Ensemble of KalmanNets for Maneuvering Target Tracking

Marco Mari
DMIF
University of Udine
Udine, Italy
ORCID: 0009-0007-6465-5961

Lauro Snidaro
DMIF
University of Udine
Udine, Italy
ORCID: 0000-0003-3828-9017

***Abstract*—Tracking a maneuvering target requires the modeling of the target's movements by multiple pre-defined mathematical models. However, the uncertainty in the target's dynamics can lead traditional model-based (MB) tracking algorithms to significant performance degradation when model mismatch occurs. To tackle this problem, we propose the use of a Recurrent Neural Network (RNN) for the purpose of learning complex target dynamics. Following the recent advances in state estimation provided by KalmanNet, a neural network-aided Kalman Filter, the proposed approach aims to exploit its tracking performance in a multiple model schema to compensate for model mismatch across maneuvers, leading to a more prompt response to motion switches. The results over a simulated set of maneuvering target trajectories demonstrate the potential of the proposed approach over the MB solution.**

***Index Terms*—Target Tracking, Kalman Filter, Recurrent Neural Network**

I. Introduction

Target state estimation is a critical task in target tracking. A precise target state estimation can be achieved through the implementation of a dependable and consistent target tracking algorithm. In the context of Bayesian tracking, the task is addressed through the application of mathematical models that accurately depict the dynamic state of the system over time.

In 1960, Emil Rudolf Kalman proposed an optimal estimator for linear Gaussian systems [1], which has since become the most commonly used method for state estimation in discrete time-variant dynamical systems, known as the Kalman Filter (KF). However, as the KF assumes the system state to be governed by a linear state-space model, many problems arise in predicting the states of non-linear dynamical systems. To address non-linear dynamics, the Extended Kalman Filter (EKF) linearizes the State Space (SS) model through its Jacobian matrix [2]. The KF's minimal complexity, along with its solid theoretical foundation, resulted in it swiftly becoming the main method of state estimation in discrete-time systems adequately described by SS models, and it has been implemented for real-life problems such as radar target tracking [2], ballistic missile trajectory estimation [3], and, remarkably, determining the location and velocity of a spacecraft during the Apollo missions [4].

Nevertheless, in numerous practical cases, obtaining precise mathematical models for tracking maneuvering targets might be challenging due to the ever-changing and unpredictable nature of the target's movements. Consequently, pre-defined mathematical models utilized in maneuvering-target tracking algorithms frequently fail to perform adequately, as the predicted motion no longer holds true. Enhancing the efficacy of these algorithms remains a significant obstacle.

Representing the target as a jump Markov process is useful for better modeling its dynamics over time, especially when these might change. Using multiple models, the target's dynamics assume the form of one SS model in a set of models, and at each time step, its motion can assume the form of one of the equations in the set. Among the algorithms representing the target's dynamics as a jump Markov process, the Interactive Multiple Models (IMM) [5] procedure has emerged as the gold standard for maneuvering target tracking.

Still, model-based algorithms require efficient modeling and identification of mathematical SS equations and the meticulous calibration of models' parameters, which makes them sensitive to errors in analytical formulation, resulting in suboptimal performances in practical scenarios where targets exhibit intricate dynamic behavior. This limits the applicability of model-based approaches when the underlying model's mathematical formulation is unclear, difficult to express analytically, or expensive to estimate.

In recent times, real-world applications have experienced notable empirical achievements through the utilization of Deep Neural Networks (DNN). Nonetheless, Recurrent Neural Network (RNN), in particular Long Short Term Memory (LSTM) [6] and Gated Recurrent Unit (GRU) [7], have been demonstrated to be effective in the analysis of sequential data [8], and they have also found application in the field of target tracking. In [9], LSTMs are employed to model the matrices within the KF, whereas in [10], they are used to predict the Gaussian distribution parameters of the target states. Nevertheless, in both of these works, the underlying model is assumed to be linear and time-invariant. Additionally, LSTMs have also been proposed for addressing motion uncertainty in the target's motion. In [11], LSTMs are used to sequentially output a state estimate from noisy observations. In [12], a bi-directional LSTM elaborates observation and estimated trajectories, which are then used to model the discrepancies between the true and estimated target trajectories by an UKF. In a similar vein, in

[13] and [14], aside from some differences, a transformer-based architecture is designed to perform tracking of maneuvering targets in a simulated Air Traffic Control (ATC) scenario, bypassing the need for proper model formulation and identification.

Although their performance has improved, using LSTMs or Transformers in an end-to-end fashion often results in a loss of interpretability, which is often an appreciated characteristic of model-based algorithms and, in some application fields, a discriminating factor in favor of their choice.

In this context, the KF recursive framework has served as a significant source of inspiration for numerous recent works, that have utilized it as a foundation for developing neural network based methods for target tracking.

In [15], a modular approach utilizing GRU to correct the fallacies inherent in the KF recursion in maneuvering target tracking is presented. The latter represents a promising hybrid approach for maneuvering target tracking, i.e., a trade-off between optimal tracking performance and sufficient model explainability.

In [16], the potential of hybrid methods for state estimation has been firmly demonstrated. However, the validity of their approach has yet to be evaluated in a scenario where the system state cannot be timely and precisely modeled, as in the maneuvering target case. In [17], a hybrid architecture makes use of an LSTM to predict the state estimate and its covariance, as in the prediction step of the traditional KF recursion for maneuvering target tracking. The predicted state and covariance are then fed into the KF workflow to perform the update step, with full knowledge of the measurement model. Nevertheless, their approach must be validated in a less controlled and more realistic environment, as this work intends to simulate, following ATC simulation settings in [12]–[14].

This paper proposes a KalmanNets fusion architecture that is capable of handling target motion uncertainty by leveraging multiple models and fusing their track information in an IMM-like fashion through a GRU module. Finally, the proposed method is compared to the KF as well as to a single KalmanNet in a simulated maneuvering target scenario to validate its improvements in the described setting.

The remainder of the paper is organized as follows: Section II describes the State Space (SS) model and the KF recursion for target state estimation; Section III addresses the design of the ensemble architecture for handling the uncertainty in the target's motion and the implementation details; Section IV describes the experimental setting and the results; Section V summarizes the work, draws some conclusions and outlines future perspectives.

II. State Space Model and Kalman Filter

In the experiments presented in Section IV, we analyze the efficacy of the proposed method in tracking maneuvering targets whose motion can be described by a jump Markov model with interchangeable SS models. An SS model is a description of a dynamical system. In this work, we consider SS models in discrete time for Gaussian systems:

$$x_t = f(x_{t-1}) + w_t,\ w_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}),\ x_t \in \mathbf{R}^n \tag{1}$$

$$z_t = h(x_t) + v_t,\quad v_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}),\quad z_t \in \mathbf{R}^m \tag{2}$$

where the state vector x_t evolves at time t with $t = 1, \dots, T$, from the previous time step through the state evolution function $f(\cdot)$ and undergoes a process noise, characterized by an additive white Gaussian noise having zero mean and covariance $\boldsymbol{Q}$. At the same time, the measurements on the maneuvering target are provided by a measurement model simulating the behavior of a Radar sensor, whose equation is:

$$\begin{aligned} z_t &= \begin{bmatrix} r_t \cdot \cos\phi_t \\ r_t \cdot \sin\phi_t \end{bmatrix}, \quad \begin{bmatrix} r_t \\ \phi_t \end{bmatrix} = \begin{bmatrix} \sqrt{x_t^2 + y_t^2} \\ \arctan\frac{y}{x} \end{bmatrix} + \begin{bmatrix} v_{t,r} \\ v_{t,\phi} \end{bmatrix} \\ v_t &= \begin{bmatrix} v_{t,r} \\ v_{t,\phi} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}), \quad \boldsymbol{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \end{aligned} \tag{3}$$

For systems whose SS model is Gaussian and Linear, namely $f(\cdot) = \boldsymbol{F}$ and $h(\cdot) = \boldsymbol{H}$, the KF represents the minimum mean squared error recursive estimator. At each time step t the recursion takes place in two steps:

1) Prediction step It computes the *a priori* state and covariance given the last time step prediction:

$$\hat{x}_{t|t-1} = \boldsymbol{F} \cdot \hat{x}_{t-1|t-1} \tag{4}$$

$$\Sigma_{t|t-1} = \boldsymbol{F} \cdot \Sigma_{t-1|t-1} \cdot \boldsymbol{F}^T + \boldsymbol{Q} \tag{5}$$

and the *a priori* statistical moments for the measurements:

$$\hat{z}_{t|t-1} = \boldsymbol{H} \cdot \hat{x}_{t|t-1} \tag{6}$$

$$S_{t|t-1} = \boldsymbol{H} \cdot \Sigma_{t|t-1} \cdot \boldsymbol{H}^T + \boldsymbol{R} \tag{7}$$

2) Update step It corrects the predictions made using the available measurement at the time step t providing the *a posteriori* state prediction along with its covariance:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + \mathcal{K}_t \cdot \Delta z_t \tag{8}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - \mathcal{K}_t \cdot S_{t|t-1} \cdot \mathcal{K}_t^T \tag{9}$$

where $\Delta z_t = z_t - \hat{z}_{t|t-1}$, called *innovation*, represents the error made by approximating the current measurement with the measurement model, and

$$\mathcal{K}_t = \Sigma_{t|t-1} \cdot H^T \cdot S_{t|t-1}^{-1} \tag{10}$$

is known as the Kalman Gain.

In the case of non-linear systems, the EKF linearizes the state evolution function $f(\cdot)$ and the measurement model $h(\cdot)$ with the respective Jacobian matrices evaluated at the time step under recursion.

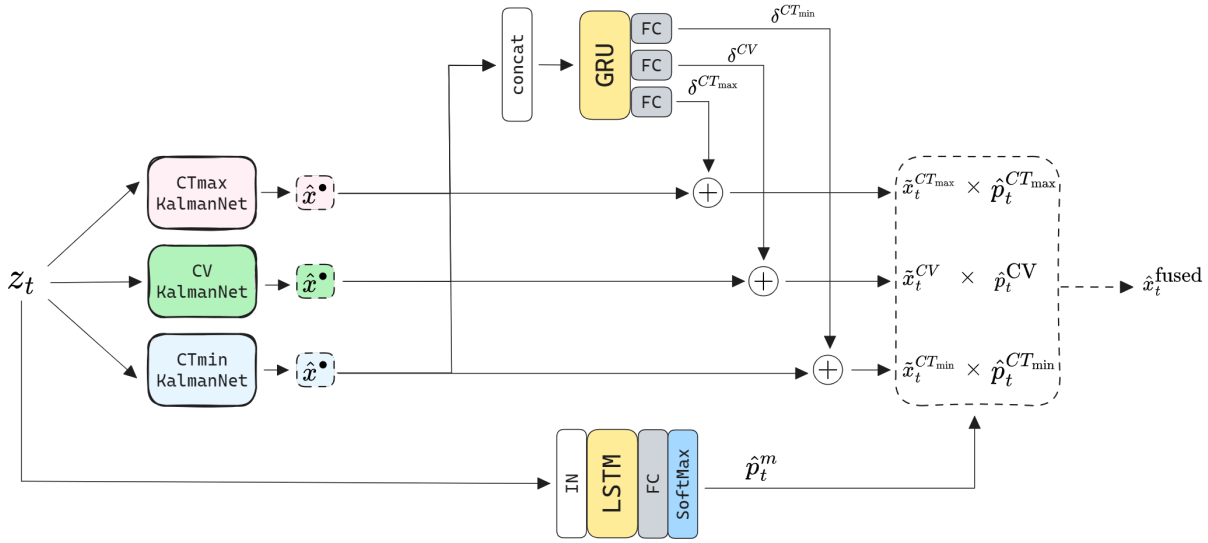

Fig. 1. Ensemble model with tracks correction and motion mode estimation.

III. Ensemble of Kalman Networks

In this Section we present an ensemble architecture for fusing Kalman Networks using different motion models. Fig. 1 illustrates the logical progression of the data.

The measurements coming from the radar sensor are first filtered by the set of Kalman Networks, each one providing an *a posteriori* state estimate. The latter are concatenated in the state dimension and passed to a GRU module. This module learns the temporal relationship of the target's maneuver.

The hidden state of the GRU module is exploited by a set of Fully-Connected (FC) Layers, named FC in Fig. 1, that use the information extracted from the GRU module to infer a correction term δ_t over each of the *a posteriori* state estimates provided by the KalmanNets $\hat{x}_t$, resulting in $\tilde{x}_t = \hat{x}_t + \delta_t$ being the corrected state estimate at time t. The choice of the GRU module over the LSTM is motivated by its demonstrated superiority in terms of performance, relative to the number of parameters it has. In fact, although the GRU has fewer parameters than the LSTM, it has been shown that the overall performance is comparable, or sometimes even better in many sequence modeling tasks. Additionally, many of the previous works cited in Section I, such as [15] and [16], have made the same choice for the upper branch in Fig. 1, which aims at estimating the correction terms for each of the KalmanNets' *a posteriori* estimates.

As the GRU embeds the temporal relationship from all of the filters, the correction term is computed by considering useful information coming from all of them rather than merely from the corrected one, improving the correction task by gaining advantage from more reliable motion models from time to time. In light of the insights gleaned from related works [12] and [14], it has been determined that residual learning enhances the convergence of design parameters' optimization and the network's capacity to infer over unseen data. Consequently, it has been selected as the preferred alternative to direct state estimation by the neural network, a strategy that has often yielded unsatisfactory results in practice.

A separate branch of the system feeds the radar sensor measurements into a LSTM, which is designed to estimate the probability of motion modes at the time t, namely $\hat{p}_t^m \in [0,1]\,,\ m \in \mathcal{M}$, being $\mathcal{M} = \{\mathrm{CV}, \mathrm{CT}_{\max}, \mathrm{CT}_{\min}\}$ the set of motion models used by the three KalmanNets and constrained by the relationship $\sum_{m\in\mathcal{M}} \hat{p}_t^m = 1$.

Although GRUs have demonstrated efficacy in KalmanNets' trace correction tasks, our experimental observations indicate that LSTMs exhibit superior performance in the context of the target motion classification task. This task involves learning temporal relationships over long sequences. LSTMs, with their enhanced capacity for memory retention and adaptation to evolving motion patterns, provide reliable accuracy in motion classification and prompt adaptation to motion switches.

The state evolution function that describes the three motion models will be discussed in detail in Section III-B. In particular, in Fig. 1 CV KalmanNet indicates a KalmanNet that tracks the target using a Nearly Constant Velocity (NCV) motion model. $\mathrm{CT}_{\max}$ KalmanNet and $\mathrm{CT}_{\min}$ KalmanNet refer to two KalmanNets that exploit a Constant Turn Rate (CTR) motion model with a constant turn rate of $+10°/s$ and $-10°/s$, respectively. These represent the maximum and minimum turn rates according to common ATC parameter settings, as described in Section III-B.

In the end, the corrected state estimates $\tilde{x}_t^m$ resulting from the filtering and correction process of each of the KalmanNets are averaged using the estimated motion mode probabilities $\hat{p}_t^m$, with resulting in the fused state $\hat{x}_t^{\mathrm{fused}}$, for $t = 1, \ldots, T$:

$$\hat{x}_t^{\mathrm{fused}} = \sum_{m\in\mathcal{M}} \hat{p}_t^m \tilde{x}_t^m \tag{11}$$

A. Loss Function

A proper loss is considered for each of the predicted correction terms by considering the ground truth gap between

each of the *a posteriori* estimate from the m-th KalmanNet and the actual trajectory: $\delta_t^{GT,m} = x_t^{GT} - \hat{x}_t^m$. Let N be the number of maneuvering trajectories in the training set and T the number of time steps in each of them, the loss for the correction task is then:

$$\mathcal{L}_{\text{corr},m} = \frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\frac{\left(\delta_{n,t}^{GT,m} - \delta_{n,t}^{m}\right)^2}{\|\delta_{n,t}^{GT,m}\|_2}, \; m \in \mathcal{M} \quad (12)$$

representing the normalized MSE between the estimated correction and the truly needed one.

The classification of the motion mode has been trained using a cross-entropy loss function, defined as follows:

$$\mathcal{L}_{CE} = -\sum_{m\in\mathcal{M}}\sum_{t=1}^{T}\bar{y}_t^m \log(\hat{p}_t^m) \quad (13)$$

where $\bar{y}_t^m$ is the one-hot encoding of the ground truth motion mode.

On the other hand, the loss required by the fusion task is the normalized MSE between the fused state and the ground truth reference, namely:

$$\mathcal{L}_f = \frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\frac{\left(x_t^{GT} - \hat{x}_t^{\text{fused}}\right)^2}{\|x_t^{GT}\|_2} \quad (14)$$

By combining the tasks together, the composed loss is then the sum of the above loss functions:

$$\mathcal{L} = (1-\gamma_{CE})\cdot\left(\mathcal{L}_f + \frac{1}{\|\mathcal{M}\|}\sum_{m\in\mathcal{M}}\mathcal{L}_{\text{corr},m}\right) + \gamma_{CE}\cdot\mathcal{L}_{CE} \quad (15)$$

where γ_{CE} is a weighting factor between the MSE contributions and the cross-entropy contribution, which balances the learning of the regression and the classification task. The normalization of the MSE terms in the combined loss is required to ensure that the MSE scales in the same range as the cross-entropy loss, preventing the gradients of the larger loss from disproportionately influencing the update of the entire architecture's parameters.

B. Data Generation and Training Algorithm

The Ensemble of KalmanNets is trained via supervised learning by providing a dataset $\mathcal{D} : \{(Z_{1,\dots,T}, X_{1,\dots,T})\}$ of simulated trajectories that constitutes the ground truth reference.

The training set is composed of $N = 5000$ trajectories lasting $T = 200$ time steps. Generation parameters, influenced by [14] and [12], are briefly summarized below.

During the simulation, the target can assume either a NCV transition model or a CTR one, that are both characterized by a linear state evolution function, namely

$$x_t = F\cdot x_{t-1} + w_t, \; w_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}) \quad (16)$$

- Nearly Constant Velocity (NCV) model

$$F = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad (17)$$

$$Q = \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & \Delta t \end{bmatrix}\sigma_q \quad (18)$$

- Constant Turn Rate (CTR) model

$$F = \begin{bmatrix} 1 & \frac{\sin\omega\Delta t}{\omega} & 0 & -\frac{1-\cos\omega\Delta t}{\omega} \\ 0 & \sin\omega\Delta t & 0 & -\sin\omega\Delta t \\ 0 & \frac{1-\cos\omega\Delta t}{\omega} & 1 & \frac{\sin\omega\Delta t}{\omega} \\ 0 & \sin\omega\Delta t & 0 & \cos\omega\Delta t \end{bmatrix} \quad (19)$$

$$Q = \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 0 & 0 \\ 0 & 0 & \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ 0 & 0 & \frac{\Delta t^2}{2} & \Delta t \end{bmatrix}\sigma_q^2 \quad (20)$$

For each trajectory, the starting point is sampled in a polar coordinate system, meaning that the initial angle with respect to the positive x axis is uniformly sampled from $\mathcal{U}(-180°, +180°)$, while the initial distance from the origin is sampled from $\mathcal{U}(50km, 150km)$. The initial velocity module is sampled in $\mathcal{U}(100\ m/s, 340\ m/s)$, while its direction is sampled in the interval $[-180°, 180°]$. The angular velocity for the CTR motion model is instead taken in the range $[-10°/s, 10°/s]$. Those sampled quantities, constituting the initial targets' state, are converted to Cartesian coordinates. The simulation is conducted over the entire time interval $t = 1, \dots, T$, with each time step separated by a time interval $\Delta t = 1\ s$. The noise parameter for the motion model σ_q is randomly selected from $\mathcal{U}(8.0\ m/s^2, 13.0\ m/s^2)$.

The measurements from the simulated radar sensor are affected by white Gaussian noise, whose covariance parameters, according to (3), are $\sigma_r = 40\ m$ and $\sigma_\phi = 0.1°$. Each trajectory in the training set is allowed to change its motion model during the simulation with a maximum number of changes fixed at 2. A subset of the generated test trajectories is illustrated in Fig. 2.

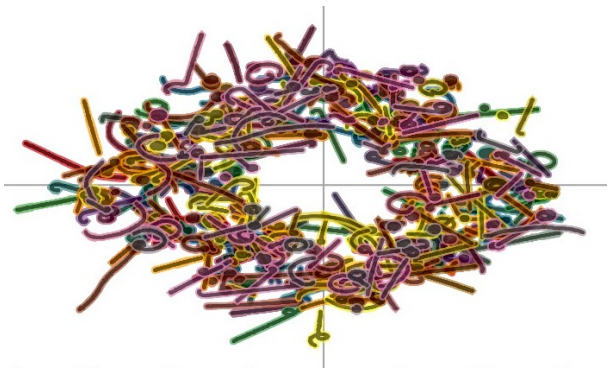

Fig. 2. Examples of training trajectories.

The optimization algorithm chosen to carry out the training is the ADAptive Moment estimation (ADAM) algorithm [18].

The learning rate is set to 0.001. The maximum number of epochs is 2000, and at the end of each epoch, the validation loss is computed: the parameter setting that gives the best validation loss is saved and used for testing purposes. The batch size for the optimization algorithm is 256 trajectories. Gradient clipping is used in order to avoid the divergence of the gradient during the training, fixing the threshold for the $\mathcal{L}_2$ norm of the gradient to 10. The weighting factor γ_{CE} for balancing the learning between regression and classification tasks is set to 0.5.

IV. Experiments and Results

The dataset used to evaluate the effectiveness of the proposed method is composed of simulated target trajectories that follow predefined maneuvering procedures. For each defined maneuver, a total of 100 Monte Carlo runs are collected by setting all of the generation parameters but σ_q, the process noise, which is randomly selected following the distribution in the Section above. In particular: $\sigma_r = 40.\ m$, $\sigma_\phi = 0.1°$.

Each of the KalmanNets used in the architecture has the settings specified in [16]. The LSTM dedicated to the estimation of motion probabilities has 2 hidden layers with 64 neurons each and the GRU dedicated to the prediction of correction terms for the KalmanNets' *a posteriori* estimates has 4 hidden layers with 64 neurons per layer. The feed-forward, fully connected layers associated to the GRU in the branch estimating the correction terms have 4 neurons in output: 2 of them predicting the position in the x-y axes, and the other 2 predicting the velocity. Instead, the GRU for the motion mode classification is followed by aFC net that outputs 3 probability logits that are then transformed into motion mode probabilities by a softmax operation.

	Sector Motion Parameters
sector 1	$70\ s$, CT $\omega = 1°/s$
sector 2	$40\ s$, CT $\omega = 9°/s$
sector 3	$90\ s$, CT $\omega = -2°/s$

TABLE I
Test trajectories' maneuver settings

The settings of the trajectories maneuvers are detailed in the Table I, and the result of the simulations is displayed in Fig. 3. We tested the performance of the KalmanNets' Ensemble against an IMM using the same motion models by measuring the RMSE in position and velocity concerning the ground truth trajectory. In the following results, time steps from 1 to 5 have been discarded to allow the IMM filter to converge after the initialization phase. Fig. 4 illustrates an example of tracking in the test scenario, while the plot in Fig. 5 summarizes the average performance in these settings for both the Ensemble approach and the IMM.

In the considered scenario, the proposed approach has proven its effectiveness in terms of RMSE, highlighting more reliable tracking performances when compared to the IMM filter. For most of the simulation, the Ensemble of the KalmanNets overcomes the fallacies of the IMM filter and provides

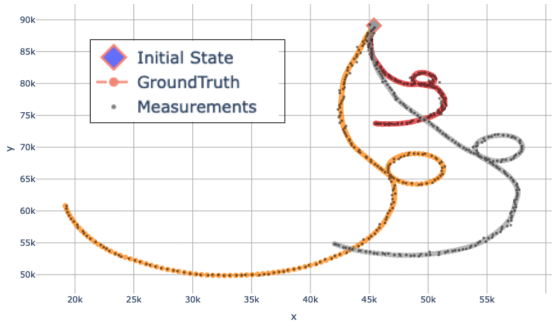

Fig. 3. Example of test trajectory.

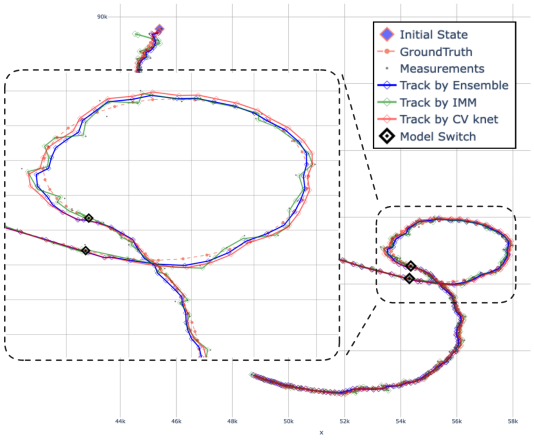

Fig. 4. Examples of tracking results on a single test sample with zoom on motion switch.

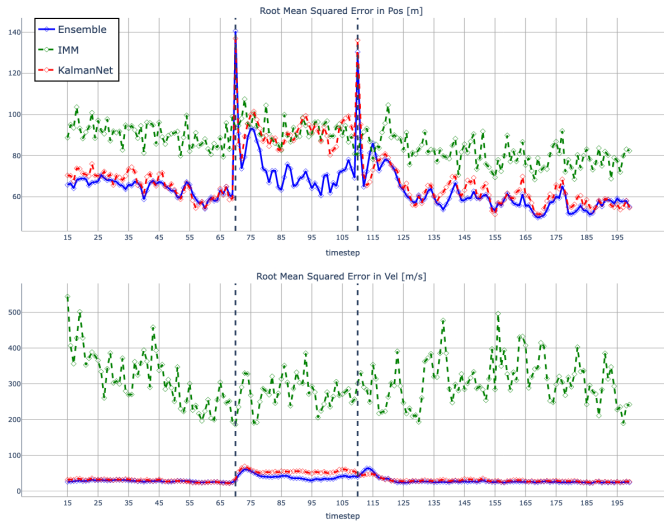

Fig. 5. Average tracking results in the testing scenario.

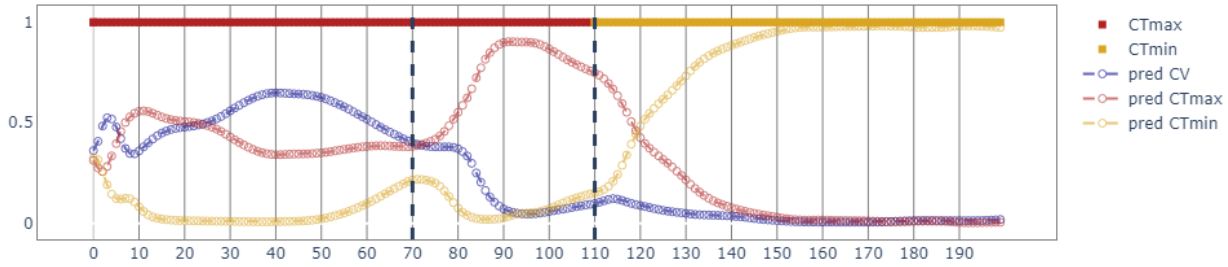

Fig. 6. Average motion prediction accuracy in the testing scenario.

more accurate tracking of the maneuvering target, and shows a better flexibility during motion uncertainty when compared to the KalmanNet using a single motion model.

When the target dynamics change, spikes in the errors can be observed. Nevertheless, the Ensemble approach shows prompt reactivity to the motion change, quickly overcoming the motion model mismatch. On the other hand, the Ensemble of KalmanNets is consistently better than the IMM filter in tracking the target velocity, which is the unobservable state due to the available sensor limitations. The average test results per sector are listed in Table II.

Test Maneuver [$\mu \pm \sigma$]

	Position RMSE	Velocity RMSE
IMM	91.16 ($\pm$7.85) m	309.21 ($\pm$105.53) m/s
	91.21 ($\pm$11.39) m	276.19 ($\pm$116.91) m/s
	89.63 ($\pm$8.32) m	271.51 ($\pm$88.39) m/s
KalmanNet [16]	70.57 ($\pm$9.46) m	32.77 ($\pm$4.82) m/s
	90.76 ($\pm$18.02) m	53.06 ($\pm$11.76) m/s
	78.02 ($\pm$11.16) m	40.84 ($\pm$7.23) m/s
Ensemble	68.22 ($\pm$10.53) m	30.40 ($\pm$5.27) m/s
	73.23 ($\pm$14.07) m	40.36 ($\pm$9.50) m/s
	69.97 ($\pm$10.45) m	34.59 ($\pm$6.44) m/s

TABLE II
TEST RESULTS IN SCENARIOS BY SECTOR.

Finally, we also tested the accuracy of the motion classification provided by the Ensemble of KalmanNets in the considered scenario. In detail, Fig. 6 shows the average accuracy over all test trajectories per time step.

Despite the performance in the initial sector of the test trajectory appearing chaotic, the motion classification accuracy reaches satisfactory performance in the rest of the sequence. It is worth noting that the poorer performance in the first sector might be explained by the high similarity of the NCV and CTR motion models for small values of ω, as for the first sector of the analyzed trajectory. Consequently, the motion classification branch of the Ensemble schema misclassified the motion model of the tracked target, predicting a NCV instead of the true CTR. However, it has correctly learned the difference between a CTR motion at a negative turn rate ω and a CTR at a positive turn rate.

V. CONCLUSIONS

In this work, we addressed the challenges posed by tracking a single maneuvering target. The proven ability of GRUs to learn and memorize temporal relationships, along with the robust performance that they have in time series prediction, has led us to formulate a GRU-based architecture to fuse a pool of neural-aided Kalman filters, i.e., the KalmanNet, each one using a different motion model to compensate for the uncertainty in the target dynamics during maneuvering target tracking.

We have tested the performance in tracking a maneuvering target in a simulated set of trajectories, proving that our solution can provide a reliable alternative when a predefined SS model cannot be formulated. It achieves satisfactory tracking performance in the considered scenario compared to fully MB solutions, such as the IMM filter, and that provides fast adaptation to maneuvers.

In future work, it might be helpful to extend the list of motion models that the target might adopt in its trajectories and to validate the ensemble strategy in this case, by focusing on the trade-off between the increased computational requirements and tracking performance. Similarly, it might be helpful to expand the set of scenarios used to train and evaluate the model.

It is crucial to highlight that the presented method does not account for the uncertainty in state predictions at any point in the recursion, which is a fundamental aspect of the KF algorithm. In contrast, KalmanNet has the potential to provide *a posteriori* covariance matrices for state predictions. Exploiting uncertainty prediction is a reasonable approach to explore in order to improve model performance, robustness, and explainability.

ACKNOWLEDGEMENTS

The activity has been funded by a PhD scholarship granted by Leonardo Electronics Division.

REFERENCES

[1] R. E. Kalman, “A new approach to linear filtering and prediction problems,” 1960.

[2] M. Gruber, *An approach to target tracking*. MIT Lincoln Laboratory, 1967.

[3] R. E. Larson, R. M. Dressler, and R. S. Ratner, *Application of the extended kalman filter to ballistic trajectory estimation*. Stanford Research Institute Menlo Park, 1967.

[4] J. D. McLean, S. F. Schmidt, and L. A. McGee, *Optimal filtering and linear prediction applied to a midcourse navigation system for the circumlunar mission*. National Aeronautics and Space Administration, 1962.

[5] H. A. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, 2014.

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[9] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long short-term memory kalman filters: Recurrent neural estimators for pose regularization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5524–5532, 2017.

[10] D. Iter, J. Kuck, P. Zhuang, and C. Learning, "Target tracking with Kalman Filtering, KNN and LSTMs," pp. 1–7, 2016.

[11] C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Information Sciences*, vol. 502, pp. 279–296, 2019.

[12] J. Liu, Z. Wang, and M. Xu, "Deepmtt: A deep learning maneuvering target-tracking algorithm based on bidirectional lstm network," *Information Fusion*, vol. 53, pp. 289–304, 2020.

[13] G. Zhao, Z. Wang, Y. Huang, H. Zhang, and X. Ma, "Transformer-based maneuvering target tracking," *Sensors*, vol. 22, no. 21, p. 8482, 2022.

[14] Y. Zhang, G. Li, X.-P. Zhang, and Y. He, "A deep learning model based on transformer structure for radar tracking of maneuvering targets," *Information Fusion*, vol. 103, p. 102120, 2024.

[15] S. Yan, Y. Liang, and B. Wang, "Multi-level deep learning kalman filter," in *2023 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 1113–1118, IEEE, 2023.

[16] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. Van Sloun, and Y. C. Eldar, "Kalmannet: Neural network aided kalman filtering for partially known dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.

[17] S. Jung, I. Schlangen, and A. Charlish, "A mnemonic kalman filter for non-linear systems with extensive temporal dependencies," *IEEE Signal Processing Letters*, vol. 27, pp. 1005–1009, 2020.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, (San Diega, CA, USA), 2015.