METHODS AND SYSTEMS FOR IDENTIFYING FALSE TRACKS

TECHNICAL FIELD

The present disclosure relates to radar track processing and in particular relates to machine learning methods for the identification of false tracks.

BACKGROUND

The performance of a radar system is defined by its ability to effectively detect and track specific objects while maintaining an acceptable false alarm rate. This balance is crucial for ensuring reliable and accurate situational awareness across various operational contexts.

A minimum detectable speed threshold ensures that the radar is sufficiently sensitive to detect relevant moving targets while limiting false positives caused by clutters. However, it prevents the detection and tracking of slower-moving targets such as small-size drones (e.g., DJI Mavic). On the other hand, reducing this threshold generates more false tracks, making it difficult to identify the real slow-moving targets of interest.

As a result, radar operators may struggle to focus on genuine targets due to difficulties in distinguishing between genuine and erroneous tracks. This can severely impact situational awareness, potentially compromising decision-making and operational effectiveness.

Current radar solutions employ advanced signal and data processing techniques to reduce false alarms. Key methods include CFAR (Constant False Alarm Rate) algorithms, Doppler filtering to separate moving targets, and pulse compression for better range resolution. Additionally, tracking filters such as Kalman filters and Multiple Hypothesis Tracking (MHT) contribute to accurate target tracking and reducing false alarms. Classification algorithms further enhance target recognition.

These integrated techniques result in highly reliable radar systems with reduced false alarms under well-defined conditions.

In an environment with a high density of false alarms, the false alarm rate is compromised, resulting in the generation of more false tracks.

SUMMARY

According to a first aspect of the present disclosure a method of identifying false tracks in a radar system is provided. The method comprises: receiving track data comprising trajectory data measurements corresponding to time points for a potential target object; for a first time window comprising at least three trajectory data measurements, inputting trajectory data measurements corresponding to the first time window into a machine learning model to generate a first prediction indication, wherein the machine learning model is trained to generate prediction indications indicating a probability that a track corresponding to trajectory data measurements is a true track; and generating a track prediction output using the first prediction indication.

The use of a machine learning method capable of identifying false tracks can ensure a reduction of these false alarms in such an environment. It is notable that the method does not actually remove false tracks, rather they indicate to the user who can make the ultimate decision on how to interpret such tracks.

In an embodiment, the machine learning model is a time series model or sequential model such as sequence model, such as a Recurrent Neural Networks (RNN), Transformer or Temporal Convolutional Networks (TCN).

In an embodiment, the method further comprises comparing the track prediction output with a threshold and generating a false track indication if the track prediction output is less than the threshold.

In an embodiment, the method further comprises comparing the track prediction output with a threshold and generating a true track indication if the track prediction output exceeds the threshold.

In an embodiment, the method further comprises, for at least one further time window comprising at least three trajectory data measurements, inputting the trajectory data measurements corresponding to the at least one further time widow into the machine learning model to generate at least one further prediction indication indicating a probability that a track corresponding to the trajectory data measurements is a true track, wherein the track prediction output is generated using the first prediction indication and the second prediction indication.

The track prediction output may be an average of the first prediction indication and the at least one further prediction indication.

According to a second aspect of the present disclosure, a method of training a machine learning model to generate a prediction indication indicating that a track in a radar system is a true track is provided. The method comprises: receiving track data; automatically labelling the track data as false tracks or true tracks; partitioning the labelled track data into time windows; and training the machine learning model using the partitioned labelled track data.

In an embodiment, the method further comprises receiving transponder data corresponding to the track data and using the transponder data to automatically label the track data as false tracks or true tracks based on correlations between the transponder data and the track data.

In an embodiment, using the transponder data to automatically label the track data comprises partitioning a space of the transponder data and track data into cells.

In an embodiment, partitioning the space of the transponder data and track data into cells comprises progressively splitting the space into smaller cells until the number of plots in each cell is less than a threshold number of plots.

In an embodiment, the transponder data is identification friend or foe data.

In an embodiment, partitioning the labelled track data into time windows comprise partitioning the labelled track data into a plurality of overlapping time windows each comprising at least 3 data points.

In an embodiment, the method further comprises selecting a subset of time windows of the partitioned labelled track data according to a target ratio of time windows corresponding to true tracks to time windows corresponding to false tracks.

In an embodiment, the machine learning model is a time series model or sequential model such as sequence model, such as a Recurrent Neural Networks (RNN), Transformer or Temporal Convolutional Networks (TCN).

According to a third aspect of the present disclosure a computer readable medium is provided which stores processor executable instructions. The processor executable instructions, when executed on a processor, cause the processor to carry out a method as set out above.

According to a fourth aspect of the present disclosure a radar data processing system for identifying false tracks is provided. The system comprises: a processor and a data storage device storing computer program instructions operable to cause the processor to: receive track data comprising trajectory data measurements corresponding to time points for a potential target object; for a first time window comprising at least three trajectory data measurements, input trajectory data measurements corresponding to the first time window into a machine learning model to generate a first prediction indication, wherein the machine learning model is trained to generate prediction indications indicating a probability that a track corresponding to trajectory data measurements is a true track; and generate a track prediction output using the first prediction indication.

In an embodiment the machine learning model is a time series model. The time series model may be a recurrent neural network.

In an embodiment, the data storage device further stores computer program instructions operable to cause the processor to compare the track prediction output

with a threshold and generate a false track indication if the track prediction output is less than the threshold.

In an embodiment, the data storage device further stores computer program instructions operable to compare the track prediction output with a threshold and generating a true track indication if the track prediction output exceeds the threshold.

In an embodiment, the data storage device further stores computer program instructions operable to, for at least one further time window comprising at least three trajectory data measurements, input the trajectory data measurements corresponding to the at least one further time widow into the machine learning model to generate at least one further prediction indication indicating a probability that a track corresponding to the trajectory data measurements is a true track, wherein the track prediction output is generated using the first prediction indication and the at least one further prediction indication.

The track prediction output may be an average of the first prediction indication and the at least one further prediction indication.

In an embodiment, the data storage device further stores computer program instructions operable to train the machine learning model to generate a prediction indication indicating that a track in a sensor system is a false track, by: receiving track data; automatically labelling the track data as false tracks or true tracks; partitioning the labelled track data into time windows; and training the machine learning model using the partitioned labelled track data.

In an embodiment, the data storage device further stores computer program instructions operable to receive transponder data corresponding to the track data and use the transponder data to automatically label the track data as false tracks or true tracks based on correlations between the transponder data and the track data.

In an embodiment, the data storage device further stores computer program instructions operable to use the transponder data to automatically label the track data by partitioning a space of the transponder data and track data into cells.

In an embodiment, the data storage device further stores computer program instructions operable to partition the space of the transponder data and track data into cells by progressively splitting the space into smaller cells until the number of plots in each cell is less than a threshold number of plots.

In an embodiment, the transponder data is identification friend or foe data.

In an embodiment, partitioning the labelled track data into time windows comprise partitioning the labelled track data into a plurality of overlapping time windows each comprising at least 3 data points.

In an embodiment, the data storage device further stores computer program instructions operable to select a sub set of time windows of the partitioned labelled track data according to a target ratio of time windows corresponding to true tracks to time windows corresponding to false tracks.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following, embodiments of the present invention will be described as non-limiting examples with reference to the accompanying drawings in which:

FIG.1 shows an example of radar track processing according to an embodiment of the present invention;

FIG.2 is a flow chart showing an overview of radar track processing according to an embodiment of the present invention;

FIG.3 is a block diagram showing a false track identification system according to an embodiment of the present invention;

FIG.4 is a flow chart showing a method of identifying a false track according to an embodiment of the present invention;

FIG.5 is a flow chart showing a method of training of a machine learning model according to an embodiment of the present invention;

FIG.6 is a flow chart showing a method of labelling tracks in an embodiment of the present invention;

FIG.7A shows tracks and Identification Friend or Foe (IFF) plots used to validate the tracks;

FIG.7B shows the space of tracks and IFF plots split into cells;

FIG.7C shows a process of correlating tracks and IFF plots;

FIG.7D shows an example of a recursive algorithm which progressively splits a space;

FIG.8 shows an example of splitting tracks into windows;

FIG.9 shows a comparison of different window sizes;

FIG.10 shows an example of grouping windows by tracks;

FIG.11 is a graph showing recall and precision curves obtained when computing the prediction of a model;

FIG.12 shows an example of use of training data for training the model, setting thresholds and testing the trained model; and

FIG.13 is a set of graphs showing metrics for different window numbers and sizes.

DETAILED DESCRIPTION

FIG.1 shows an example of radar track processing according to an embodiment of the present invention. FIG.1 shows a system in which a radar on a ship 10 progressively

receives (for example, every 1-2 seconds) a measurement of the target 12 (here, an aircraft).

In the present disclosure, the following terminology is used. A plot refers to a target point corresponding to a measurement by a radar. A window is a succession of a fixed number of consecutive plots. Windows may be used during prediction of whether a track is a true track or a false track and also during training of a machine learning model to carry out the prediction. A track refers to the outcome of radar signal and data processing, representing a target trajectory.

In order to make a prediction on whether a track is real track or a false track, it is necessary to wait for three measurements to form a first window14 and make a first prediction 16. Then, with each new measurement, the window is shifted by one step to make an additional prediction. Thus, after receiving six measurements, four successive predictions are made.

As shown in FIG.1, for each window 14, track data such as radar cross section (rcs), speed, altitude, fuzzy logic and range are extracted and used as inputs to a machine learning model. The machine learning model is trained to output an indication whether the track is a false track or a real track.

FIG. 2 is a flow chart showing an overview of radar track processing according to an embodiment of the present invention. As shown in FIG.2, a radar detection chain 20 processes raw radar data which may comprise a set of indications of detections including angles and distances. The detection chain may also provide data on range, velocity and radar cross section. The detections are then formed into plots 22. A tracking algorithm 24 processes the plots 22 to generate tracks 26. Window partitioning 28 divides the tracks 26 into windows 30 which are input into a machine learning model for model prediction 32. The model prediction 32 generates notification indicating a probability that a track is a false track. This information is displayed to an operator who can then make a decision 34 on how to interpret the track.

FIG.3 shows a false track identification system according to an embodiment of the present invention. The false track identification system 100 is a computer system with

memory that stores computer program modules which implement false track identification methods according to embodiments of the present invention. As mentioned above, the false track identification system 100 may be integrated into a radar system, or alternatively may process track data from a radar system.

The false track identification system 100 comprises a processor 110, a working memory 112, a data interface 114, an output interface 116, program storage 120, and data storage 130. The processor 110 may be implemented as one or more central processing unit (CPU) chips. The program storage 120 is a non-volatile storage device such as a hard disk drive which stores computer program modules. The computer program modules are loaded into the working memory 112 for execution by the processor 110. The data interface 114 allows a track data to be received by the false track identification system 100 for processing. The data interface 114 is an interface which allows data to be received from and sent to other computer systems. The data interface 114 may be a wireless network interface such as a Wi-Fi or Bluetooth interface, or alternatively may be a wired interface. The output interface 116 may be implemented as a data interface which allows indications to be output to a radar system for display or alternatively may be implemented as part of the display of a radar system.

The program storage 120 stores a false track prediction module 122, and a training module 124. In the example shown in FIG.3, the false track identification system 100 has program storage which stores both a false track prediction module 122 and a training module 124, however, it will be appreciated that a system for identifying false tracks may be implemented with just a false track prediction module 122 and such a system may use a pre-trained machine learning model to operate. Similarly, a system may also be implemented without the false track prediction module 122 and such a system may be configured to train the machine learning model for use by a separate system. The computer program modules cause the processor 110 to execute various data processing which is described in more detail below. The program storage 120 may be referred to in some contexts as computer readable storage media and/or non-transitory computer readable media. As depicted in FIG.3, the computer program modules are distinct modules which perform respective functions implemented by the false track prediction system 100. It will be appreciated that the boundaries between

these modules are exemplary only, and that alternative embodiments may merge modules or impose an alternative decomposition of functionality of modules. For example, the modules discussed herein may be decomposed into sub-modules to be executed as multiple computer processes, and, optionally, on multiple computers. Moreover, alternative embodiments may combine multiple instances of a particular module or sub-module. It will also be appreciated that, while a software implementation of the computer program modules is described herein, these may alternatively be implemented as one or more hardware modules (such as field-programmable gate array(s) or application-specific integrated circuit(s)) comprising circuitry which implements equivalent functionality to that implemented in software.

The data storage 130 stores a machine learning model 132. The machine learning model 132 is trained to generate prediction indications for radar tracks or parts of radar tracks which indicate a probability that the tracks are false tracks. The machine learning model 132 may be a sequence model, such as a Recurrent Neural Network (RNN), Transformers or a Temporal Convolutional Networks (TCN).

Although the false track prediction system 100 is described with reference to a computer, it should be appreciated that the false track prediction system 100 may be formed by two or more computers in communication with each other that collaborate to perform a task. For example, but not by way of limitation, an application may be partitioned in such a way as to permit concurrent and/or parallel processing of the instructions of the application. Alternatively, the data processed by the application may be partitioned in such a way as to permit concurrent and/or parallel processing of different portions of a data set by the two or more computers. In an embodiment, virtualization software may be employed by the false track prediction system 100 to provide the functionality of a number of servers that is not directly bound to the number of computers in the false track prediction system 100. In an embodiment, the functionality disclosed above may be provided by executing the application and/or applications in a cloud computing environment. Cloud computing may comprise providing computing services via a network connection using dynamically scalable computing resources. A cloud computing environment may be established by an enterprise and/or may be hired on an as-needed basis from a third party provider.

FIG.4 is a flow chart showing a method of identifying a false track according to an embodiment of the present invention. The method 400 shown in FIG.4 is carried out by the false track prediction system 100 shown in FIG.3.

In step 402, the data interface 114 of the false track prediction system 100 receives track data. The track data comprises trajectory data measurements corresponding to time points for a target object.

In step 404, the false track prediction module 122 is executed by the processor 110 of false track prediction system 100 to input the trajectory data measurements corresponding to a time window into the machine learning model 132. The time window comprises at least 3 trajectory data measurements each corresponding to different time points. In response to the trajectory data measurements for the time window, the machine learning model generates a prediction indication corresponding to the time window. The prediction indication indicates a probability that the track corresponding to the 3 trajectory data points is a true track.

As shown in FIG.4, the step 404 may be repeated for successive time windows. For example, the step 404 may be carried out for a first time window to generate a first prediction indication and then for a second time window to generate a second prediction indication. The first time window and the second time window may overlap. For example, if the first time window comprises three trajectory data measurements, the second time window may comprise three trajectory data points including the later two trajectory data points of the first time window.

In step 406, the prediction calculated in step 404 is added to a list of prediction indications.

In step 408, a track prediction output is generated using the prediction indication(s) generated in in the list of prediction indications. For example, the track prediction output may be generated using a single prediction indications, or may be generated using multiple prediction indications. When the track prediction output is generated using multiple prediction indications it may be generated as the average of the

prediction indications, for example the average of the first prediction indication and the second prediction indication mentioned above.

It will be appreciated that the method 400 shown in FIG.4 may be executed in real time or close to real time as track data is received. Therefore, when only a small amount (for example three trajectory data measurements) of data is available, then an initial track prediction output is generated based on that small amount of data. As more data becomes available, then the track prediction output may be re-generated and refined using the additional data.

In the section above, the logic of the grouping is described, which leverages this succession of correlated predictions. However, averaging is not the only solution: more sophisticated formulas can also be considered, which can be added at the end of the chain to provide a cumulative score and a confidence value.

The proposed solution provides an additional filtering solution that enhances the relevance of the overall filtering process in an environment with a high density of false alarms.

This layer can be integrated within a radar system or externally. An external integration offers seamless compatibility with existing solutions, enhancing and optimizing the overall filtering process for improved performance. It also ensures compatibility with legacy radar systems, ensuring wider usability and an extended operational lifespan.

Using a time series machine learning method like a Recurrent Neural Network (RNN) allows for greater robustness, adaptability and precision. A Recurrent Neural Network (RNN) is well suited to time series data and can identify time series patterns.

The solution processes tracks and their associated plot data points. These tracks can be automatically labelled as false or true based on heuristic criteria. This offers a significant advantage, as labelling measurement data within radar processing typically requires manual effort by one or more experts, which is very time-consuming. Automatic labelling offers the potential for complete end-to-end process automation,

encompassing labelling, training, performance evaluation, and model selection. This can ensure faster deployment and upgrading of the solution.

The solution can be integrated within a radar system or externally. An external implementation offers seamless compatibility with existing solutions, enhancing and optimizing the overall filtering process for improved performance. It also ensures compatibility with legacy radar systems, thereby providing wider usability and an extended operational lifespan.

On the other hand, an internal implementation within a radar system provides an all-in-one integrated solution, which is more suitable for a new radar system to be developed.

One important consideration is the dataset for training, validating and selecting the machine learning model. Radar data generally remains the property of the end user, and this data is not necessarily shared with the radar manufacturer. Therefore, to obtain a larger dataset that is adapted to the end user's operations with the radar, it is preferable to consider an external implementation of the solution. The end user can benefit from end-to-end process automation, allowing re-training and testing of the machine learning model with new data whenever required, thus enabling continuous improvement of the solution.

FIG.5 is a flow chart showing a method of training of a machine learning model according to an embodiment of the present invention. The method 500 shown in FIG.5, is used to train the machine learning model 132 stored in the data storage 130 of the false track identification system 100. The method 500 may be implemented by the training module 124 stored in the program storage 120 of the false track identification system 100.

As described above, the present disclosure provides a method for identifying radar false alarms using a machine learning model such as a Recurrent Neural Network (RNN) well-suited to time series data, which is generalizable to different types of radars and adaptable to various contexts. To achieve this, features from a considerable number of labelled tracks, obtained from observations of the same radar,

are collected. The machine learning model is then trained on windows extracted from the tracks, taking into account the temporal factor. Once the model is trained, the software can be easily inserted between the radar output and the display intended for the operator, enabling the operator to filter the tracks shown at their convenience.

As shown in FIG.5, the input to the method 500 is a dataset 502 which comprises radar detections. Recordings are extracted 504 from the dataset 502. Then tracks are labelled 506. The labelled tracks are preprocessed 508 and partitioned into windows 510. The machine learning model is trained 512 on the partitioned windows. Then the performance of the machine learning model is evaluated 514 and if the evaluation is successful, the model 516 is output for use.

ML models trained on this data may potentially be limited by the algorithm. Directly using the labelling algorithm to predict the class of the tracks could also be conceivable. It is therefore important to remember that two types of features are being dealt with: the track features such as number of updates, update rate, length of the track and the plot features such as the radar cross-section (RCS), the speed and the position.

The automatic labelling algorithm uses only track features to label the tracks, as these features are already very indicative of their nature. The ML models, however, use only plot features since the objective is to make a prediction at any given timestamp in the track. As an example, it is not possible in a real-life scenario to know the total length of a track when only a few plots have been received.

It has been observed that the features used for the automatic labelling algorithm and the ML models are fundamentally different. Furthermore, even if desired, the track features could not be used in the ML models. Hence, no label leakage is present in this case.

FIG.6 is a flow chart showing a method of labelling tracks in an embodiment of the present invention. The method 600 shown in FIG.6 corresponds to an example of step 506 shown in FIG.5. At the start 602, the method takes a track as an input. In step 604 the maximum number of consecutive updates is compared with a threshold (the threshold may be, for example 5). If the maximum number of consecutive updates is

less than the threshold, the track is labelled as a false track in step 606. If the maximum number of consecutive updates is greater than the threshold, the method moves to step 608. In step 608, the number of updates is compared with a threshold (which may be, for example 20). If the number of updates is greater than the threshold, then the track is labelled as a real track in step 610. If the number of updates is less than the threshold, then the method moves to step 612 in which the track is labelled as a false track.

The method described above could be applied in several other cases where a notion similar to the track-plot relation exists, often found in the inhomogeneous temporal dataset. Here in FIG.6, length of the tracks and the updates were considered but the same reasoning would work with different track-level features such as the global shape of the track or the percentage of plots correlated by a transponder system such as IFF (Identification Friend or Foe, a transponder based system to obtain positions of an aircraft) plot within the track.

FIG.7A shows tracks and Identification Friend or Foe (IFF) plots used to validate the tracks. As shown in FIG.7A, the radar used includes an IFF component which outputs timestamped IFF plots 710 alongside the regular plots 720. The general idea is to make use of this information to validate the plots by assuming that a real plot is likely to have an IFF plots close by. This case is described as a space and time correlation between both plots. For example, in the upper right corner of the diagram, one track seems to have no IFF plots that correlate. This could be explained either by this specific track being a false alarm; or a real object that turned off its IFF.

FIG.7B shows the space of tracks and IFF plots s split into cells. In practice, to list the plots correlated by the IFF, the space is split into numerous smaller cells as shown in FIG.7B. The cells 730 define the IFF plots s zone 732, which contains all the IFF plots 734 that will be tested. Then the spatial maximal distance is defined for a plot to be spatially correlated by an IFF plot. This value allows the creation of the track plots zone 736: a slightly larger cell that could contain plots correlated.

Once an individual cell is defined, each IFF plot is looped over to look for track plots inside its ellipsoid. Not only does the plot have to be close enough in terms of space,

but also in terms of time. Let us define $distance_{space}$ and $distance_{time}$ the space and time maximal distances for a plot to be correlated. Then, a plot p defined by (p_x, p_y, p_t) is said to be correlated by an IFF plot i (i_x, i_y, i_t) if and only if:

$$\left(\frac{i_x - p_x}{distance_{space}}\right)^2 + \left(\frac{i_y - p_y}{distance_{space}}\right)^2 + \left(\frac{i_t - p_t}{distance_{time}}\right)^2 \leq 1$$

In the case where one IFF plot correlates multiple plots, we need to check if they all belong to the same track before defining them as truly correlated. This is illustrated in FIG.7C. FIG.7C shows a process of correlating tracks and IFF plots.

The objective of the cell division is to reduce the time complexity of the correlation search. Indeed, one recording contains dozens of thousands of plots for both IFF and regular plots. A direct approach would take hours. However, partitioning the space into homogeneous cells as shown above might not be the optimal solution as the points are not homogeneously distributed.

To address this issue, a recursive algorithm may be introduced that progressively splits the space until every cells has less than a given fixed number of plots inside.

FIG.7D shows an example of a recursive algorithm which progressively splits a space. FIG.7D shows five iterations of this algorithm. At each step, the cells that contain too many plots are identified (here highlighted in grey) and further split.

The result of the process is a list of the correlated plots that can be grouped by tracks to give a correlation percentage. This value, with the track length and track update rate can then be used to create an improved automatic labelling algorithm. Any other feature at the track-level can also be included depending on what is available in the radar. For example, if the radar does not have IFF plots or a notion of update but another track feature called $feature_1$, the automatic labelling algorithm could use $feature_1$ and the track length only.

The proposed solution relies on treating the tracks as time-series data. This approach takes into account not only the absolute values of features but also how these values evolve over time. These criteria allow the model to establish classification rules.

FIG.8 shows an example of splitting tracks into windows. In the example shown in FIG.7, the initial dataset includes 25 rows of data. The window size is 7 and the overlap between windows is 2. As shown in FIG.8, four window are extracted.

More generally, all tracks are split into windows containing a fixed number of consecutive measured plots, with a relatively constant time interval. This number, called the window size, must be the same for false track samples, real track samples, and test data. The values tested for this parameter range from 3 to 12. However, different values can be tested depending on the need. In fact, a smaller window size will allow for earlier predictions, as fewer plots are expected.

FIG.9 shows a comparison of different window sizes. For each track, a certain number n (here 1, 2, 3 or 4) of windows are taken into account. The first n windows give the n predictions and the average is outputted to get the performance on each track. It is pretty clear that increasing the window size can improve the performance on some specific metric (here we chose the recall).

For the training data, an overlap for the windows may be chosen. Overlap is defined here as the number of plots in a window that will also be shared with the next window. An overlap of zero means each plot belongs to only one window, while an overlap of one causes the windows to slightly overlap. This criterion serves two purposes: first, it creates a form of data augmentation since we obtain more samples as input to the model.

Second, the overlap introduces a form of diversity by exposing the model to multiple variations of the same underlying data, with slightly different start and end points for each window. This leads to a better understanding of how the model should classify borderline cases.

The dataset is based on radar tracks recordings with the default value of the minimum detectable velocity threshold (some of which include a high density of alarms due to environmental factors), as well as with lower values.

Once the window format and their overlap are chosen, the next step is to select the proportion of false alarms relative to the real track samples. In reality, the radar detects way less false alarm plots than plots corresponding to real objects. Therefore, it is essential to find a balance between a total class imbalance (unmodified proportion), which may hinder the model's ability to learn how to recognize false alarms, and a perfect balance (the same number of false and real samples), which could negatively affect the model's ability to generalize across different types of real targets.

This class proportion criteria is called the ratio. The proposed solution offers significant flexibility, allowing for easy training of multiple models by simply changing the ratio, and then comparing the performance of the models based on well-chosen metrics. Below is a table describing the general change in performance when increasing the dataset parameters for each algorithm. For example, as seen previously, the window size of the dataset samples is positively correlated with the model performance.

	window size				**overlap**				**ratio**			
session	DT	XGB	RF	RNN	DT	XGB	RF	RNN	DT	XGB	RF	RNN
w4o2r3	**+**					**+**	**+**	**+**		**—**	**—**	
w7o2r2	**+**		**+**	**+**				**+**		**—**	**—**	**—**
w10o1r3	**+**	**+**	**+**	**+**		**—**				**—**		

In the above table, DT stands for Decision Tree, XGB for Extreme Gradient Boosting, RF for Random Forest and RNN for Recurrent Neural Network.

During the testing, ratio values ranging from 1 to 5 were used. In the raw dataset, the value tends to be closer to 20 but this would make the training data much bigger in terms of size complexity. In practice, it was observed that a ratio of 2 is often enough to achieve good generalization. Of course, the performance evaluation is always done using a realistic ratio, materialized by the ratio value of -1 (which means raw dataset ratio) to avoid any bias.

The features selected for training the models play a crucial role. Naturally, when training a model with a set of *n* features, it is essential to ensure that these exact *n* features can be found for all the samples that will be tested later. However, tests have shown that some features are more valuable than others for making predictions. In our case, we use several dozen features, but only about ten are truly necessary. These include, among others, geographic features (Local Geographical Coordinates: position relative to the ship, speed, and their covariance), and quality features (track quality, detection quality, RCS). This means that the solution would be well-suited for other radar systems with different fields of application. In the table below, the features are listed by order of importance for the RNN models.

Name	**Description**
pv_cov	Magnitude of the covariance between position and velocity in all three directions.
v_cov	Magnitude of the covariance between the two by two velocities in all three directions.
p_cov	Magnitude of the covariance between the two by two positions in all three directions.
v_std	Magnitude of the velocity standard deviation in all three directions.
arm_alert	Boolean notifying an ARM (Anti Radar Missile) alert.
velo_decrease	Boolean notifying if the velocity is rapidly decreasing.
rcs	Radar Cross Section of the target.
height	Altitude of the target relatively to the ground.
range	Distance of the target to the radar.
helicopter	Boolean notifying if the target is possibly a helicopter.
atc	Automatic Track Labelling features, also called fuzzy logic: contains 32 features.
track_strength	Track strength type.
p_std	Magnitude of the position standard deviation in all three directions.
priority	Track priority enumerated type.

To better describe the construction of the datasets, the nature of the algorithm that was used for an example implementation: RNN (Recurrent Neural Network) is considered.

RNNs are machine learning models designed to handle sequential data. Unlike traditional neural networks, RNNs have internal connections that allow them to retain a “memory” of previous inputs in the sequence. This enables them to capture temporal dependencies and detect patterns in the data, which is particularly useful for tasks such as time series prediction, such as a sequence of radar track plots. At each time step, an RNN processes the current input while considering the hidden state from the previous step, updating this state to pass it to the next step. This recurrent structure allows RNNs to handle sequences of variable lengths. Therefore, this class of algorithms can accept samples in two-dimensional form, like the windows described above.

FIG.10 shows an example of grouping windows by tracks. The method involves taking the first n windows of each track, calculating the prediction for each window using the model, and then averaging these predictions. This results in a value that is much more representative of the track, without needing to know all the individual plots that make up the track.

FIG.11 is a graph showing recall and precision curves obtained when computing the prediction of a model. The objective is to place the threshold in such a way as to maximize the performance. For this, the worst performance is minimized by setting the threshold so that sensitivity and specificity are roughly equal. To maximize the proportion of correct answers, the maximum of the minima of the two curves are chosen. Here, this optimum is achieved with a threshold of 43%, offering an 84% performance for both classes.

FIG.12 shows an example of use of training data for training the model, setting thresholds and testing the trained model. It was found that using 50 to 200 recordings for each model was a good compromise between dataset representability and size complexity. Another way to put it is considering the samples instead: generally speaking, datasets with 10 000 samples for false alarms gave satisfying results. Of

course, in the end product, better use all the recordings to achieve the best performance and generalization possible. In our example, the model is trained using 100 recordings. The model is then tested on 400 recordings under real-world conditions (using the grouping logic described above). Half of the samples are used to adjust the threshold and the rest is used to evaluate the improvement in terms of accuracy.

FIG.13 is a set of graphs showing metrics for different window numbers and sizes. IN FIG.13, the four metrics (threshold, expectation, recall and precision) are represented for window sizes ranging from 3 to 6 and various numbers of first windows from 1 (so only the first window of each track) to 20 (meaning virtually the entire track). On top of each cell is the optimal threshold, here ranging from 30% to 60% depending on the model. This shows how important the optimization of the threshold is, because it varies a lot from a model to the other.

The proposed solution provides an additional filtering solution that enhances the relevance of the overall filtering process in an environment with a high density of false alarms. This layer can be integrated within a radar system or externally. An external integration offers seamless compatibility with existing solutions, enhancing and optimizing the overall filtering process for improved performance. It also ensures compatibility with legacy radar systems, ensuring wider usability and an extended operational lifespan.

Finally, the solution provides track assessment probabilities (real track versus false track) to aid the operator in decision-making. No decisions are made by the supervised learning algorithm and no tracks are suppressed, ensuring that the responsibility for actions remains with the operator. This is particularly important in the context of defense radar systems, where the reliability and understandability of the decisions made by a machine learning algorithm are crucial.

Whilst the foregoing description has described exemplary embodiments, it will be understood by those skilled in the art that many variations of the embodiments can be made within the scope and spirit of the present invention.

CLAIMS

1. A method of identifying false tracks in a radar system, the method comprising:

receiving track data comprising trajectory data measurements corresponding to time points for a potential target object;

for a first time window comprising at least three trajectory data measurements, inputting trajectory data measurements corresponding to the first time window into a machine learning model to generate a first prediction indication, wherein the machine learning model is trained to generate prediction indications indicating a probability that a track corresponding to trajectory data measurements is a true track; and

generating a track prediction output using the first prediction indication.

2. The method according to claim 1, wherein the machine learning model is a time series model.

3. The method according to claim 2, wherein the time series model is a recurrent neural network.

4. The method according to any preceding claim, further comprising comparing the track prediction output with a threshold and generating a false track indication if the track prediction output is less than the threshold.

5. The method according to any one of claims 1 to 3, further comprising comparing the track prediction output with a threshold and generating a true track indication if the track prediction output exceeds the threshold.

6. The method according to any preceding claim, further comprising, for at least one further time window comprising at least three trajectory data measurements, inputting the trajectory data measurements corresponding to the at least one further time window into the machine learning to generate at least one further prediction indication, wherein the track prediction output is generated using the first prediction indication and the at least one further prediction indication.

7. The method according to claim 6, wherein the track prediction output is an average of the first prediction indication and the at least one further prediction indication.

8. A method of training a machine learning model to generate a prediction indication indicating that a track in a radar system is a true track, the method comprising:

receiving track data;

automatically labelling the track data as false tracks or true tracks;

partitioning the labelled track data into time windows; and

training the machine learning model using the partitioned labelled track data.

9. The method according to claim 8, further comprising receiving transponder data corresponding to the track data and using the transponder data to automatically label the track data as false tracks or true tracks based on correlations between the transponder data and the track data.

10. The method according to claim 9, wherein using the transponder data to automatically label the track data comprises partitioning a space of the transponder data and track data into cells.

11. The method according to claim 10, wherein partitioning the space of the transponder data and track data into cells comprises progressively splitting the space into smaller cells until the number of plots in each cell is less than a threshold number of plots.

12. The method according to any one of claims 9 to 11, wherein the transponder data is identification friend or foe data.

13. The method according to any one of claims 9 to 12, wherein partitioning the labelled track data into time windows comprise partitioning the labelled track data into a plurality of overlapping time windows each comprising at least 3 data plots.

14. The method according to any one of claims 9 to 13, further comprising selecting a sub set of time windows of the partitioned labelled track data according to a target ratio

of time windows corresponding to true tracks to time windows corresponding to false tracks.

15. The method according to any one of claims 8 to 14, wherein the machine learning model is a time series model such as a recurrent neural network.

16. A computer readable medium storing processor executable instructions which when executed on a processor cause the processor to carry out a method according to any one of claims 1 to 15.

17. A radar data processing system for identifying false tracks, the system comprising: a processor and a data storage device storing computer program instructions operable to cause the processor to:

receive track data comprising trajectory data measurements corresponding to time points for a potential target object;

for a first time window comprising at least three trajectory data measurements, input trajectory data measurements corresponding to the first time window into a machine learning model to generate a first prediction indication, wherein the machine learning model is trained to generate prediction indications indicating a probability that a track corresponding to trajectory data measurements is a true track; and

generate a track prediction output using the first prediction indication.

18. The radar data processing system according to claim 17, wherein the machine learning model is a time series model.

19. The radar data processing system according to claim 18, wherein the time series model is a recurrent neural network.

20. The radar data processing system according to any one of claims 17 to 19, wherein the data storage device further stores computer program instructions operable to cause the processor to compare the track prediction output with a threshold and generate a false track indication if the track prediction output is less than the threshold.

21. The radar data processing system according to any one of claims 17 to 19, wherein the data storage device further stores computer program instructions operable to compare the track prediction output with a threshold and generating a true track indication if the track prediction output exceeds the threshold.

22. The radar data processing system according to any one of claims 17 to 21, wherein the data storage device further stores computer program instructions operable to, for at least one further time window comprising at least three trajectory data measurements, input the trajectory data measurements corresponding to the at least one further time window into the machine learning model to generate at least one further prediction indication, wherein the track prediction output is generated using the first prediction indication and the at least one further prediction indication.

23. The radar data processing system according to claim 22, wherein the track prediction output is an average of the first prediction indication and the at least one further indication.

24. The radar data processing system according to any one of claims 17 to 23, wherein the data storage device further stores computer program instructions operable to train the machine learning model to generate a prediction indication indicating that a track in a sensor system is a false track, by: receiving track data; automatically labelling the track data as false tracks or true tracks; partitioning the labelled track data into time windows; and training the machine learning model using the partitioned labelled track data.

25. The radar data processing system according to claim 24, wherein the data storage device further stores computer program instructions operable to receive transponder data corresponding to the track data and use the transponder data to automatically label the track data as false tracks or true tracks based on correlations between the transponder data and the track data.

26. The radar data processing system according to claim 25, wherein the data storage device further stores computer program instructions operable to use the transponder

data to automatically label the track data by partitioning a space of the transponder data and track data into cells.

27. The radar data processing system according to claim 27, wherein the data storage device further stores computer program instructions operable to partition the space of the transponder data and track data into cells by progressively splitting the space into smaller cells until the number of plots in each cell is less than a threshold number of plots.

28. The radar data processing system according to any one of claims 25 to 27, wherein the transponder data is identification friend or foe data.

29. The radar data processing system according to any one of clams 25 to 28, wherein partitioning the labelled track data into time windows comprise partitioning the labelled track data into a plurality of overlapping time windows each comprising at least 3 data points.

30. The radar data processing system according to any one of clams 25 to 29, wherein the data storage device further stores computer program instructions operable to select a sub set of time windows of the partitioned labelled track data according to a target ratio of time windows corresponding to true tracks to time windows corresponding to false tracks.