Defence Science Journal, Vol. 74, No. 5, September 2024, pp. 690-700, DOI : 10.14429/dsj.74.19683

Unscented Kalman Filters Integrated with Deep Learning Approaches for Active Sonar Based 2D Underwater Target Tracking

Uwigize Patrick[#], S. Koteswara Rao[#], B. Omkar Lakshmi Jagan[!,*], M. Kavitha Lakshmi[$] and Thayyaba Khatoon Mohammed[^]

[#]*Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur - 522 302, India*

[!]*Department of Computer Science and Engineering, Vignan's Institute of Information Technology, Duvvada, Visakhapatnam - 530 049, India*

[$]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur -522 302, India*

[^]*Department of Artificial Intelligence and Machine Learning, School of Engineering, Malla Reddy University, Maisammaguda, Dulapally, Hyderabad - 500 043, India*

[*]*E-mail: omkarjagan@gmail.com*

ABSTRACT

This manuscript proposes a new approach to track 2D targets using a combination of machine learning algorithms and the Unscented Kalman filter (UKF). The approach makes use of active sonar sensors to measure range and bearing, which are used to predict the target's course and speed. So far in the literature of target tracking, researchers assumed covariance matrix of the noise in sonar measurements. In this manuscript, it is tried to estimate the same using deep learning algorithms. The Machine Learning algorithms, such as multilayer perceptron, convolutional neural network, long-short term memory, and gated recurrent unit, are employed to approximate the covariance of the noise in the input measurements. Simultaneously, the Unscented Kalman Filter (UKF) is utilised to mitigate the noise in the measurements and to estimate the position and speed of the target. The results are quantified through Monte Carlo simulations in a simulated underwater environment. The measurements are assumed to conform to a normal Gaussian distribution with a mean of zero. The findings indicate that LSTM has superior performance compared to the other models. Nevertheless, it is important to note that the results are constrained in their applicability due to the restricted set of variables employed for training the machine learning models.

Keywords: Nonlinear filtering; Statistical signal processing; Deep learning; Recurrent neural network; Time series prediction

1. INTRODUCTION

ACTIVE sonar systems, underwater vehicles and acoustic imaging equipment use sound waves to detect targets underwater and extract information about their characteristics. There are different techniques for transmitting and receiving sound waves in an acoustic environment, such as pulse-echo, phase difference measurement, and frequency modulated continuous wave. These techniques have diverse applications, including marine navigation, object detection, ranging and acoustic imaging. After receiving the signal, various signal processing techniques, such as band-pass, matched, and adaptive filters, are then used to extract useful information from the received signal. These techniques are essential in improving the accuracy and reliability of the extracted data[1].

The task of target tracking involves estimating the target state based on measurements only, which is challenging due to measurement noise, unmodeled dynamics, and non-linearities[2]. To address these challenges, various methods have been developed, such as Kalman, particle filters and machine learning algorithms etc.[3-7]. Current research on active sonar systems aims to enhance the detection and localization of targets by using advanced signal processing techniques such as beamforming[7]. This method involves creating a beam by combining signals from multiple sensors directed towards the target, which improves the signal-to-noise ratio and enhances the measurement accuracy.

Deep Neural Networks (DNNs) are a type of machine learning models that have become widely popular because of their capacity to autonomously acquire intricate connections between inputs and outputs from extensive datasets[9]. There are several types of DNNs, including MLP[10], CNN[11], LSTM[12], and GRU[13], each with unique architectural features and training procedures. MLP (Multi-Layer Perceptron) and CNN (Convolutional Neural Network) are types of feedforward networks that operate by sequentially processing incoming data through a series of layers, without any loops or feedback connections. However, LSTM and GRU are types of recurrent neural networks (RNNs) that may effectively simulate sequential data by preserving an internal state or memory. However, all four types of neural networks - MLP, CNN,

Received : 12 December 2023, Revised : 22 April 2024
Accepted : 31 July 2024, Online published : 19 September 2024

LSTM, and GRU - are well-suited for processing sequential data and can model dependencies between measurements and true values to improve tracking accuracy[14].

RNNs are designed to maintain an internal state that is updated with each new input in the sequence, allowing them to use information from previous inputs to inform their prediction for the current input[15]. This internal state, along with the current input, is used to compute the output for the current time step. In contrast, feedforward neural networks such MLPs the computation is performed layer by layer, with each layer's output serving as the input to the next layer[10]. This makes them ideal for modeling dependencies between the elements of a sequence, especially in problems where the current output depends on previous inputs.

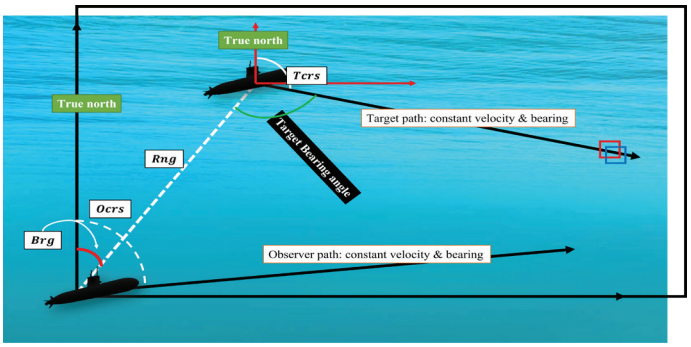
This article describes methods for tracking targets using bearing and range measurements from an active surveillance sonar mounted on a submarine to predict target states—course and speed. The UKF incorporated with machine learning was introduced which provided good results in tracking mechanism. Particle Filter (PF) with its variants as nonlinear non gaussian state estimation methods was able to handle non-Gaussian measurement noise[16,17]. However, the disadvantage of the PF is that it requires many particles to be used, which increases computation time. Other methods such as sequential Monte Carlo methods and hidden Markov models were proposed to track maneuvering targets, but their implementation was difficult to track the targets[7].

So far, the methods developed for target tracking, assume that the covariance matrix of noise in sonar measurements was known. But, in reality this is not available. Hence, using deep learning algorithms the covariance matrix of input measurements is found out, and the same is used in UKF. The deep learning algorithms used are MLP, CNN, LSTM and GRU. Python is used to realize and implement UKF and deep learning algorithms. The results of these methods are tabulated and compared. Active sonar systems are of paramount importance in the realm of underwater target tracking as they utilize chaotic sensor measurements to approximate target attributes such as course and speed. Nevertheless, current tracking techniques that employ Kalman filters or particle filters operate under the presumption that the attributes of the noise, which are symbolized by the covariance matrix, are predetermined. This information is frequently unavailable or inaccurate in the actual world, which results in tracking errors. A novel approach is proposed in this article to tackle this challenge. The system utilizes deep learning algorithms such as MLP, CNN, LSTM, and GRU in an effort to directly learn the noise covariance matrix from the sonar measurements. This obviates the necessity for prequalification regarding the attributes of the noise, which may result in enhanced precision and resilience in the tracking of the target.

This article is divided into several sections. Section I covers the introduction part and the challenges of target tracking, while Section II describes about target tracking in acoustic environment, Section III delves into the mathematical formulations of two proposed networks, the Recurrent Neural Network (RNN) and the Unscented Kalman Filter (UKF). Section IV provides description about machine leaning algorithms implementation, Section V outlines the simulation design, while Section VI discusses how machine learning algorithms are integrated into the UKF. The simulation results and analysis are presented in Section VII, and Section VIII draws conclusions based on the findings presented in the preceding sections.

2. TARGET TRACKING IN ACOUSTIC ENVIRONMENT

The unpredictable and rapidly changing nature of a moving target's motion makes tracking it a complex and non-linear task. In addition, moving in water is difficult for underwater vehicles due to the density of water, hydrodynamic forces, and the physical characteristics of the vehicle. These factors, including size, weight, buoyancy, propulsion system, and hydrodynamic properties, all affect a vehicle's maneuverability in water, making it a challenging environment to navigate reducing the speeds of UVs. The sonar system is mounted on the submarine and is being used to track a target as both the submarine and the target are moving in the direction shown in Fig. 1.



β: Bearing, Tcrs: Target course, Ocrs: Observer course, r: Range, TBA: Target bearing angle

Figure 1. Observer - target movement in acoustic environment for tracking purpose.

3. MATHEMATICAL MODELLING

In traditional tracking approaches, the movement of the target and the data collected about it are typically represented using mathematical equations that describe the target's behavior. The models employed are commonly derived from preexisting knowledge regarding the dynamics and behavior of the object. It is expected that these models are realistic enough to accurately depict the movements and observations of the target. The objective of the tracking process is to infer the target's state over time by utilizing these models and the observations of the target. The precision of the tracking outcomes relies on the precision of the model and the excellence of the observations. Target model can be described as follows:

$$X_{\eta+1} = \zeta_{T_\eta}(X_\eta) + \rho_\eta, T_\eta \in \{1,2,.....\eta\} \quad (1)$$

This is a state transition equation in the context of dynamic systems. Here, $X_{\eta+1}$ is the state of the system at time step η, ζ_{T_η} being non-linear function that describes the system's dynamics, ρ_η is an additive noise term, and T_η is a discrete random variable that determines which of the η possible dynamics the system follows at time η.

The Eqn. (1) describes how the state of the system evolves over time, given the current state X_η at time η and the function ζ_{T_η}, which represents the system's dynamics. The random variable T_η allows for the possibility of multiple different dynamics being present in the system, and the noise term ρ_η represents any unmodeled or unknown effects that influence the system's evolution. The measurements are modeled as a function of the true target states, which can be expressed as in the following Eqn.:

$$Y(\eta) = H_\eta(\chi_\eta) + V_\eta \quad (2)$$

This Eqn. (2) is an observation equation. Here, Y(η) s the observation of the system at time step η, H_η is a non-linear function that maps the state of the system to its observation, and it is determined by sensor, χ_η is the state of the system, and V_η is an additive noise term. The observation equation describes how the system's observations are related to its state.

Target tracking estimates target's state using a sequence of measurements. This is done by calculating the predicted and conditional probability densities. The conditional density gives the best estimate of the target's state. The predicted density can be calculated recursively using the Bayesian formula:

$$p(X_\eta/Y_{1:\eta-1}) = \int p(X_\eta/X_{1:\eta-1}) p(X_{\eta-1}/Y_{1:\eta-1}) d\chi_{\eta-1} \quad (3)$$

$$p(X_\eta/Y_\eta) = \frac{1}{p(Y_\eta/Y_{\eta-1})} p(Y_\eta/X_\eta) p(X_\eta/Y_{\eta-1}) \quad (4)$$

Here, $p(X_\eta/X_{1:\eta-1})$ is the posterior density of the target at time η given observations up to time η-1, $p(X_\eta/X_{1:\eta-1})$ is the posterior density of the target at time η-1 given observations up to time η-1, and $p(X_{\eta-1}/Y_{1:\eta-1})$ is the transition density or the probability distribution of the target's state at time η given its state at time η-1. $p(X_\eta/Y_\eta)$ is the likelihood of observing the measurement Y_η given the target state X_η, $p(Y_\eta/Y_{\eta-1})$ is the predicted state density or the probability distribution of the target state at time η given all previous measurements $p(Y_\eta/Y_{1:\eta-1})$ and $p(X_\eta/Y_{\eta-1})$ is the marginal likelihood or the probability of observing the measurement Y_η given all previous measurements $p(Y_\eta/Y_{1:\eta-1})$.

This system exhibits uniform motion with constant velocity, and it was modeled using a second order system following normal distribution with zero-mean. The covariance of the noise was appropriately adjusted to account for the normally small linear accelerations as follows:

$$X_{\eta+1} = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} X_\eta + \begin{bmatrix} \tau^2 & 0 \\ \tau & 0 \\ 0 & \tau^2 \\ 0 & \tau \end{bmatrix} \rho_{\eta+1} \quad (5)$$

Here, the coordinates and corresponding velocities in a 2-dimension environment are denoted by $X_\eta = [x \;\; \dot{x} \;\; \gamma \;\; \dot{\gamma}]$ and τ represents the sampling interval.

3.1 Recurrent Neural Networks Architecture

ANNs are in fact computing systems encourage by the way the brain works, designed to learn, and perform tasks by learning from examples. DNNs are a type of ANN containing multiple hidden layers connecting inputs and output layers allowing them to extract more complex features and achieve more accurate results. RNNs are another type of ANN that can use its internal memory to process sequences of input and generate corresponding sequences of output. RNNs are suitable for tasks that involve tracking and processing sequences of data.

RNNs are specifically designed to handle input that is presented in a sequence, such as time series or text data. They achieve this by maintaining a state that enables the retention of information from previous time steps. A conventional architecture of a RNN containing hidden layers that processes sequential input data. The hidden layer has connections that allow the output of the previous input to be fed back into the current input. This allows RNN to learn patterns in the input data over time. The mathematical formula for a conventional RNN architecture with the given variable substitutions can be represented as:

$$tf(\lambda) = \sigma(W_{hh} * tf(\lambda-1) + W_{hx} * x(\lambda) + b_h) \quad (6)$$

$$of(\lambda) = \sigma(W_{yh} * tf(\lambda) + b_y) \quad (7)$$

Where at λ, $tf(\lambda)$ is the hidden, $x(\lambda)$ is the input, $of(\lambda)$ is the output , Weight matrices (i.e., W_{hh}, W_{hx} and W_{yh}) are responsible for regulating the transmission of data among the input, hidden, and output layers. Bias terms (i.e., b_h and b_y) are additional values that are incorporated into the calculations of the hidden and output layers. σ is a non-linear activation function, such as the sigmoid or *tanh* function, that adds non-linearity to the network.

The Rectified Linear Unit (ReLU) activation function was employed in this study due to its computational efficiency, sparsity, and ability to circumvent the vanishing gradient problem. ReLU is mathematically given by $f(x)=max(0,x)$, x being the input to the activation function and $f(x)$ is the output. In this function, the output is zero for any input x that is less than or equal to zero, and the output is equal to the input for any input x that is greater than zero. This makes ReLU a piecewise linear function with a rectified or flattened output for negative inputs and a linear output for positive inputs.

The use of the hidden state $tf(\lambda-1)$ in the calculation of the hidden state $tf(\lambda)$ allows the RNN to remember previous inputs and learn patterns in the input data over time.

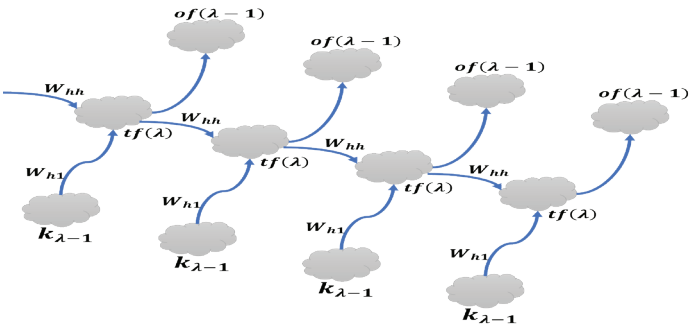


Figure 2. Recurrent neural network illustration.

The network is composed of input layers, hidden layers, and output layers, where the hidden layers serve as the network's memory unit which is given in Fig. 2. At each iteration of the input sequence, the network receives the current input and updates its hidden state using both the current input and the prior hidden state.

After being trained then only the neural network can generate the desired deterministic output. Let our RNN take such sequence of time series data k_1, k_2,…, k_λ, as input to return output sequence tf_1, tf_2, tf_3,…, tf_λ. If we have M number sequences, this can be described as follows:

$$S=\left\{\begin{matrix}\left(k_1^{(m)},v_1^{(m)}\right),\left(k_2^{(m)},v_2^{(m)}\right),\\ \left(k_3^{(m)},v_3^{(m)}\right),\\ ....,\left(k_{L_m}^{(m)},v_{L_m}^{(m)}\right)\end{matrix}\right\}\frac{M}{m=1} \tag{8}$$

An RNN's parameters, which include input sequences i and required output t, can be determined by minimizing the following cost function.

$$K(\Phi)=\frac{1}{M}\sum_{n=1}^{M}\sum_{\tau=1}^{L_m} dl(\mathfrak{t}f(\lambda),v_\lambda) \tag{9}$$

The cost function involves a predefined divergence measure, such as the Euclidean distance, between x and y, represented as $dl(x,y)$. The network parameters denoted by Φ and represented by $\Phi=\{w_{hi}, w_{hh}, w_{oh}\}$. These parameters are typically determined through the stochastic gradient descent algorithm, which involves calculating the gradient of the cost function. From a probabilistic perspective, the RNN aims to estimate the conditional probability $p(v_1,v_2,\ldots,v_\tau/k_1,k_2,\ldots,k_\tau)$ using deterministic functions. The computation of the RNN involves parameterizing this conditional probability as a product of conditional probabilities, which can be expressed as:

$$p\left(v_1,v_2,...,v_\tau/k_1,k_2,...,k_\tau\right)= \prod p\left(v_1/v_1,...,v_{\{\tau\}},v_1,...,v_{\{\tau-1\}}\right) \tag{10}$$

This involves estimating the probability of each target output t_i given the input sequence $k_1,k_2,\ldots,k_\tau$ and the previously generated targets $v_1,\ldots,v_{\{\tau-1\}}$.

3.2 Deep Neural Networks Architecture

A DNN is composed of several layers of neurons, which include input layers, hidden layers, and output layers. Every individual neuron inside the network employs an activation function to process its inputs and transmits the outcome to the subsequent layer of neurons. The equations for a feedforward DNN can be written as:

$$z_l=W_l*a_{l-1}+b_l \tag{11}$$

$$a_l=g\left(z_l\right) \tag{12}$$

where, W is the weight matrix between layer l and layer l-1, b_l is the bias vector for layer l, a_l is the output of layer 1 after applying the activation function g to its inputs z_l. The forward pass through the network can be computed recursively as:

$$a_0=x \tag{13}$$

$$a_l=g\left(z_l\right)=g\left(W_l*a_{l-1}+b_l\right) \tag{14}$$

for $l=1,2,....,L$

where, x is the input to the network and L is the number of layers in the network.

During the training process, the network receives instruction to minimize a loss function that quantifies the discrepancy between the predicted output of the network and the desired target output. Commonly, this task is accomplished by employing gradient descent or a related technique, such as backpropagation.

4. MACHINE LEARNING IMPLEMENTATION

Prior to modeling, the authors applied standard scaling to each feature individually. This was done to ensure equal contribution of all characteristics to the model's performance and to prevent any single feature from overpowering the others. Standard scaling is a process of scaling the data in such a way that each feature has an average of zero and a variance of one.

Next, the data was split into x and y and generated 3D data [samples, time steps, n features] as required by CNN, LSTM and GRU. The experiment was done with different time steps, and it was found that 10-time steps provided better results. This involved breaking the data into windows of 10-time steps and treating each window as a single input to the model. MLP was exceptional because it doesn't require 3D data, so the data was flattened back to 2D data.

Several model architectures and different parameters for each ML algorithm were tested, and the one providing better results was considered and explained below. Each model was building sequentially, and specifics are explained below:

4.1 MLP

The neural network architecture comprised of four densely connected layers. The initial dense layer consisted of 200 units utilizing a ReLU activation function, which was then followed by another dense layer containing 100 units and employing the same ReLU activation function. The third dense layer consisted of 64 units and utilized a ReLU activation function. Ultimately, the output layer consisted of a dense layer containing a number of units that matched the total number of classes in the target variable. The ReLU activation function is frequently employed in deep learning models and is highly effective in mitigating the issue of vanishing gradients.

4.2 CNN

The Neural network model comprised six layers. The initial layer consisted of a 1-dimensional convolutional layer with 200 filters, a kernel size of 2, and a ReLU activation function. The second layer consisted of a 1D max pooling layer with a pool size of 2, which was then followed by a flatten layer that transformed the output of the preceding layer into a one-dimensional array. The fourth and fifth levels consisted of thick layers with 100 and 64 units, respectively, and both employed the ReLU activation function. Ultimately, the sixth and final layer consisted of a compact layer with several units, each corresponding to the number of classes in the target variable. This layer served as the output layer of the model.

4.3 LSTM

The architecture of the model consists of three LSTM layers, each stacked on top of each other. The first layer has 200 units, the second layer has 100 units, and the third layer has 64 units. All of these levels utilize the ReLU activation function. Stacked refers to the property where the output of each recurrent layer has a same number of time steps as the input sequence. The output layer consisted of a dense layer

with 3 units, corresponding to the number of classes in the target variable. This layer served as the final layer of the model. This model architecture is well-suited for handling sequential data and accurately predicting the class labels of the target variable. LSTM layers possess the ability to acquire and retain information about long-term relationships, making them highly prevalent in a range of applications, including speech recognition and natural language processing.

4.4 GRU

The recurrent neural network model was constructed utilising stacked gated recurrent unit layers. The model consists of three GRU layers with 200, 100, and 64 units, respectively. All layers utilize the ReLU activation function, except for the last layer. The output of each GRU layer is sequentially sent to the subsequent GRU layer, with the final layer's output serving as the model's overall output. The output layer is a dense layer with 3 classes in the target variable, serving as the final layer of the model. This model architecture is well-suited for analysing sequential input and accurately predicting the class labels of the target variable. GRU layers possess the ability to acquire long-term dependencies and share similarities with LSTM layers, however with a reduced number of parameters, resulting in faster training.

To enhance the precision and efficiency of the training process, the learning rate was decreased for all the models. The initial learning rate of 0.001 was reduced to 0.0001. The learning rate governs the magnitude of the increments the model makes during optimization, and a high learning rate might lead to the model surpassing the optimal solution or diverging completely.

Another technique used to prevent overfitting was early stopping. This method was implemented by monitoring the validation loss. If the validation loss stopped improving, the training process would stop to prevent the model from continuing to train and improve performance on the training data at the expense of generalization performance on unseen data.

Given that this is a regression problem, various loss functions can be employed. However, the decision was made to use Mean Squared Error (MSE) as the loss function. The MSE calculates the average of the squared differences between the predicted and actual values. It has been demonstrated to be a superior option for regression situations.

After training, the data was scaled, so it was crucially important to inverse scale it to go back to the original range. This was done to ensure that the predictions are meaningful in the original units of measurement.

Finally, the trained models were saved along with the scalers so that they could be retrieved and reused without having to retrain them every time predictions were made. This was done to make the process more efficient and faster, especially when making predictions in real-time applications such as the UKF.

5. SYSTEM DESIGN

In this paper, the author designed a Python-based simulator to generate synthetic data that mimics the behavior of an underwater real-world system. The generated data was used to test and validate the algorithm and train machine learning models, as actual data was not available. Without actual data, simulators can provide a cost-effective and safe way to test and refine an approach under a wide range of scenarios and conditions. Moreover, since machine learning requires a large amount of data, the synthetic data for various scenarios provided insight into the performance of the machine learning algorithm. Table 1 presents an example of the initial conditions used to generate the dataset for the 2D space for the target tracking simulator design. The simulator, given these initial conditions and many others (300 in total), produced enough independent features (range and bearing measurements), as well as dependent features (standard deviation of noise in range and bearing measurements). These features are used for training and evaluating machine learning models in the simulation.

Moreover, the simulator was also used along with the UKF to predict the target's position, course and speed, given the standard deviation of noise in measurements using machine learning algorithms. Overall system is presented in the block diagram as given in Fig. 3.

Table 1. Few of several simulated initial conditions

Sc No	Brg	Tcrs	Ocrs	Rng	Vt	Vo	σ_R	σ_B
1	136	63	93	5028	8	5	12.506	0.020
2	298	73	43	4295	9	11	14.255	0.028
3	325	51	21	4957	11	7	10.141	0.023
4	189	164	194	4498	17	6	11.523	0.025
5	176	222	192	5317	19	11	11.335	0.006
6	153	216	186	5668	14	7	9.382	0.027
7	162	74	104	5841	7	5	9.258	0.009
8	161	243	213	5827	12	4	15.902	0.017
9	16	355	385	3877	17	4	13.591	0.039
10	261	177	207	5890	19	10	15.763	0.015

Note: Sc no: scenario number, Brg: starting bearing (deg), Rng: starting range (m), Vt: Target's velocity (m/s), Vo: observer's velocity (m/s), σ_R and σ_B are standard deviation of noise in range and bearing measurements respectively.

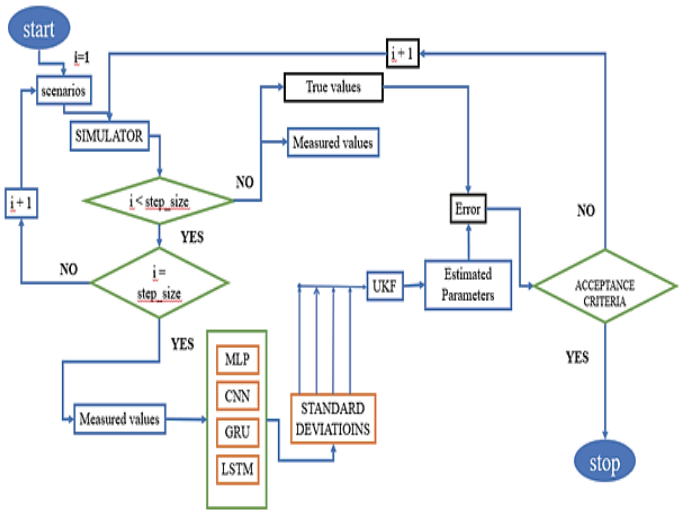


Figure 3. Descriptive block diagram of the process.

The simulation was run for a specific time interval, in this case, 10 sec, which matches the time steps considered during training. During the initial 10 sec of the simulation, the measured bearing and range were buffered and the UKF did not operate. Later, the buffered measurements were used along with saved models and scalers to make a prediction of standard deviation. The buffered data was first transformed into 3D data to match the requirements of the GRU, LSTM and CNN models but not MLP. These predictions were then used in the UKF to predict the state of the unknown target, following the block diagram shown in Fig. 3.

6. UKF BASED TARGET TRACKING

UKF-based target tracking is a widely employed method in several domains such as robotics, aerospace, and military. It is used to estimate the location and velocity of a moving target. This technique is particularly useful when the measurements are subject to noise or uncertainty. In this paper, bearing and range measurements from a sonar sensor are available, which can be used to estimate the position, course and speed of a target.

The UKF, is a recursive Bayesian filter employed for state estimation in dynamic systems. The UKF employs a collection of sigma points to estimate the probability distribution of the system state, hence enhancing the precision of the estimation. The method comprises two primary stages: prediction and update. During the prediction step, the UKF utilises a motion model to forecast the subsequent state of the target. The process involves computing the average and covariance of the sigma points and advancing them in time using the motion model. The resultant forecasted state estimation is subsequently employed as the preliminary estimation for the update phase.

During the update step, the UKF uses the available sonar measurements to update the state estimate of the target. This is done by first predicting the expected measurement using the predicted state estimate and then comparing it to the actual measurement obtained from the sonar sensor. The disparity between the anticipated and real data is subsequently employed to revise the state estimation by means of the Kalman filter equations.

Through repetitive repetition of the prediction and update processes, the UKF is capable of accurately tracking the location and velocity of the target. This is particularly crucial in situations where the target's movement is not in a straight line, or the measurements are susceptible to noise or uncertainty.

This is particularly crucial in situations where the target's movement is not in a straight line, or the measurements are affected by noise or ambiguity. The subsequent actions are executed during the implementation of the UKF algorithm for target tracking.

6.1 Sigma Point Generation

Calculate the sigma points around the current state estimate:

$$\mathrm{X} = \left[x, x + \sqrt{(L+\lambda)}P, x - \sqrt{(L+\lambda)}P \right] \tag{15}$$

where, x is the current state estimate, P is the state covariance matrix, L is a scaling parameter, and λ is the offset parameter.

6.2 Sigma Point Propagation

Propagate the sigma points through the process model to obtain the predicted state estimate and covariance:

$$x_{hat} = \sum w_i * f(\mathrm{X}) P_{hat}$$
$$= \sum w_i * \left[f(\mathrm{X}_i) - x_{hat} \right] \left[f(\mathrm{X}_i) - x_{hat} \right]^T + Q \tag{16}$$

where, f is the process model, Q is the process noise covariance matrix, w_i are the weights of the sigma points, and X_i are the propagated sigma points.

$$Q = \begin{bmatrix} d * {}_s^2 & 0 & 0 & d * \lambda_s^3 / 2 \\ 0 & d * \lambda_s^2 & 0 & 0 \\ 0 & 0 & d * \lambda_s^3 / 4 & 0 \\ d * \lambda_s^3 / 2 & 0 & 0 & d * \lambda_s^3 / 4 \end{bmatrix} \tag{17}$$

And d should be a small positive number, for example d=10^{-5}, λ_s is time steps.

6.3 Measurement Prediction

Predict the measurement from the predicted state estimate:

$$z_{hat} = \sum w_i * h(\mathrm{X}_i) \tag{18}$$

where, h is the measurement model.

6.4 Innovation Calculation

Calculate the innovation, or the difference between the actual measurement and the predicted measurement:

$$y = z - z_{hat} \tag{19}$$

where z is the actual measurement.

6.5 Measurement Update

Calculate the cross-covariance between state and measurement:

$$P_{xz} = \sum w_i * \left[\mathrm{X}_i - x \right] \left[h(\mathrm{X}_i) - z_{hat} \right]^T \tag{20}$$

6.6 Calculate the Innovation Covariance Matrix

$$S = \sum w_i * \left[h(\mathrm{X}_i) - z_{hat} \right] \left[h(\mathrm{X}_i) - z_{hat} \right]^T + R \tag{21}$$

where, R is the measurement noise covariance matrix.

6.7 Calculation of Kalman Gain

$$K = P_{xz} * S^{-1} \tag{22}$$

Calculate the updated state estimate and covariance:

$$x = x_{hat} + K * yP$$
$$= P_{hat} - K * S * K^T \tag{23}$$

where x is the updated state estimate, P is the updated state covariance, K is the Kalman gain, and y is the innovation.

7. INCORPORATION OF ML MODELS INTO UKF

The accuracy of a UKF's state estimation and covariance update is influenced by the correct estimation of the standard deviation. An underestimation can lead to overconfidence in the estimate, while overestimation can lead to over-conservatism, both resulting in poor performance. To optimise UKF's performance, it is necessary to estimate the standard deviation

using prior knowledge of the system and measurement sensors. In UKF, the assumed standard deviation is represented by the process noise covariance matrix and the measurement noise covariance matrix, which are used to compute the sigma points during the unscented transformation step to estimate the state and covariance of the system. To design a machine learning based UKF, various parameters and features must be considered, including raw sensor data, sensor configuration, physical constraints, environmental factors, quality and quantity of training data, and ML algorithm parameters. These parameters are included in the standard deviation for the implementation of the UKF.

8. SIMULATION ANALYSIS AND RESULTS

A 2D space target tracking simulator was created to track a target using measurements of range and bearing, with noise added to the measurements. Machine learning models such as MLP, CNN, LSTM, and GRU were used to forecast standard deviations in range and bearing. These predictions, along with the measurements, were used in an Unscented Kalman Filter (UKF) to predict the unknown states of the target. The Monte Carlo method was used to model nonlinearities in the target motion and measurement models for more accurate predictions. The results of the 10th scenario from the table are shown. Moreover, the initial target's state vector is computed using the initial distance of 6500 mtr and the velocity of 10 meters per second. The state vector is assumed to follow a Gaussian distribution with a zero mean; therefore, the following expression is derived:

$$X_s(0) = \begin{bmatrix} 15.0 & 15.0 & 6500 * \sin\beta & 6500 * \cos\beta \end{bmatrix} \quad (24)$$

Starting covariance matrix

$$P\begin{pmatrix} 0 \\ 0 \end{pmatrix} = diagonal\left[4 * \frac{X_s(k)^2}{12} \right] \quad (25)$$

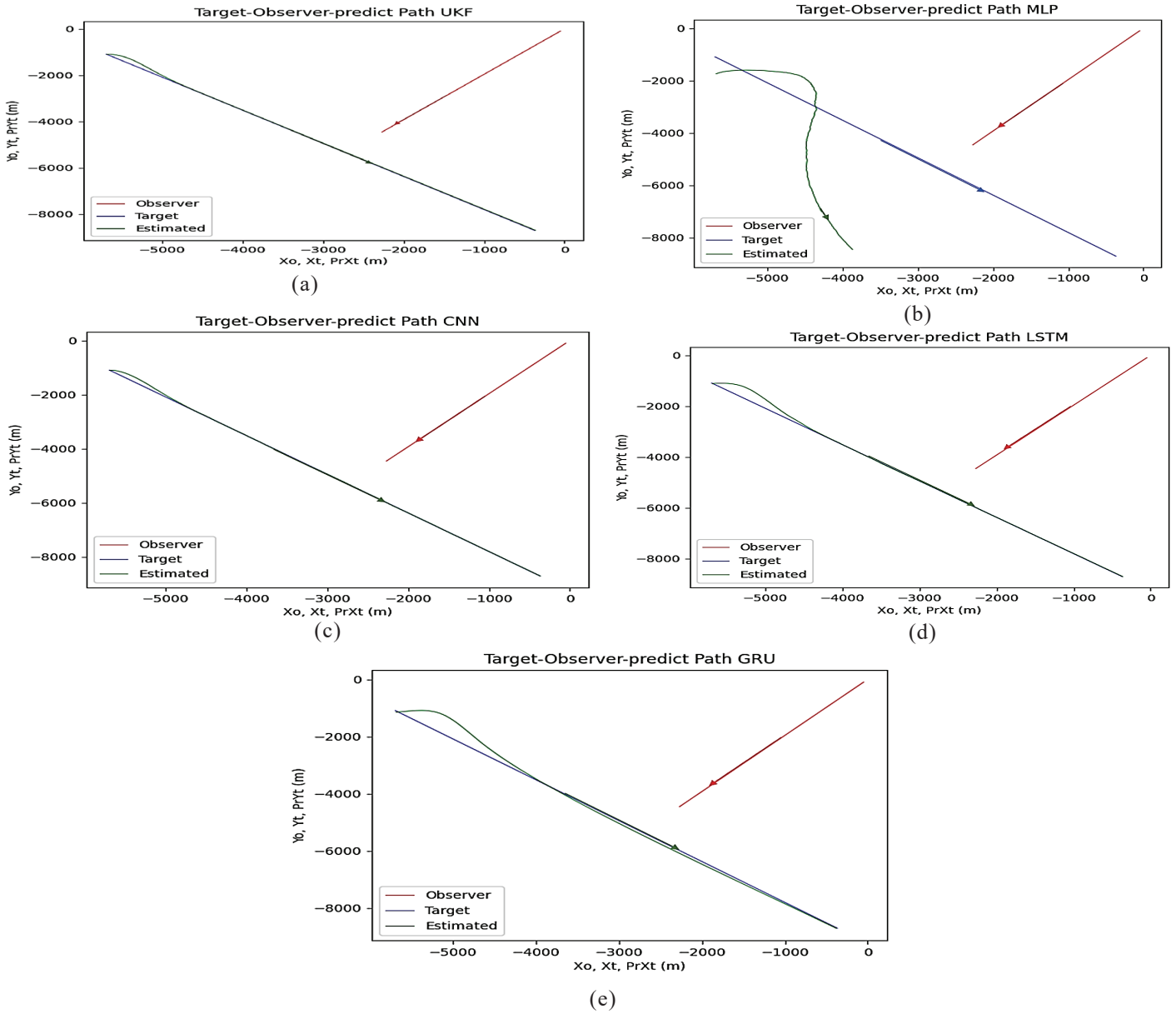


Figure 4. Observer, target's true, and estimated paths using (a) UKF only, (b) UKF with MLP, (c) UKF with CNN, (d) UKF with LSTM and (e) UKF with GRU

Table 2. Performance comparison of LSTM, GRU, CNN and MLP models for target tracking: Results of a test case

		Forecasted			
	True	**MLP**	**CNN**	**LSTM**	**GRU**
Sigma R	10.005	11.1987	10.969	10.122	10.156
Sigma B	0.0045	0.02364	0.005174	0.001249	0.003013

Multiple Monte Carlo simulations were employed to address randomness and improve estimates, as a single experiment would be insufficient. Machine learning algorithms estimated standard deviation and UKF was used to predict unknown target states. Estimated target path was plotted against true target paths and observer. Moreover, Root Mean Square (RMS) error values corresponding to target's estimated was also plotted. Graphs below depict analysis outcomes.

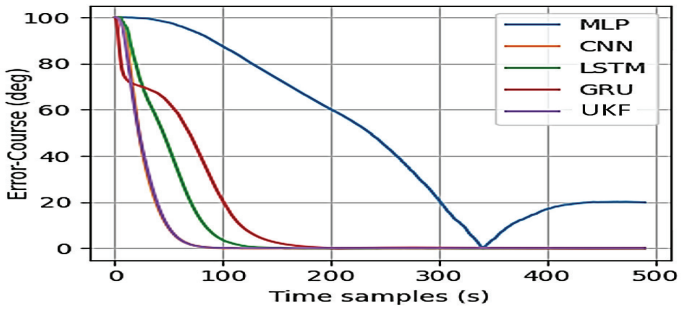
In Fig. 4 (a), the true path followed by the target and observer is displayed, along with the estimated target path using the UKF algorithm alone. The standard deviations used in the UKF algorithm are determined solely based on human intuition and experience and are assumed to be in the range of 0° to 3°. Despite this, the UKF algorithm alone has successfully estimated the target's unknown states, indicating its effective performance.

However, it is crucial to note that relying solely on human judgment to determine the standard deviations used in the UKF algorithm may not always result in optimal performance. It is often advantageous to utilize data-driven approaches to estimate the uncertainty in the system, rather than relying on subjective assessments based on experience and intuition. Such an approach can lead to improved accuracy and robustness in the estimation process.

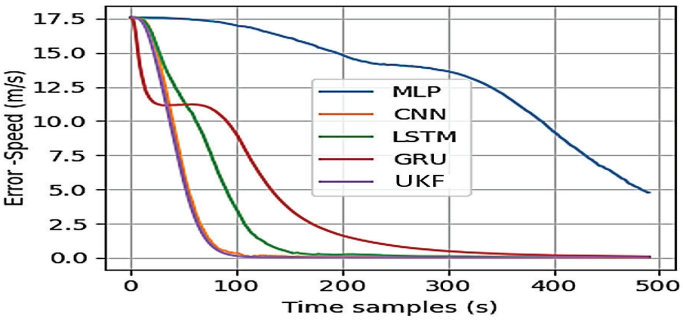
Figure 4 (a), (b), (c) and (d) display the true path of the target and observer, as well as the estimated target path obtained using different machine learning algorithms incorporated into the UKF algorithm. The specific machine learning algorithms used were the Multilayer Perceptron, Convolutional Neural Network, Gated Recurrent Unit, and Long Short-Term Memory.

Authors want to compare how different algorithms perform in terms of their error convergence when they are used in conjunction with the Unscented Kalman Filter, as well as when the UKF algorithm is used on its own.

The main objective of incorporating these machine learning algorithms was to address the issue of relying on human intuition to estimate the standard deviation. This reliance on human intuition can lead to problems and inaccuracies in estimating the standard deviation. By using pre-trained machine learning algorithms to learn patterns in measurements, the estimation of the standard deviation can be improved and the reliance on human intuition can be removed.



(a)



(b)

Figure 5. Comparison in error convergence for all algorithms when used along with UKF and when UKF algorithm is only used, (a) RMS Error in course; and (b) RMS error in speed.

Figure 5(a) and (b) serves to compare the performance of the different machine learning algorithms incorporated into the UKF algorithm in estimating the target path. This comparison allows for the determination of which machine learning algorithm performs best in estimating the target path and how well it compares to the true path. However, the purpose was not achieved because of limited number of features and the results of MLP clearly demonstrates this.

8.1 Limitation

When analysing time series data, the number of features in a machine learning model can have a significant impact on its ability to make accurate predictions. If the number of features is too low, the model may not fully capture the complex relationships and patterns within the data. This can lead to incomplete representation of the data, decreased predictive power, and increased risk of overfitting. In this paper, the authors encountered this issue due to limited number of features within the initial model, and it was unable to capture the underlying patterns and relationships in the data. To address this an elevation measurement was incorporated as an additional feature and evaluated the results. Table 3 shows the simulated scenario.

Where, inr: starting range, Elev: starting elevation, opi: observer's starting pitch, tpi: target's starting pitch. sr, sb, se: are standard deviation of range, bearing and elevation respectively.

Table 3. Simulation results

scno	ocrs	inr	β	Vt	Vo	Tcrs	Elev	opi	tpi	sr	sb	se
1	45	3000	45	9	5	255	45	45	110	10	0.34	0.24

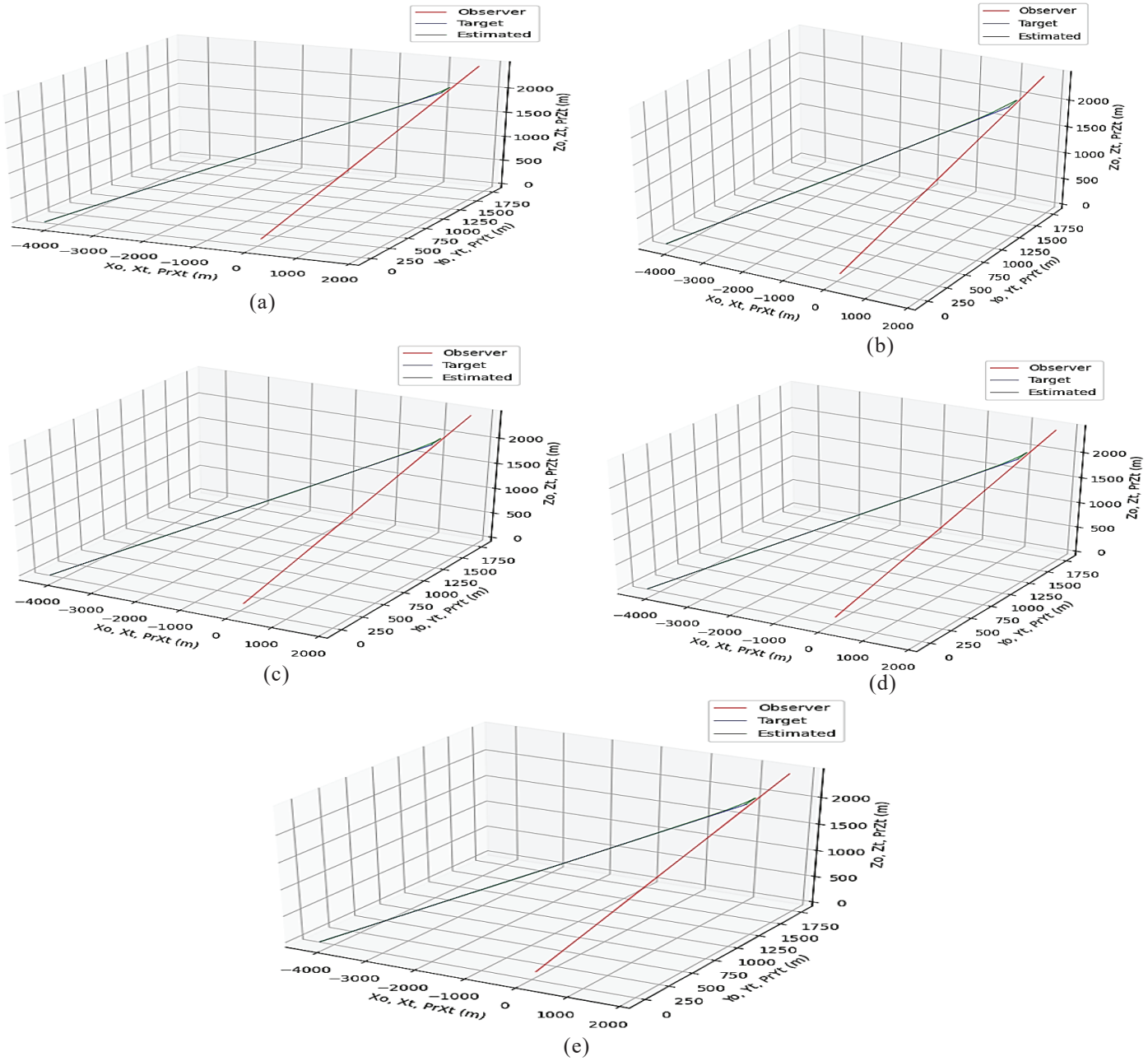


Figure 6. Observer, target's true, and estimated paths using (a) UKF with CNN, (b) UKF with LSTM, (c) UKF with MLP, (d) UKF only, and (e) UKF with GRU.

Bearing starting bearing, trgtv: Starting target's velocity, obsrv: Observer's starting velocity, trgtc: Starting target's course, elev: starting elevation, opi: Observer's starting pitch, tpi: Target's starting pitch, sr, sb, se: Standard deviation of range, bearing, and elevation respectively.

Findings revealed that including this additional feature significantly improved the model's ability to capture the complexities of the time series data hence the prediction was improved, affected the UKF to use correctly estimated Standard deviations which ultimately provided accurate estimation of target unknown states. Figure 6 represents the obtained results.

Figure 6(a) shows the true and estimated paths of the target and observer using the Unscented Kalman Filter algorithm only which means the standard deviations used in the algorithm are based on human intuition, assumption, and experience and may not always result in optimal performance. Figure 6(b), (c) and (d) display the true and estimated paths obtained using different machine learning algorithms, including multilayer perceptron, convolutional neural network, gated recurrent unit and long short-term memory.

Text emphasises that relying solely on human intuition to estimate the standard deviation can lead to inaccuracies and problems, and using pre-trained machine learning algorithms can remove the reliance on human intuition and improve the estimation of the standard deviation. It should be noted that data-driven approaches, such as machine learning algorithms, to estimate the uncertainty in the system and improve the estimation process's accuracy and robustness is shown in Fig. 7 in the form of RMS errors. We can assess how the error convergence of various algorithms is affected using

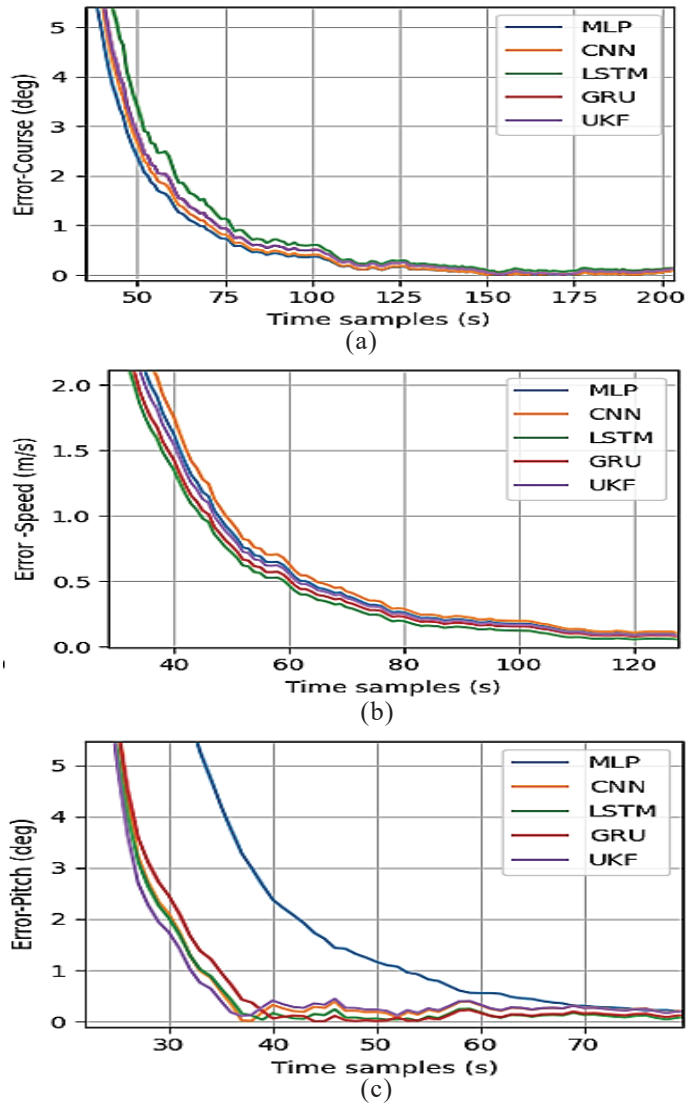


Figure 7. Comparison in error convergence for all algorithms when used along with UKF and when UKF algorithm is only used. (a) RMS Error in course, (b) RMS Error in pitch, and (c) MSE Error in speed.

the Unscented Kalman Filter both in combination with other algorithms and on its own.

9. CONCLUSION

The authors of a study found that if range and bearing measurements are available, a solution can be obtained for underwater target tracking using UKF without machine learning algorithms in six minutes. When deep learning is used, in other hand, the solution time increases to ten minutes. By adding elevation measurements, the convergence time for deep learning can be reduced to under six minutes, which suggests that the number of features was the limitation. However, the reduction in time comes at the cost of increased complexity and a longer overall convergence time. Since Elevation measurements are typically not used in underwater target tracking, the authors concluded that if an acceptable analytical solution can be obtained within the required time frame without using deep learning, it is not necessary to use it as deep learning is more complex and takes longer. While MLP can be used for time series prediction, it may not be able to capture the complex patterns and dependencies present in the data. LSTM, GRU, and CNN have more advanced structures that are designed for time series analysis and have been shown to be more effective at capturing the complexity of the data.

REFERENCES

1. Havelock, D.; Kuwano, S. & Vorländer, M. Editors. Handbook of signal processing in acoustics. New York: Springer; 2008 Oct 26.
2. Luo, J.; Han, Y. & Fan, L. Underwater acoustic target tracking: A review. *Sensors*., 2018, **18**(1), 112. doi: 10.3390/s18010112.
3. Murthy, A.S.; Rao, S.K.; Naik, K.S.; Das, R.P.; Jahan, K. & Raju, K.L. Tracking of a manoeuvering target ship using radar measurements. *Indian J. Sci. Technol.*, 2015, **8**(28), 1-6. doi: 10.17485/ijst/2015/v8i28/73788
4. Divya, G.N. & Rao, S.K. Implementation of ensemble Kalman filter algorithm for underwater target tracking. *J. Control and Decision*. 2022, 1-10. doi: 10.1080/23307706.2022.2092039
5. Jagan, B.O.L. & Rao, S.K. Measure of nonlinearity for underwater target tracking using hull-mounted sensor. *Int. J. Intelligent Comput. Cybernetics*. 2022, **15**(3), 333-44. doi: 10.1108/IJICC-08-2021-0167
6. Rao, S.K.; Lakshmi, M.K.; Jahan, K.; Naga, D.G. & Omkar, B.L.J. Acceptance criteria of bearings-only passive target tracking solution. *IETE J. Res.*, 2023, **69**(5), 2874-85. doi: 10.1080/03772063.2021.1906769
7. Raju, K.L.; Rao, S.K.; Das, R.P.; Santhosh, M.N. & Murthy, A.S. Passive target tracking using unscented kalman filter based on monte carlo simulation. *Indian J. Sci. Technol*. 2015, **8**(29), 1-7. doi: 10.17485/ijst/2015/v8i29/76981
8. Sharaga, N.; Tabrikian, J. & Messer, H. Optimal cognitive beamforming for target tracking in MIMO radar/sonar. *IEEE J. Selected Topics in Signal Process.* 2015, **9**(8), 1440-50. doi: 10.1109/JSTSP.2015.2467354
9. Montavon, G.; Samek, W. & Müller, K.R. Methods for interpreting and understanding deep neural networks. *Digital Signal Process*. 2018, **73**, 1-5. doi: 10.1016/j.dsp.2017.10.011
10. Ramchoun, H.; Idrissi, M.J.; Ghanou, Y. & Ettaouil, M. Multilayer perceptron: Architecture optimization and training with mixed activation functions. In Proceedings of the 2nd international Conference on Big Data, Cloud and Applications. 2017, Article number 71, PP. 1-6. doi: 10.1145/3090354.3090427
11. Albawi, S.; Mohammed, T.A. & Al-Zawi, S. Understanding of a convolutional neural network. *In* 2017 international conference on engineering and technology (ICET). 2017, PP. 1-6. doi: 10.1109/ICEngTechnol.2017.8308186
12. Yu, Y.; Si, X.; Hu, C. & Zhang, J. A review of recurrent

neural networks: LSTM cells and network architectures. *Neural Comput*. 2019, **31**(7), 1235-70.
doi: 10.1162/neco_a_01199

13. Shen, G.; Tan, Q.; Zhang, H.; Zeng, P. & Xu, J. Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Comput., Sci*. 2018, **131**, 895-903.
doi:10.1016/j.procs.2018.04.298
14. Brownlee, J. Deep learning for time series forecasting: Predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery; 2018 Aug 30.
15. Salehinejad, H.; Sankar, S.; Barfett, J.; Colak, E. & Valaee S. Recent advances in recurrent neural networks. arXiv preprint arXiv:1801.01078. 2017.
doi: 10.48550/arXiv.1801.01078
16. Branco, R. Beyond the Kalman filter; Partical filter for tracking application. British Library Cataloging in Publication data. 2004, 318.
17. Aidala, V. & Hammel, S. Utilization of modified polar coordinates for bearings-only tracking. *IEEE Transact. on Automatic Control*. 1983, **28**(3), 283-94.
doi: 10.1109/TAC.1983.1103230

CONTRIBUTORS

Mr Uwigize Patrick is a Researcher and Engineer, who holds a Bachelor's degree in Electronics and Telecommunication Engineering from the National University of Rwanda. His research interests include: Signal processing, wireless communication networks and machine learning for signal processing.
His contribution in the current study is software development.

Dr S. Koteswara Rao is working as a Professor at the Department of ECE, Koneru Lakshmaiah Education Foundation (Deemed to be University), Vaddeswaram, Guntur, Andhra Pradesh, India. He obtained PhD from College of Engineering, Andhra University, A.P.
His contribution in the current study is conceptualisation, methodology development and supervision.

Dr B. Omkar Lakshmi Jagan is working as an Associate Professor in Department of Electrical and Electronics Engineering, Vignan's Institute of Information Technology (A), Duvada, Visakhapatnam, Andhra Pradesh, India. He obtained PhD from the Koneru Lakshmaiah Education Foundation (Deemed to be University), Vaddeswaram, Guntur, A.P., India. His areas of research include: Statistical signal processing, biomedical signal processing and image processing technologies.
His contribution in the current study is methodology and mathematical modelling.

Dr M. Kavitha Lakshmi obtained PhD in KL Demeed to be university, Vijayawada. She is currently working as Research Associate at DRDO-NSTL, sponsered Project at the Dept. of ECE, Koneru Lakshmaiah Education Foundation (Deemed to be University), Vaddeswaram, Guntur, Andhra Pradesh, India.
His contribution in the current study is validation and software coding.

Dr Thayyaba Khatoon Mohammad is a Professor in the Department of CSE-AIML at Malla Reddy University. She obtained PhD from JNTUH. Her area of expertise is Artificial intelligence, nature language processing, network security and cloud computing.
His contribution in the current study is editing and reviewing the concepts.