengukuran pada ISPU. Error yang dihasilkan masing-masing model dapat dilihat pada Tabel 1 dan 2.



Gambar 13 Prediksi dengan algoritma ELM

Tabel 1 Perbandigan Prediksi pada Algoritma LSTM, GRU, dan ELM (64 Neuron)

Evaluasi Error (64 neuron)	LSTM	GRU	ELM
MAE	7.656798752737634	7.656798752737634	7.692530378976657
RMSE	11.568377681030887	11.568377681030887	10.632805597373052

Tabel 2 Perbandigan Prediksi pada Algoritma LSTM, GRU, dan ELM (128 Neuron)

Evaluasi Error (128 neuron)	LSTM	GRU	ELM
MAE	8.496233202854302	8.496233202854302	7.690751248550094
RMSE	13.335054272315594	13.335054272315594	10.632131346576447

#### 4. KESIMPULAN

Dengan mengacu pada hasil dan pembahasan dapat di lihat bahwa ketiga algoritma mampu memberikan hasil yang terbaik berdasarkan pembelajaran yang telah diberikan. Walaupun untuk kasus pada algoritma LSTM dan GRU memiliki hasil yang sama persis tetapi bila jumlah neuron ditingkatkan, misal dari 64 ke 128 atau 256 ke 512 hasil yang terlihat cukup memuaskan dengan evaluasi error dari *Mean Absolute Error* dan *Root Mean Square Error* yang mengecil ditiap kali jumlah neuron ditetapkan.

*Extreme Learning Machine* memberikan keunggulan dengan lebih cepat dalam proses komputasi serta hasil yang lebih maksimal. Hal ini dapat memungkinkan penerapan ELM yang lebih luas serta pengembangannya untuk melakukan prediksi

terhadap *Indeks Standar Pencemaran Udara* (ISPU) dalam membantu mencegah meningkatnya polusi udara agar upaya penyelesaian masalah melalui penggunaan teknologi di Indonesia semakin luas..

## 5. SARAN

Untuk saran dalam penelitian selanjutnya adalah dengan penggunaan model yang sama tetapi data yang akan diuji berupa *multivariate* data dengan sedikit perbedaan pada pengujian model dengan meningkatkan performa model dalam melakukan prediksi.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada keluarga yang telah memberi dukungan doa dan semangat terhadap penelitian ini.

## DAFTAR PUSTAKA

- [1] Y. Karyadi and H. Santoso, “Prediksi Kualitas Udara Dengan Metoda LSTM, Bidirectional LSTM, dan GRU”.
- [2] A. Basuki and F. Abdurrachman Bachtiar, “METODE DETEKSI INTRUSI MENGGUNAKAN ALGORITME EXTREME LEARNING MACHINE DENGAN CORRELATION-BASED FEATURE SELECTION,” vol. 8, no. 1, pp. 103–110, 2021, doi: 10.25126/jtiik.202183358.
- [3] Khalis Sofi, Aswan Supriyadi Sunge, Sasmithoh Rahmad Riady, and Antika Zahrotul Kamalia, “PERBANDINGAN ALGORITMA LINEAR REGRESSION, LSTM, DAN GRU DALAM MEMPREDIKSI HARGA SAHAM DENGAN MODEL TIME SERIES,” *SEMINASTIKA*, vol. 3, no. 1, pp. 39–46, Nov. 2021, doi: 10.47002/seminastika.v3i1.275.
- [4] W. Hastomo, N. Aini, A. Satyo, B. Karno, and L. M. R. Rere, “Metode Pembelajaran Mesin untuk Memprediksi Emisi Manure Management,” 2022.
- [5] A. A. Ningrum *et al.*, “ALGORITMA DEEP LEARNING-LSTM UNTUK MEMPREDIKSI UMUR TRANSFORMATOR,” vol. 8, no. 3, pp. 539–548, 2021, doi: 10.25126/jtiik.202184587.
- [6] F. Hamdi Bahar, N. Indah Sari, and dan Armin Lawi, “Konferensi Nasional Ilmu Komputer (KONIK) 2021 Klasifikasi Suara Kucing dan Anjing Menggunakan LSTM-GRU dan ANN-BP”, [Online]. Available: <https://www.kaggle.com/mmoreaux/audio-cats-and-dogs>.
- [7] I. Alkahfi and K. Chiuloto, *Prosiding SNASTIKOM: Seminar Nasional Teknologi Informasi & Komunikasi Penerapan Model Gated Recurrent Unit Pada Masa Pandemi Covid-19 Dalam Melakukan Prediksi Harga Emas Dengan Menggunakan Model Pengukuran Mean Square Error*.
- [8] J. J. Pangaribuan, “MENDIAGNOSIS PENYAKIT DIABETES MELITUS DENGAN MENGGUNAKAN METODE EXTREME LEARNING MACHINE,” 2016.
- [9] A. Hartono and A. Muliani Harahap, “Volume 6 ; Nomor 1,” *Januari*, pp. 127–134, 2023, [Online]. Available: <https://ojs.trigunadharma.ac.id/index.php/jsk/index>
- [10] J. J. Pangaribuan, C. Tedja, and S. Wibowo, “PERBANDINGAN METODE ALGORITMA C4.5 DAN EXTREME LEARNING MACHINE UNTUK MENDIAGNOSIS PENYAKIT JANTUNG KORONER,” 2019.