Pengembangan *Dead Reckoning* Berbasis *Multilayer Perceptron* (MLP) yang Diimplementasikan di *Raspberry Pi Pico*

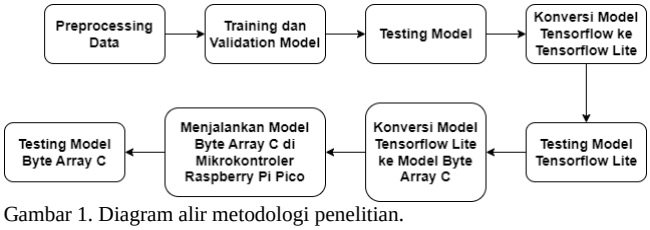
Gagah Putra Haryo Satrio, Dion Hayu Fandiantoro, dan Hany Boedinugroho
Departemen Teknik Komputer, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: dion@its.ac.id

***Abstrak*—Penentuan posisi sangatlah penting untuk pergerakan robot dalam ruangan tertutup (*indoor*). Salah satu penentuan posisi yang dapat digunakan adalah *dead reckoning*. *Dead reckoning* adalah sistem navigasi berdasarkan lokasi awal relatif yang diketahui kemudian secara bertahap mengintegrasikan jarak yang ditempuh dan arah perjalanan untuk mengetahui titik lokasi selanjutnya. Karena termasuk sistem navigasi inersia yang menggunakan *Inertial Measurement Unit* (IMU), untuk melakukan prediksi posisi maka dilakukan perhitungan gabungan dari sensor giroskop, akseleromoter, dan magnetometer. Pada penelitian ini, akan menerapkan model *deep learning*, yaitu *Multilayer Perceptron* (MLP). Sehingga dapat memberikan prediksi posisi berdasarkan data dari sensor IMU. Kemudian dari model yang telah dibuat akan dimasukkan ke mikrokontroler. Penggunaan mikrokontroler karena ukurannya yang kecil sehingga dapat diterapkan di robot kecil. Berbagai macam mikrokontroler saat ini, diantaranya Arduino nano, STM32F103C8T6, dan *Raspberry Pi Pico*. Pada penelitian ini memilih menggunakan mikrokontroler *Raspberry Pi Pico* karena memiliki beberapa keunggulan dari segi memori flash, SRAM dan kecepatan prosesornya dibandingkan mikrokontroler lainnya. Hal ini untuk menunjang penggunaan model *Multilayer Perceptron* ke dalam mikrokontroler. Model *Multilayer Perceptron* yang dibuat akan bervariasi untuk menemukan model yang ideal untuk sistem *Dead reckoning*. Berdasarkan hasil dari pengujian model *Multilayer Perceptron* ke perangkat mikrokontroler *Raspberry Pi Pico*, maka dihasilkan model yang ideal, yaitu model *Multilayer Perceptron* tanpa berbasis *time series* dengan 3 *hidden* layer dengan nodes masing-masing sebesar 250, 125, dan 30. Model ini dipilih karena menghasilkan nilai metrik *R-Square* sebesar 0,7452684 pada pengujian di perangkat mikrokontroler *Raspberry Pi Pico*.**

***Kata Kunci*—*Dead Reckoning, Inertial Measurement Unit, Multilayer Perceptron, Raspberry Pi Pico*.**

I. PENDAHULUAN

SISTEM navigasi atau *dead reckoning* adalah sistem navigasi yang berbasis pada lokasi awal yang diketahui, yang secara bertahap mengintegrasikan jarak yang ditempuh dan arah perjalanan untuk menentukan titik lokasi berikutnya [1]. Untuk membuat sistem *dead reckoning* yang memanfaatkan sensor IMU, telah dikembangkan algoritma sensor *fusion* (juga disebut sebagai filter) untuk memberikan estimasi orientasi yang akurat dan dapat diandalkan. Kalman Filter (KF) adalah filter yang paling umum digunakan yang memanfaatkan pembacaan *accelerometer* dan *magnetometer* untuk memperbaiki estimasi laju perubahan orientasi yang dihitung melalui integrasi *strap-down* dari pembacaan giroskop [2]. Untuk membuat sistem *dead reckoning* yang memanfaatkan sensor IMU, dapat juga menggunakan Filter Madgwick. Filter Madgwick adalah algoritma *Attitude and Heading Reference System* (AHRS) yang menggunakan pengukuran percepatan dan medan magnetik untuk mengoreksi *drift* giroskop [3]. Metode sensor *fusion* ini bergantung pada dua estimasi orientasi: kecepatan sudut (giroskop) yang digabungkan dengan estimasi orientasi sebelumnya, dan estimasi orientasi berdasarkan algoritma gradien penurunan untuk mengukur orientasi relatif terhadap gravitasi [3].



Gambar 1. Diagram alir metodologi penelitian.

Kemajuan teknologi yang pesat memberikan peluang baru untuk menggunakan pendekatan baru dalam menggunakan sistem *Dead reckoning*. Penggunaan model *deep learning* telah banyak diimplementasikan di berbagai bidang. Penggunaan *deep learning* dapat menjadi salah satu cara untuk memprediksi posisi dari data sensor IMU. Model *deep learning* yang dipilih adalah *Multilayer Perceptron* (MLP) dengan variasi pada lapisan tersembunyi untuk mencari model yang paling cocok.

II. METODOLOGI PENELITIAN

A. Metodologi

Diagram alir metodologi penelitian disajikan pada Gambar 1. Terdapat 8 bagian yang akan dijelaskan dari penelitian ini.

B. Pengolahan Data

Dataset yang digunakan dalam penelitian ini diperoleh dari rekaman sensor GPS dan IMU yang tersedia di situs web Zenodo dengan nama "Realistic GPS and IMU data". Dataset ini terdiri dari 143.318 titik data. Dalam dataset ini, terdapat 15 variabel per titik data yang akan digunakan dalam pembuatan model MLP seperti pada Tabel 1.

Untuk menggunakan data dalam pembuatan model *deep learning*, nilai-nilai dalam setiap dataset di atas perlu dinormalisasi terlebih dahulu. Dalam penelitian ini akan membuat dua buah macam model *Multilayer Perceptron*, tanpa berbasis *time series* dan berbasis *time series.* Fungsi MinMax Scaler dari pustaka *scikit-learn* dapat digunakan untuk melakukan normalisasi data untuk model tanpa berbasis *time series* maka rentang data dibuat menjadi -1 dan 1, sedangkan untuk model berbasis *time series* maka rentang data dibuat menjadi 0 dan 1. Normalisasi data bertujuan untuk menghilangkan perbedaan skala dan meningkatkan kecepatan konvergensi algoritma *machine learning* yang digunakan. Dataset yang telah dinormalisasi kemudian dapat digunakan

Tabel 1.
Parameter Dataset dari Situs Web Zenodo

No	Keterangan Parameter
1	Arah sudut
2	Sudut rotasi per menit
3	Kecepatan horizontal dalam kilometer per jam
4	Lintang (*latitude*)
5	Bujur (*longitude*)
6	Sudut *roll*
7	Sudut *pitch*
8	Sudut *yaw*
9	Laju rotasi *roll* dalam radian per menit
10	Laju rotasi *pitch* dalam radian per menit
11	Laju rotasi *yaw* dalam radian per menit
12	Percepatan akselerasi sumbu-x (m/s²)
13	Percepatan akselerasi sumbu-y (m/s²)
14	Percepatan akselerasi sumbu-z (m/s²)
15	Laju perubahan sudut *yaw* dalam derajat per menit

dalam pembuatan model *deep learning*.

Terdapat variabel-variabel, yaitu "lat degrees" dan "lon degrees" yang perlu diubah atau diubah menjadi satuan meter. Jadi, parameter *latitude* akan menjadi parameter X, dan parameter *longitude* akan menjadi parameter Y. Parameter X dan Y ini akan mewakili posisi relatif pada sumbu x dan y, masing-masing. Untuk melakukan konversi ini, digunakan rumus Haversine. Rumus Haversine adalah persamaan yang umum digunakan dalam navigasi untuk menghitung jarak lingkaran besar antara dua titik di permukaan bumi berdasarkan koordinat garis bujur dan garis lintangnya. Rumus Haversine adalah sebagai berikut:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right)$$
$$c = 2\cdot\tan^{-1}\left(\sqrt{\frac{a}{1-a}}\right) \quad (1)$$
$$d = R\cdot c$$

dengan ϕ adalah lintang (*latitude*), λ adalah bujur (*longitude*), dan R adalah radius bumu (R = 6.371 km).

C. Training dan Validation Model

Untuk model MLP tanpa berbasis *time series* akan di-*training* menggunakan *activation function* tanh dengan variasi jumlah *hidden layers* dan jumlah *nodes.* Pada pembuatan model ini menggunakan jumlah *epoch* sebesar 25 dan jumlah *batch size* sebesar 50. Dari hasil model *training* yang dibuat kemudian divalidasi menggunakan dataset validasi.

Untuk model MLP berbasis *time series* akan di-*training* menggunakan *activation function linear* dengan variasi jumlah *hidden layers*, jumlah *nodes*, dan panjang *window*. Pada pembuatan model ini menggunakan jumlah *epoch* sebesar 100 dan jumlah *batch size* sebesar 100. Dari hasil model *training* yang dibuat kemudian divalidasi menggunakan dataset validasi.

Parameter *input* yang digunakan pada model MLP adalah 10 parameter dari 15 parameter yang digunakan seperti pada Tabel 2. Parameter *output* yang digunakan pada model *Multilayer Perceptron* adalah 5 parameter dari 15 parameter yang digunakan Tabel 3. Pada model yang telah dibuat akan dievaluasi menggunakan dataset *testing* yang sudah disiapkan. Hasil dari *testing* model ini akan menghasilkan nilai metrik *R-Square*, *Mean Absolute Error* (MAE), dan *Mean Squared Error* (MSE).

D. Konversi Model Tensorflow ke Model Tensorflow Lite

Dari model *Tensorflow* yang telah dibuat, maka selanjutnya akan dilakukan konversi dari model *Tensorflow* ke model *Tensorflow Lite*. Hal ini dilakukan karena model *Tensorflow* tidak dapat diterapkan secara langsung ke perangkat mikrokontroler. Untuk dapat menerapkan model MLP di perangkat mikrokontroler maka perlu dikonversi menjadi model *Tensorflow Lite*.

Tabel 2.
Parameter *Input* Model

No	Keterangan Parameter
1	Sudut *roll*
2	Sudut *pitch*
3	Sudut *yaw*
4	Laju rotasi *roll* dalam radian per menit
5	Laju rotasi *pitch* dalam radian per menit
6	Laju rotasi *yaw* dalam radian per menit
7	Percepatan akselerasi sumbu-x (m/s²)
8	Percepatan akselerasi sumbu-y (m/s²)
9	Percepatan akselerasi sumbu-z (m/s²)
10	Laju perubahan sudut *yaw* dalam derajat per menit

Tabel 3.
Parameter *output* model

No	Keterangan parameter
1	Arah sudut
2	Sudut rotasi per menit
3	Kecepatan horizontal dalam kilometer per jam
4	X (posisi relatif pada sumbu-x)
5	Y (posisi relatif pada sumbu-y)

E. Testing Model Tensorflow Lite

Model-model *Tensorflow* yang telah dikonversi menjadi model *Tensorflow Lite* perlu dilakukan evaluasi model. Hal ini dilakukan untuk meninjau kinerja model *Tensorflow Lite*, apakah mengalami penurunan akurasi model. Untuk evaluasi model *Tensorflow Lite* dilakukan di perangkat laptop dengan menggunakan dataset *testing* yang sudah ada sebelumnya. Penggunaan dataset *testing* ini untuk menguji model *Tensorflow Lite* terhadap data baru. Dari evaluasi model *Tensorflow Lite* ini akan diperoleh nilai metrik *R-Square*, MSE, dan MAE.

F. Konversi Model Tensorflow Lite ke Model Byte Array C

Model-model *Tensorflow Lite* perlu dilakukan konversi kembali ke model *byte array* C. Hal ini dilakukan karena lingkungan mikrokontroler pada umumnya tidak memiliki sistem operasi yang mendukung penerapan *Tensorflow Lite* secara langsung. Konversi model *Tensorflow Lite* ke *byte array* C menggunakan perintah unix "xxd -i". Dalam proses konversi ini, tidak ada manipulasi atau pengolahan data model yang terjadi. Konversi hanya mengubah cara model tersebut direpresentasikan dalam bentuk *byte array* yang dapat digunakan dalam bahasa C. Pada konversi ke *byte array* C ini ukuran model akan meningkat.

G. Menjalankan Model Byte Array C di Mikrokontroler Raspberry Pi Pico

Setelah model *deep learning* menjadi *byte array* C, maka model akan dimasukkan ke dalam program mikrokontroler. Program IDE yang digunakan untuk memprogram mikrokontroler *Raspberry Pi Pico*, yaitu Arduino IDE. Dibutuhkan juga pustaka tambahan untuk dapat menggunakan model *byte array* C di mikrokontroler *Raspberry Pi Pico*, yaitu Arduino_*Tensorflow Lite*. Secara khusus, penulis tidak menemukan pustaka *Tensorflow Lite* untuk mikrokontroler Pi Pico yang dapat dijalankan di program Arduino IDE. Sehingga pustaka Arduino_*Tensorflow Lite* perlu sedikit penyesuaian

Tabel 4.
Model MLP yang Dibuat

Nama Model	Jumlah *Hidden Layer*	*Nodes Hidden Layer* Pertama	*Nodes Hidden Layer* Kedua	*Nodes Hidden Layer* Ketiga	*Nodes Hidden Layer* Keempat
Model I	2	100	80	-	-
Model II	2	100	100	-	-
Model III	3	100	75	25	-
Model IV	3	200	100	75	-
Model V	3	250	125	30	-
Model VI	4	250	125	30	5

Tabel 5.
Nilai Akurasi dan *Loss* Model yang Dibuat

Nama Model	Nilai Akurasi	Nilai *Loss*
Model I	82,97%	1,73%
Model II	81,61%	1,84%
Model III	84,68%	1,16%
Model IV	84,48%	0,97%
Model V	85,28%	1,04%
Model VI	85,64%	1,00%

Tabel 6.
Nilai Metrik *R-Square*, MSE, dan MAE

Nama Model	Nilai *R-Square*	Nilai MSE	Nilai MAE
Model I	0,64481797	0,01734278	0,06523952
Model II	0,36315539	0,01844258	0,06850198
Model III	0,83365380	0,01158553	0,05076780
Model IV	0,77585215	0,00966025	0,04840861
Model V	0,86989581	0,01042153	0,04816222
Model VI	0,87422422	0,01004757	0,04655807

Tabel 7.
Model *Multilayer Perceptron* berbsasis *time series* yang dibuat

Nama Model	Jumlah *Hidden Layer*	Panjang *Window*	*Nodes Hidden Layer* Pertama	*Nodes Hidden Layer* Kedua	*Nodes Hidden Layer* Ketiga
Model I	2	5	100	80	-
Model II	2	5	100	100	-
Model III	3	5	100	75	25
Model IV	2	3	100	80	-
Model V	2	3	100	100	-
Model VI	3	3	100	75	25
Model VII	2	1	100	80	-
Model VIII	2	1	100	100	-
Model IX	3	1	100	75	25

agar dapat dijalankan di mikrokontroler Pi Pico. Penyesuaian ini dilakukan karena penulis mengalami hambatan dalam meng-*compile* model *byte array* C ke mikrokontroler *Raspberry Pi Pico*. Secara *default* pustaka Arduino_ *Tensorflow Lite* ditujukan untuk mikrokontroler Arduino Nano 33 BLE.

H. Testing Model Byte Array C

Model-model *Tensorflow Lite* yang telah dikonversi menjadi model *byte array* C perlu dilakukan evaluasi model. Hal ini dilakukan untuk meninjau kinerja model *byte array* C, apakah mengalami penurunan akurasi model. Untuk evaluasi model *byte array* C dilakukan dengan mengirim data secara serial dari laptop ke mikrokontroler Pi Pico, kemudian hasil yang diperoleh dari mikrokontroler Pi Pico dikirim kembali ke laptop secara serial dan disimpan ke dalam *file* csv. Data yang digunakan, yaitu dataset *testing* yang ada sebelumya. Penggunaan dataset *testing* ini untuk menguji model *Tensorflow Lite* terhadap data baru. Dari evaluasi model *Tensorflow Lite* ini akan diperoleh nilai metrik *R-Square*, MSE, dan MAE.

III. HASIL PENGUJIAN

A. Model Tensorflow Tanpa Berbasis Time Series

Untuk model *Multilayer Perceptron* tanpa berbasis *time series* yang dibuat ada enam model. Model-model tersebut pada Tabel 4. Dari model *Tensorflow Multilayer Perceptron* yang di-*training* mendapatkan nilai akurasi dan *loss* model seperti pada Tabel 5. Dari model yang dibuat di atas kemudian dilakukan *testing* model. Sehingga akan didapatkan nilai metrik dari *R-Square*, MSE, dan MAE. Berikut nilai metrik yang diperoleh dari model-model *Tensorflow* seperti pada Tabel 6.

B. Model Tensorflow Berbasis Time Series

Untuk model MLP berbasis *time series* yang dibuat ada sembilan model. Model-model tersebut disajikan pada Tabel 7. Dari model *Tensorflow* MLP yang di-*training* mendapatkan nilai akurasi dan *loss* model seperti pada Tabel 8.

Dari model yang dibuat di atas kemudian dilakukan *testing* model. Sehingga akan didapatkan nilai metrik dari *R-Square*, MSE, dan MAE. Berikut nilai metrik yang diperoleh dari model-model *Tensorflow* disajikan pada Tabel 9.

C. Hasil Konversi Model Multilayer Perceptron ke Model Tensorflow Lite

Setelah semua model MLP dikonversi ke *Tensorflow Lite*, maka dilakukan evaluasi model. Evaluasi model ini ditujukan untuk meninjau kinerja model mengalami penurunan atau sama seperti sebelum model dikonversi. Evaluasi model dilakukan dengan menjalankan model *Tensorflow Lite* di laptop. Untuk data yang digunakan untuk evaluasi model

Tabel 8.
Nilai Akurasi dan *Loss* Model yang Dibuat

Nama Model	Nilai Akurasi	Nilai *Loss*
Model I	67,34%	1,63%
Model II	67,42%	1,63%
Model III	67,41%	1,63%
Model IV	67,46%	1,64%
Model V	67,44%	1,65%
Model VI	67,46%	1,64%
Model VII	67,47%	1,64%
Model VIII	67,46%	1,66%
Model IX	67,46%	1,66%

Tabel 9.
Nilai Metrik *R-Square*, MSE, dan MAE

Nama Model	Nilai *R-Square*	Nilai *MSE*	Nilai *MAE*
Model I	0,7140122	0,0266163	0,0811515
Model II	0,7123290	0,0267729	0,0811187
Model III	0,7094931	0,0270368	0,0819708
Model IV	0,7042375	0,0275260	0,0815221
Model V	0,6997363	0,0279449	0,0820153
Model VI	0,6982917	0,0280793	0,0813656
Model VII	0,6581889	0,0318117	0,0880789
Model VIII	0,6564381	0,0319746	0,0881551
Model IX	0,6532546	0,0322709	0,0883021

menggunakan data *testing* yang sudah ada. Berikut nilai-nilai metrik *R-Square*, MSE, dan MAE yang diperoleh dari model MLP *Tensorflow Lite* seperti pada Tabel 10.

D. Hasil Konversi Model Multilayer Percetron Berbasis Time Series ke Model Multilayer Perceptron Berbasis Time Series Tensorflow Lite

Setelah semua model MLP dikonversi ke *Tensorflow Lite*, maka dilakukan evaluasi model. Evaluasi model ini ditujukan untuk meninjau kinerja model mengalami penurunan atau sama seperti sebelum model dikonversi. Evaluasi model dilakukan dengan menjalankan model *Tensorflow Lite* di laptop. Untuk data yang digunakan untuk evaluasi model menggunakan data *testing* yang sudah ada. Berikut nilai-nilai metrik *R-Square*, MSE, dan MAE yang diperoleh dari model MLP *Tensorflow Lite* disajikan pada Tabel 11.

E. Hasil Konversi Model Multilayer Perceptron Tensorflow Lite ke Model Byte Array C

Setelah model *Multilayer Perceptron* dapat di *compile* di mikrokontroler Pi Pico, maka selanjutnya mengevaluasi model. Evaluasi model in ditujukan untuk meninjau kinerja model mengalami penurunan atau sama seperti model sebelum dikonversi. Untuk skema pengujian evaluasi model ini, yaitu, data dikirim secara serial dari laptop ke mikrokontroler Pi Pico, kemudian hasil yang di peroleh dari mikrokontroler Pi Pico dikirim kembali ke laptop secara serial dan disimpan ke dalam *file* csv. Untuk data yang digunakan, yaitu data *testing* yang sudah ada. Berikut nilai-nilai metrik *R-Square*, *Mean Squared Error*, dan *Mean Absolute Error* yang diperoleh dari model *Multilayer Perceptron Tensorflow Lite byte array* C disajikan pada Tabel 12.

F. Hasil Konversi Model Multilayer Perceptron Berbasis Time series Tensorflow Lite ke Model Byte Array C

Setelah model *Multilayer Perceptron* dapat di *compile* di mikrokontroler Pi Pico, maka selanjutnya mengevaluasi model. Evaluasi model in ditujukan untuk meninjau kinerja model mengalami penurunan atau sama seperti model sebelum dikonversi. Untuk skema pengujian evaluasi model ini, yaitu data dikirim secara serial dari laptop ke mikro-

Tabel 10.
Nilai Metrik *R-Square*, MSE, dan MAE

Nama Model	Nila *R-Square*	Nilai MSE	Nilai MAE
Model I	0,6448178	0,01734278	0,06523952
Model II	0,3631551	0,01844258	0,06850198
Model III	0,8336538	0,01158553	0,05076780
Model IV	0,7758512	0,00966025	0,04840861
Model V	0,8698958	0,01042153	0,04816222
Model VI	0,8742242	0,01004757	0,04655807

Tabel 11.
Nilai Metrik *R-Square*, MSE, dan MAE

Nama Model	Nilai *R-Square*	Nilai MSE	Nilai MAE
Model I	0,7147268	0,0266150	0,0811478
Model II	0,7130478	0,0267717	0,0811148
Model III	0,7102189	0,0270356	0,0819670
Model IV	0,7049765	0,0275267	0,0815183
Model V	0,7004864	0,0279436	0,0820115
Model VI	0,6990455	0,0280780	0,0813618
Model VII	0,6590429	0,0318102	0,0880746
Model VIII	0,6572964	0,0319731	0,0881509
Model IX	0,6541209	0,0322694	0,0882980

Tabel 12.
Nilai Metrik *R-Square*, MSE, dan MAE

Nama Model	Nila *R-Square*	Nilai MSE	Nilai MAE
Model I	0,4839825	0,02655625	0,08023244
Model II	0,2507435	0,02766978	0,08326039
Model III	0,6325080	0,02295309	0,07022192
Model IV	0,6037505	0,02223413	0,06964164
Model V	0,7452684	0,02409519	0,06963067
Model VI	0,7375073	0,02274129	0,06784162

kontroler Pi Pico, kemudian hasil yang diperoleh dari mikrokontroler Pi Pico dikirim kembali ke laptop secara serial dan disimpan ke dalam *file* csv. Untuk data yang digunakan, yaitu data *testing* yang sudah ada. Berikut nilai-nilai metrik *R-Square*, MSE, dan MAE yang diperoleh dari model *Multilayer Perceptron Tensorflow Lite byte array* C disajikan pada Tabel 13.

IV. PEMBAHASAN

Untuk dapat melihat sejauh mana performa model yang dihasilkan dari model *Tensorflow* hingga ke model *byte array* C dapat dilihat dari nilai metrik *R-Square* yang dihasilkan. Pembahasan yang pertama, yaitu model MLP tanpa berbasis *time series* yang hasilnya disajikan pada Tabel 14.

Tabel 14 menampilkan nilai *R-Square* pada setiap model. Saat model *Multilayer Perceptron* dikonversi menjadi model *Tensorflow Lite* nilai metrik *R-Square* yang dihasilkan sama dengan model *Multilayer Perceptron* keras *Tensorflow*. Hal ini terjadi karena konversi dari model *Multilayer Perceptron* ke model *Tensorflow Lite* tidak menerapkan kuantisasi maupun optimasi. Tetapi, nilai metrik *R-Square* yang dihasilkan oleh model *Tensorflow Lite* yang dikonversi menjadi *byte array* C mengalami penurunan. Penurunan ini disebabkan karena nilai *input* dataset yang digunakan untuk testing memiliki angka desimal berjumlah di antara 16 sampai 18 angka yang kemudian dibulatkan menjadi 8 angka desimal saja. Hal ini dilakukan karena adanya keterbatasan jumlah data yang bisa diterima program *Raspberry Pi Pico* saat pengiriman data secara serial dari laptop ke mikrokontroler. Batasan ini menyebabkan model mengalami penurunan performa berdasarkan nilai metrik *R-Square* yang dihasilkan seperti disajikan pada Gambar 2.

Tabel 15 menampilkan persentase penurunan nilai metrik *R-Square* dari model *Multilayer Perceptron Tensorflow Lite*

Tabel 13.
Nilai Metrik *R-Square*, MSE, dan MAE

Nama Model	Nilai *R-Square*	Nilai MSE	Nilai MAE
Model I	0,7140157	0,0266159	0,0811496
Model II	0,7123274	0,0267731	0,0811181
Model III	0,7094901	0,0270371	0,0819691
Model IV	0,7042322	0,0275265	0,0815222
Model V	0,6997376	0,0279448	0,0820127
Model VI	0,6982882	0,0280797	0,0813628
Model VII	0,6581917	0,0318114	0,0880893
Model VIII	0,6564378	0,0319747	0,0881672
Model IX	0,6532597	0,0322704	0,0882904

Tabel 14.
Nilai Metrik *R-Square* pada Setiap Konversi Model MLP

Nama Model	Model *Tensorflow*	Model *Tensorflow Lite*	Model *Byte Array* C
Model I	0,64481797	0,6448178	0,4839825
Model II	0,36315539	0,3631551	0,2507435
Model III	0,83365380	0,8336538	0,6325080
Model IV	0,77585215	0,7758512	0,6037505
Model V	0,86989581	0,8698958	0,7452684
Model VI	0,87422422	0,8742242	0,7375073

Tabel 15.
Persentase Penurunan Model *Tensorflow Lite* ke Model *Byte Array* C Berdasarkan Nilai Metrik *R-Square*

Nama Model	Model *Tensorflow Lite*	Model *Byte Array* C	Persentase Penurunan
Model I	0,6448178	0,4839825	24,94%
Model II	0,3631551	0,2507435	30,95%
Model III	0,8336538	0,6325080	24,13%
Model IV	0,7758512	0,6037505	22,18%
Model V	0,8698958	0,7452684	14,33%
Model VI	0,8742242	0,7375073	15,64%

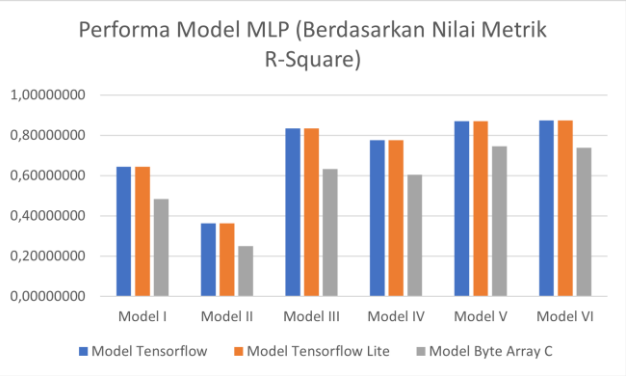
Tabel 16.
Nilai Metrik *R-Square* pada Setiap Konversi Model MLP Berbasis *Time Series*

Nama Model	Model *Tensorflow*	Model *Tensorflow Lite*	Model *Byte Array* C
Model I	0,7140122	0,7147268	0,7147268
Model II	0,7123290	0,7130478	0,7130478
Model III	0,7094931	0,7102189	0,7102189
Model IV	0,7042375	0,7049765	0,7049765
Model V	0,6997363	0,7004864	0,7004864
Model VI	0,6982917	0,6990455	0,6990455
Model VII	0,6581889	0,6590429	0,6590429
Model VIII	0,6564381	0,6572964	0,6572964
Model IX	0,6532546	0,6541209	0,6541209

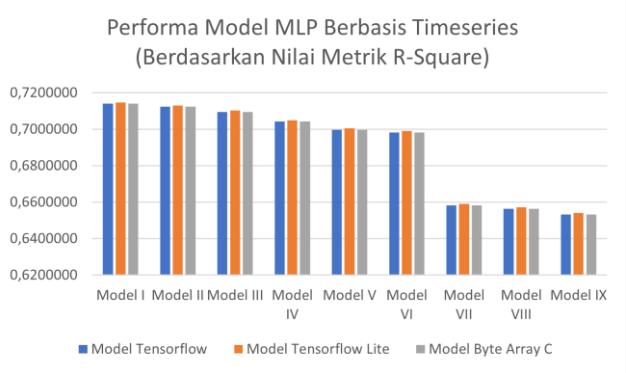
ke model MLP *byte array* C. Pada model pertama, yaitu model I mengalami penurunan nilai metrik *R-Square* sebesar 24,94%. Kemudian model kedua, yaitu model II mengalami penurunan nilai metrik *R-Square* sebesar 30,95%. Kemudian model ketiga, yaitu model III mengalami penurunan nilai metrik *R-Square* sebesar 24,13%. Kemudian model IV mengalami penurunan nilai metrik *R-Square* sebesar 22,18%. Kemudian model ketiga, yaitu model V mengalami penurunan nilai metrik *R-Square* sebesar 14,33%. Kemudian model VI mengalami penurunan nilai metrik *R-Square* sebesar 15,64%.

Pembahasan yang kedua, yaitu model *Multilayer Perceptron* berbasis *time series*. Pada Tabel 16 dapat dilihat bahwa konversi dari model *Tensorflow* ke model *Tensorflow Lite* menghasilkan nilai metriks *R-Square* yang hampir sama. Nilai yang hampir sama ini menjelaskan bahwa model *Tensorflow Lite* tidak mengalami penurunan performa dari model *Tensorflow* saat dilakukan konversi. Begitu juga saat model *Tensorflow Lite* dikonversi ke model *byte array* c tidak mengalami penurunan performa seperti pada Gambar 3.

Berbeda dengan hasil performa yang diperoleh dari model



Gambar 2. Grafik nilai metrik *R-Square* model MLP.



Gambar 3. Grafik nilai metrik *R-Square* model MLP berbasis *time series*.

MLP tanpa berbasis *time series* yang mengalami penurunan, untuk model MLP berbasis *time series* tidak mengalami penurunan performa dikarenakan dataset yang digunakan pada saat *training* model. Setelah dataset dinormalisasi, kemudian dataset mengalami operasi pembulatan di mana untuk dataset *input* mengalami pembulatan hingga 2 angka desimal dan dataset *output* mengalami pembulatan 3 angka desimal. Hal ini dilakukan karena ada keterbatasan jumlah data yang bisa diterima program *Raspberry Pi Pico* saat pengiriman data secara serial. Oleh karena itu model yang diujikan di mikrokontroler *Raspberry Pi Pico* tidak mengalami penurunan performa karena jumlah angka desimal data *input* yang masuk ke model sama.

V. KESIMPULAN

Keseluruhan model *Multilayer Perceptron* (MLP) yang telah dibuat tanpa berbasis *time series*, saat dilakukan pengujian di perangkat mikrokontroler *Raspberry Pi Pico* mengalami penurunan akurasi model dari model *Tensorflow Lite* ke model *Byte Array* C yang dapat terlihat dari nilai metrik *R-Square* yang dihasilkan. Untuk model I mengalami penurunan sebesar 24,94%, model II mengalami penurunan sebesar 30,95%, model III mengalami penurunan sebesar 24,13%, model IV mengalami penurunan sebesar 22,18%, model V mengalami penurunan sebesar 14,33%, dan model VI mengalami penurunan sebesar 15,64%. Kemudian Keseluruhan model MLP yang berbasis *time series*, saat dilakukan pengujian di perangkat mikrokontroler *Raspberry Pi Pico* tidak mengalami penurunan akurasi model hal ini dapat dilihat dari nilai metrik *R-Square* antara model *Multilayer Perceptron* berbasis *time series Tensorflow Lite* dengan model MLP berbasis *time series byte array* C. Sehingga Model paling ideal yang dihasilkan untuk pene-rapan sistem *Dead reckoning* pada mikrokontroler *Raspberry Pi Pico*,

yaitu model MLP tanpa berbasis *time series* dengan 3 *hidden* layer dengan *nodes* masing-masing sebesar 250, 125, dan 30, karena menghasilkan nilai metrik *R-Square* sebesar 0,7452684 pada pengujian di perangkat mikrokontroler *Raspberry Pi Pico*.

DAFTAR PUSTAKA

[1] J. Lin, C. Zou, L. Lan, S. Gu, and X. An, "Deep Heading Estimation for Pedestrian Dead Reckoning," in *International Conference on Big Data Mining and Information Processing (BDMIP)*, Qingdao, 2020, vol. 1656, no. 1, p. 12009. doi: 10.1088/1742-6596/1656/1/012009.

[2] M. Nazarahari and H. Rouhani, "A full-state robust extended kalman filter for orientation tracking during long-duration dynamic tasks using magnetic and inertial measurement units," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1280–1289, 2021, doi: 10.1109/TNSRE.2021.3093006.

[3] M. L. Hoang, A. Pietrosanto, S. D. Iacono, and V. Paciello, "Pre-Processing Technique for Compass-less Madgwick in Heading Estimation for Industry 4.0," in *IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Dubrovnik, 2020, pp. 1–6. doi: 10.1109/I2MTC43012.2020.9128969.