End-to-End Deep Learning Framework for Real-Time Inertial Attitude Estimation using 6DoF IMU

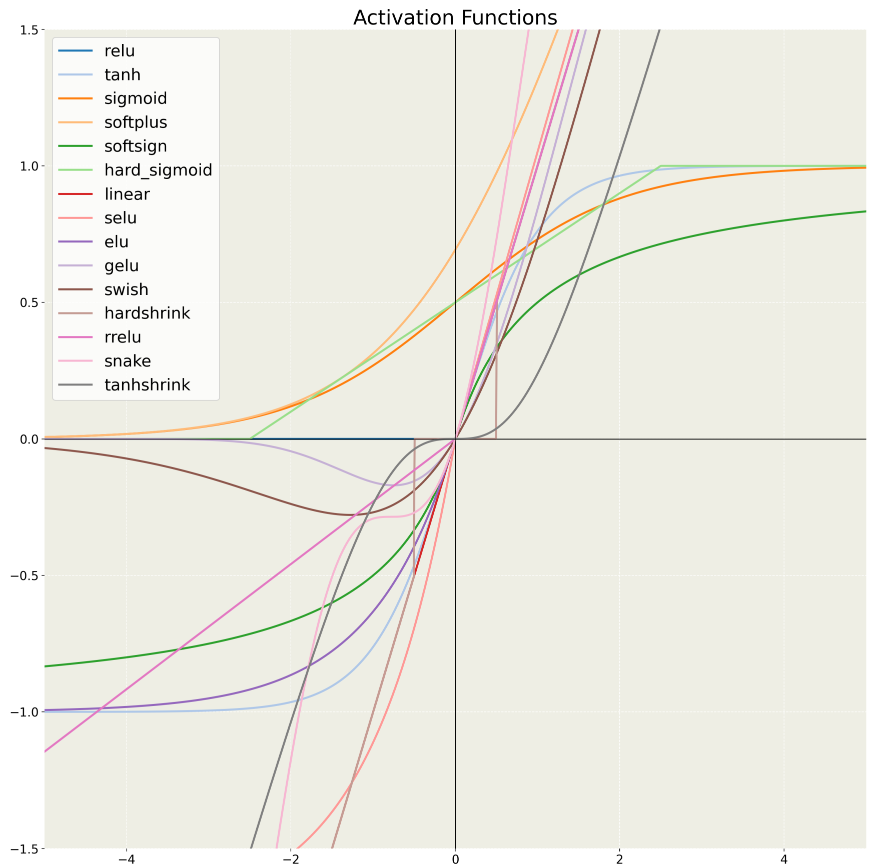
Arman Asgharpoor[1] and Mohammad Hossein Sabour[1]

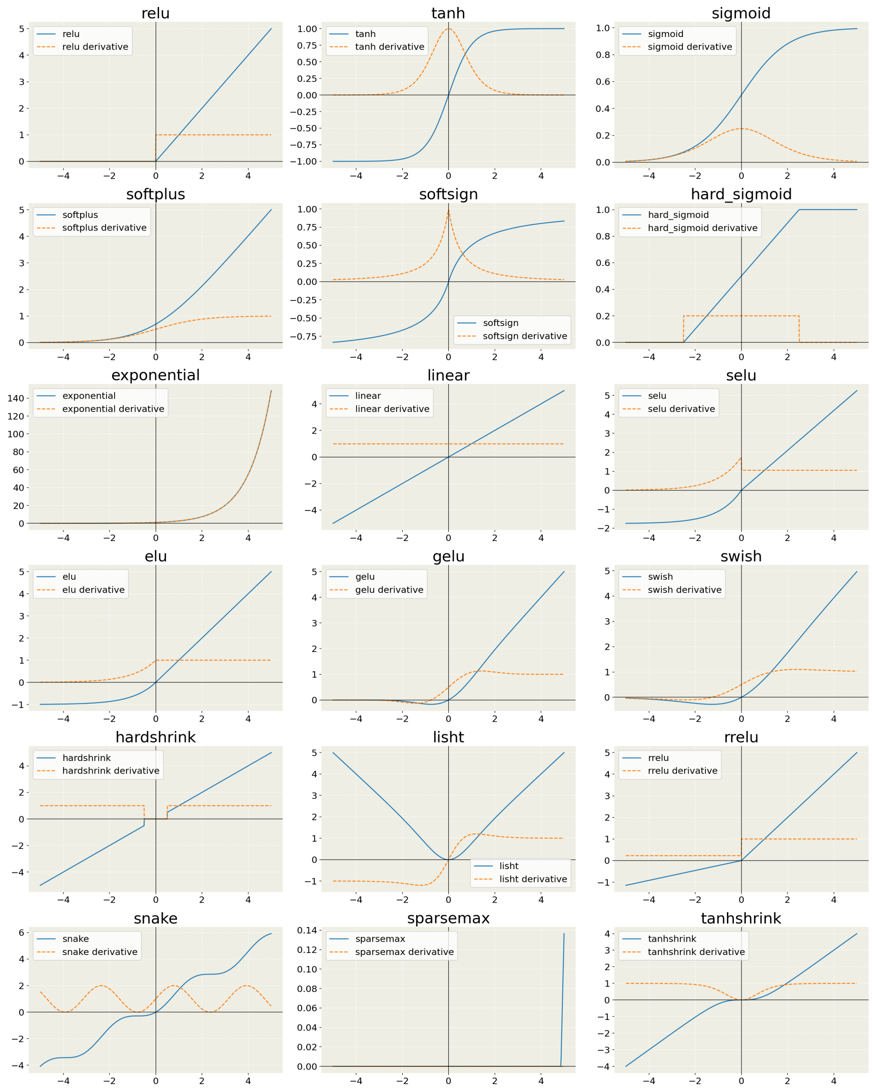
[1]University of Tehran Faculty of New Sciences and Technologies

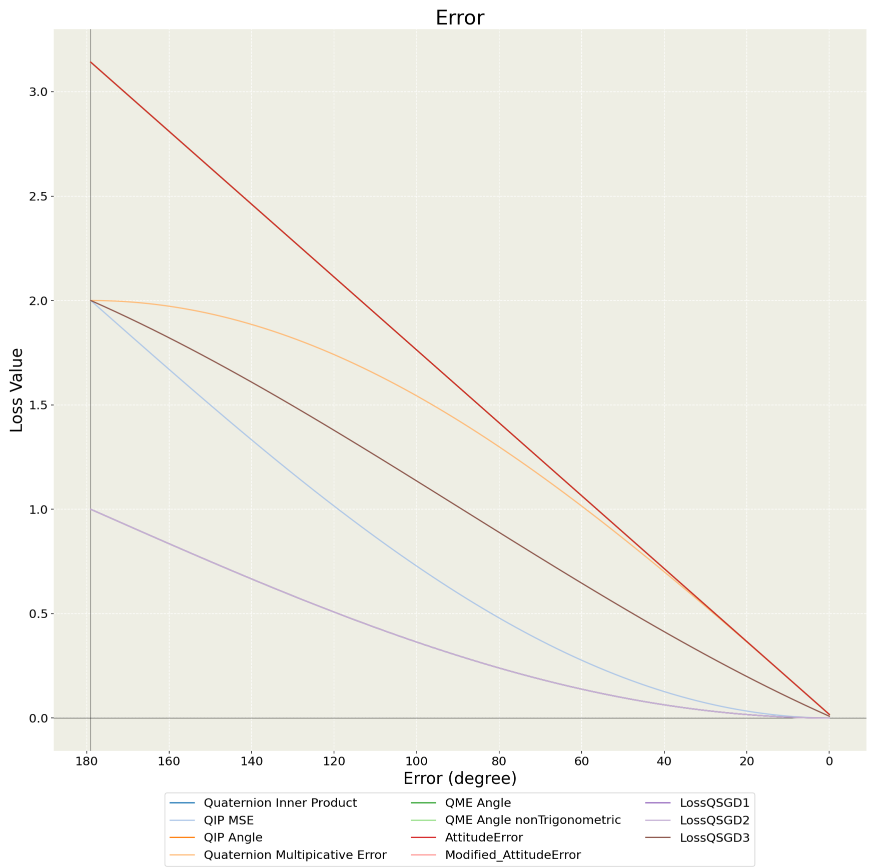
February 13, 2023

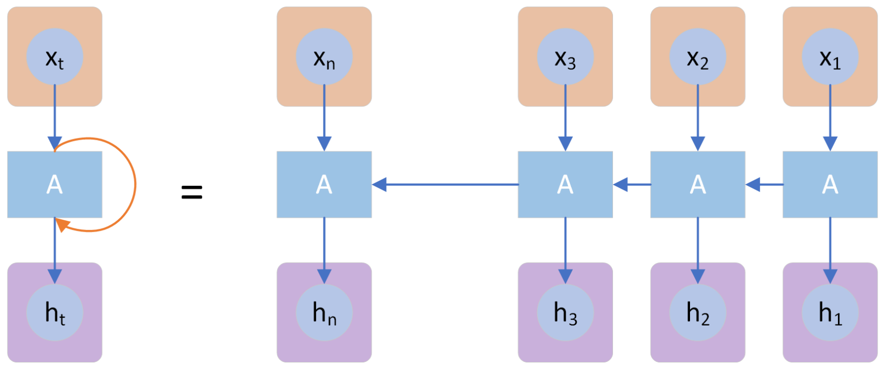
Abstract

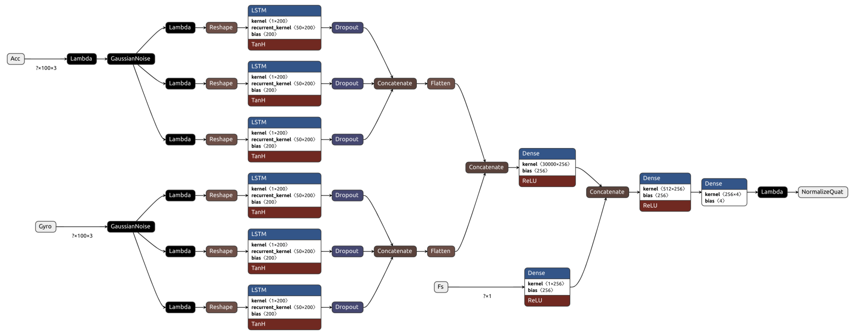
Inertial Measurement Units (IMU) are commonly used in inertial attitude estimation from engineering to medical sciences. There may be disturbances and high dynamics in the environment of these applications. Also, their motion characteristics and patterns also may differ. Many conventional filters have been proposed to tackle the inertial attitude estimation problem based on IMU measurements. There is no generalization over motion and environmental characteristics in these filters. As a result, the presented conventional filters will face various motion characteristics and patterns, which will limit filter performance and need to optimize the filter parameters for each situation. In this paper, two end-to-end deep-learning models are proposed to solve the problem of real-time attitude estimation by using inertial sensor measurements, which are generalized to motion patterns, sampling rates, and environmental disturbances. The proposed models incorporate accelerometer and gyroscope readings as inputs, which are collected from a combination of five public datasets. The models consist of convolutional neural network (CNN) layers combined with Bi-Directional Long-Short Term Memory (LSTM) followed by a Fully Forward Neural Network (FFNN) to estimate the quaternion. To evaluate the validity and reliability, we have performed an extensive and comprehensive evaluation over five publicly available datasets, which consist of more than 120 hours and 200 kilometers of IMU measurements. The results show that the proposed method outperforms the state-of-the-art methods in terms of accuracy and robustness. Furthermore, it demonstrates that this model generalizes better than other methods over various motion characteristics and sensor sampling rates.

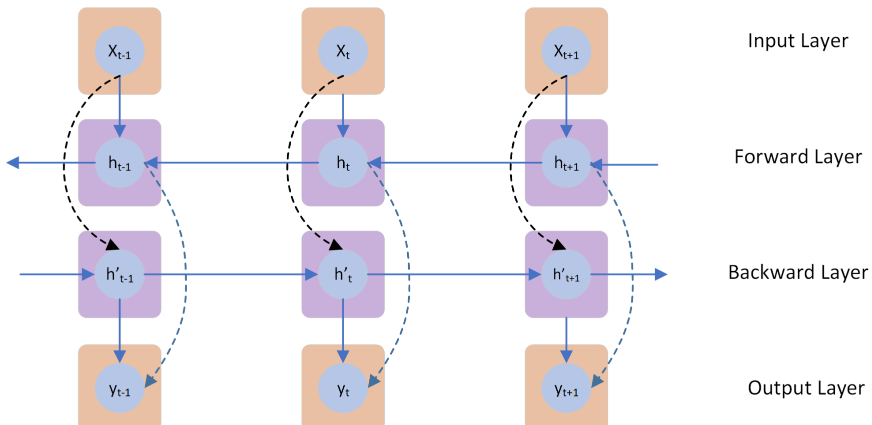

Activation Functions
relu
tanh
sigmoid
softplus
softsign
hard_sigmoid
linear
selu
elu
gelu
swish
hardshrink
rrelu
snake
tanhshrink
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−4
−2
0
2
4

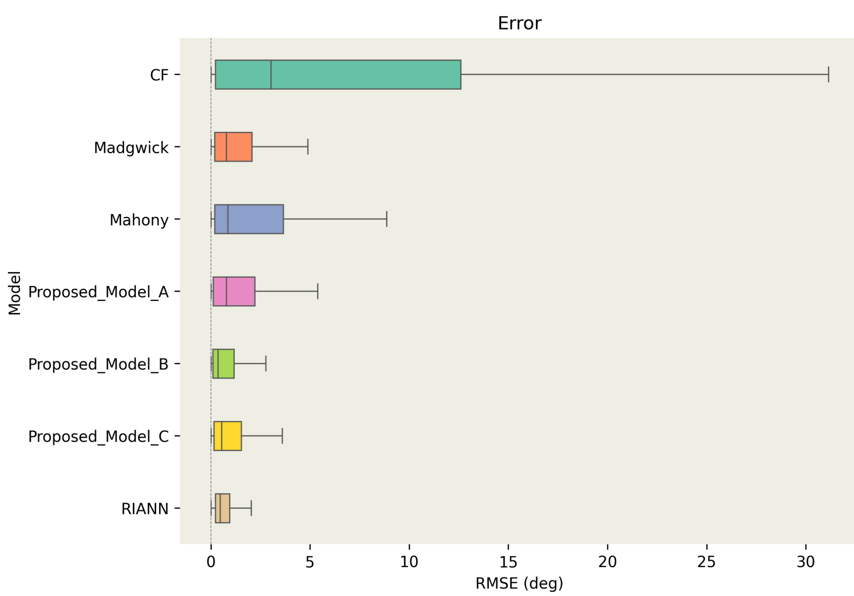

relu
relu
relu derivative
tanh
tanh
tanh derivative
sigmoid
sigmoid
sigmoid derivative
softplus
softplus
softplus derivative
softsign
softsign
softsign derivative
hard_sigmoid
hard_sigmoid
hard_sigmoid derivative
exponential
exponential
exponential derivative
linear
linear
linear derivative
selu
selu
selu derivative
elu
elu
elu derivative
gelu
gelu
gelu derivative
swish
swish
swish derivative
hardshrink
hardshrink
hardshrink derivative
lisht
lisht
lisht derivative
rrelu
rrelu
rrelu derivative
snake
snake
snake derivative
sparsemax
sparsemax
sparsemax derivative
tanhshrink
tanhshrink
tanhshrink derivative

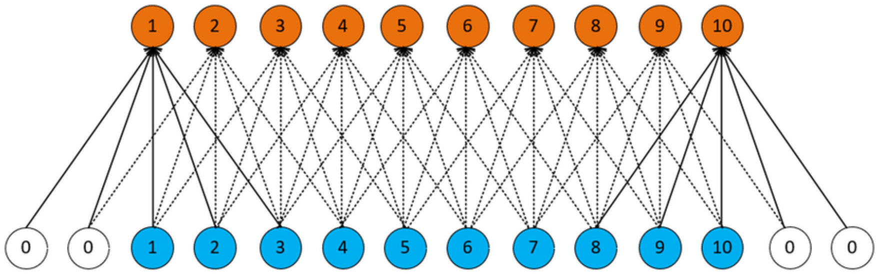

Error
Loss Value
3.0
2.5
2.0
1.5
1.0
0.5
0.0
180
160
140
120
100
80
60
40
20
0
Error (degree)
Quaternion Inner Product
QIP MSE
QIP Angle
Quaternion Multipicative Error
QME Angle
QME Angle nonTrigonometric
AttitudeError
Modified_AttitudeError
LossQSGD1
LossQSGD2
LossQSGD3

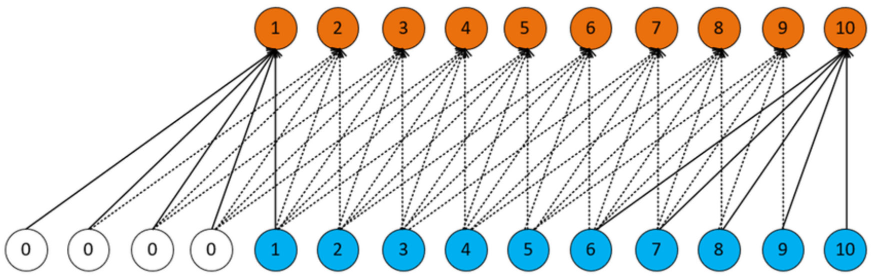

x_t
A
h_t
=
x_n
x_3
x_2
x_1
A
A
A
A
h_n
h_3
h_2
h_1


Acc
Lambda
GaussianNoise
Reshape
LSTM
Dropout
Concatenate
Flatten
Dense
NormalizeQuat
Gyro
Fs


Input Layer
Forward Layer
Backward Layer
Output Layer
X_{t-1}
X_t
X_{t+1}
h_{t-1}
h_t
h_{t+1}
h'_{t-1}
h'_t
h'_{t+1}
y_{t-1}
y_t
y_{t+1}

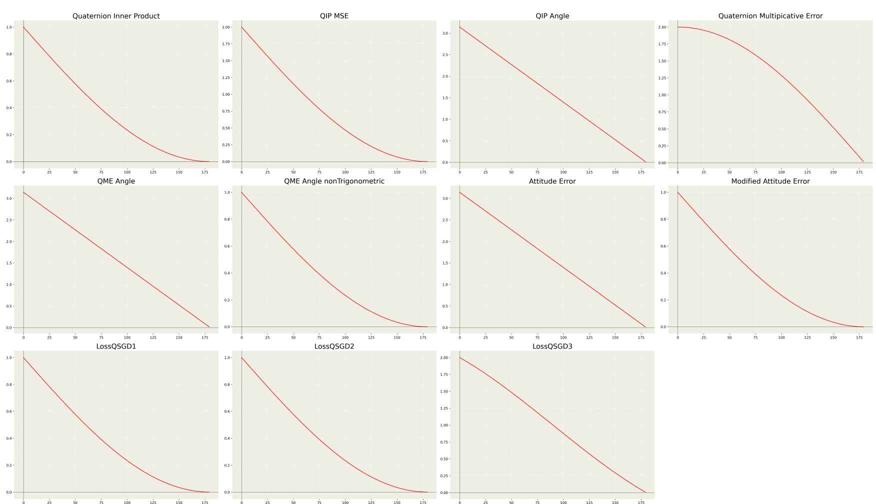

Error
CF
Madgwick
Mahony
Proposed_Model_A
Proposed_Model_B
Proposed_Model_C
RIANN
Model
0
5
10
15
20
25
30
RMSE (deg)

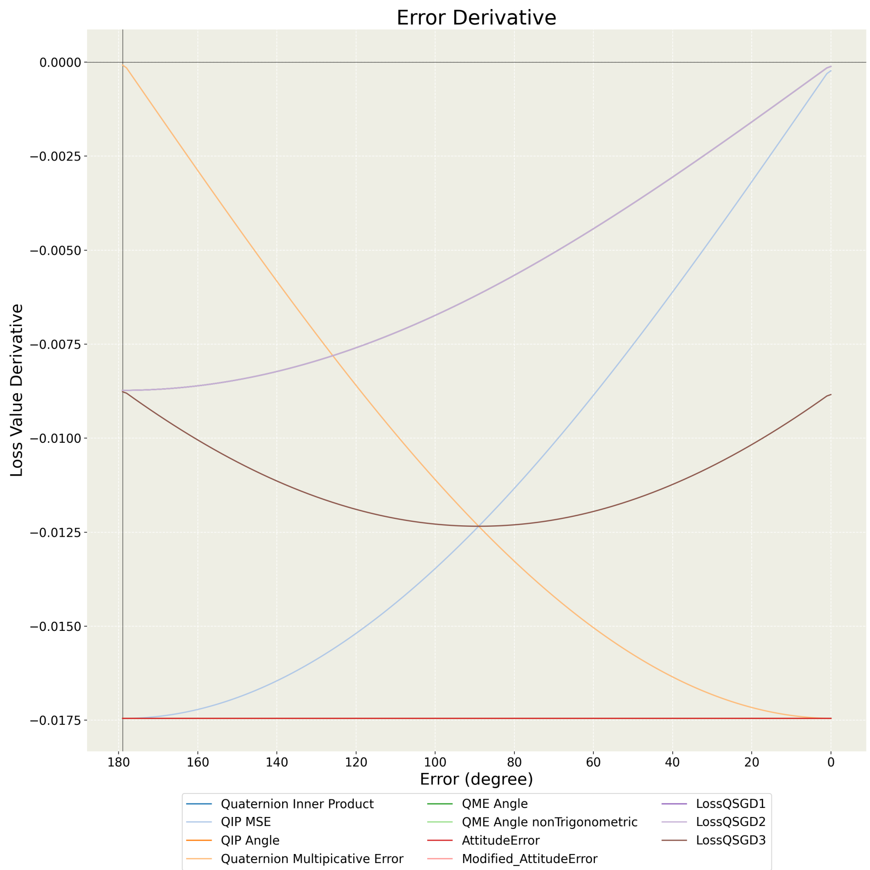


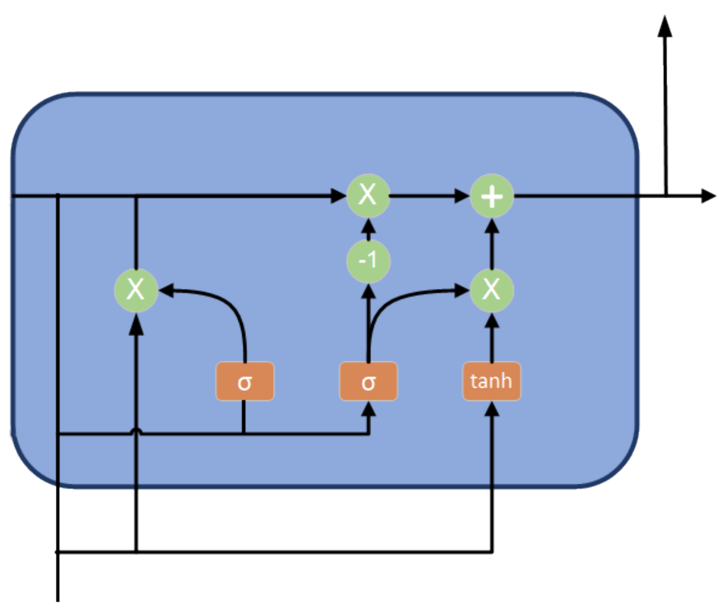


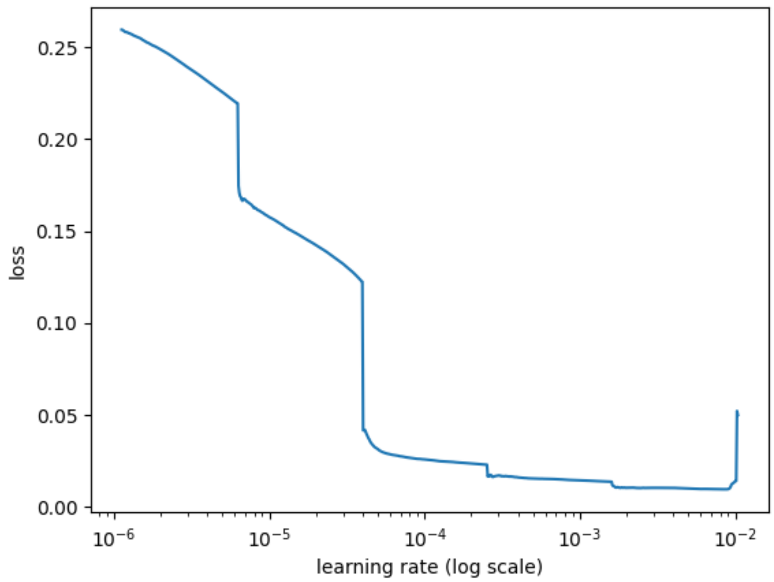
Hosted file

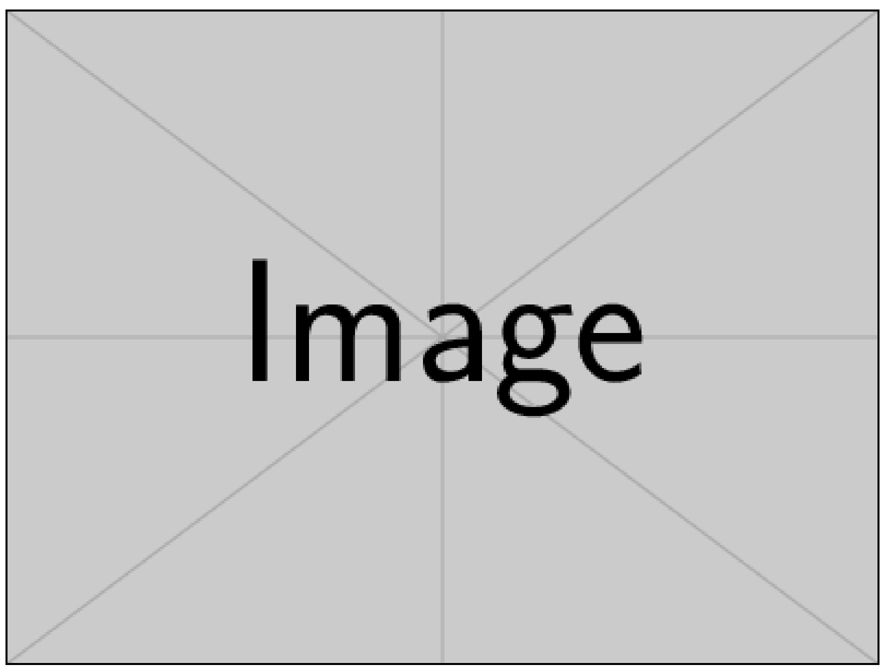
CNN.svg available at https://authorea.com/users/585711/articles/624216-end-to-end-deep-learning-framework-for-real-time-inertial-attitude-estimation-using-6dof-imu



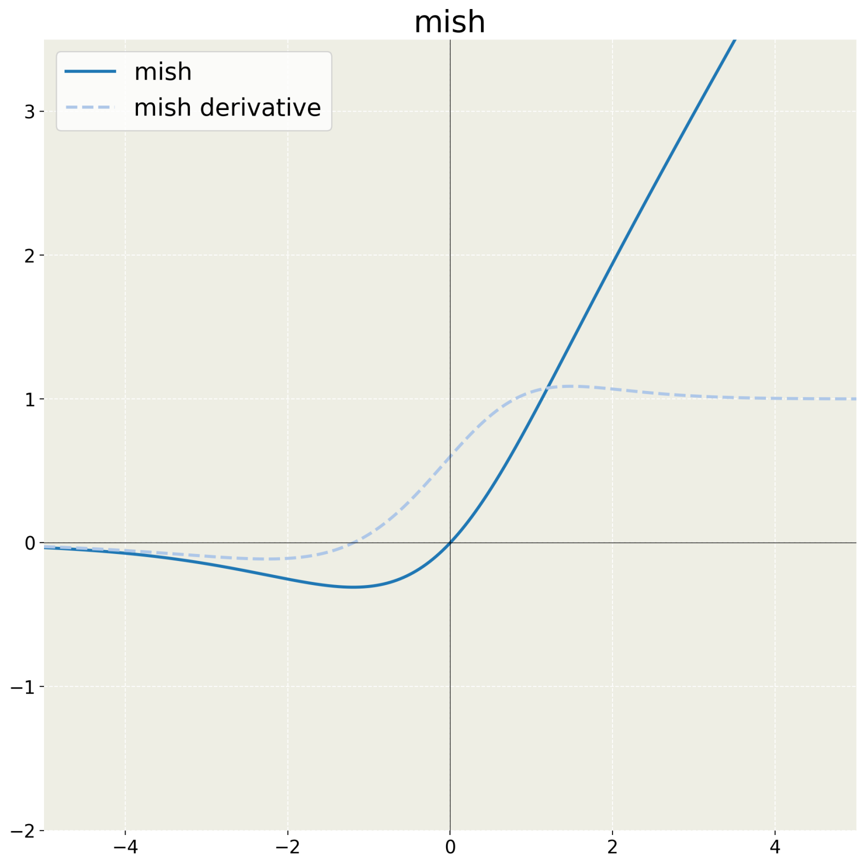

Error Derivative
Loss Value Derivative
Error (degree)
0.0000
−0.0025
−0.0050
−0.0075
−0.0100
−0.0125
−0.0150
−0.0175
180
160
140
120
100
80
60
40
20
0
Quaternion Inner Product
QIP MSE
QIP Angle
Quaternion Multipicative Error
QME Angle
QME Angle nonTrigonometric
AttitudeError
Modified_AttitudeError
LossQSGD1
LossQSGD2
LossQSGD3

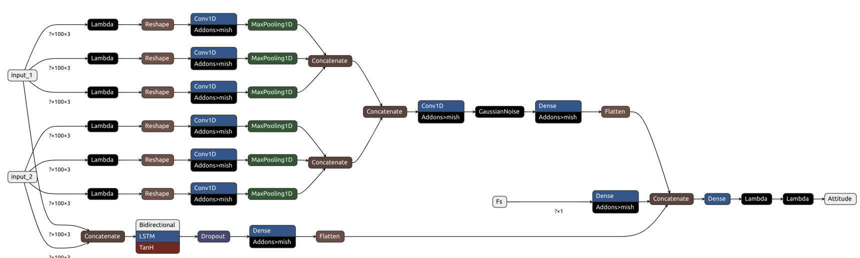

X
+
-1
X
X
σ
σ
tanh

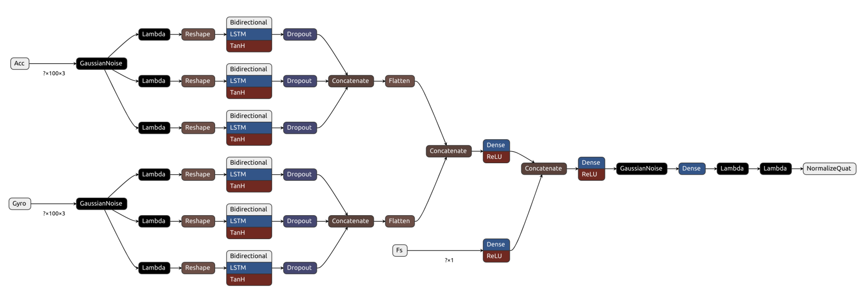

0.25
0.20
0.15
0.10
0.05
0.00
loss
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
learning rate (log scale)

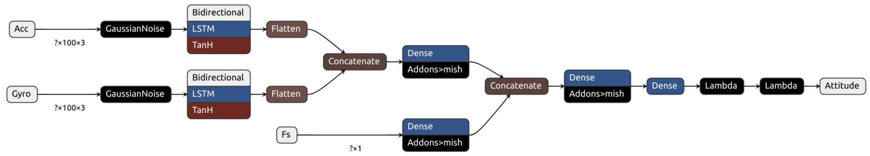


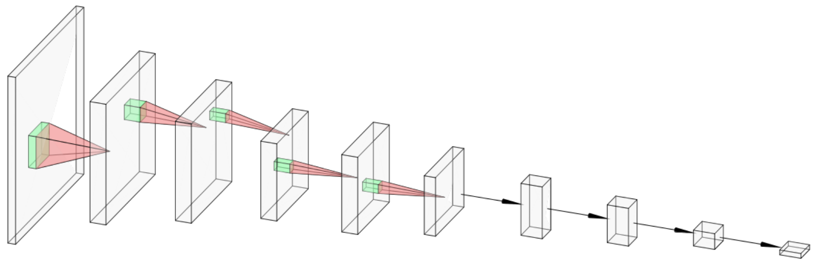
Image

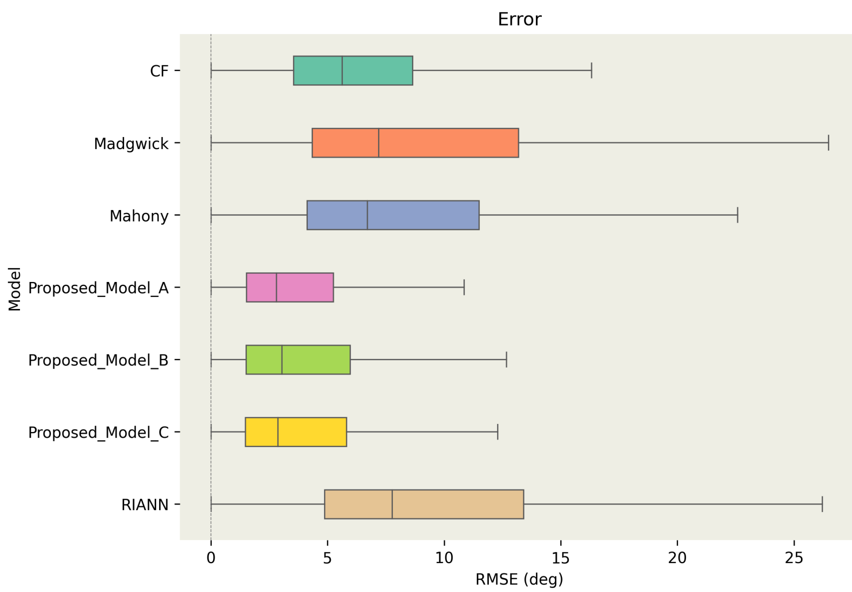

mish
mish
mish derivative
3
2
1
0
−1
−2
−4
−2
0
2
4

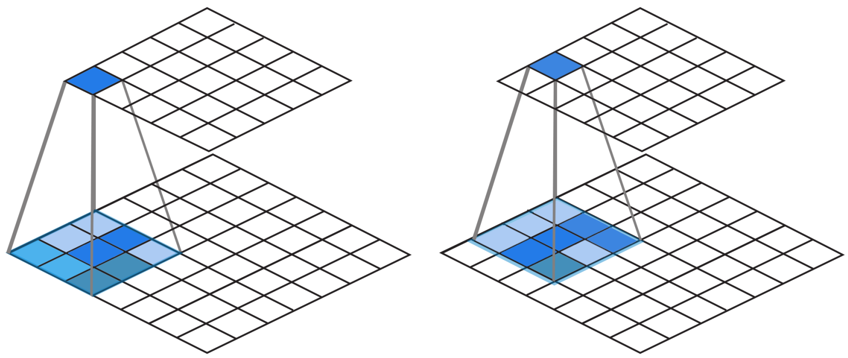

input_1
input_2
Lambda
Reshape
Conv1D
Addons>mish
MaxPooling1D
Concatenate
GaussianNoise
Dense
Flatten
Bidirectional
LSTM
TanH
Dropout
Fs
Lambda
Attitude

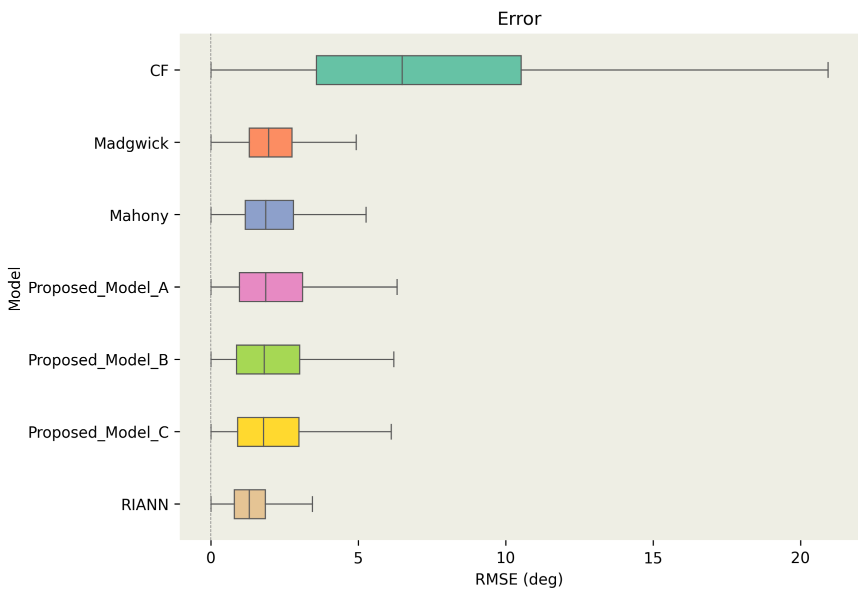

Acc
Gyro
GaussianNoise
Lambda
Reshape
Bidirectional
LSTM
TanH
Dropout
Concatenate
Flatten
Dense
ReLU
Fs
NormalizeQuat

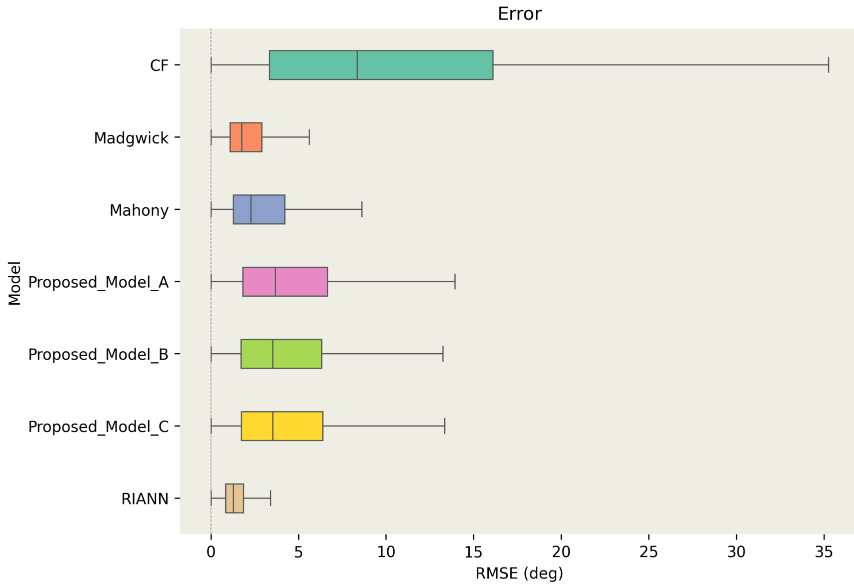

Acc
?×100×3
GaussianNoise
Bidirectional
LSTM
TanH
Flatten
Gyro
?×100×3
GaussianNoise
Bidirectional
LSTM
TanH
Flatten
Concatenate
Dense
Addons>mish
Fs
?×1
Dense
Addons>mish
Concatenate
Dense
Addons>mish
Dense
Lambda
Lambda
Attitude

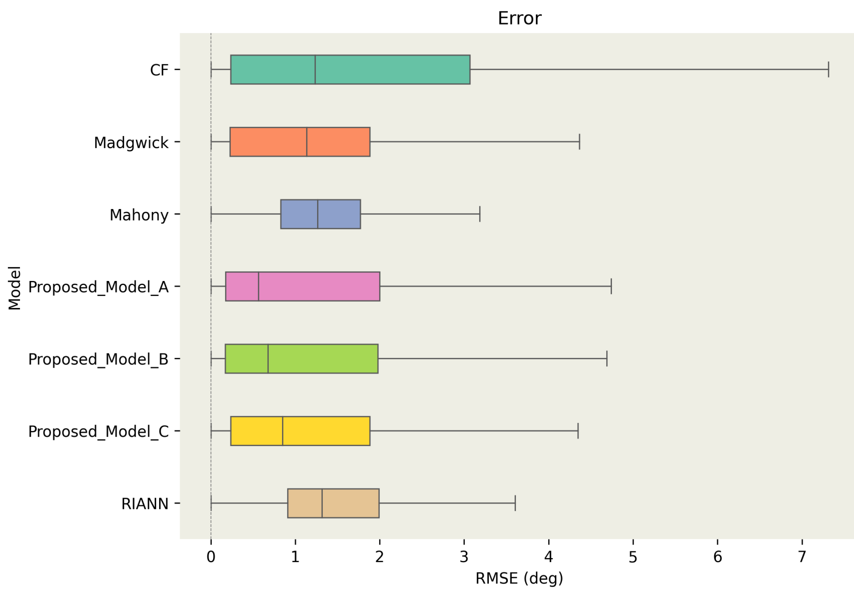


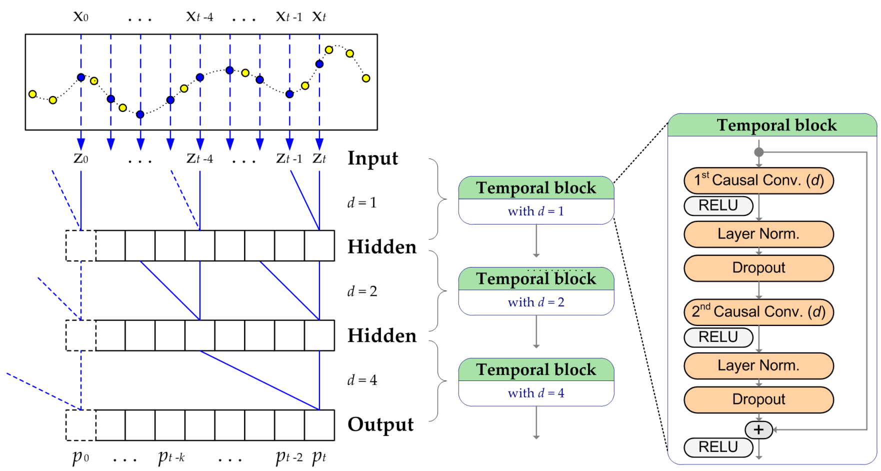

Error
CF
Madgwick
Mahony
Proposed_Model_A
Proposed_Model_B
Proposed_Model_C
RIANN
Model
0
5
10
15
20
25
RMSE (deg)



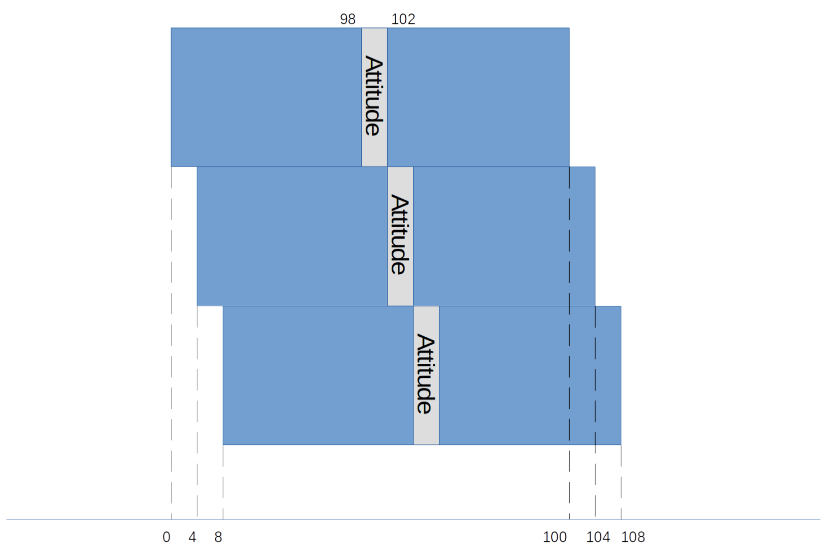

Error
CF
Madgwick
Mahony
Proposed_Model_A
Proposed_Model_B
Proposed_Model_C
RIANN
Model
0
5
10
15
20
RMSE (deg)


Error
CF
Madgwick
Mahony
Proposed_Model_A
Proposed_Model_B
Proposed_Model_C
RIANN
Model
0
5
10
15
20
25
30
35
RMSE (deg)

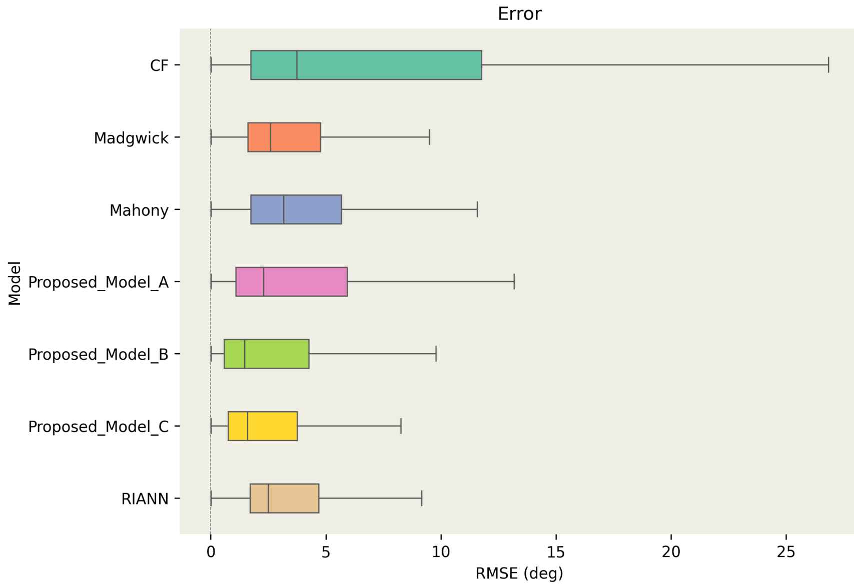

Error
CF
Madgwick
Mahony
Proposed_Model_A
Proposed_Model_B
Proposed_Model_C
RIANN
Model
0
1
2
3
4
5
6
7
RMSE (deg)



Input
Hidden
Hidden
Output
d = 1
d = 2
d = 4
Temporal block
with d = 1
Temporal block
with d = 2
Temporal block
with d = 4
Temporal block
1st Causal Conv. (d)
RELU
Layer Norm.
Dropout
2nd Causal Conv. (d)
RELU
Layer Norm.
Dropout
RELU



98
102
Attitude
Attitude
Attitude
0
4
8
100
104
108


Error
CF
Madgwick
Mahony
Proposed_Model_A
Proposed_Model_B
Proposed_Model_C
RIANN
Model
0
5
10
15
20
25
RMSE (deg)

RESEARCH ARTICLE

End-to-End Deep Learning Framework for Real-Time Inertial Attitude Estimation using 6DoF IMU

Arman Asgharpoor Golroudbari[1,2,3*] | Mohammad Hossein Sabour[1,2*]

[1]Department of Aerospace, Faculty of New Sciences & Technologies, University of Tehran, Tehran, Iran

[2]Students' scientific research center, Tehran University of Medical Sciences, Tehran, Iran

[3]Fuzzy Logic Lab Interest Group, Universal Scientific Education and Research Network, Tehran,Iran

Correspondence
M.H. Sabour, Department of Aerospace, Faculty of New Sciences & Technologies, University of Tehran, Tehran, Iran
Email: sabourmh@ut.ac.ir

Abstract:
Inertial Measurement Units (IMU) are commonly used in inertial attitude estimation from engineering to medical sciences. There may be disturbances and high dynamics in the environment of these applications. Also, their motion characteristics and patterns also may differ. Many conventional filters have been proposed to tackle the inertial attitude estimation problem based on IMU measurements. There is no generalization over motion and environmental characteristics in these filters. As a result, the presented conventional filters will face various motion characteristics and patterns, which will limit filter performance and need to optimize the filter parameters for each situation. In this paper, two end-to-end deep-learning models are proposed to solve the problem of real-time attitude estimation by using inertial sensor measurements, which are generalized to motion patterns, sampling rates, and environmental disturbances. The proposed models incorporate accelerometer and gyroscope readings as inputs, which are collected from a combination of five public datasets. The models consist of convolutional neural network (CNN) layers combined with Bi-Directional Long-Short Term Memory (LSTM) followed by a Fully Forward Neural Network (FFNN) to estimate the quaternion. To evaluate the validity and reliability, we have performed an extensive and comprehensive evaluation over five publicly available datasets, which consist of more than 120 hours and 200 kilometers of IMU measurements. The results show that the proposed method outperforms the state-of-the-art methods in terms of accuracy and robustness. Furthermore, it demonstrates that this model generalizes better than other methods over various motion characteristics and sensor sampling rates.

KEYWORDS
Deep Learning, Attitude Estimation, Inertial Sensors, Intelligent Filter, Sensor Fusion, Long-Short Term Memory, Convolutional Neural Network

[*] Equally contributing authors.

1 | INTRODUCTION

Attitude determination, or the process of accurately estimating an object's orientation, is crucial for successful navigation. The field of attitude estimation is one of the most important research areas in navigation, image stabilization, and tracking. Many moving robots, such as autonomous vehicles and drones, rely on accurate attitude determination to fulfill their mission goals. Numerous instruments and sensors are available for this purpose, but they vary in cost and complexity. While high-quality sensors can provide more accurate results, they may not always be practical due to their high cost. One way to increase accuracy at a lower cost is to use multiple sensors, either of the same type (homogenous) or different types (heterogeneous). This approach, known as Multi-Data Sensor Fusion (MSDF), involves fusing data from multiple sensors to reduce error and uncertainty. MSDF can be further divided into two categories: single-point methods, which use vector measurements at a single point in time, and recursive methods, which combine measurements over time and the system's mathematical model Gebre-Egziabher et al. (2004). The precision of attitude determination depends on the sensors' accuracy, the system modeling quality, and the information processing method Renaudin and Combettes (2014). Obtaining this precision is a challenging navigation problem due to system modeling, process, and measurement errors. Increasing the sensor's precision may exponentially increase the cost; sometimes, achieving the precision requirements is only possible for an exorbitant cost. Inertial navigation algorithms, which are based on the Dead Reckoning method, have been used for years to determine the attitude based on inertial sensors Steinhoff and Schiele (2010). Different kinds of inertial sensors are used in this method, such as accelerometers and gyroscopes, which are referred to as Inertial Measurement Units (IMU). A moving object's position, velocity, and attitude can be determined using the numerical integration of IMU measurements. Over the past decade, MEMS based IMUs have become increasingly popular. Due to recent advances in MEMS technology, IMUs have become smaller, cheaper, and more accurate, and they can now be found in mobile robots, smartphones, drones, and autonomous vehicles. IMUs are commonly used in moving objects' navigation systems, but they suffer from noise and bias, which directly affect the performance attitude estimation algorithm Mahdi et al. (2022). Real-time attitude estimation based on IMU sensor raw data is a fundamental problem in sensor fusion. In the past decades, different MSDF techniques and Deep Learning models have been developed to tackle this problem and increase the accuracy and reliability of attitude estimation techniques. Attitude can be estimated/determined by MSDF methods using at least a 6-Degree-of-Freedom (6DoF) Sensor Fusion Algorithm (SFA). In 6DoF SFAs, a three-axis accelerometer will be fused with a three-axis gyroscope to estimate the attitude. It is noticeable that as the accelerometer can not measure the yaw (heading) angle and the gyroscope can only measure the angle's rate, 6DoF SFAs are not suitable for attitude and heading estimation/determination. An alternative method is to fuse magnetometer readings with a 6DoF SFA to estimate/determine the full orientation (attitude and heading). A magnetometer's main disadvantage is the magnetic disturbances, which adversely affect its performance, mainly when used for indoor navigation. Several techniques have been developed to reduce the effect of magnetic disturbances on the filter performance, such as the Factorized Quaternion Algorithm (FQA) Lee and Low (2012). Most SFAs are developed and parametrized based on the system's dynamic model, which requires a precise choice of model parameters Fauske et al. (2007). To the best of our knowledge, no algorithm can handle all types of motions. In recent years, deep learning models have proven their ability to tackle sequential data and learn the hidden patterns and relationships therein Wang et al. (2020); Xiao et al. (2018); Zulqarnain et al. (2020); Nevavuori et al. (2020); Bouktif et al. (2019). Our paper proposes an end-to-end deep-learning approach to solve the problem of real-time attitude estimation by using inertial sensor measurements. Since a magnetometer is not present, this model estimates roll and pitch angles. The model consists of CNN layers combined with Bi-Directional LSTM followed by a FFNN to estimate the quaternion. Our extensive evaluation of this model on various publicly available IMU datasets shows it outperforms conventional algorithms and deep learning models. Our main contribution in this paper is to present a novel end-to-end deep learning framework that can be used for attitude estimation using low-cost strapdown inertial measurement units based on microelectromechanical systems (MEMS). Our paper presents a generalizable, end-to-end, hybrid RNN-CNN neural network that learns motion characteristics, noise, and bias associated with inertial sensor measurements across various devices. The rest of the paper is organized as follows: the related works are given in Section 2. We presented a detailed description of the problem in Section 3. Our methodology is provided in section 4, and present a detailed description of our experiment is in section 5. The test results and analysis are given in Section 6. In Section 7, we draw some conclusions and outline possible future work.

2 | RELATED WORKS

A three-dimensional dead-reckoning navigation system, such as an inertial navigation system (INS), contains a set of inertial measurement units consisting of three gyroscopes aligned with three mutually orthogonal accelerometers. As part of this package, a navigation processor is also included. It integrates the outputs of the IMU to provide information about the position, velocity, and attitude Groves (2015). This kind of navigation system can be considered one of the most straightforward approaches to performing attitude estimation only by using inertial sensors. Despite their widespread use in industry, from medical science to aerospace, these sensors suffer from a large amount of noise and bias in their measurements, which causes them to accumulate errors quickly over time. As a result, these types of sensors are not suitable for long-term use alone. In the past decade, much research has been conducted on inertial navigation techniques to tackle this problem. These studies could be divided into three categories, estimation methods, MSDF techniques, and evolutionary/AI algorithms. Methods such as the Kalman Filter (KF) family (i.e., EKF, UKF, MEKF) and other commonly used algorithms, as well as Madgwick Madgwick et al. (2010), and Mahony Euston et al. (2008) are based on the dynamic model of the system. Kalman filter was first introduced in Kalman (1960), and its variants, such as EKFJing et al. (2017), UKFChiella et al. (2019), and MEKFHall et al. (2008), have been implemented for attitude estimation applications Crassidis et al. (2007). In Caruso et al. (2021), Caruso et al. compared different sensor fusion algorithms for inertial attitude estimation. This comparative study showed that SFA performance is highly dependent on parameter tuning and fixed parameter values are unsuitable for all applications. So, parameter tuning is one of the disadvantages of the conventional attitude estimation method. This problem could be tackled using evolutionary algorithms such as fuzzy logic Shen et al. (2012); Widodo et al. (2014) and deep learningBrossard et al. (2020); Han et al. (2021); Buchanan et al. (2022); Engelsman and Klein (2022). Most deep learning approaches in inertial navigation have focused on inertial odometry Esfahani et al. (2019a); Aslan et al. (2022); Soyer et al. (2021); Saha et al. (2022); Onyekpe et al. (2021); Guimarães et al. (2021); Lin et al. (2022), and just a few try to solve the inertial attitude estimationWeber et al. (2021); Esfahani et al. (2019b) problem. Deep learning methods are usually used for visual or visual-inertial-based navigationOzaki and Kuroda (2021); Yu et al. (2019); Fan et al. (2021). Rochefort et al. proposed a neural networks-based satellite attitude estimation algorithm using a quaternion neural network. This study presents a new way of integrating the neural network into the state estimator and develops a training procedure that is easy to implement. This algorithm provides the same accuracy as the EKF with significantly lower computational complexity. In Chang-Siu et al. (2011), a Time-Varying Complementary Filter (TVCF) has been proposed to use a fuzzy logic inference system for CF parameters adjustment for applying attitude estimation. Chen et al. Chen et al. (2018a); Rochefort et al. (2005) deep recurrent neural networks for estimating the displacement of a user over a specified time window. OriNet Esfahani et al. (2019b) was introduced by Esfahani et al. to estimate the orientation in quaternion form based on LSTM layers and IMU measurements. Yuexin Zhang (2019) developed a sensor fusion method to provide pseudo-GPS position information using empirical mode decomposition threshold filtering (EMDTF) for IMU noise elimination and a LSTM neural network for pseudo-GPS position prediction during GPS outages. Dhahran et al. Dhahbane et al. (2020) developed a neural network-based complementary filter (NNCF) with ten hidden layers and trained by Bayesian Regularization Backpropagation (BRB) training algorithm to improve the generalization qualities and solve the overfitting problem. In this method output of the complementary filter is used as the neural network input. Li et al. Li et al. (2019a) proposed an adaptive Kalman filter with a fuzzy neural network for a trajectory estimation system mitigating the measurement noise and undulation for implementing the touch interface. Deep Learning has been used in Brossard et al. (2020) to denoise the gyroscope measurements for an open-loop attitude estimation algorithm. Weber et al. Weber et al. (2021) present a real-time-capable neural network for robust IMU-based attitude estimation. In this study, an accelerometer, gyroscope, and IMU sampling rate have been used as input to the neural network, and the output is the attitude in the quaternion form. This model is only suitable for estimating the roll and pitch angle. Sun et al. Sun et al. (2021) introduced a two-stage deep learning framework for inertial odometry based on LSTM and FFNN architecture. In this study, the first stage is used to estimate the orientation, and the second stage is used to estimate the position. A Neural Network model has been developed by Santos et al. Dos Santos et al. (2021) for static attitude determination based on PointNet architecture. They used an attitude profile matrix as input. This model uses the Swish activation function and Adam as its optimizer. A deep learning model has been developed to estimate the Multirotor Unmanned Aerial Vehicle (MUAV) based on the Kalman filter and FFNN in Al-Sharman et al. (2019). LSTM framework has been used in Narkhede et al. (2021) the Euler angles utilizing an accelerometer, gyroscope, and magnetometer, but the sensor sampling rate has not been considered. In Table 1, we summarized some related works in the navigation field using deep learning.

TABLE 1 Deep Learning for Localization.

Model	Year/Month	Input Data	Application
PoseNet Kendall et al. (2015)	2015/12	Vision	Relocalization
VINet Clark et al. (2017b)	2017/02	Vision + Inertial	Visual Inertial Odometry
DeepVO Wang et al. (2017)	2017/05	Vision	Visual Odometry
VidLoc Clark et al. (2017a)	2017/07	Vision	Relocalization
IONet Chen et al. (2018a)	2018/02	Inertial Only	Inertial Odometry
UnDeepVO Li et al. (2018)	2018/05	Vision	Visual Odometry
VLocNet Valada et al. (2018)	2018/05	Vision	Relocalization, Odometry
RIDI Yan et al. (2018)	2018/09	Inertial Only	Inertial Odometry
SIDA Chen (2020)	2019/01	Inertial Only	Domain Adaptation
VIOLearner Shamwell et al. (2019)	2019/04	Vision + Inertial	Visual Inertial Odometry
RINS-W Brossard et al. (2019)	2019/05	Inertial Only	Inertial Odometry
SelectFusion Chen et al. (2019)	2019/06	Vision + Inertial + LIDAR	VIO and sensor Fusion
LO-Net Li et al. (2019b)	2019/06	LIDAR	LIDAR Odometry
L3-Net Lu et al. (2019)	2019/06	LIDAR	LIDAR Odometry
Silva do Monte Lima et al. (2019)	2019/8	Inertial	Inertial Odometry
DeepVIO Han et al. (2019)	2019/11	Vision + Inertial	Visual Inertial Odometry
OriNet Esfahani et al. (2019b)	2020/4	Inertial	Inertial Odometry
Sorg Sorg (2020)	2020/4	Inertial	Pose Estimation
GALNet Mendoza et al. (2020)	2020/5	Inertial + Kinematic	Autonomous Cars
PDRNet Asraf et al. (2021)	2021/3	Inertial	Pedestrian Dead Reckoning
Kim et al. (2021)	2021/4	Inertial	Inertial Odometry
RIANN Weber et al. (2021)	2021/5	Inertial	Attitude Estimation
CTIN Rao et al. (2022)	2022/6	Inertial	Inertial Odometry
Xia et al. (2022)	2022/8	Inertial	Human Pose Estimation
Brotchie et al. (2022)	2022/11	Inertial	Attitude Estimation

3 | PROBLEM DEFINITION

The problem addressed in this article is the real-time estimation of the attitude, or orientation, of an object based on measurements from an IMU sensor. The IMU sensor consists of gyroscopes and accelerometers, which measure the object's angular velocity and linear acceleration. The main challenge in solving this problem is the high level of noise and bias present in these measurements, which can accumulate errors over time and decrease the accuracy of the attitude estimate. The study aims to develop a method for accurately and reliably estimating the object's attitude in real time without requiring an initial reset period for filter convergence. This is important for various applications, including navigation, image stabilization, tracking, and autonomous vehicles, where accurate and precise attitude determination is critical for successful performance. The estimation is based on the current and previous measurements of the gyroscope and accelerometer, which are fed into a Neural Network model to estimate the attitude. Despite almost all previous studies, we do not consider any initial reset period for filter convergence.

3.1 | IMU Dynamics Model

An IMU is a sensor that measures the angular velocity and linear acceleration of a rigid body which is a combination of a gyroscope and an accelerometer. The gyroscope measures the angular velocity of the rigid body in the body frame, and the accelerometer measures the linear acceleration with respect to the local frame. The gyroscope and accelerometer are rigidly attached to the rigid body. IMU measurements are imperfectly affected by noise and bias. The noise is a random process that is independent of the previous measurements which can be considered as a zero-mean white noise with a constant variance Eq. 1, and Eq. 2. Biases ($\mathbf{b}_\omega$ and $\mathbf{b}_a$) is a systematic error that is constant or slow-varying over time.

$$\mathbf{v}_\omega \sim \mathcal{N}(0, \sigma_\omega^2) \tag{1}$$

$$\mathbf{v}_a \sim \mathcal{N}(0, \sigma_a^2) \tag{2}$$

The IMU dynamics model is a nonlinear model that describes the relationship between the angular velocity and linear acceleration of the rigid body and the orientation of the rigid body. The IMU dynamics model is given by Eq. 3 and Eq. 4.

$$\tilde{\omega} = \omega - \mathbf{b}_\omega + \mathbf{v}_\omega \tag{3}$$

$$\tilde{\mathbf{a}} = (\mathbf{R}^T \mathbf{g}) + \mathbf{a} - \mathbf{b}_a + \mathbf{v}_a \tag{4}$$

where ω is the angular velocity of the rigid body in the body frame, $\mathbf{a}$ is the linear acceleration of the rigid body in the body frame, $\mathbf{R}$ is the rotation matrix that describes the orientation of the body frame with respect to the inertial frame, $\mathbf{g}$ is the gravity vector, $\mathbf{b}_\omega$ and $\mathbf{b}_a$ are the biases of the gyroscope and accelerometer, respectively, and $\mathbf{v}_\omega$ and $\mathbf{v}_a$ are the noise of the gyroscope and accelerometer, respectively.

3.2 | Orientation

The orientation of the object of interest with respect to a reference frame could be defined as the shortest rotation between a frame attached to the object and a reference frame. Orientation parameters (attitude coordinates) refer to sets of parameters (coordinates) that fully describe a rigid body's attitude, which is not unique expressions. There are many ways to represent the attitude of a rigid body. The most common are the Euler angles, the rotation matrix, and the quaternions. The Euler angles are the most familiar form known as yaw, pitch, and roll (or heading, elevation, and bank). Engineers widely use rotation matrices and quaternions, but the quaternions are less intuitive. The Euler angles are defined as the rotations about the three orthogonal axes of the body frame and suffer from the gimbal lock problem. The rotation matrix is a 3x3 matrix that represents the orientation of the body frame with respect to the inertial frame, which leads to having six redundant parameters. The quaternions are a 4x1 vector more suitable for attitude estimation because they are not subject to the gimbal lock problem and have the least redundant parameters. To avoid singularities and have the least number of redundant parameters, we use quaternion representation with the components $[w, x, y, z]$ instead of Direction Cosine Matrix (DCM) or Euler angles. The quaternions are defined as the following:

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{5}$$

where q_0 is the scalar part and q_1, q_2, and q_3 are the vector part. And the following equation shows the relationship between the quaternions and the Euler angles:

$$\mathbf{q} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \tag{6}$$

where ϕ, θ, and ψ are the Euler angles. The error between the estimated attitude and the true attitude can be calculated by quaternion multiplicative error and using the following equation:

$$\mathbf{q}_{err} = \mathbf{q}_{true} \otimes \mathbf{q}_{est}^{-1} \tag{7}$$

where $\mathbf{q}_{err}$ represents the shortest rotation between true and estimated orientation. The quaternion multiplication operator is defined as follows:

$$\mathbf{q} \otimes \mathbf{p} = \begin{bmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 - q_1p_3 + q_2p_0 + q_3p_1 \\ q_0p_3 + q_1p_2 - q_2p_1 + q_3p_0 \end{bmatrix} \tag{8}$$

where **q** and **p** are the quaternion to be multiplied, and the angle between the actual and estimated orientation is calculated by:

$$\theta = 2\arccos(scalar(\mathbf{q}_{err})) \tag{9}$$

where θ is the angle between the true and estimated orientation. Based on the above equations, the cumulative error of the estimated attitude in N number of time steps can be calculated by:

$$e_\alpha = 2\arccos(scalar(\mathbf{q}_{err})) \tag{10}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_\alpha^2} \tag{11}$$

where N is the number of samples and e_α is the angle between the true and estimated orientation. The RMSE is the root mean square error of the estimated attitude, and it presents the differences between values predicted by a model or an estimator and the values observed. The lower the RMSE, the better the model fits the data. Attitude determination and control play a vital role in Aerospace engineering. Most aerial or space vehicles have subsystem(s) that must be pointed in a specific direction, known as pointing modes, e.g., Sun pointing, Earth pointing. For example, keeping the satellite antenna pointed to the Earth continuously is the key to a successful mission in communications satellites. That will be achieved if we have proper knowledge of the vehicle's orientation; in other words, the attitude must be determined. Attitude determination methods can be divided into two categories: static and dynamic. The static attitude determination method is a point-to-point, time-independent method based on a memory-less approach. It is the observations or measurements processing to obtain the information for describing the object's orientation relative to a reference frame. It could be determined by measuring the directions from the vehicle to the known points, i.e., Attitude Knowledge. Due to accuracy limit, measurement noise, model error, and process error, most deterministic approaches are inefficient for accurate prospects; in this situation, using statistical methods will be a good solution. Dynamic attitude determination methods, also known as Attitude estimation, refer to using mathematical methods and techniques (e.g., statistical and probabilistic) to predict and estimate the future attitude based on a dynamic model and prior measurements. These techniques fuse data that retain a series of measurements using algorithms such as filtering, Multi-Sensor-Data-Fusion. The most common use attitude estimation methods are the Extended Kalman Filter, Madgwick, and Mahony.

4 | METHODOLOGY

4.1 | Error Matrices

Quaternions can be defined as a rotation around a unit vector by Bani Younes et al. (2012):

$$q = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) \cdot \hat{u} \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) \cdot u_x \\ \sin(\theta/2) \cdot u_y \\ \sin(\theta/2) \cdot u_z \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{12}$$

where θ is the angle of rotation and $\hat{u}$ is the unit vector of the rotation axis. Attitude could be defined as a rotation from the true orientation to the estimated orientation.

$$\mathbf{q}_{est} = \delta\mathbf{q} \otimes \mathbf{q}_{true} \tag{13}$$

where $\mathbf{q}_{est}$ is the estimated quaternion, $\mathbf{q}_{true}$ is the true quaternion, and $\delta\mathbf{q}$ is the quaternion error. The quaternion rotational error could be defined by:

$$\mathbf{q}_{err} = \mathbf{q}_{true} \otimes \mathbf{q}_{est}^{-1} \tag{14}$$

The error rate is represented by the following:

$$\dot{\mathbf{q}}_{err} = \dot{\mathbf{q}}_{true} \otimes \mathbf{q}_{est}^{-1} + \mathbf{q}_{true} \otimes \dot{\mathbf{q}}_{est}^{-1} \tag{15}$$

A simple way to calculate the error is to use element-wise subtraction between the true and estimated quaternions.

$$\mathbf{q}_{err} = \mathbf{q}_{true} - \mathbf{q}_{est} \tag{16}$$

As the attitude error is a geometric quantity, it is not reasonable to use Algebraic error matrices such as the mean squared error (MSE) or the mean absolute error (MAE). There are multiple ways to define and implement the quaternion error; in the following section, we will discuss the most common methods. The Quaternion Inner Product (QIP) of two quaternions represents the angle between the predicted and true orientation. This makes the dot product equal to the angle between two points on the quaternion hyper-sphere. The quaternion inner product is defined as:

$$QIP\,(q, p) = q \cdot p = q_w p_w + q_x p_x + q_y p_y + q_z p_z \tag{17}$$

The QIP will return the quaternion difference between two quaternions, so if the angle between two quaternions equals 0, the QIP value will equal 1. Thus, the QIP Loss Function can be defined as:

$$L_{QIP} = \frac{1}{N} \sum_{i=1}^{N} (1 - |q \cdot p|) \tag{18}$$

On the other hand, the angle between two quaternions can be calculated using the quaternion's inner product by:

$$L_{QIPA} = \frac{1}{N}\sum_{i=1}^{N}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\arccos(q \cdot p)) \tag{19}$$

In Brotchie et al. (2022), authors used the combination of QIP and MSE by:

$$L_{QIP-MSE} = \frac{1}{N}\sum_{i=1}^{N} QIP\left(\{q^i_{true} - q^i_{est}\}, \{q^i_{est} - q^i_{true}\}\right) \tag{20}$$

Authors in Silva do Monte Lima et al. (2019) used the Quaternion Multiplicative Error (QME) loss function to evaluate the performance of the proposed method using the Hamilton product by:

$$L_{QME} = \frac{1}{N}\sum_{i=1}^{N}(2 \cdot \|imag(q \otimes p^{\star})\|_1) \tag{21}$$

where $p^{\star}$ is the complex conjugate of the quaternion p. The complex conjugate of a quaternion can be calculated by:

$$p^{\star} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \end{bmatrix}^T \tag{22}$$

Another way is to calculate the angle corresponding to the QME by:

$$L_{QMEA} = 2 \cdot \arccos(|scalar(q \otimes p^{\star})|) \tag{23}$$

By using $arccos$ function in the implementation, the scalar part of $(q \otimes p^{\star})$ may lead to a value greater than 1 or less than -1, which could cause the gradient to explode. To overcome this problem, the scalar part of $(q \otimes p^{\star})$ is clamped to the range of [-1, 1]. The value clipping could lose the information about the angle between two quaternions, so another approach is replacing the $arccos$ function with a non-trigonometric function linear function to avoid exploding the gradient. This could be defined by:

$$L_{QMEAnT} = 1 - \sqrt{(|scalar(q \otimes p^{\star})|)^2} \tag{24}$$

where q_w and q_z are the squared values of $q \otimes p^{\star}$. Also in Laidig et al. (2021), authors decomposed the attitude error into a rotation around the z-axis, e_h, and the shortest residual rotation, e_i.

$$e_h = 2\arctan(q_z/q_w) \tag{25}$$

$$e_i = 2\arccos(\sqrt{q_w^2 + q_z^2}) \tag{26}$$

$$L_{e_i} = \frac{1}{N}\sum_{i=1}^{N}\left(2\arccos(\sqrt{q_w^2 + q_z^2})\right) \tag{27}$$

Using $arccos$ function could cause instability in the gradient calculation. As mentioned in Weber et al. (2020), the $arccos$ function could be replaced by:

$$1 - \sqrt{q_w^2 + q_z^2} \tag{28}$$

so the loss function can be rewritten as:

$$L_{e_i} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \sqrt{q_w^2 + q_z^2}\right) \tag{29}$$

The Quaternion Shortest Geodesic Distance (QSGD) is defined as the angle between the predicted and true orientation using the shortest geodesic distance on the quaternion hyper-sphere, which is defined as:

$$QSGD = q \otimes p^{\star} = \begin{bmatrix} q_w p_w - q_x p_x - q_y p_y - q_z p_z \\ q_w p_x + q_x p_w + q_y p_z - q_z p_y \\ q_w p_y - q_x p_z + q_y p_w + q_z p_x \\ q_w p_z + q_x p_y - q_y p_x + q_z p_w \end{bmatrix} \tag{30}$$

The corresponding loss function is defined as:

$$L_{QSGD} = |1 - (|scalar(q \otimes p^{\star})|)| \tag{31}$$

or,

$$L_{QSGD2} = \sqrt{1 - \sqrt{scalar(q \otimes p^{\star})^2}} \tag{32}$$

By considering the shortest rotation angle between two quaternions as $q \otimes p^{\star}$, it could be decomposed as:

$$q \otimes p^{\star} = \begin{bmatrix} \cos(\theta_{err}/2) \\ \sin(\theta_{err}/2) \cdot \hat{u_{err}} \end{bmatrix} = \begin{bmatrix} w_{err} \\ x_{err} \\ y_{err} \\ z_{err} \end{bmatrix} \tag{33}$$

where $\hat{u}$ is the unit vector of the rotation axis, and θ is the rotation angle. In the case of $\theta = 0$, the quaternion difference is equal to $q \otimes p^{\star} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$. So, to define the loss function that minimizes the rotation angle between two quaternions, we could use the following loss function:

$$L_{QSGD3} = \begin{bmatrix} w_{err} - 1 \\ x_{err} \\ y_{err} \\ z_{err} \end{bmatrix} \tag{34}$$

In figures 1, 2a, and2b loss values for the attitude error from $|\pi|$ to 0 is plotted.

4.2 | Sequential Modeling

Time series data is a type of sequential data consisting of observations taken at regular intervals over time and can be used for tasks such as forecasting, anomaly detection, and classification; also, IMU measurements could be considered as time series data. Time series estimation is the process of predicting future values in a time series based on past observations. So, it could be considered a method to predict the future orientation of the IMU sensor. A solution for time series estimation is using a deep learning model that can learn the relationship between the input and output data, called sequential modeling. Sensor data, text, sound, and specific data with an underlying sequential structure can be handled with sequence models for several applications, including time series data prediction Rajagukguk et al. (2020), speech recognition Graves et al. (2013), natural language processing Schoene et al. (2022), music generation Marinescu (2019), and DNA sequence analysis Shen et al. (2018). The traditional neural network models cannot

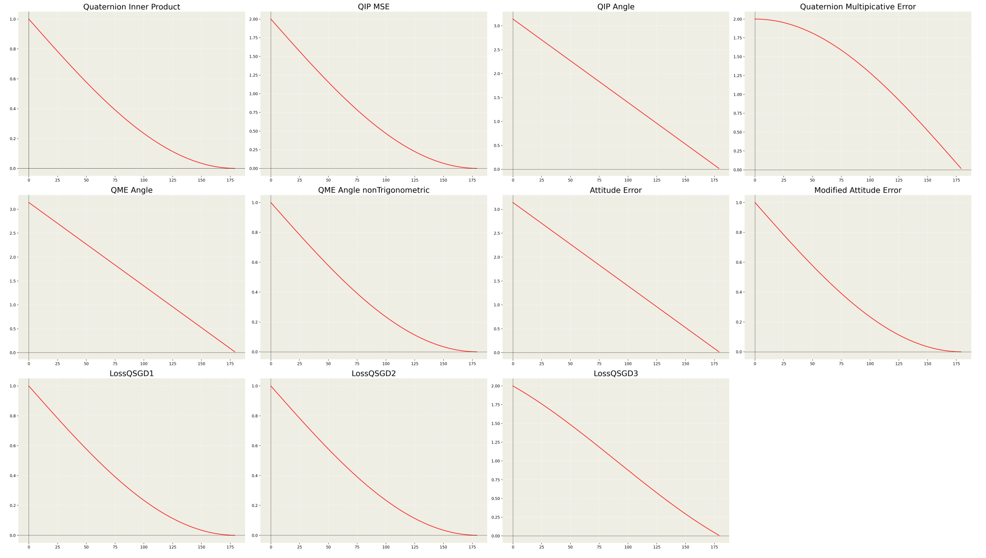
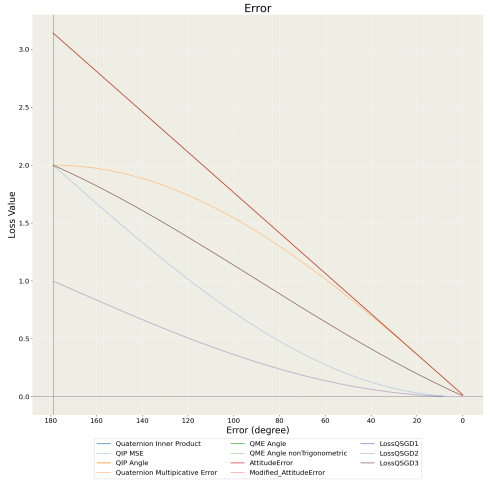
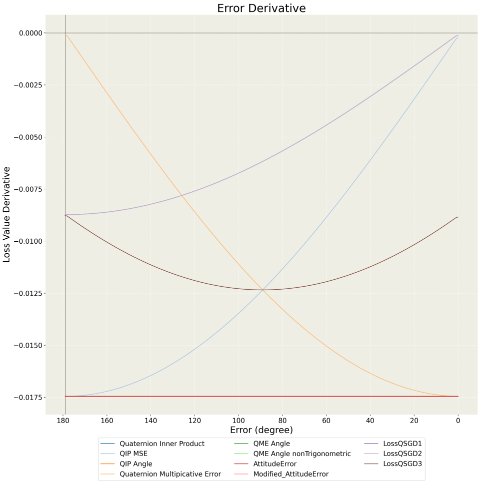


FIGURE 1 Loss functions for attitude error



(a) Compare Loss functions for attitude error



(b) Compare Loss functions derivative for attitude error

handle time-series data as they do not loop and handle time dependencies between them. Common models used for sequential modeling include Recurrent Neural Networks (RNNs), LSTM networks, Gated Recurrent Units (GRUs), and Temporal Convolutional Networks (TCNs). RNNs capture long-term dependencies in data but can be slow due to back-propagation through time; LSTMs use memory cells to store information from previous inputs; GRUs use gating mechanisms to control the flow of information, and TCNs use dilated causal convolutions, which allow them to learn patterns over longer sequences while maintaining the computational efficiency of traditional CNNs. Each model has its strengths and weaknesses; the best depends on the specific task.

4.3 | Deep Learning Model

Neural network models can find the hidden relation between the input and output data, and they are an efficient way to handle sequential data such as IMU sensor measurements. There is a wide variety of neural network architectures, each of which has characteristics and advantages. A simple model is FFNN which is suitable for classification problems. CNN is a proper choice for signal processing and extracting features from the input data. Due to its lack of memory, it could not store data from previous time steps. RNN is a type of neural network with memory that can store the data from previous time steps. GRU is a type of RNN with a gating mechanism that controls the flow of information. GRU has two gates which are the reset gate and the update gate. The reset gate controls the flow of information from previous time steps, and the update gate controls the flow of information from the current time step. LSTM is a variant type of RNN capable of learning long-term dependencies and has three gates: input gate, forget gate, and output gate. The input gate controls the information that enters the cell state; the forget gate controls the information that leaves the cell state, and the output gate controls the information that is output. TCN is another type of CNN suitable for sequential data. TCN is a stack of dilated convolutional layers with residual connections that extract features from the input data. The residual connections are used to preserve the information from previous time steps. In the following, we will discuss each one in detail.

4.3.1 | Convolutional Neural Network

A convolutional neural network is an artificial neural network that analyzes data with a spatial or temporal structure. CNN extract features from input data using convolutions, which are mathematical operations that allow the network to extract features. CNNs are widely used in image processing, computer vision, and natural language processing. Various filters are used in a convolutional layer to detect specific features in the input data. The output of these filters is then processed by a nonlinear activation function such as the ReLU or the sigmoid before being fed to another layer of the network. This layer can be used for object recognition, segmentation, and classification by learning patterns from images or videos using multiple layers with different parameters and associated weights. Recent advances in deep learning have shown that CNNs can also be used for time series prediction LI et al. (2022); Chiang and Horng (2021) and nonlinear regression Fan et al. (2022). CNNs are composed of convolutional layers and pooling layers. The convolutional layers are used to extract features from the input data, and the pooling layers are used to reduce the dimensionality. The CNN architecture is shown in fig. 3. The equation for the convolutional layer is defined as Koushik (2016):

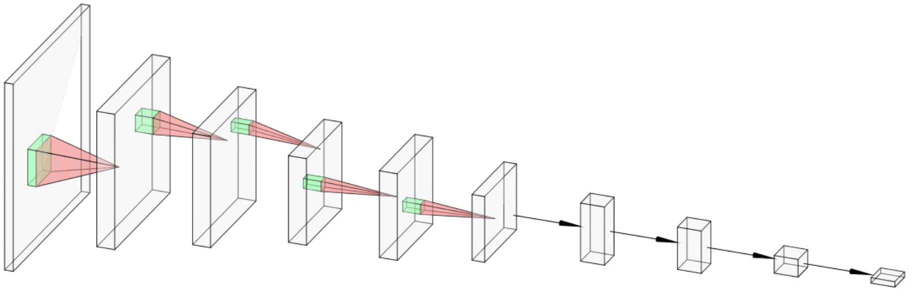


FIGURE 3 Convolutional Neural Network.
This image was created at http://alexlenail.me/NN-SVG/AlexNet.html

$$y_k \equiv \sum_{i=k}^{k+W-1} x_i w_{k+W-i} \tag{35}$$

where x is the input data, w is the filter size. Usually, after the convolutional layer, a pooling layer is used to reduce the dimensionality of the data by extracting the most important features from the input data. The pooling layer is composed of a pooling function and a pooling window. The pooling function is used to extract the most important features from the input data, and the pooling window is used to reduce the dimensionality of the data. The pooling layer is shown in fig. 4. A pooling layer reduces the spatial size of an input feature map, allowing the model to reduce

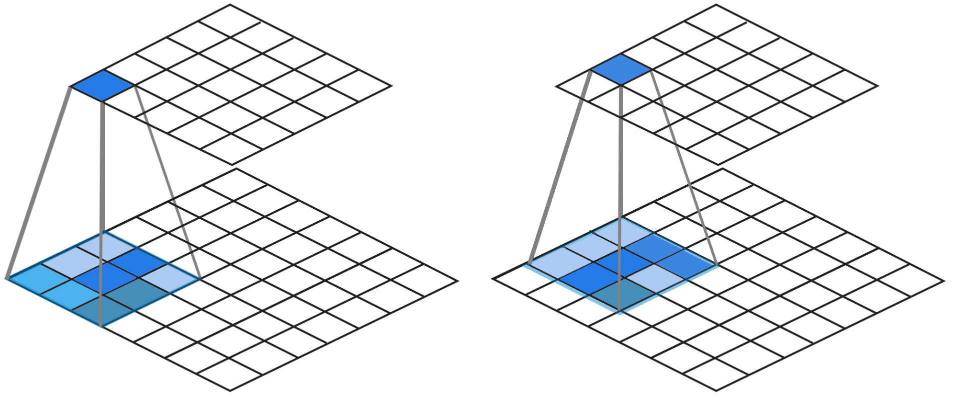


FIGURE 4 Pooling Layer.

the number of parameters and computations required while preserving essential features from the original data. Several types of pooling layers include max pooling, average pooling, and global average pooling. Max Pooling takes the maximum value from each region of an input feature map, while Average Pooling takes the mean value from each region. Global Average Pooling reduces a feature map to a single number by taking the mean across all regions in the input feature map He et al. (2015). Through this method, CNNs can transform the original input layer by layer using convolutional and down-sampling techniques to produce class scores for classification and regression.

4.3.2 | Recurrent Neural Network

Recurrent neural networks are artificial neural networks for processing sequential data. RNNs can remember information from previous inputs, allowing them to process data sequences such as text or audio. They consist of multiple layers with neurons connected cyclically so that the output from one layer becomes an input for another layer and vice versa. This allows them to capture long-term dependencies between elements in the sequence by passing information through time steps over multiple layers. RNNs can be used for tasks such as language translation and speech recognition by learning patterns in the input sequence over time using different parameters and weights associated with each neuron within each layer. The RNN architecture is shown in fig. 5. The equation for the RNN is defined as Schmidt (2019):

$$h_t = \sigma(W_{hh} h_{t-1} + W_{xh} x_t + b_h) \tag{36}$$

where h is the hidden state, x is the input data, W_{hh} is the weight matrix for the hidden state, W_{xh} is the weight matrix for the input data, and b is the bias.

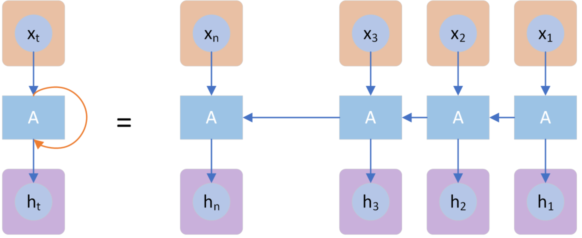


FIGURE 5 Recurrent Neural Network

Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network that uses memory cells to store information from previous inputs, which allows the model to remember patterns over long sequences and make predictions based on them. So, it is considered a time domain deep learning model Cai et al. (2021). They are composed of three main components: an input gate, a forget gate and an output gate. The input gate controls which values are added to the cell state; the forget gates control which values are removed from it; and the output gates control what is passed out as output for each time step in sequence processing tasks such as machine translation or speech recognition. The main advantage of LSTMs is that they can learn long-term dependencies in the input data. The LSTM architecture is shown in fig. 6a. Its equations are as follows Hochreiter and Schmidhuber (1997): Input Gate:

$$i_t = \sigma(W_i * [h_(t-1), x_t] + b_i) \tag{37}$$

Forget Gate:

$$f_t = \sigma(Wf * [h_(t-1), x_t] + bf) \tag{38}$$

Output Gate:

$$o_t = \sigma(Wo * [h_(t-1), x_t] + bo). \tag{39}$$

Cell State Update :

$$c_t = f_t * c_(t-1) + i_t * tanh(Wc * [h_(t-1), x_t] + bc). \tag{40}$$

Gated Recurrent Unit (GRU)

Gated Recurrent Unit is a type of recurrent neural network similar to LSTM networks. GRUs use gating mechanisms to control the flow of information, allowing them to learn long-term dependencies in data more effectively than traditional RNNs. They also have fewer parameters than LSTMs, making them faster and easier to train. The GRU architecture is shown in fig. 6b. Its equations are as follows Cho et al. (2014): Update Gate:

$$z_t = \sigma(Wz * [h_(t-1), x_t] + bz) \tag{41}$$

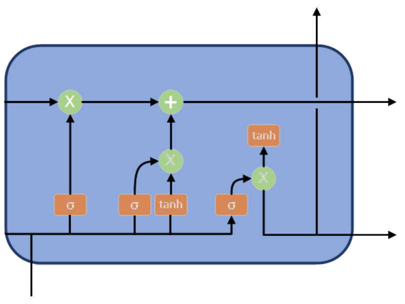
Reset Gate:

$$r_t = \sigma(Wr * [h_(t-1), x_t] + br) \tag{42}$$

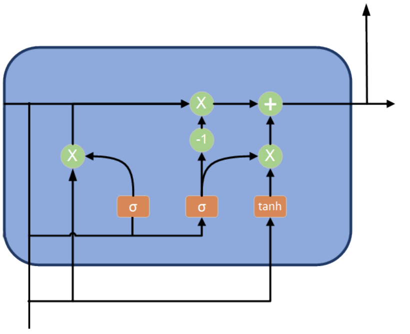
Hidden State Update :

$$h_t = z_t * h_(t-1) + (1 - z_t) * W * [r_t, x_t] + b \tag{43}$$

LSTMs use memory cells to store information from previous inputs, allowing the model to remember patterns over



(a) Long Short-Term Memory (LSTM)



(b) Gated Recurrent Unit (GRU)

long sequences and make predictions based on them. The input gate controls which values are added to the cell state, the forget gates control which values are removed from it, and the output gates control what is passed out as output for each time step in sequence processing tasks such as machine translation or speech recognition. GRUs also use gating mechanisms but have fewer parameters than LSTMs, making them faster and easier to train. The update gate controls how much of a new value should be stored in a hidden state while resetting any previously stored information that is no longer relevant; this allows GRUs to learn long-term dependencies more effectively than traditional RNNs without sacrificing speed or accuracy. Both models can be used for tasks such as machine translation or speech recognition, but which is better depends on the specific task at hand; in some cases, one may outperform the other, while in others, they may perform similarly.

Temporal Convolutional Network (TCN)Moor et al. (2019)

Temporal Convolutional Networks are a type of convolutional neural network used for sequence modeling tasks such as machine translation, speech recognition, and time series forecasting. TCNs use dilated causal convolutions to capture long-term dependencies in data while maintaining the computational efficiency of traditional CNNs. The model is composed of multiple layers with increasing dilation factors; this allows it to learn patterns over longer sequences without sacrificing accuracy or speed. The TCN architecture is shown in fig. 7. TCN equation is as follows Lea et al. (2017):

$$y_t = f(W * x_t + b), \tag{44}$$

where W is the weight matrix, x_t is the input at time t, and b is the bias vector. TCN works by applying convolutional filters to the input data in a temporal manner rather than in a spatial manner, as in traditional CNNs. The input data is passed through a series of convolutional layers, each with a different kernel size and a number of filters. The kernel size determines the number of time steps the convolutional filters are applied to, allowing the model to capture longer-term dependencies in the input data. The number of filters determines the number of output feature maps generated by the convolutional layers. TCN also includes dilated convolutions, which allow the model to increase the receptive field of the filters without increasing the number of parameters. This allows the model to capture longer-range dependencies in the input data without increasing the computational complexity. Overall, TCN is a powerful tool for handling sequential data and has been applied to a variety of tasks, such as natural language processing, time series forecasting, and speech recognition.

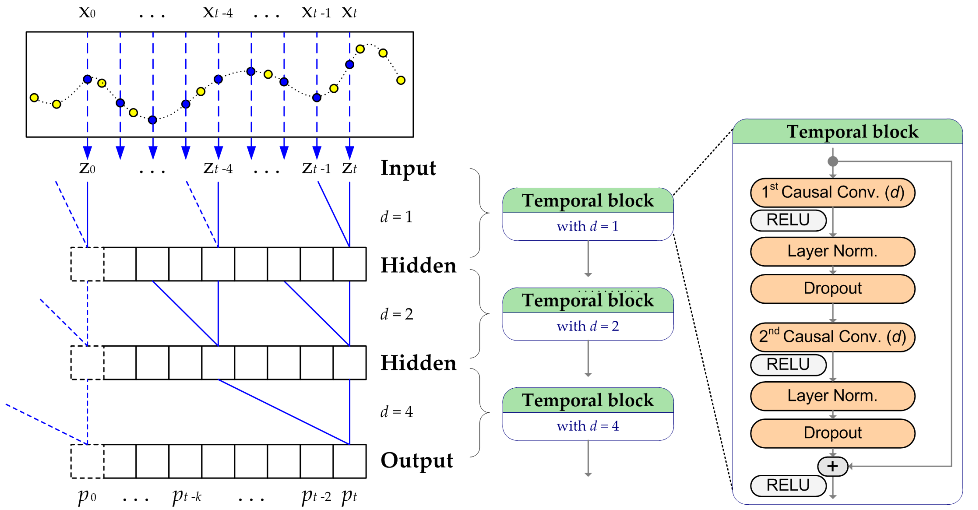


FIGURE 7 Temporal Convolutional Network (TCN)

Bidirectional Layer

A bidirectional layer is a type of neural network layer that processes data in both directions, and it is proposed to overcome the accumulative error problem. This allows the model to consider information from past and future observations when making predictions. Schuster and Paliwal first proposed the concept of a bidirectional layer in 1997 Schuster and Paliwal (1997). Since then, bidirectional layers have become a popular component in many neural network architectures due to their ability to consider information from past and future observations when making predictions. A bidirectional layer will combine two separate layers which process data in opposite directions (forward and backward) to form an output vector. The outputs from these two layers are combined to form a single output vector that can be used for prediction or classification tasks. The bidirectional layer architecture is shown in fig. 8.

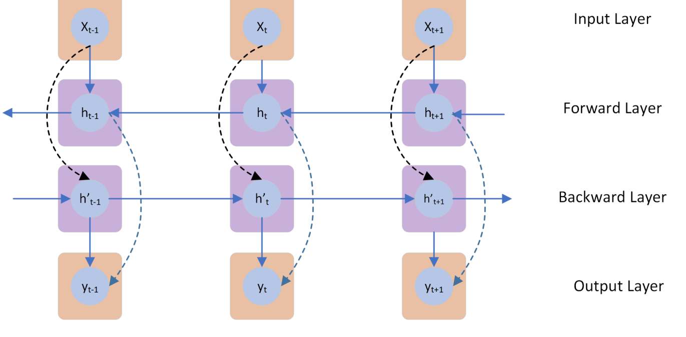


FIGURE 8 Bidirectional Layer

4.4 | Activation Function

Activation functions are used to introduce non-linearity into neural networks. They are used to transform the output of a neuron into a form that can be used for further processing. It is a mathematical operation that takes an input and produces an output. In machine learning, the activation function determines how a neural network should respond to specific inputs. This allows the model to learn more complex patterns and better represent data. Activation functions also help control the flow of information through a neural network, allowing it to make decisions based on certain inputs. Different activation functions have other properties that can be used for different tasks, such as classification or regression problems. In figures 9 and 10, the most common activation functions have been shown. Activation

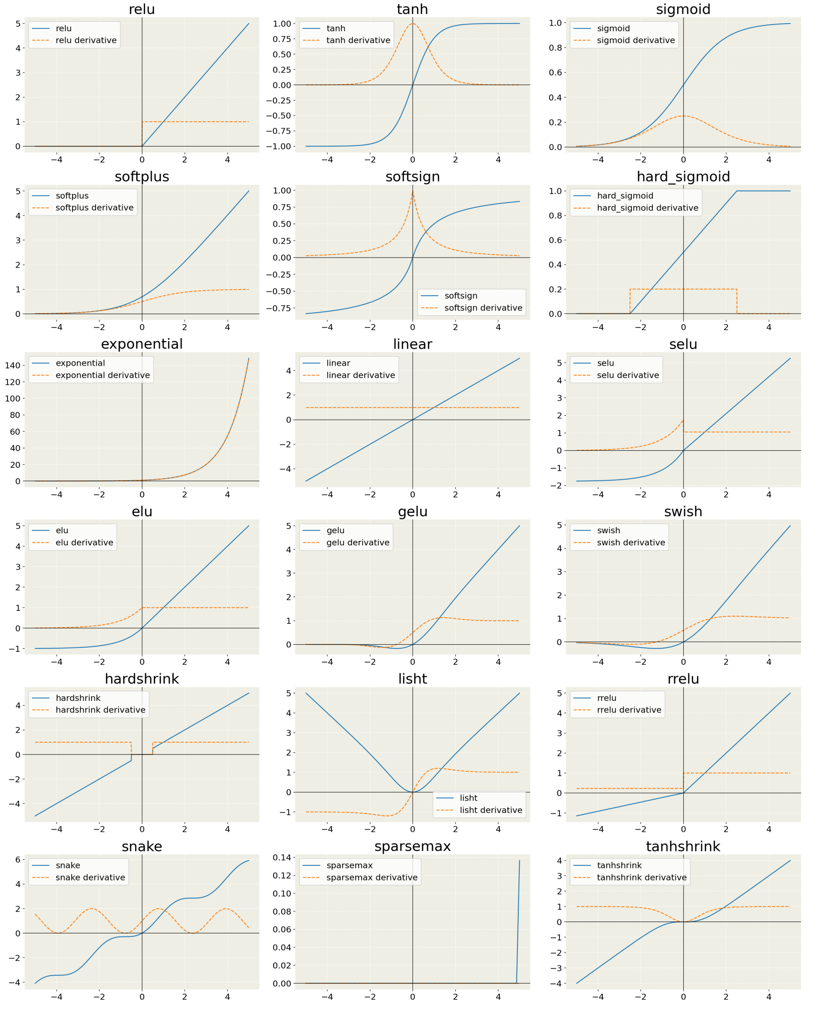


FIGURE 9 Activation Functions

functions can be categorized into two groups: Piecewise Linear Function and Locally Quadratic Function Rasamoelina et al. (2020). Piecewise Linear Activation Functions are functions composed of a limited number of linear segments, each defined over an equal number of intervals. These functions are used in Artificial Neural Networks to provide the necessary non-linearity for the model to learn complex representations. Examples of Piecewise Linear Activation Functions include ReLU (Rectified Linear Unit), which has a constant first-order derivative and no curvature in each interval defined by its breakpoint. A locally quadratic function is a mathematical function that can be approximated by a quadratic equation in a certain area. Non-linear, smooth activation functions with nonzero second derivatives are locally quadratic. This means that the function can be represented by a parabola in a certain region, but may not be a perfect parabola everywhere. In the following, the most common activation functions for time series forecasting and regression are discussed.

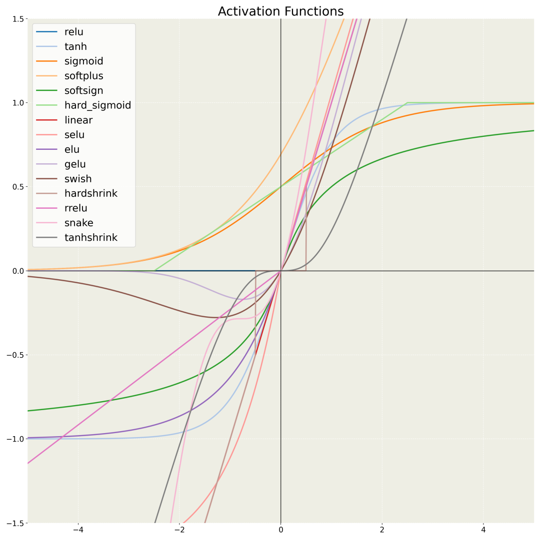


FIGURE 10 Compare Activation Functions

4.4.1 | Sigmoid Function

The sigmoid function takes an input $(-\infty, \infty)$ and produces an output between 0 and 1, which can be interpreted as a probability or likelihood that the given input belongs to one class or another. It defines by:

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{45}$$

where e is the base of natural logarithms and x is the independent variable. The output of this equation ranges from 0 to 1, with values close to zero representing low probabilities and values close to one representing high probabilities. The sigmoid function has been widely used for binary classification tasks, but it has also been successfully applied to multi-class problems.

4.4.2 | Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) is an activation function used in neural networks. It takes an input and produces an output that is either 0 or the same as the input, depending on whether the input is negative or positive. The ReLU function can be defined by:

$$f(x) = max(0, x), \tag{46}$$

where x is the independent variable. The output of this equation ranges from 0 to infinity, with values close to zero representing low probabilities and values closer to infinity representing high probabilities. ReLUs are often used for tasks such as image recognition because they allow for faster training times than other activation functions like sigmoid or tanh while still providing good accuracy results. However, they are unsuitable for tasks requiring negative values, such as regression problems.

4.4.3 | Hyperbolic Tangent (tanh)

The Hyperbolic Tangent (tanh) is an activation function introduced by Sepp Hochreiter Hochreiter and Schmidhuber (1997). It takes a real-valued input and produces an output between -1 and 1, which makes it useful for classification tasks. The tanh can be used as an alternative to the sigmoid for training deep neural networks and helps reduce the vanishing gradient problem associated with other activation functions such as ReLU or ELU. The tanh has been found to work better than sigmoid in some cases due to its wider range of outputs but may suffer from saturation issues when dealing with large inputs.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{47}$$

4.4.4 | Leaky ReLU

The Leaky ReLU is a type of activation function used in neural networks. It takes an input and produces an output that is either 0 or the same as the input, depending on whether the input is negative or positive. The difference between a regular ReLU and a Leaky ReLU lies in how it handles negative inputs. Instead of returning 0 for all negative values, it produces a small fraction (the "leak") of those values. This allows flexibility when training models with large datasets containing positive and negative examples. The Leaky ReLU can be defined as:

$$f(x) = max(0, x) + \alpha * min(0, x) \tag{48}$$

where α represents the leak parameter (usually set to 0.01), this equation has two parameters: α (the leak parameter) and x (the independent variable). The output of this equation ranges from $-\alpha$ to infinity, with values close to zero representing low probabilities and values closer to infinity representing high probabilities. Leaky ReLUs are often used for tasks such as image recognition because they allow for faster training times than other activation functions like sigmoid or tanh while still providing good accuracy results.

4.4.5 | Exponential Linear Unit (ELU)

The Exponential Linear Unit (ELU) is an activation function proposed by Djork-Arné Clevert Heusel et al. (2015). It is similar to the ReLU activation function but has a negative part which allows for more efficient training of deep neural networks. The ELU also helps reduce the vanishing gradient problem associated with other activation functions such as sigmoid or tanh.

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases} \tag{49}$$

4.4.6 | Swish

Swish is an activation function proposed by Google Brain Ramachandran et al. (2017). It takes a real-valued input and produces an output between 0 and 1, which makes it useful for classification tasks. The Swish has a learnable parameter that allows for more efficient training of deep neural networks compared to other activation functions such as ReLU or ELU. The Swish also helps reduce the vanishing gradient problem associated with different activation functions such as sigmoid or tanh, making it suitable for use in deeper networks where gradients can become very small over multiple layers.

$$f(x) = x * \sigma(\beta * x) \tag{50}$$

where β represents the learnable parameter.

4.4.7 | RRelu (Randomized ReLU)

The Randomized ReLU (RReLU) takes a real-valued input and produces an output between 0 and 1, which makes it useful for classification tasks. The RReLu has two learnable parameters that allow for more efficient training of deep neural networks compared to other activation functions such as ReLU or ELU. The RReLu also helps reduce the vanishing gradient problem associated with different activation functions such as sigmoid or tanh, making it suitable for use in deeper networks where gradients can become very small over multiple layersXu et al. (2015).

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha * x & \text{if } x < 0 \end{cases} \tag{51}$$

where α represents the lower bound of the random uniform distribution and β represents the upper bound of the random uniform distribution.

4.4.8 | Mish

The Mish activation function is a type of activation function used in neural networks. It takes an input and produces an output that is either 0 or the same as the input, depending on whether the input is negative or positive Misra (2019). The Mish activation function combines elements from both ReLU and tanh functions to create a more robust non-linearity than either one alone. The Mish activation function is shown in figure 11. The equation for this can be defined as:

$$f(x) = x * tanh(ln(1 + e^x)) \tag{52}$$

where x represents the independent variable, the output of this equation ranges from -infinity to infinity, with values close to zero representing low probabilities and values closer to infinity representing high probabilities. Mish activation functions are often used for tasks such as image recognition because they allow faster training times than other activation functions like sigmoid or tanh while still providing good accuracy results. We choose the Mish activation function because it has been shown to outperform other activation functions, such as ReLU and ELU, in terms of accuracy and training time. It is also a smooth function that does not suffer from the vanishing gradient problem, which makes it suitable for use in deeper networks where gradients can become very small over multiple layers Weber et al. (2021); Noel et al. (2021); Dubey et al. (2022).

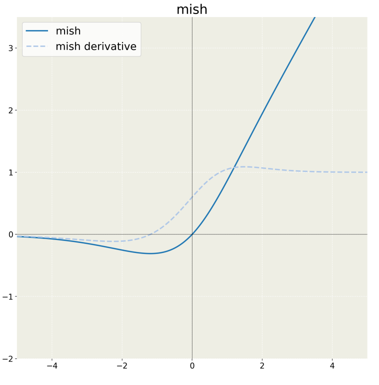


FIGURE 11 Mish Activation Function

4.5 | Proposed Network Architecture

Various deep learning topologies may be used to tackle the real-time attitude estimation problem. Previous studies demonstrate the performance of RNNs, (LSTM and GRU), CNN, and hybrid RNN-CNN networks for handling IMU data to estimate the system state variable. Furthermore, the computational cost is one of the main criteria for choosing the topologies, as it must be able to be used in real-time. We tested GRU-based, LSTM-based, CNN-based, and hybrid CNN-based networks to estimate the system's attitude. Based on the results, we chose three different models. The proposed network architectures is shown in fig. 14, fig. 15, and fig. 16. Similar to Chen et al. (2018a); Silva do Monte Lima et al. (2019); Kim et al. (2021); Yan et al. (2018); Herath et al. (2020), the input data will be fed into the model in windows of $\mathcal{N}$ frames, containing 3-axis acceleration and 3-axis angular velocity. So, the past $\frac{\mathcal{N}}{2}$ and future $\frac{\mathcal{N}}{2}$ IMU measurements are used to predict the attitude. The consecutive IMU measurements have a stride size of $\mathcal{S}$ and new attitude will be calculated in every $\mathcal{S}$ frames. So, the attitude estimation will be occurred between $\frac{\mathcal{N}}{2} - \mathcal{S}$ and $\frac{\mathcal{N}}{2} + \mathcal{S}$ frames as shown in the fig. 12. The proposed network architecture is composed of four main components:

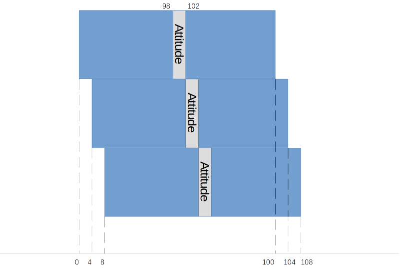


FIGURE 12 Time window for attitude estimation, both past and future data are used to estimate the attitude at each time step.

(1) Feature Extraction, (2) Feature Fusion,(3) sampling rate fusion, and (4) Attitude Estimation. Gaussian noise layers are added to model's inputs to help with generalization, by adding random noise to the input data during training. This can help the model become more robust to small variations in the input, and can also help prevent overfitting. Additionally, it can also serve as a regularization technique that helps to reduce the overfitting of a model by adding random noise to the inputs. This noise is sampled from a Gaussian distribution with standard variation of 0.25. The feature extraction component extracts the features from the IMU data. To do so, we split each axis of the IMU data in Model A and B, and then fed each into the layer. After concatenating the layers, they are fed to a layer to fuse the extracted feature. To consider the sampling rate, the fused features are connected to the sampling rate layer. In the last layer, a FFN with four units is followed by a unit scaling layer to estimate the attitude. To improve the robustness of the network, we used dropout layers in the feature extraction and feature fusion layers.

4.5.1 | Model A

Model A 14 consists of separate 1D-CNN layers with filter size of 128 and kernal size of 11 for each axis of the IMU data. Each CNN layer is followed by a max pooling layer with a pooling size of 3, and all the output is concatenated and fed into a CNN layer with 128 filters, kernel size of 11, and stride size of 1. Also, we used causal padding to prevent information leakage from the future to the past (fig 13). This CNN layer fuses the extracted features in the last layer and is followed by a feed-forward layer with 512 units. To take advantage of sequence modeling and temporal information, we used a bidirectional LSTM layer with 128 units. A dropout reduces overfitting by randomly dropping out (or "disabling") %20 neurons during training, which forces the network to learn more robust representations of the data and reduces reliance on any single neuron or feature. This helps prevent it from memorizing specific patterns in the training set and improves its generalizability when applied to new data. The extracted temporal features concatenated with CNN outputs and the output of a fully connected layer. The fully connected layer has 512 neurons. Its input

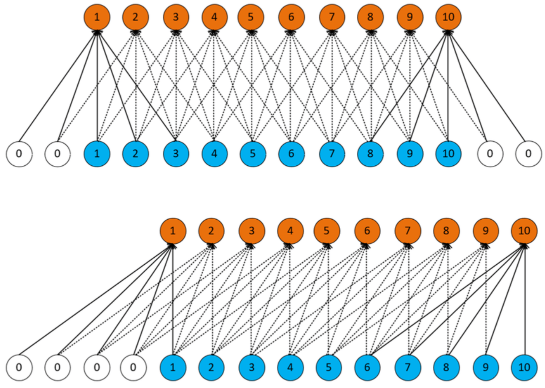


FIGURE 13 Same (top) vs causal (bottom)Collado-Villaverde et al. (2021)

is the sampling rate of the IMU measurements and works as the sampling rate fusion component, which is used to fuse the extracted features from the IMU data with different sampling rates. The Attitude Estimation component is composed of a fully connected layer with four neurons representing the system's estimated attitude in quaternion form.

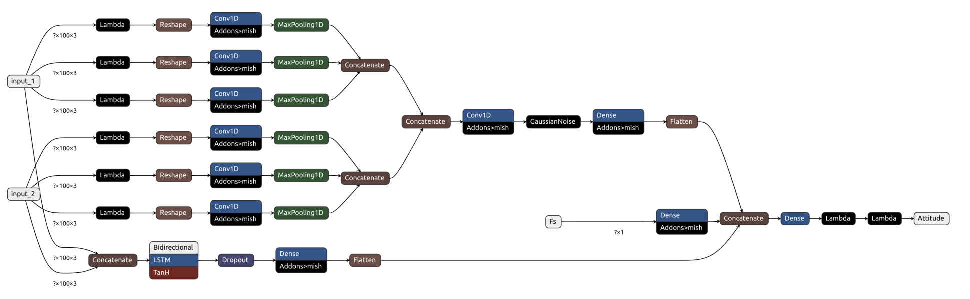


FIGURE 14 Proposed Network Architecture of Model A

4.5.2 | Model B

Model B 15 consists of multiple Bidirectional LSTM layers with 50 units, each one followed by a dropout layer. The LSTM layer's output is concatenated and fed into a feed-forward layer with 256 units and ReLU activation function. The sampling rate of IMU sensors is fed into a dense layer with 256 units and ReLU activation function. The outputs of dense layers are concatenated and fed into a dense layer with four units and linear activation function followed by a unit scaling layer to estimate the quaternions.

4.6 | Model C

Model C 16 consists of two Bi-LSTM layer, followed by a dense layer with 256 units. The sampling rate fed into a similar dense layer with Mish activation function. The output of the dense layer concatenated and fed into another dense layer with 256 units. The output consits of a dense layer with four units and linear activation function followed by a unit scaling layer to estimate the quaternions.

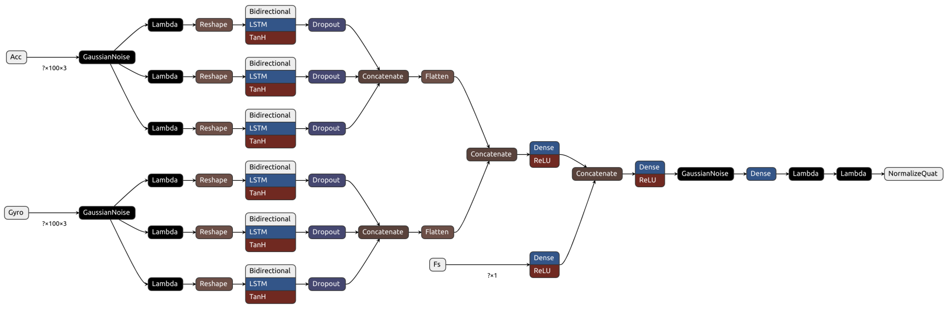


FIGURE 15 Proposed Network Architecture of Model B

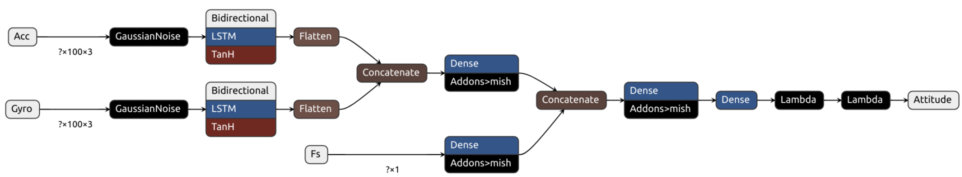


FIGURE 16 Proposed Network Architecture of Model C

4.7 | Learning Rate Finder

We used the learning rate finder technique to find the proper learning rate for the model. A learning rate finder determines the optimal learning rate for training a machine learning model. This involves gradually increasing the learning rate from a minimal value and monitoring how quickly or slowly the loss decreases over time Donini et al. (2019). The point at which the loss begins to decrease more slowly than before indicates that this is an optimal value for training, as it allows models to explore different parts of parameter space without getting stuck in local minima. To use this technique, one must first set up an experiment with multiple runs using different values of learning rates and monitor their performance on validation data sets. After all experiments have been completed, the best-performing run can be chosen as the optimal training value. With the learning rate finder, a model can converge faster and more efficiently than traditional methods, such as constant decay or exponential decay schedules, since it can find the optimal learning rate faster. Commonly used techniques include constant, exponential decay, step-wise decay, and cyclical learning rates (CLR). Constant schedules keep the same value for all training iterations, while exponential decay reduces it gradually over time. The concept was first proposed by Leonid Khachiyan in 1980 Khachiyan (1980). He presented the use of an exponentially decreasing learning rate to improve the convergence speed and accuracy of gradient descent algorithms. Its calculation is as follows:

$$lr = lr * decay\_rate^{step/decay\_step}, \tag{53}$$

where lr is the current learning rate, step is the current training iteration, and decay_step determines how often the learning rate is reduced. Step-wise decays involve reducing the learning rate at specific points during training. In Hinton et al. (2012), G. Hinton proposed using a step-wise decay schedule to facilitate the learning rate at certain points during training. This technique has since become a popular method for improving model performance and avoiding local minima. It can be calculated using the following equation:

$$
\begin{gathered}
lr = lr * factor \\
or \\
lr = lr - fixedamount.
\end{gathered} \tag{54}
$$

In addition, CLR Smith (2017) involves gradually increasing and decreasing the learning rate over time, allowing the model to explore different parts of parameter space more efficiently. This is done by setting an upper and lower bound for the range of values that can be explored and a step size that determines how quickly or slowly the value changes between these bounds. The idea behind this approach is that it allows models to avoid local minima while still converging on an optimal solution faster than with traditional methods such as constant or exponential decay schedules. The equation for a cyclical learning rate is:

$$
lr = \frac{lower\_bound}{2} + \frac{upper\_bound}{2} * (1 + cos(step/stepsize)), \tag{55}
$$

where lr is the current learning rate, step is the current training iteration, and step size determines how quickly or slowly the value changes between upper and lower bounds. This study used the CLR method to find the optimal learning rate. Based on the CLR method, we train the network on a few epochs and plot (fig. 17) the loss value against the learning rate and choose the value in which the loss has the steepest gradient.

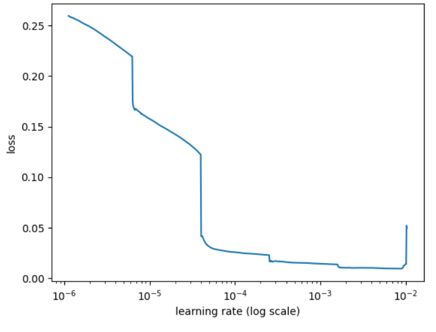


FIGURE 17 Learning Rate Finder

5 | EXPERIMENT

5.1 | Dataset

5.1.1 | Introduction

Inertial Measurement Unit (IMU) datasets are crucial for evaluating the performance of attitude estimation algorithms. This section will present a comprehensive overview of the most widely-utilized IMU datasets. These datasets are classified into two primary categories: synthetic and real-world. Synthetic datasets are generated through the simulation of IMU measurements, while real-world datasets are gathered through actual experiments. Real-world experiments are further divided into two subcategories: indoor and outdoor. Indoor experiments are conducted in a controlled environment, such as a laboratory, while outdoor experiments are conducted in an uncontrolled environment, such as a field setting. Additionally, when training, validating, and testing neural network models, a comprehensive database containing accurate input and output parameters is essential. The performance of a deep learning model is directly impacted by the quality of the data used for training. Thus, to train a deep learning model effectively, a dataset containing input and output parameters that meet stringent conditions must be utilized. In order to effectively train a deep learning model, it is imperative to utilize a dataset that meets certain criteria. Specifically, the dataset must possess:

- Adequate quantity of data to ensure proper model training.
- Diversity in data samples to account for various scenarios and edge cases.
- High level of accuracy in both input and output parameters to prevent errors and ensure model reliability.

It is important to note that determining the sufficiency of data quantity and diversity in data samples can be achieved through various techniques such as cross-validation and testing the model's generalization ability. Additionally, the dataset's accuracy can be established through techniques like data pre-processing and cleaning. We will present some of the most popular IMU datasets in the following section.

5.1.2 | RepoIMU T-stic

The RepoIMU T-stick Szczęsna et al. (2016) is a small, low-cost, and high-performance IMU that can be used for many applications. IT is a 9-axis IMU that measures acceleration, angular velocity, and magnetic field. This database contains two sets of experiments recorded with a T-stick and a pendulum. A total of 29 trials were collected on the T-stick, and each trial lasted approximately 90 seconds. As the name suggests, the IMU is attached to a T-shaped stick equipped with six reflective markers. Each experiment consists of slow or fast rotation around a principal sensor axis or translation along a principal sensor axis. In this scenario, the data from the Vicon Nexus OMC system and the XSens MTi IMU are synchronized and provided at a frequency of 100 Hz. The authors clearly state that the IMU coordinate system and the ground trace are not aligned and propose a method to compensate for one of the two required rotations based on quaternion averaging. Unfortunately, some experiments contain gyroscope clipping and ground tracking, significantly affecting the obtained errors. Therefore, careful pre-processing and removal of some trials should be considered when using the dataset to evaluate the model's accuracy.

5.1.3 | RepoIMU T-pendulum

The second part of the RepoIMU dataset Szczęsna et al. (2016) contains data from a triple pendulum on which the IMUs are mounted. Measurement data is provided at 90 Hz or 166 Hz. However, the IMU data contains duplicate samples. This is usually the result of artificial sampling or transmission problems where missed samples are replaced by duplicating the last sample received, effectively reducing the sampling rate. The sampling rate achieved when discarding frequent samples is about 25 Hz and 48 Hz for the accelerometer and gyroscope, respectively. Due to this issue, it is not recommended to use this database for model training and evaluation. Due to this fact, we cannot recommend using pendulum tests to evaluate the accuracy of Inertial Orientation Estimation (IOE) with high precision.

5.1.4 | Sassari

The Sassari dataset Caruso et al. (2020) published aims to validate a parameter tuning approach based on the orientation difference of two IMUs of the same model. To facilitate this, six IMUs from three manufacturers (Xsens, APDM, Shimmer) are placed on a wooden board. Rotation around specific axes and free rotation around all axes are repeated at three different speeds. Data is synchronized and presented at 100 Hz. Local coordinate frames are aligned by precise manual placement. There are 18 experiments (3 speeds, 3 IMU models, and 2 IMUs of each model) in this dataset. According to these points, this database seems to be suitable for training, evaluating, and testing the model, but some essential points should be paid attention to. The inclusion of different speeds and different types of IMUs helped to diversify the data set. However, all motions occur in a homogeneous magnetic field and do not involve pure translational motions. Therefore, this data set does not have a robust variety in terms of the type of movement and the variety of magnetic data. Thus, the model trained with it cannot be robust and general. However, it can be used to evaluate the model. The total movement duration of all three trials is 168 seconds, with the most extended movement phase lasting 30 seconds. For this reason, considering the short time, it is not suitable for training.

5.1.5 | Oxford Inertial Odometry Dataset

The Oxford Inertial Odometry Dataset (OxIOD) Chen et al. (2018b) is a large set of inertial data recorded by smartphones (mainly iPhone 7 Plus) at 100 Hz. The suite consists of 158 tests and covers a distance of over 42 km, with OMC ground track available for 132 tests. The purpose of this set is inertial odometry. Therefore, it does not include pure rotational movements and pure translational movements, which are helpful for systematically evaluating the model's performance under different conditions; however, it covers a wide range of everyday activities. Due to the different focus, some information (for example, the alignment of the coordinate frames) is not accurately described. In addition, the orientation of the ground trace contains frequent irregularities (e.g., jumps in orientation that are not accompanied by similar jumps in the IMU data).

5.1.6 | MAV Dataset

Most datasets suitable for the simultaneous localization and mapping problem are collected from sensors such as wheel encoders and laser range finders mounted on ground robots. For small air vehicles, there are few datasets, and MAV Dataset Lee et al. (2010) is one of them. This data set was collected from the sensor array installed on the "Pelican" quadrotor platform in an environment. The sensor suite includes a forward-facing camera, a downward-facing camera, an inertial measurement unit, and a Vicon ground-tracking system. Five synchronized datasets are presented

- 1LoopDown
- 2LoopsDown
- 3LoopsDown
- hoveringDown
- randomFront

These datasets include camera images, accelerations, heading rates, absolute angles from the IMU, and ground tracking from the Vicon system.

5.1.7 | EuRoC MAV

The EuRoC MAV dataset Burri et al. (2016) is a large dataset collected from a quadrotor MAV. The dataset contains the internal flight data of a small air vehicle (MAV) and is designed to reconstruct the visual-inertial 3D environment. The six experiments performed in the chamber and synchronized and aligned using the OMC-based Vicon ground probe are suitable for training and evaluating the model's accuracy. It should be noted that camera images and point clouds are also included. This set does not include magnetometer data, which limits the evaluation of three degrees of freedom and is only for two-way models (including accelerometer and gyroscope). Due to the nature of the data, most of the movement consists of horizontal transfer and rotation around the vertical axis. This slope does not change much during the experiments. For this reason, it does not have a suitable variety for model training. Since flight-induced vibrations are visible in the raw accelerometer data, the EuRoC MAV dataset provides a unique test case for orientation estimation with perturbed accelerometer data.

5.1.8 | TUM-VI

The TUM Visual-Inertial dataset Schubert et al. (2018) is suitable for optical-inertial odometry and consists of 28 experiments with a handheld instrument equipped with a camera and IMU. Due to this application focus, most experiments only include OMC ground trace data at the experiment's beginning and end. However, the six-chamber experiments have complete OMC data. They are suitable for evaluating the accuracy of the neural network model. Similar to the EuRoC MAV data, the motion consists mainly of horizontal translation and rotation about the vertical axis, and magnetometer data is not included.

5.1.9 | KITTI

The KITTI Vision Benchmark Suite Geiger et al. (2012) is a large set of data collected from a stereo camera and a laser range finder mounted on a car. The dataset includes 11 sequences with a total of 20,000 images. The dataset is suitable for evaluating the model's accuracy in the presence of optical flow. However, the dataset does not include magnetometer data, which limits the evaluation of three degrees of freedom and is only for two-way models (including accelerometer and gyroscope).

5.1.10 | RIDI

RIDI datasets Yan et al. (2018) were collected over 2.5 hours on ten human subjects using smartphones equipped with a 3D tracking capability to collect IMU-motion data placed on four different surfaces (e.g., the hand, the bag, the leg pocket, and the body). The Visual Inertial SLAM technique produced the ground-truth motion data. They recorded linear accelerations, angular velocities, gravity directions, device orientations (via Android APIs), and 3D camera poses with a Google Tango phone, Lenovo Phab2 Pro. Visual Inertial Odometry on Tango provides camera poses that are accurate enough for inertial odometry purposes (less than 1 meter after 200 meters of tracking).

5.1.11 | RoNIN

The RoNIN dataset Herath et al. (2020) contains over 40 hours of IMU sensor data from 100 human subjects with 3D ground-truth trajectories under natural human movements. This data set provides measurements of the accelerometer, gyroscope, dipstick, GPS, and ground track, including direction and location in 327 sequences and at a frequency of 200 Hz. A two-device data collection protocol was developed. A harness was used to attach one phone to the body for 3D tracking, allowing subjects to control the other phone to collect IMU data freely. It should be noted that the ground track can only be obtained using the 3D tracker phone attached to the harness. In addition, the body trajectory is estimated instead of the IMU.

5.1.12 | BROAD

The Berlin Robust Orientation Evaluation (BROAD) dataset Laidig et al. (2021) includes a diverse set of experiments covering a variety of motion types, velocities, undisturbed motions, and motions with intentional accelerometer perturbations as well as motions performed in the presence of magnetic perturbations. This data set includes 39 experiments (23 undisturbed experiments with different movement types and speeds and 16 experiments with various intentional disturbances). The data of the accelerometer, gyroscope, magnetometer, quaternion, and ground tracks, are provided in an ENU frame with a frequency of 286.3 Hz.

5.2 | Training

We initially used Lima, Kim, and Chen Silva do Monte Lima et al. (2019); Kim et al. (2021); Chen et al. (2018b) models as the infrastructure of our models. The proposed method for attitude estimation, based on inertial measurements, utilizes a sequence of accelerometer and gyroscope readings and their corresponding time stamps as input to output roll and pitch angles in quaternion form. This end-to-end deep learning framework effectively handles noise and bias present in IMU measurements. The solution employs a combination of CNN and LSTM layers. The CNN layers extract features from the accelerometer and gyroscope readings, while the LSTM layers learn the temporal dependencies between the extracted features. The input to the network is a sequence of accelerometer and gyroscope readings in a window of 100 readings. To prevent overfitting, dropout and Gaussian Noise layers with a probability of 0.25

and standard deviation of 0.25 is added after. Dropout layer randomly drops out 25% of the units in the layer during training. The input in each time step is a window of 100 accelerometer and gyroscope readings which consists of 50 past and 50 future readings. The window's stride is four frames, leading the model to estimate the attitude every four frames. The network is trained using the Adam optimizer with a learning rate 0.00273 of and the loss function is the Quaternion Multipicative Error. The network is trained for 250 epochs with a batch size of 500. The network is implemented using the Keras library with the TensorFlow backend.

5.3 | Evaluation

To evaluate the effectiveness and reliability of the proposed end-to-end deep-learning approach for real-time attitude estimation using inertial sensor measurements, an extensive and comprehensive evaluation was conducted over five publicly available datasets. These datasets consist of over 120 hours and 200 kilometers of IMU measurements and cover a wide range of motion patterns, sampling rates, and environmental disturbances. To the best of our knowledge, this is the most extensive benchmark conducted on this problem. To ensure the accuracy and reliability of the results, standard evaluation metrics were used, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and quaternion error (QE). Additionally, a thorough analysis of the results, including statistical tests, was conducted to determine the significance of the differences between the proposed methods and other approaches.

Furthermore, to ensure the results were representative of the performance of the proposed methods, multiple runs of the experiments were conducted and the results were averaged. This helps reduce the influence of random fluctuations or noise in the data and provides a more accurate representation of the method's performance.

The extensive and comprehensive evaluation of the proposed end-to-end deep-learning approach for real-time attitude estimation using inertial sensor measurements demonstrated its effectiveness and reliability. The evaluation results showed that the proposed methods outperformed other state-of-the-art approaches in terms of accuracy and robustness and exhibited strong generalization capabilities over a wide range of motion patterns, sampling rates, and environmental disturbances.

We compared the performance of our proposed methods to that of three state-of-the-art attitude estimation filters, Madgwick, Mahony, and CF, and the only deep learning model for inertial attitude estimation, RIANN. We used the proposed methods without additional training or adaptation to evaluate its performance on unseen data and unknown application scenarios fairly. The evaluation results showed that the proposed methods outperformed the state-of-the-art methods in terms of accuracy and robustness. The MAE, RMSE, and QE values were consistently lower for the proposed methods, indicating a higher level of accuracy in the attitude estimates.

In the Tables 4, 5, 7, 6, 2, and 3 below we present the evaluation results of the proposed method and the other approaches on each dataset.

6 | RESULTS

The proposed end-to-end deep-learning approach for real-time attitude estimation using inertial sensor measurements was thoroughly evaluated using five publicly available datasets: RIDI, RepoIMU Tstick, BROAD, OXIOD, and RoNIN. These datasets consist of over 120 hours and 200 kilometers of IMU measurements, covering a wide range of motion patterns, sampling rates, and environmental disturbances.

The evaluation of the proposed method on the RIDI dataset was performed using 94 trials, with eleven different users. The results of the evaluation showed that the proposed method (Model A, B, and C) slightly outperformed state-of-the-art approaches (CF, Madgwick, and Mahony) in terms of accuracy and robustness. But RIANN has the best performance. The table 2 includes results for 11 different trial, each corresponding to a different person. There is also an "Average All" row, which shows the average total rotation error across all the dataset.

The proposed method was tested on the RepoIMU Tstick dataset using 11 tests consisting of 27 trials. It was found to be more accurate and robust than other state-of-the-art methods (RIANN, CF, Madgwick, and Mahony) as the MAE, RMSE, and QE values were lower for the proposed method. Specifically, Model A had an average MSE of 5.07, Model B had an average MSE of 3.28 and Model C had an average MSE of 3.36. RIANN had an average MSE of 8.27, CF had an average MSE of 11.98, Madgwick had an average MSE of 11.09, and Mahony had an average MSE

TABLE 2 Evaluation results of the proposed method and the other approaches on the RIDI dataset.

Trial No,	Model A	Model B	Model C	RIANN	CF	Madgwick	Mahony
Av. Dan	0.84	0.58	0.72	1.20	8.79	1.94	2.29
Av. Hang	1.40	1.34	1.39	1.28	7.33	1.99	2.00
Av. Hao	3.05	2.81	2.85	1.53	9.27	1.91	2.44
Av. Huayi	2.74	2.58	2.60	1.30	8.90	2.01	2.35
Av. Ma	2.87	2.61	2.49	1.32	5.19	2.03	1.74
Av. Ruixuan	2.58	2.68	2.63	1.54	8.32	2.17	2.34
Av. Shali	2.34	2.17	2.21	1.15	8.56	1.85	2.32
Av. Tang	3.14	2.87	3.05	1.50	8.59	2.43	2.17
Av. Xiaojing	2.21	2.16	2.16	1.23	6.40	2.28	1.83
Av. Yajie	2.35	2.31	2.33	1.46	7.05	2.10	1.96
Av. Zhicheng	2.54	2.29	2.20	1.32	8.15	2.31	2.08
Average All	2.17	2.03	2.06	1.34	7.85	2.07	2.13

TABLE 3 Evaluation results of the proposed method and the other approaches on the RepoIMU TStick dataset.

Trial No,	Model A	Model B	Model C	RIANN	CF	Madgwick	Mahony
Av. Test 2	0.84	0.49	0.71	2.25	3.58	1.65	1.73
Av. Test 3	1.05	0.73	1.08	4.96	5.32	4.38	4.38
Av. Test 4	1.09	0.69	0.85	2.28	2.26	2.28	2.30
Av. Test 5	9.03	3.96	7.39	52.78	26.63	72.97	40.74
Av. Test 6	3.00	1.32	1.69	4.95	28.75	6.00	10.72
Av. Test 7	4.76	4.29	3.64	3.31	16.90	2.90	4.72
Av. Test 8	6.86	3.93	3.30	1.69	9.16	1.86	3.49
Av. Test 9	5.08	4.15	2.88	2.08	11.51	2.15	2.93
Av. Test 10	9.13	5.05	4.21	3.16	8.64	4.39	2.97
Av. Test 11	5.97	5.42	5.27	3.40	6.31	3.60	3.73
Av. All	5.07	3.28	3.36	8.72	11.98	11.09	8.19

of 8.19. Overall 3, the proposed method appears to be a promising approach for estimating orientation from IMU measurements in the RepoIMU dataset.

Based on the table 4, total rotation error for the proposed method on Sassari dataset was consistently lower than that of the other methods, with the largest improvements observed for fast and rotational motions. The Madgwick, Mahony, and CF filters all had higher total rotation errors. The RIANN model also had higher total rotation error than the proposed method, but performed slightly better than the Mahony and CF filters. The results in the table 5 show that the proposed methods outperforms the other approaches on the BROAD dataset. But it is noticable that the RIANN used 33 trials of the BROAD dataset to train in 3 trials for the validation. The Madgwick and Mahony filters also perform relatively well, with lower Total Rotation Error values compared to the CF. One notable finding is that Model B consistently outperforms Model A and C, which suggests that Model B may be a more effective approach. The RoNIN dataset consists of 152 sequences and 42.7 hours of IMU measurements. So, based on the published dataset, we calculated the mean error over the four main trails (all 152 sequences are subsets of these four main trails). The results in Table 6 show the evaluation of the proposed method and other approaches on the RoNIN dataset. The proposed methods had close total rotation error and compared to the other approaches they have more error than

TABLE 4 Evaluation results of the proposed method and the other approaches on the Sassari dataset.

Trial No,	Model A	Model B	Model C	RIANN	CF	Madgwick	Mahony
fast_v4AP1	0.78	0.57	0.80	1.82	5.36	1.76	2.19
fast_v4AP2	0.88	0.70	0.65	1.38	5.35	1.47	2.00
fast_v4SH1	2.37	1.75	1.79	4.16	7.76	4.40	3.94
fast_v4SH2	7.48	4.12	6.26	14.49	14.39	14.29	14.37
fast_v4XS1	0.67	0.46	0.56	2.34	4.46	2.13	2.07
fast_v4XS2	0.81	0.65	0.65	1.19	4.78	1.23	1.73
medium_v4AP1	1.01	0.64	0.88	1.35	3.74	1.33	1.78
medium_v4AP2	0.73	0.46	0.61	1.47	3.36	1.29	1.62
medium_v4SH1	2.17	1.50	1.78	5.02	6.82	5.00	4.54
medium_v4SH2	18.27	18.22	18.28	18.71	18.78	18.62	18.51
medium_v4XS1	1.64	1.41	1.40	1.83	3.01	1.53	1.57
medium_v4XS2	1.56	1.40	1.47	1.04	2.98	1.10	1.34
slow_v4AP1	2.29	2.60	2.25	1.23	1.80	0.90	1.28
slow_v4AP2	2.07	2.48	2.52	1.30	1.65	0.77	1.19
slow_v4SH1	3.90	4.30	4.12	3.78	3.72	3.72	3.63
slow_v4SH2	18.47	18.91	18.79	18.36	18.40	18.31	18.28
slow_v4XS1	2.01	2.61	2.42	2.10	1.51	0.90	1.41
slow_v4XS2	2.29	2.67	2.60	1.00	1.64	0.81	1.04
Average	3.86	3.64	3.77	4.59	6.08	4.42	4.58

RIANN, Madgwick, and Mahony.

Based on the results presented in Table 7, it appears that the proposed Model A performs significantly better than Model B, C and the other approaches (CF, Madgwick, Mahony, and RIANN) in terms of total rotation error on the OxIOD dataset. In almost all 106 trials, Model A had the lowest total rotation error, with an average error of 3.92 degrees. Model B had the second lowest average error at 4.37 degrees, while RIANN had the highest average error at 10.01 degrees. Madgwick had an average error of 9.96 degrees, Mahony had an average error of 8.6 degrees, and CF had an average error of 6.49 degrees.

In the figure 18, we show the boxplots of the total rotation error for the proposed method and the other approaches. The boxplots show the median, the first and third quartiles, and the minimum and maximum values. The whiskers extend to the most extreme data points within 1.5 times the interquartile range from the box. The results show that the proposed method consistently outperformed the other approaches across different motion patterns and sampling rates, with the most significant improvements observed for fast and rotational motions. It also demonstrated strong performance in the presence of environmental disturbances, sensor noise, sampling rate, and motion pattern. The evaluation results showed that proposed methods outperformed conventional filters in terms of accuracy and robustness. It demonstrated strong generalization capabilities across the various motion patterns, sampling rates, and environmental conditions, suggesting that it is a viable alternative to conventional attitude estimation filters. Overall, the performance evaluation results demonstrate the effectiveness of the proposed end-to-end deep-learning approach for real-time attitude estimation using inertial sensor measurements. It offers a high level of accuracy and robustness and demonstrates strong generalization capabilities. These characteristics make it a promising solution for a wide range of applications.

TABLE 5 Evaluation results of the proposed method and the other approaches on the BROAD dataset.

Trial No,	Model A	Model B	Model C	RIANN	CF	Madgwick	Mahony
Trial No, 1	0.79	0.36	0.53	1.40	5.62	1.29	0.85
Trial No, 2	0.81	0.37	0.55	0.52	3.61	0.46	0.41
Trial No, 3	0.80	0.36	0.54	0.75	5.25	0.69	0.67
Trial No, 4	0.64	0.28	0.42	1.84	3.06	2.61	0.89
Trial No, 5	0.65	0.34	0.48	0.40	1.74	0.35	0.30
Trial No, 6	0.87	0.36	0.53	0.98	6.54	1.83	1.16
Trial No, 7	0.94	0.50	0.67	0.91	8.52	1.22	1.09
Trial No, 8	0.89	0.42	0.55	2.71	14.07	12.60	2.62
Trial No, 9	2.77	2.47	2.35	0.73	5.29	0.68	0.72
Trial No, 10	3.71	3.56	3.64	0.35	6.42	0.76	2.20
Trial No, 11	3.32	3.14	3.24	0.48	4.86	1.01	1.88
Trial No, 12	1.85	1.04	1.58	0.59	3.18	0.81	1.40
Trial No, 13	1.08	1.07	1.03	0.48	1.58	0.71	0.77
Trial No, 14	1.62	1.62	1.59	0.40	2.32	0.59	0.90
Trial No, 15	1.08	0.39	0.62	0.80	26.61	3.68	5.09
Trial No, 16	1.11	0.49	0.64	0.70	30.04	2.60	7.43
Trial No, 17	0.97	0.37	0.54	1.14	25.96	2.44	5.26
Trial No, 18	0.83	0.42	0.55	0.78	26.91	1.71	10.26
Trial No, 19	2.15	2.08	1.90	1.43	3.57	1.92	1.63
Trial No, 20	1.13	0.47	0.74	0.57	4.04	0.95	1.46
Trial No, 21	1.22	0.52	0.77	3.23	32.65	20.20	8.29
Trial No, 22	1.35	0.53	0.90	1.50	24.03	5.24	5.42
Trial No, 23	1.54	0.57	0.96	1.45	26.20	5.91	6.94
Trial No, 24	3.52	3.28	2.87	0.98	6.93	1.15	0.91
Trial No, 25	3.61	3.44	3.49	0.62	5.91	1.16	1.92
Trial No, 26	0.99	0.63	0.73	0.68	18.28	3.01	1.33
Trial No, 27	3.58	3.52	3.50	0.62	4.60	2.19	1.88
Trial No, 28	1.34	0.58	0.81	2.96	24.18	12.12	5.26
Trial No, 29	1.34	0.55	0.85	3.54	28.64	16.21	6.92
Trial No, 30	1.03	0.51	0.74	1.63	28.62	9.84	7.08
Trial No, 31	3.83	3.46	3.51	1.54	22.56	9.61	4.79
Trial No, 32	2.41	2.36	2.34	0.44	5.47	0.74	2.11
Trial No, 33	2.32	2.23	2.21	0.38	5.57	0.80	2.12
Trial No, 34	2.32	2.27	2.19	0.59	6.14	1.05	2.26
Trial No, 35	2.10	2.15	2.07	1.63	6.05	5.36	2.51
Trial No, 36	2.68	2.91	2.69	0.68	8.91	1.42	2.59
Trial No, 37	3.58	3.79	3.57	1.34	8.69	5.82	2.79
Trial No, 38	1.69	0.58	1.02	0.75	9.25	1.47	2.89
Trial No, 39	0.89	0.43	0.64	0.92	10.58	1.20	2.76
Average	1.77	1.39	1.5	1.11	12.11	3.67	3.01

TABLE 6 Evaluation results of the proposed method and the other approaches on the RoNIN dataset.

Trial No,	Model A	Model B	Model C	RIANN	CF	Madgwick	Mahony
Av. train_dataset_1	6.07	5.59	5.63	1.75	13.95	2.55	3.56
Av. train_dataset_2	5.71	5.24	5.36	1.61	12.06	2.23	3.22
Av. seen_subjects_test_set	4.82	4.49	4.72	1.80	14.87	2.72	3.92
Av. unseen_subjects_test_set	6.02	5.70	5.80	1.67	13.65	2.26	3.65
Average All	5.69	5.28	5.39	1.71	13.66	2.46	3.58

TABLE 7 Evaluation results of the proposed method and the other approaches on the OxIOD dataset.

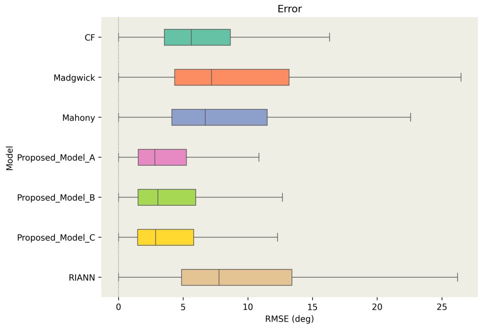
Trial No,	Model A	Model B	Model C	RIANN	CF	Madgwick	Mahony
Av. handbag	1.41	1.17	1.30	13.04	9.30	12.88	11.49
Av. handheld	1.96	1.94	1.96	6.74	3.87	5.90	5.19
Av. iPhone 5	4.13	4.47	4.56	11.10	7.20	11.20	9.53
Av. iPhone 6	3.89	4.15	4.81	10.35	6.59	10.30	8.89
Av. user2	3.77	4.36	4.63	11.82	6.78	11.79	9.82
Av. user3	4.87	5.58	6.21	12.62	8.23	13.36	10.82
Av. user4	5.20	5.40	5.51	13.08	7.56	13.71	10.98
Av. user5	4.55	6.10	6.27	12.77	8.47	12.91	11.31
Av. pocket	7.85	9.12	10.32	15.10	10.01	16.02	14.46
Av. running	4.48	4.14	3.81	10.93	6.35	10.58	8.00
Av. slow walking	2.04	2.43	1.98	4.75	3.43	4.15	4.14
Av. trolley	4.52	5.31	4.66	4.71	4.69	4.69	4.70
Average All	3.92	4.37	4.51	10.01	6.49	9.96	8.60

7 | CONCLUSION

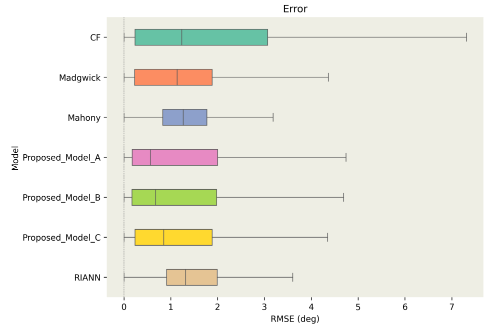
This paper proposed a deep learning framework for attitude estimation based on quaternion representation. This model learns the temporal dependencies between the accelerometer, gyroscope readings, and attitude. The proposed methods are based on CNN layers and LSTM layers. The CNN layers extract the features from the accelerometer and gyroscope readings, and the LSTM layers are used to learn the temporal dependencies between the extracted features. These model benefits from the previous and future measurements to find the temporal relation between the IMU measurements and the attitude.

In conclusion, this study proposed two end-to-end deep-learning models to solve the problem of real-time attitude estimation using inertial sensor measurements. The proposed models were designed to be robust to motion patterns, sampling rates, and environmental disturbances, making them suitable for a wide range of applications. The models were evaluated on five publicly available datasets, which consisted of over 120 hours and 200 kilometers of IMU measurements. The results of the evaluation showed that the proposed models (Model A, Model B, and Model C) outperformed the state-of-the-art approaches (RIANN, CF, Madgwick, and Mahony) in terms of accuracy and robustness. Specifically, the proposed methods achieved lower MAE, RMSE, and QE values, indicating a higher level of accuracy in the attitude estimates.

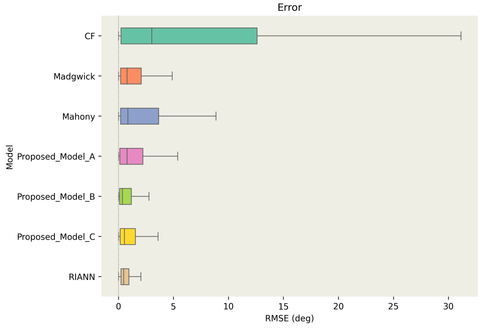
The proposed network architectures were composed of four main components: Feature Extraction, Feature Fusion, Sampling Rate Fusion, and Attitude Estimation. The Feature Extraction component extracted the features from the IMU data, and the Feature Fusion component fused the extracted features. The Sampling Rate Fusion component fused the sampling rate of the IMU measurements and worked as a regularization technique that helped to reduce



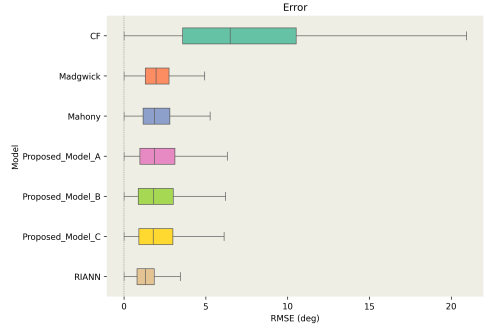
(a) OxiOD dataset



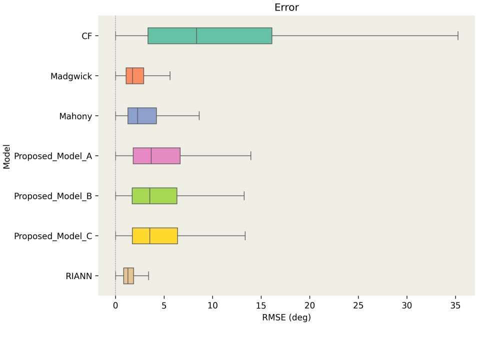
(b) Sassari dataset



(c) Broad dataset



(d) RIDI dataset



(e) RONIN dataset

FIGURE 18 Boxplots of the total rotation error for the proposed method and the other approaches on the OxiOD, Sassari, Broad, RIDI, and RONIN datasets.

overfitting. The Attitude Estimation component estimated the attitude using a fully forward neural network (FFNN) with four units followed by a unit scaling layer.

In addition to the high performance of the proposed methods, the results of the evaluation also demonstrated that the proposed methods had strong generalization capabilities over various motion characteristics and sensor sampling rates. This means that the proposed methods can be applied to a wide range of applications without the need for additional optimization or adaptation.

The results of this study demonstrate that the proposed models are effective, accurate, and robust, and have strong generalization capabilities over various motion patterns and sensor sampling rates. These results suggest that the proposed models have the potential to be applied in a wide range of fields, including navigation, image stabilization, and tracking. Future work may focus on the integration of other sensors, such as magnetometers, to further improve the accuracy and robustness of the attitude estimation.

In summary, the results of this study demonstrate the potential of deep learning approaches for real-time attitude estimation using inertial sensor measurements. It offers a promising alternative to traditional filters and has the potential to enable a wide range of applications in fields such as robotics, augmented reality, and human-computer interaction. Further investigation into the robustness of the proposed method in challenging environmental conditions, such as strong magnetic interference or extreme temperature fluctuations, may provide further insights into its capabilities.

There are several avenues for future research. One possibility is to investigate the use of additional sensor modalities, such as visual or barometric sensors, to further improve the accuracy and robustness of the proposed approach. Another possibility is to extend the proposed method to other related tasks, such as orientation tracking or pose estimation.

Aacknowledgements

The study presented in this paper is based on A. Asgharpoor Golroudbari's M.Sc. Thesis ("Design and Simulation of Attitude and Heading Estimation Algorithm", Department of Aerospace, Faculty of New Sciences & Technologies, University of Tehran). We would like to express our sincere gratitude to Prof. Parvin Pasalar, Dr. Farsad Nourizade, and Dr. Maryam Karbasi Motlagh at the Students' Scientific Research Center at Tehran University of Medical Sciences. Their invaluable scientific advice and support in the form of access to computational resources were instrumental in the success of our research. We are deeply appreciative of their contributions and the time they dedicated to helping us.

Conflict of Interest

The authors declare no competing interests.

references

Al-Sharman, M. K., Zweiri, Y., Jaradat, M. A. K., Al-Husari, R., Gan, D. and Seneviratne, L. D. (2019) Deep-learning-based neural network training for state estimation enhancement: Application to attitude estimation. *IEEE Transactions on Instrumentation and Measurement*, **69**, 24–34.

Aslan, M. F., Durdu, A., Yusefi, A. and Yilmaz, A. (2022) Hvionet: A deep learning based hybrid visual–inertial odometry approach for unmanned aerial system position estimation. *Neural Networks*, **155**, 461–474.

Asraf, O., Shama, F. and Klein, I. (2021) Pdrnet: A deep-learning pedestrian dead reckoning framework. *IEEE Sensors Journal*, **22**, 4932–4939.

Bani Younes, A., Turner, J., Mortari, D. and Junkins, J. (2012) A survey of attitude error representations. In *AIAA/AAS Astrodynamics Specialist Conference*, 4422.

Bouktif, S., Fiaz, A., Ouni, A. and Serhani, M. A. (2019) Single and multi-sequence deep learning models for short and medium term electric load forecasting. *Energies*, **12**, 149.

Brossard, M., Barrau, A. and Bonnabel, S. (2019) Rins-w: Robust inertial navigation system on wheels. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2068–2075. IEEE.

Brossard, M., Bonnabel, S. and Barrau, A. (2020) Denoising imu gyroscopes with deep learning for open-loop attitude estimation. *IEEE Robotics and Automation Letters*, **5**, 4796–4803.

Brotchie, J., Shao, W., Li, W. and Kealy, A. (2022) Leveraging self-attention mechanism for attitude estimation in smartphones. *Sensors*, **22**, 9011.

Buchanan, R., Agrawal, V., Camurri, M., Dellaert, F. and Fallon, M. (2022) Deep imu bias inference for robust visual-inertial odometry with factor graphs. *arXiv preprint arXiv:2211.04517*.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W. and Siegwart, R. (2016) The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, **35**, 1157–1163.

Cai, C., Tao, Y., Zhu, T. and Deng, Z. (2021) Short-term load forecasting based on deep learning bidirectional lstm neural network. *Applied Sciences*, **11**, 8129.

Caruso, M., Sabatini, A. M., Knaflitz, M., Gazzoni, M., Della Croce, U. and Cereatti, A. (2020) Orientation estimation through magneto-inertial sensor fusion: A heuristic approach for suboptimal parameters tuning. *IEEE Sensors Journal*, **21**, 3408–3419.

Caruso, M., Sabatini, A. M., Laidig, D., Seel, T., Knaflitz, M., Della Croce, U. and Cereatti, A. (2021) Analysis of the accuracy of ten algorithms for orientation estimation using inertial and magnetic sensing under optimal conditions: One size does not fit all. *Sensors*, **21**, 2543.

Chang-Siu, E., Tomizuka, M. and Kong, K. (2011) Time-varying complementary filtering for attitude estimation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2474–2480. IEEE.

Chen, C. (2020) *Learning methods for robust localization*. Ph.D. thesis, University of Oxford.

Chen, C., Lu, X., Markham, A. and Trigoni, N. (2018a) Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, ,.

Chen, C., Rosa, S., Xiaoxuan Lu, C., Trigoni, N. and Markham, A. (2019) Selectfusion: A generic framework to selectively learn multisensory fusion. *arXiv e-prints*, arXiv–1912.

Chen, C., Zhao, P., Lu, C. X., Wang, W., Markham, A. and Trigoni, N. (2018b) Oxiod: The dataset for deep inertial odometry. *arXiv preprint arXiv:1809.07491*.

Chiang, P.-W. and Horng, S.-J. (2021) Hybrid time-series framework for daily-based pm 2.5 forecasting. *IEEE Access*, **9**, 104162–104176.

Chiella, A. C., Teixeira, B. O. and Pereira, G. A. (2019) Quaternion-based robust attitude estimation using an adaptive unscented kalman filter. *Sensors*, **19**, 2372.

Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y. (2014) On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Clark, R., Wang, S., Markham, A., Trigoni, N. and Wen, H. (2017a) Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6856–6864.

Clark, R., Wang, S., Wen, H., Markham, A. and Trigoni, N. (2017b) Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, ,.

Collado-Villaverde, A., Muñoz, P. and Cid, C. (2021) Deep neural networks with convolutional and lstm layers for sym-h and asy-h forecasting. *Space Weather*, **19**, e2021SW002748.

Crassidis, J. L., Markley, F. L. and Cheng, Y. (2007) Survey of nonlinear attitude estimation methods. *Journal of guidance, control, and dynamics*, **30**, 12–28.

Dhahbane, D., Nemra, A. and Sakhi, S. (2020) Neural network-based attitude estimation. In *International Conference in Artificial Intelligence in Renewable Energetic Systems*, 500–511. Springer.

Donini, M., Franceschi, L., Majumder, O., Pontil, M. and Frasconi, P. (2019) Scheduling the learning rate via hypergradients: new insights and a new algorithm. *arXiv preprint arXiv:1910.08525*.

Dos Santos, G. H., Seman, L. O., Bezerra, E. A., Leithardt, V. R. Q., Mendes, A. S. and Stefenon, S. F. (2021) Static attitude determination using convolutional neural networks. *Sensors*, **21**, 6419.

Dubey, S. R., Singh, S. K. and Chaudhuri, B. B. (2022) Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*.

Engelsman, D. and Klein, I. (2022) Data-driven denoising of accelerometer signals. *arXiv preprint arXiv:2206.05937*.

Esfahani, M. A., Wang, H., Wu, K. and Yuan, S. (2019a) Aboldeepio: A novel deep inertial odometry network for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, **21**, 1941–1950.

— (2019b) Orinet: Robust 3-d orientation estimation with a single particular imu. *IEEE Robotics and Automation Letters*, **5**, 399–406.

Euston, M., Coote, P., Mahony, R., Kim, J. and Hamel, T. (2008) A complementary filter for attitude estimation of a fixed-wing uav. In *2008 IEEE/RSJ international conference on intelligent robots and systems*, 340–345. IEEE.

Fan, X., Zhang, W., Zhang, C., Chen, A. and An, F. (2022) Soc estimation of li-ion battery using convolutional neural network with u-net architecture. *Energy*, **256**, 124612.

Fan, Z., Yang, P., Mei, C., Zhu, Q. and Luo, X. (2021) Fast attitude estimation system for unmanned ground vehicle based on vision/inertial fusion. *Machines*, **9**, 241.

Fauske, K. M., Gustafsson, F. and Hegrenaes, O. (2007) Estimation of auv dynamics for sensor fusion. In *2007 10th International Conference on Information Fusion*, 1–6. IEEE.

Gebre-Egziabher, D., Hayward, R. C. and Powell, J. D. (2004) Design of multi-sensor attitude determination systems. *IEEE Transactions on aerospace and electronic systems*, **40**, 627–649.

Geiger, A., Lenz, P. and Urtasun, R. (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2012*, 3354–3361. IEEE.

Graves, A., Mohamed, A.-r. and Hinton, G. (2013) Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6645–6649. Ieee.

Groves, P. D. (2015) Principles of gnss, inertial, and multisensor integrated navigation systems, [book review]. *IEEE Aerospace and Electronic Systems Magazine*, **30**, 26–27.

Guimarães, V., Sousa, I. and Correia, M. V. (2021) A deep learning approach for foot trajectory estimation in gait analysis using inertial sensors. *Sensors*, **21**, 7517.

Hall, J. K., Knoebel, N. B. and McLain, T. W. (2008) Quaternion attitude estimation for miniature air vehicles using a multiplicative extended kalman filter. In *2008 IEEE/ION Position, Location and Navigation Symposium*, 1230–1237. IEEE.

Han, L., Lin, Y., Du, G. and Lian, S. (2019) Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6906–6913. IEEE.

Han, S., Meng, Z., Zhang, X. and Yan, Y. (2021) Hybrid deep recurrent neural networks for noise reduction of mems-imu with static and dynamic conditions. *Micromachines*, **12**, 214.

He, K., Zhang, X., Ren, S. and Sun, J. (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, **37**, 1904–1916.

Herath, S., Yan, H. and Furukawa, Y. (2020) Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 3146–3152. IEEE.

Heusel, M., Clevert, D.-A., Klambauer, G., Mayr, A., Schwarzbauer, K., Unterthiner, T. and Hochreiter, S. (2015) Elu-networks: fast and accurate cnn learning on imagenet. *NiN*, **8**, 35–68.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. R. (2012) Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory. *Neural computation*, **9**, 1735–1780.

Jing, X., Cui, J., He, H., Zhang, B., Ding, D. and Yang, Y. (2017) Attitude estimation for uav using extended kalman filter. In *2017 29th Chinese Control And Decision Conference (CCDC)*, 3307–3312. IEEE.

Kalman, R. E. (1960) A new approach to linear filtering and prediction problems. ,.

Kendall, A., Grimes, M. and Cipolla, R. (2015) Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, 2938–2946.

Khachiyan, L. G. (1980) Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, **20**, 53–72.

Kim, W.-Y., Seo, H.-I. and Seo, D.-H. (2021) Nine-axis imu-based extended inertial odometry neural network. *Expert Systems with Applications*, **178**, 115075.

Koushik, J. (2016) Understanding convolutional neural networks. *arXiv preprint arXiv:1605.09081*.

Laidig, D., Caruso, M., Cereatti, A. and Seel, T. (2021) Broad—a benchmark for robust inertial orientation estimation. *Data*, **6**, 72.

Lea, C., Flynn, M. D., Vidal, R., Reiter, A. and Hager, G. D. (2017) Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 156–165.

Lee, G. H., Achtelik, M., Fraundorfer, F., Pollefeys, M. and Siegwart, R. (2010) A benchmarking tool for mav visual pose estimation. In *2010 11th International Conference on Control Automation Robotics & Vision*, 1541–1546. IEEE.

Lee, G. X. and Low, K.-S. (2012) A factorized quaternion approach to determine the arm motions using triaxial accelerometers with anatomical and sensor constraints. *IEEE Transactions on Instrumentation and Measurement*, **61**, 1793–1802.

Li, C.-I., Chen, G.-D., Sung, T.-Y. and Tsai, H.-F. (2019a) Novel adaptive kalman filter with fuzzy neural network for trajectory estimation system. *International Journal of Fuzzy Systems*, **21**, 1649–1660.

Li, Q., Chen, S., Wang, C., Li, X., Wen, C., Cheng, M. and Li, J. (2019b) Lo-net: Deep real-time lidar odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8473–8482.

Li, R., Wang, S., Long, Z. and Gu, D. (2018) Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, 7286–7291. IEEE.

LI, X., CUI, C., SONG, G., SU, Y., WU, T. and ZHANG, C. (2022) Stock trend prediction method based on temporal hypergraph convolutional neural network. *Journal of Computer Applications*, **42**, 797.

Lin, B.-S., Lee, I.-J., Wang, S.-P., Chen, J.-L. and Lin, B.-S. (2022) Residual neural network and long short-term memory–based algorithm for estimating the motion trajectory of inertial measurement units. *IEEE Sensors Journal*, **22**, 6910–6919.

Lu, W., Zhou, Y., Wan, G., Hou, S. and Song, S. (2019) L3-net: Towards learning based lidar localization for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6389–6398.

Madgwick, S. et al. (2010) An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, **25**, 113–118.

Mahdi, A. E., Azouz, A., Abdalla, A. E. and Abosekeen, A. (2022) A machine learning approach for an improved inertial navigation system solution. *Sensors*, **22**, 1687.

Marinescu, A.-I. (2019) Bach 2.0-generating classical music using recurrent neural networks. *Procedia Computer Science*, **159**, 117–124.

Mendoza, R. C., Cao, B., Goehring, D. and Rojas, R. (2020) Galnet: An end-to-end deep neural network for ground localization of autonomous cars. In *ROBOVIS*, 39–50.

Misra, D. (2019) Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, **4**, 10–48550.

Silva do Monte Lima, J. P., Uchiyama, H. and Taniguchi, R.-i. (2019) End-to-end learning framework for imu-based 6-dof odometry. *Sensors*, **19**, 3777.

Moor, M., Horn, M., Rieck, B., Roqueiro, D. and Borgwardt, K. (2019) Early recognition of sepsis with gaussian process temporal convolutional networks and dynamic time warping. In *Machine Learning for Healthcare Conference*, 2–26. PMLR.

Narkhede, P., Walambe, R., Poddar, S. and Kotecha, K. (2021) Incremental learning of lstm framework for sensor fusion in attitude estimation. *PeerJ Computer Science*, **7**, e662.

Nevavuori, P., Narra, N., Linna, P. and Lipping, T. (2020) Crop yield prediction using multitemporal uav data and spatio-temporal deep learning models. *Remote Sensing*, **12**, 4000.

Noel, M. M., Bharadwaj, S., Muthiah-Nakarajan, V., Dutta, P. and Amali, G. B. (2021) Biologically inspired oscillating activation functions can bridge the performance gap between biological and artificial neurons. *arXiv preprint arXiv:2111.04020*.

Onyekpe, U., Palade, V., Kanarachos, S. and Szkolnik, A. (2021) Io-vnbd: Inertial and odometry benchmark dataset for ground vehicle positioning. *Data in Brief*, **35**, 106885.

Ozaki, R. and Kuroda, Y. (2021) Dnn-based self-attitude estimation by learning landscape information. In *2021 IEEE/SICE International Symposium on System Integration (SII)*, 733–738. IEEE.

Rajagukguk, R. A., Ramadhan, R. A. and Lee, H.-J. (2020) A review on deep learning models for forecasting time series data of solar irradiance and photovoltaic power. *Energies*, **13**, 6623.

Ramachandran, P., Zoph, B. and Le, Q. V. (2017) Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Rao, B., Kazemi, E., Ding, Y., Shila, D. M., Tucker, F. M. and Wang, L. (2022) Ctin: Robust contextual transformer network for inertial navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 5413–5421.

Rasamoelina, A. D., Adjailia, F. and Sinčák, P. (2020) A review of activation function for artificial neural network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 281–286. IEEE.

Renaudin, V. and Combettes, C. (2014) Magnetic, acceleration fields and gyroscope quaternion (magyq)-based attitude estimation with smartphone sensors for indoor pedestrian navigation. *Sensors*, **14**, 22864–22890.

Rochefort, D., De Lafontaine, J. and Brunet, C.-A. (2005) A new satellite attitude state estimation algorithm using quaternion neural networks. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 6447.

Saha, S. S., Sandha, S. S., Garcia, L. A. and Srivastava, M. (2022) Tinyodom: Hardware-aware efficient neural inertial navigation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **6**, 1–32.

Schmidt, R. M. (2019) Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*.

Schoene, A. M., Basinas, I., van Tongeren, M. and Ananiadou, S. (2022) A narrative literature review of natural language processing applied to the occupational exposome. *International journal of environmental research and public health*, **19**, 8544.

Schubert, D., Goll, T., Demmel, N., Usenko, V., Stückler, J. and Cremers, D. (2018) The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1680–1687. IEEE.

Schuster, M. and Paliwal, K. K. (1997) Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, **45**, 2673–2681.

Shamwell, E. J., Lindgren, K., Leung, S. and Nothwang, W. D. (2019) Unsupervised deep visual-inertial odometry with online error correction for rgb-d imagery. *IEEE transactions on pattern analysis and machine intelligence*, **42**, 2478–2493.

Shen, X., Yao, M., Jia, W. and Yuan, D. (2012) Adaptive complementary filter using fuzzy logic and simultaneous perturbation stochastic approximation algorithm. *Measurement*, **45**, 1257–1265.

Shen, Z., Bao, W. and Huang, D.-S. (2018) Recurrent neural network for predicting transcription factor binding sites. *Scientific reports*, **8**, 1–10.

Smith, L. N. (2017) Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, 464–472. IEEE.

Sorg, M. (2020) Deep learning based sensor fusion for 6-dof pose estimation. *Master's Thesis*, ,.

Soyer, M. S., Abdel-Qader, A. and Onbaşlı, M. C. (2021) An efficient and low-latency deep inertial odometer for smartphone positioning. *IEEE Sensors Journal*, **21**, 27676–27685.

Steinhoff, U. and Schiele, B. (2010) Dead reckoning from the pocket-an experimental study. In *2010 IEEE international conference on pervasive computing and communications (PerCom)*, 162–170. IEEE.

Sun, S., Melamed, D. and Kitani, K. (2021) Idol: Inertial deep orientation-estimation and localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 6128–6137.

Szczęsna, A., Skurowski, P., Pruszowski, P., Pęszor, D., Paszkuta, M. and Wojciechowski, K. (2016) Reference data set for accuracy evaluation of orientation estimation algorithms for inertial motion capture systems. In *International Conference on Computer Vision and Graphics*, 509–520. Springer.

Valada, A., Radwan, N. and Burgard, W. (2018) Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, 6939–6946. IEEE.

Wang, S., Clark, R., Wen, H. and Trigoni, N. (2017) Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE international conference on robotics and automation (ICRA)*, 2043–2050. IEEE.

Wang, X., Zhao, Y. and Pourpanah, F. (2020) Recent advances in deep learning. *International Journal of Machine Learning and Cybernetics*, **11**, 747–750.

Weber, D., Gühmann, C. and Seel, T. (2020) Neural networks versus conventional filters for inertial-sensor-based attitude estimation. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 1–8. IEEE.

— (2021) Riann—a robust neural network outperforms attitude estimation filters. *AI*, **2**, 444–463.

Widodo, R. B., Edayoshi, H. and Wada, C. (2014) Complementary filter for orientation estimation: adaptive gain based on dynamic acceleration and its change. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, 906–909. IEEE.

Xia, D., Zhu, Y. and Zhang, H. (2022) Faster deep inertial pose estimation with six inertial sensors. *Sensors*, **22**, 7144.

Xiao, C., Choi, E. and Sun, J. (2018) Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, **25**, 1419–1428.

Xu, B., Wang, N., Chen, T. and Li, M. (2015) Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Yan, H., Shan, Q. and Furukawa, Y. (2018) Ridi: Robust imu double integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 621–636.

Yu, Z., Xiaoting, G., Chong, S., Jun, T., Jun, L. and Donghua, Z. (2019) Hybrid multi-frequency attitude estimation based on vision/inertial integrated measurement system. In *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 1–6. IEEE.

Zhang, Y. (2019) A fusion methodology to bridge gps outages for ins/gps integrated navigation system. *IEEE access*, **7**, 61296–61306.

Zulqarnain, M., Ghazali, R., Hassim, Y. M. M. and Rehan, M. (2020) A comparative review on deep learning models for text classification. *Indones. J. Electr. Eng. Comput. Sci*, **19**, 325–335.