Data-Driven Object Tracking: Integrating Modular Neural Networks into a Kalman Framework

Christian Alexander Holz, Christian Bader, Markus Enzweiler, Matthias Drüppel

***Abstract*—This paper presents novel Machine Learning (ML) methodologies for Multi-Object Tracking (MOT), specifically designed to meet the increasing complexity and precision demands of Advanced Driver Assistance Systems (ADAS). We introduce three Neural Network (NN) models that address key challenges in MOT: (i) the Single-Prediction Network (SPENT) for trajectory prediction, (ii) the Single-Association Network (SANT) for mapping individual Sensor Object (SO) to existing tracks, and (iii) the Multi-Association Network (MANTa) for associating multiple SOs to multiple tracks. These models are seamlessly integrated into a traditional Kalman Filter (KF) framework, maintaining the system's modularity by replacing relevant components without disrupting the overall architecture. Importantly, all three networks are designed to be run in a real-time, embedded environment. Each network contains less than 50k trainable parameters.**

Our evaluation, conducted on the public KITTI tracking dataset, demonstrates significant improvements in tracking performance. SPENT reduces the Root Mean Square Error (RMSE) by 50% compared to a standard KF, while SANT and MANTa achieve up to 95% accuracy in sensor object-to-track assignments. These results underscore the effectiveness of incorporating task-specific NNs into traditional tracking systems, boosting performance and robustness while preserving modularity, maintainability, and interpretability.

I. Introduction

THE ongoing evolution of ADAS has brought the need for precise and reliable MOT into the spotlight [1]–[11]. In complex and dynamic environments, as encountered in urban traffic, it is crucial to accurately capture and predict the positions of multiple objects already in the early timestamps of detection – a key challenge in assisted and automated driving. In the commonly used Tracking-by-Detection (TbD) paradigm, a tracker fuses detected SO to create consistent object tracks over time. A crucial step within this paradigm is the association of the incoming measured SO with their corresponding existing object tracks to update their properties. If no association can be made, new object tracks must be initialized [1]–[3], [12], [13]. Tracking frameworks form the heart of ADAS that are used in millions of vehicles around the globe. The vast majority of these frameworks rely on classical approaches such as the KF or its variants [1]–[3]. These classical tracking theories have the great benefit of being modular and interpretable. The task is split into clearly separated subtasks such as the prediction of currently tracked objects and the association with newly measured ones.

However, development for automated driving is highly complex [14], [15]. In the automotive industry, tracking systems are typically developed through a software platform that is designed to support a range of vehicle models, each of which will have varying sensor placements, system configurations, or even entirely different sensor suites. These systems must perform reliably across a wide spectrum of driving scenarios, which can introduce performance challenges in specific situations. Traditional solutions often rely on heuristics and manual parameter tuning, making the software cumbersome to maintain and difficult to extend. Moreover, these hand-engineered methods lack automated optimization, resulting in suboptimal performance in complex driving conditions. To address these limitations, we propose a data-driven tracking framework that allows for fine-tuning for specific configurations, thereby improving both maintainability and adaptability.

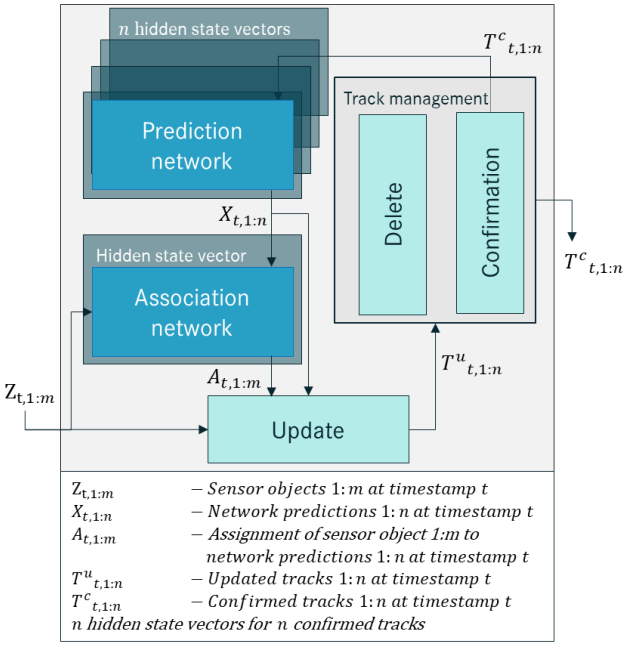

Fig. 1: This schematic representation shows the integration of two NNs (highlighted in dark blue) within a TbD framework. The "Association network" can be implemented using either SANT or MANTa. It works in tandem with the prediction network SPENT, which takes tracked objects $T^c_{t,1:n}$ and predicts them to the next timestamp as $X_{t,1:n}$. The predicted objects are then associated with sensor observations $Z_{t,1:m}$ by SANT or MANTa.

C. Holz is with Daimler Truck AG, Research and Advanced Development, Stuttgart, Germany

C. Bader is with Daimler Truck AG, Research and Advanced Development, Stuttgart, Germany

M. Enzweiler is with the Institute for Intelligent Systems, University of Applied Sciences, Esslingen, Germany

M. Drüppel is with the Center for Artificial Intelligence, Baden-Württemberg Cooperative State University (DHBW), Stuttgart, Germany

II. Contributions and overview

Our primary contribution is the development, single evaluation and joint integration of three novel NN that we label:

(i) SPENT (Single-Prediction Network) which predicts the states of tracked objects.
(ii) SANT (Single-Association-Network) which associates one incoming sensor object to all currently tracked objects.
(iii) MANTa (Multi-Association Network) which associates multiple incoming sensor objects to all currently tracked objects.

These networks were specifically designed for real-time, embedded inference. Each of them has fewer than 50k trainable parameters. Fig. 1 provides an overview of our approach in which the association network can be implemented using either SANT or MANTa. The input for the proposed prediction network (i) is up to m SOs $Z_{t,1:m}$ at timestamp t. If new objects are detected by the sensors, these are stored in a k-dimensional state vector containing information such as object position (x, y), yaw angle and object dimensions (length and width) (for $k = 5$). SPENT predicts all currently tracked objects $X_{t,1:n}$ (up to n) to the next timestamp where they are used as input to either the SANT or MANTa association networks. These provide the association matrix $A_{t,1:m}$, that is used to update the tracks ${T^u}_{t,1:n}$ using the corresponding sensor objects or create new tracked objects. The Track Management can then decide to delete tracks, that were not updated for a specific amount of time and send out tracks ${T^c}_{t,1:n}$ to the next higher software component that have been confirmed by sensor objects. (i) In contrast to the KF [3], our proposed prediction network is capable of predicting the state of individual objects without the need for a predefined heuristic prediction model. The self-learning, data-driven approach enables adaptability to various scenarios and the ability to effectively handle non-linearities and habits of road users. Many conventional tracking systems rely on static methods for data association. Commonly used algorithms like the Hungarian Algorithm (HA) [16] require heuristics and fixed thresholds. (ii) Our proposed SANT and (iii) MANTa replace the calculation of a distance metric for the corresponding assignment by employing ML in order to resolve situations unclear for traditional approaches. Both our prediction network and our association networks can be developed and evaluated as stand-alone models. For a throughout evaluation, we proceed by integrating them into an existing tracking system and demonstrate their performance through multiple tests and comparisons with established methods. This work provides new insights and advancement in the development of ADAS tracking systems by applying ML.

III. Related work

A. State prediction for tracked objects

One fundamental problem in TbD frameworks is the prediction of the states of the already tracked objects. In many approaches Kalman filters and their variants have proven to be effective for state prediction [1], [2], [17]. However, they reach their limits in more complex scenarios, particularly in the presence of non-linear motion patterns and interactions among multiple objects [18]–[20]. Ristic et al. [18] highlight the limitations of Kalman Filters in handling nonlinear and non-Gaussian cases, introducing Particle Filters as a potential alternative. Julier et al. [19] extend the standard Kalman Filter with the Unscented Kalman Filter (UKF) to better address nonlinear motion models. Wan et al. [20] propose the use of Gaussian Mixture Models for tracking multiple objects in cluttered environments. In this work, we introduce a novel MOT approach that leverages ML to overcome these challenges. We specifically focus on the development and implementation of NNs, which can enable more precise and flexible data-driven object state predictions.

B. Association of sensor objects to tracks

Another key challenge for TbD trackers is data association. The widely used Global Nearest Neighbor (GNN) algorithm, often implemented via the HA [16], assigns detections to tracks by minimizing a distance metric. However, it only considers current observations, ignoring temporal continuity and motion patterns, which can lead to errors in complex scenarios such as crossing objects or noisy sensor data. While methods like the Joint Probabilistic Data Association (JPDA) [21] evaluate the likelihood of all possible assignments, they are computationally expensive.

ML-based approaches have been proposed to address these limitations. The temporal information in tracks can, for example, be leveraged through the attention mechanisms as used in [8], [22] or Recurrent Neural Networks (RNNs) as developed in [4], [5]. The latter is what we are also pursuing in this work. Similar to the problem statement by Mertz et al. [5], the aim of our work is to develop a data-based approach that can learn to completely solve the combinatorial non-deterministic polynomial time (NP) hard optimization problem of data association. In the context of this work, we put forward the hypothesis that a Gated Recurrent Unit (GRU)-based association network can be designed and trained without forcing a concrete distance metric, as discussed in the next chapter.

C. Real time applications

Tracking systems for ADAS run on embedded devices in real-time. They must provide immediate state prediction of objects directly after their first detection [1], [23], making offline tracking methods like [24] unsuitable. Consequently, modern multi-object tracking approaches rely on online methods, which do not have access to future sensor data. These methods estimate object-track similarity based on predicted positions or object features like appearance.

a) Kalman Filter based tracker: Our ML models are integrated into a KF-based tracking system, widely used for its robust performance and interpretability [1]–[3], [17]–[20]. Bewley et al. [1] introduced an efficient multi-object tracking method combining a KF with the Hungarian Algorithm. Seidenschwarz et al. [2] proposed a simpler approach based on visual cues like color, shape, and motion for object tracking and frame-to-frame association, avoiding the complexity of many modern trackers.

b) Recurrent Neural Network based Tracker: Similar to our methodologies, RNN-based approaches for online multi-target tracking have been introduced in [4], [5], [11], [25]. Specifically, the work by Mertz et al. [5] focuses on data association within a TbD framework. Their proposed DeepDA model, a Long Short-Term Memory (LSTM)-based Deep Data Association (DeepDA) Network, is designed to learn and execute the task of associating objects across frames. This model's ability to discern association patterns directly from data enables a robust and reliable tracking outcome, even in environments with significant disturbances. Mertz et al. employ a distance matrix, derived from the euclidean distance measure, as the input for the DeepDA network. This innovative approach effectively supersedes traditional association algorithms, such as the Hungarian Algorithm. It is inferred that the euclidean distance measure served as a foundation not only for generating the ground truth (GT) training data (i.e., distance matrices) but also for the subsequent evaluation process. However, this is not explicitly stated. In our work, we want to enable the network to follow a completely data-driven association logic without forcing a concrete distance metric.

c) Attention Mechanism based Tracker: In this paper, we analyze tracks using Long Short-Term Memory (LSTM)-based models. An alternative architecture would be the attention mechanism [26], utilized in various studies [6]–[8], [22], [27], [28]. Hung et al. [8] focus on soft data association (SoDA), enabling probabilistic associations and accounting for uncertainties by aggregating information from all detections within a temporal window. This approach allows the model to learn long-term, interactive relationships from large datasets without complex heuristics. However, since tracking tasks involve real-time data processing with relatively short sequences, RNNs can efficiently handle this without the overhead of calculating attention weights for every input, making them more computationally efficient for short to medium-length sequences.

IV. Tracking with ML-based Prediction and Association Networks

We apply the TbD paradigm, in which a tracker fuses object detections to generate object tracks that are consistent over time. In our study a KF framework was implemented following the computational ideas of Vo et al. [17]. To enable our models to incorporate temporal information, we use LSTM [29] and BiLSTM network layers [30], both for the prediction and the association of the sensor objects to the existing tracks.

A. Single Prediction Network (SPENT)

Our approach uses the hidden states of the LSTM layer as an information repository for each object. SPENT operates in an open-loop manner, predicting future states based on past data. This allows the network to predict the most likely state of an object for the next timestamp.

a) Data preprocessing: In the development of our model, Ground Truth (GT) data comprising vehicle tracks (cars and vans) from the KITTI dataset [31] was utilized. We extracted 635 tracks, filtering out shorter tracks using a 3-frame threshold, resulting in 624 tracks. This ensures the network receives tracks with sufficient timestamps, a common practice in tracking systems [32]. The track lengths vary from a minimum of 4 frames to a maximum of 643 frames (Sequence 20, ID 12).

To enhance model generalization and foster convergence during training, we normalized the state values of tracks at time t (predictors) and at time $t+1$ (targets) in accordance with the methodology outlined in [33]. This normalization process standardizes the distribution of both predictors and targets to have a mean of zero and a unit variance. The mean values $\vec{\mu}$ and standard deviations $\vec{\sigma}$ for each state in the state vectors $\vec{Z}_t$ were computed across all tracks. For this, all tracks were joined together, leading to one pseudo-track with a total number of timestamps N:

$$\vec{\mu} = \frac{1}{N}\sum_{t=1}^{N}\vec{Z}_t \quad \text{and} \quad \vec{\sigma} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(\vec{Z}_t - \vec{\mu})^{2^*}}, \qquad (1)$$

where the square $(...)^{2^*}$ and the square-root must be applied element wise.

In our approach, we apply pre-padding as described by Reddy et al. [34], who examined the impact of padding strategies on sequence-based NNs. They highlight that while both pre-padding and post-padding are feasible, the choice significantly affects performance, especially for LSTM-based networks, where maintaining sequence context is critical. To

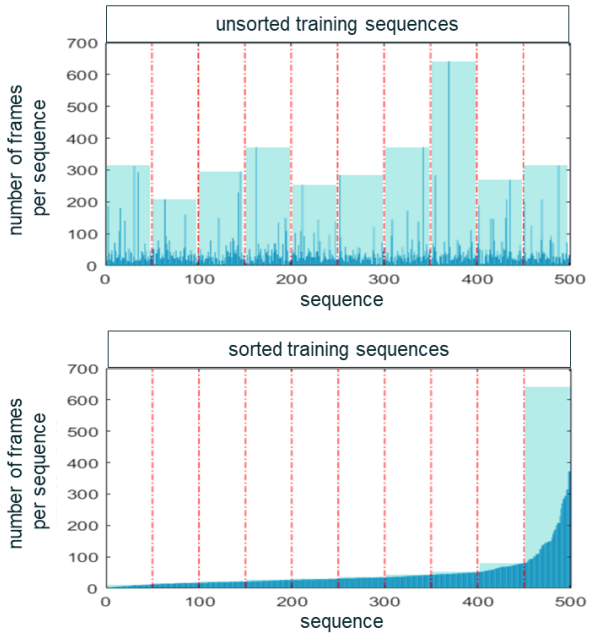

Fig. 2: Analysis of sequence padding: unsorted vs. sorted data. This figure illustrates the impact of sequence padding on LSTM training based on the sorting of input data. The upper panel shows that unsorted data requires extensive padding to equalize batch sequence lengths, increasing computational overhead. In contrast, the lower panel demonstrates that sorting data by length before batching significantly reduces the necessary padding.

manage varying sequence lengths, we pad shorter sequences with tokens, ensuring all sequences in a batch have uniform length.

We used zeros as padding tokens, added to the end of sequences as needed, allowing efficient batch processing and consistent training without affecting model performance due to varying sequence lengths. As noted by Reddy et al. [34], while padding introduces noise, it is crucial for aligning sequences in mini-batches for LSTM training. To reduce this noise, we sorted the training dataset by sequence length before applying mini-batch padding. This method, illustrated in Fig. 2, significantly minimizes the padding (shown in turquoise) required for each mini-batch, which was essential for achieving convergence during training.

b) Network architecture: As depicted in Fig. 3, the schematic illustration of the generic structure of the prediction network illustrates how the architecture is adeptly designed to address the challenges of real-time state prediction. This representation highlights the strategic deployment of the LSTM layer for storing and processing object-specific information, facilitating accurate and timely predictions of object states.

The foundational layer of our SPENT is an LSTM layer, where hidden states are dynamically updated at each timestamp based on incoming measurement data. This mechanism allows continuous correction throughout each track's sequence, enhancing predictive accuracy. The number of hidden units correlates with the amount of information retained over time, as shown in Fig. 3. These hidden states encapsulate information from all preceding timestamps, ensuring comprehensive temporal understanding. Following the LSTM layer, we incorporate a Batch-Normalization layer which accelerates training and promotes convergence by mitigating internal covariate shift [35]. Next is a Rectified Linear Unit (ReLU) activation layer [36], which applies a non-linear threshold operation, setting values below zero to zero. During training, a Dropout layer randomly nullifies input elements with a specified probability, regularizing the model and preventing overfitting [37]. The architecture concludes with a Fully Connected (FC) layer, which integrates insights from previous layers, with its dimensionality aligned to the number of required output variables [38].

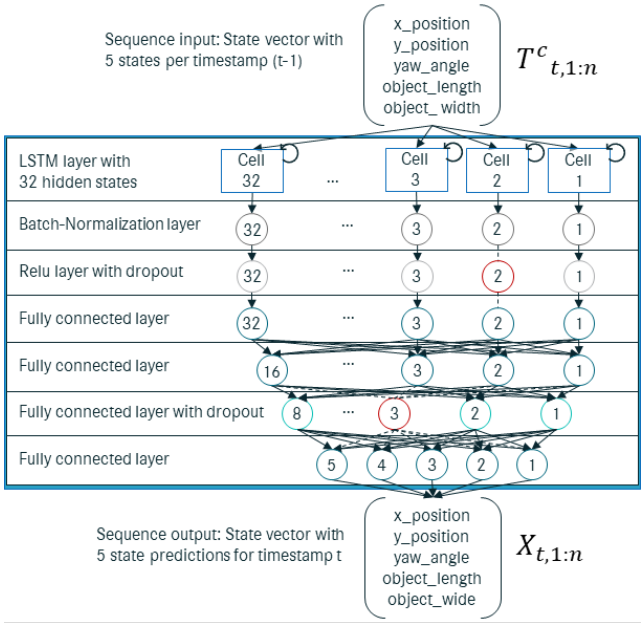

Fig. 3: Schematic representation of the generic structure of SPENT.

Our model's loss function is based on the Mean Squared Error (MSE) metric, which is calculated for each state value prediction. The MSE quantifies the average squared discrepancy between the predicted and actual target values. We chose MSE because it provides a clear and direct measure of how closely our predictions align with the true states, making it an effective metric for optimizing our model's state predictions.

For one single prediction the Mean Squared Error (MSE) is given by

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(Z_i - X_i)^2, \tag{2}$$

where k is the length of the predicted state vector (here $k = 5$: x, y, $\dot{x}$, $\dot{y}$, yaw angle), Z_i are the entries of the ground truth state vector (KITTI cars and vans tracks) and X_i the respective entries of the predicted state vector from our network. During training, the cost function is evaluated for one mini-batch with several sequences and a total number of N timestamps. It is calculated as half the mean-square-error of the predictions added up for each timestamp, normalized over all timestamps. The factor of $\frac{1}{2}$ simplifies the gradient during backpropagation:

$$cost = -\frac{1}{2}\frac{1}{N}\sum_{t=1}^{N}\frac{1}{k}\sum_{i=1}^{k}(Z_{t,i} - X_{t,i})^2, \tag{3}$$

where $Z_{t,i}$ refers to the i-th entry in the state vector at timestamp t.

B. Single Association Network (SANT)

Our association network uses a data-driven approach to solve the NP-hard data association problem [16], [21], which traditionally requires significant computational effort for optimal solutions [16]. Unlike conventional methods [4], [5], our SANT model eliminates the need for a predefined distance matrix. Instead, it directly processes the current tracks and newly detected sensor objects, matching each new object to a track. This allows the network to autonomously learn its association strategy from training data, replacing the Hungarian Algorithm and a defined distance metric with a learning-based method.

a) Data preprocessing: In the formulation of SANT, we conceptualized data association as a temporally structured challenge, adopting a sequence-to-vector paradigm. In general, m incoming sensor objects need to be associated with n tracks. For SANT we focus on the association of a singular sensor object ($m = 1$), denoted as $Z_{(t,m=1)}$, to n tracks, represented as $X_{(t,1:n)}$. These tracks were extracted from the KITTI dataset. We note, however, that genuine hand-labelled GT data for this specific association problem is not available. We rather use the existing tracks to generate synthetical GT data for the data association. The input to SANT consists of a single sensor object and a set of tracks (see top part of Fig. 4). The output is a one-hot vector encoding the association of the

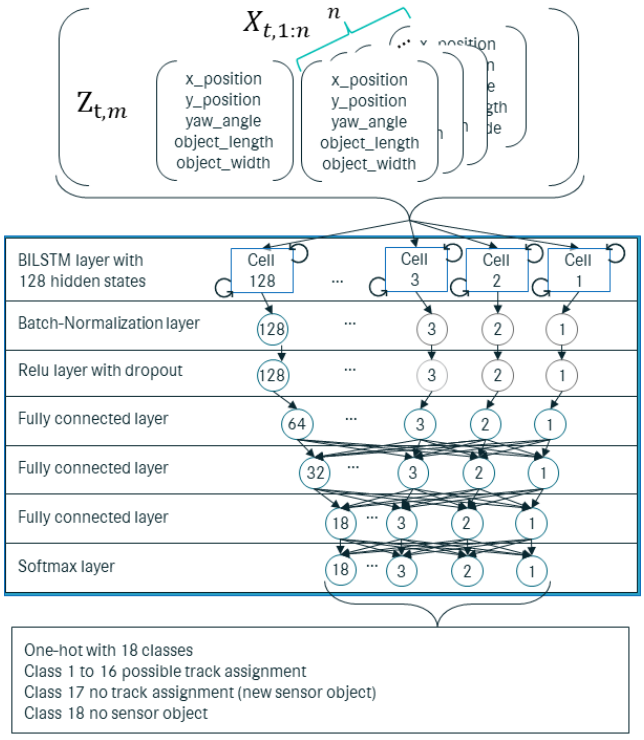

Fig. 4: Schematic representation of the network structure of SANT. Here $m = 1$, so one SO is associated to m existing tracks.

incoming SO to one of the tracks, or to none. To generate the synthetic GT, we take a set of tracks, then randomly choose one of them and take the next state vector at the next timestamp of that single track as the incoming new SO. The association result can be extracted from the corresponding track that it was taken out of. To simulate more realistic sensor data, artificial noise was introduced. This process was designed to reflect the inherent inaccuracies and uncertainties present in real-world sensor measurements. To achieve this, we add noise to the state vector of the incoming SOs with a maximum of 3%. As shown in Fig. 4 the data format was created accordingly to enable index-based track assignment for SANT. The actual number of tracks can vary between 0 and a maximum of $n = 16$ objects, as is given by the KITTI dataset using our selected objects (cars and vans) [31]. The size of the input matrix therefore corresponds to $k \times (n+1)$, where $k = 5$ is the number of state values for our work. The columns of the matrix contains the state vectors of the n tracks and the state vector of the incoming sensor object. A deeper analysis of our data is presented in the MANTa section.

b) Network architecture: The network as depicted in Fig. 4 is designed as a sequence-to-classification network. At each timestamp, a matrix is passed as input holding both the information of the one to-be-assigned SO and the multiple currently-tracked objects. Architectures with RNN, GRU, LSTM and Bidirectional Long Short-Term Memory (BiLST) layers were tested. The best performing architecture was experimentally determined to be a BiLST layer as shown in Fig. 4. We are using the work introduced by Hochreiter et al. [29], who demonstrated the ability to capture the context from both ends of the sequence by combining the outputs of two LSTM layers that pass the information in opposite directions. The resulting architecture is called BiLST. The output mode has been configured in the BiLST layer, so that the layer is able to receive a sequence as input and calculate an output vector. This form of dimension reduction is necessary in order to carry out the classification. The final FC layer specifies the number of classes via the number of output values. The class probabilities are calculated in the softmax layer by applying the softmax function. The cross-entropy cost function is utilized to quantify the discrepancy between the network's probabilistic predictions and the ground truth values, a method particularly suited for tasks involving categorically exclusive classes. This approach employs one-hot encoding to transform class representations into binary vector formats.

We calculate the cost function as the average of the cross-entropy losses for each prediction, relative to its corresponding target value:

$$cost = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log(\hat{y}_{i,j}), \tag{4}$$

where N denotes the total number of association samples (i.e., total number of timestamps), C represents the number of class categories, $y_{i,j} \in [0, 1]$ is the GT indicator for whether class j is the correct classification for sample i, and $\hat{y}_{i,j}$ is the predicted probability that sample i belongs to class j, as derived from the softmax function output.

C. Multi-Association Network (MANTa)

The development of the SANT demonstrated that data-driven association logic can be effectively learned by a deep learning model. Building upon this foundation, we developed MANTa with the objective to create a network capable of associating multiple (m) sensor objects with multiple (n) tracks in a single operational step. With MANTa, the following association scenarios can be addressed:

- **1 to n** - one SO to n tracks
- **m to 1** - m SOs to one track
- **m to n** - m SOs to n tracks
- **m to 0** - m SOs to no tracks

a) Data preprocessing: The association dataset of MANTa was created according to the described objective. Fig. 5 shows the input data structure with corresponding association tasks for timestamp 85 of sequence 20 of the KITTI dataset. Equivalent data were extracted across all sequences. All extracted tracks were each modified with noise as described in section IV-A and then normalized. Fig. 5 shows an assignment example with non-noisy SOs. The association result is given by the one-hot vector at the bottom. For each sensor object the network calculates a 1×18 one-hot vector containing its association result to the existing tracks. For a maximum of $T_{\max} = 16$ tracks, this results in an output vector of size 16×18. The ordering of the SOs is randomized per timestamp and the resulting association input matrix has dimension $F_{total} \times T_{\max}$, where F_{total} represents the total number of features (here $2 \times k = 10$, with k being the number

Fig. 5: MANTa, data structure, shows the non-noisy SOs to enable a visual assignment and increase understanding of the association procedure. Seven tracks are extracted from the KITTI dataset for the given timestamp of sequence 20. Eight sensor objects are generated in pseudo-random order. The one-hot vector shows the GT assignment of the first sensor object to the track at position two.

of values per state vector). The one-hot vector depicted in Fig. 5 shows the GT assignment of the first sensor object to the track at position two.

There are seven tracks in the timestamp of the sequence shown. For each track a corresponding SO is available. Additionally, a new SO was detected, leading to a total of eight sensor objects. The GT assignment of the first sensor object is shown in the one-hot vector at the bottom of Fig. 5. The assignment output can be thought of a matrix with dimensions maximum number of tracks $T_{\max} = 16$ and the number of possible assignment classes $C = 18$. Where each field can either be zero (no assignment) or one (assignment). For numerical reasons, we unfold this matrix to a one-hot vector of dimension $O_{\max} = 288 = 16 \cdot 18$. The assignment classes result from the described index class 1 to 16 and additional degrees of freedom. One degree of freedom of the assignment represents the case that no measurement exists, another that the measurement should not be assigned.

As for SANT, the cross-entropy cost function calculates the cross-entropy loss between network predictions $\hat{y}_{i,j}$ and target values $y_{i.j}$ for the unique assignment task for mutually exclusive classes. The already introduced one-hot vector is used to represent the class in binary form in a vector and thus generate a 1-to-$O_{\max}$ code. The following formula is used to calculate the cross-entropy loss values for each timestamp t:

$$loss_t = -\sum_{i=1}^{T_{\max}} \sum_{j=1}^{C} \left[y_{i,j} \ln(\hat{y}_{i,j}) + (1 - y_{i,j}) \ln(1 - \hat{y}_{i,j}) \right]. \tag{5}$$

Then, all scalars obtained per timestamp are summarized and divided by the number of samples N of a mini-batch for the cost function:

$$cost = \frac{1}{N} \sum_{t=1}^{N} loss_t. \tag{6}$$

b) Network architecture: The schematic representation Fig. 6 shows the developed network architecture for the simultaneous association of a large number of sensor objects to a large number of tracks. This is what we call a MANTa.

The BiLST layer processes the input data as already explained for SANT. The task of associating a SO list with a track list requires a separate network part for each track. This extension is labelled accordingly in Fig. 6. For each track (from 1 to $T_{\max} = 16$), the MANTa has been developed

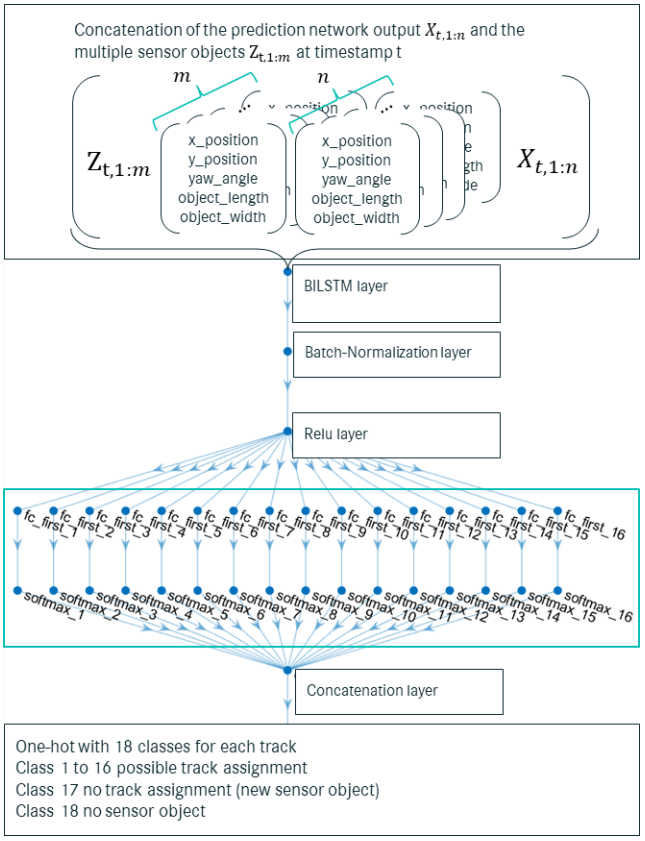

Fig. 6: Schematic representation of the generic network structure of MANTa.

with the fully connected, softmax stack that was introduced for SANT. Each softmax output consists of a vector with $C = 18$ elements, which represents the most probable assignment. This means that a single assignment can be realized for each track. The vectors 1 to $T_{\max}$ are linked together in the Concatenation layer. This creates a vector with 288 elements, whereby 18 elements each represent the most probable assignment of a SO to a track.

V. Experimental Evaluation

The developed networks were modularly integrated into the Tracking-by-Detection framework, replacing classical algorithms. SPENT substitutes the state predictions of the Kalman

– Dataset –	Training	Validation	Testing
Number of tracks	562	31	31
Model	**RMSE**		
SPENT	0.025	0.027	0.029
KF	0.066	0.065	0.066

TABLE I: Comparison of the Root Mean Square Error (RMSE) for SPENT and a KF framework implemented by the Daimler Truck Research Group following [1], [3].

Filter by directly estimating predictions per timestamp, eliminating the need for a predefined dynamics model.

Our recurrent network updates its internal hidden states at each timestamp, capturing temporal dependencies and enabling accurate state predictions without external correction. This approach, well-suited for real-time applications, presents a strong alternative to traditional methods. Using the KITTI dataset, focusing on vehicle tracks (cars and vans) divided into training, validation, and testing sets, we evaluated SPENT's performance. As first benchmark, we take a KF framework implemented by the Daimler Truck Research Group following [1], [3], which achieves an RMSE of 0.066 across 31 tracks (Table II) on the testing set. Our SPENT model reduces the RMSE by more than half to 0.029 using the identical datasets. For positional predictions the average deviation is 42 centimeters on the x-axis and 23 centimeters on the y-axis.

For the association of sensor objects to existing tracks, we developed SANT to replace classical methods like GNN. By substituting the distance metric and the Hungarian assignment procedure with a learned, data-driven assignment logic, SANT achieves an accuracy of 95% on a testing dataset with 391 samples (representing 5% of the total dataset with 7827 association samples). This approach not only simplifies the assignment process but also allows for adaptability in diverse scenarios, where classical methods may lack flexibility.

Expanding on SANT, MANTa addresses the limitations of a single object to track assignment by enabling multi-object assignments within each timestamp. This means MANTa is trained to assign a list of SOs to a list of tracks in a single operational step on the same dataset as SANT and also achieved an assignment accuracy of 95% for the six most frequently occurring association sets, i.e., the sets with one to six tracks per timestamp. It performs much worse (14%) for assignment scenarios with more tracks, which were less present in the training data.

In the context of the entire KITTI dataset, which includes both car and van objects, MANTa achieves an average association accuracy of 80%. This polarized performance with limitations for seven or more tracks was primarily attributed

TABLE II: Data association results for the SANT and MANTa networks. MANTa's average accuracy is 80%.

–	Simultaneous tracks per timestamp	
	1 to 6	7 to 16
Model	**Association accuracy**	
SANT	95%	95%
MANTa	95%	14%

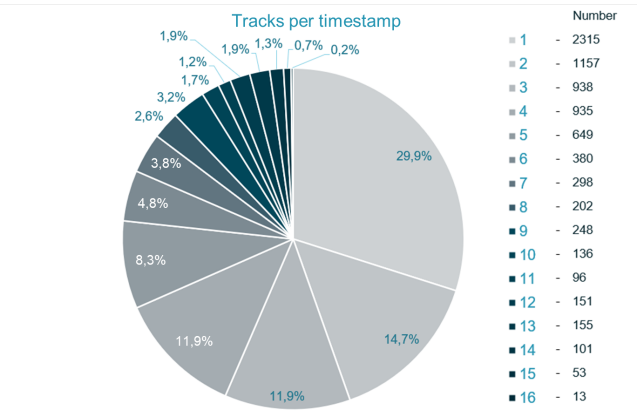

Fig. 7: The diagram shows the distribution of the number of existing tracks per timestamp. For the KITTI dataset, which includes both car and van objects. It reveals that timestamps containing one to six tracks constitute 81.5% of the samples (6374 of 7827). The legend shows the absolute number of samples which contain the respective number of tracks. In 29.9% of the samples, the data contains one track, in 14.7% two tracks. Only 18.5% (1448 of 7827 samples) of the samples contain more than six tracks, which leads to an unbalanced dataset and makes learning the association for MANTa for 6+ tracks harder.

to the characteristics of the extracted data. As illustrated in Fig. 7, the distribution of the number of existing tracks per timestamp reveals significant insights.

Notably, timestamps containing exactly one track constitute nearly one-third of the entire dataset, accounting for 29.9% of the data, which corresponds to an absolute count of 2315 samples. Timestamps containing one to six tracks constitute 81.5% of the samples and are therefore 6374 of 7827 samples. Consequently, tests were conducted using a reduced dataset with one to six tracks per timestamp to demonstrate the multi-association capability of the network. MANTa correctly assigns 95% of the dataset for timestamps containing one to six tracks. This result points to MANTa's proficiency in handling data given the appropriate dataset, as SANT also achieves a validation accuracy of approximately 95% across the entire KITTI dataset (including cars and vans). The primary advantage of MANTa over SANT is its ability to assign multiple sensor objects to multiple tracks in a single operational step.

VI. Conclusion and Future Work

In this paper, we introduced machine learning methodologies that enhance MOT for ADAS at the object list level. We designed, trained and evaluated three novel NNs: (i) SPENT for the prediction of tracked objects, (ii) SANT for the association of one new SO to a list of tracks and (iii) MANTa for the association of multiple SOs to a list of tracked objects. All networks were developed for real-time embedded applications with none having more than 50k trainable parameters. Our approach leaves the general structure of a KF framework intact, preserving modularity, interpretability and the ability to test each component separately. This work lays a foundation for future ADAS research, highlighting the potential of data-driven software development in overcoming the limitations of classical algorithms, which in practice often need to include heuristics. Future research topics will include: (a) testing of

the models on other datasets, to evaluate their generalization capabilities, (b) the quantification of uncertainties to improve decision making and (c) the investigation of multitasking networks to optimize the context-dependent tracking of multiple objects.

ACKNOWLEDGMENTS

We would like to express our sincere thanks to the organizations that provided financial support for this research project, namely the Daimler Truck AG and the Baden-Württemberg Cooperative State University (DHBW) Stuttgart.

REFERENCES

[1] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Sort: Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[2] J. Seidenschwarz, G. Brasó, V. Serrano, I. Elezi, and L. Leal-Taixé, "Simple cues lead to a strong multi-object tracker," *Paper*, 2022.

[3] J. Krejcí, O. Kost, O. Straka, and J. Duník, "Bounding box dynamics in visual tracking: Modeling and noise covariance estimation," *2023 26th International Conference on Information Fusion (FUSION)*, pp. 1–6, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261126559

[4] A. Milan, S. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," *Paper*, 2016.

[5] H. Liu, H. Zhang, and C. Mertz, "Deepda: Lstm-based deep data association network for multi-targets tracking in clutter," in *2019 22nd International Conference on Information Fusion (FUSION)*, 2019, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9011217

[6] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *ICCV*, 2017.

[7] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," *Paper*, 2022.

[8] W.-C. Hung, H. Kretzschmar, T.-Y. Lin, Y. Chai, and R. Yu, "Soda: Multi-object tracking with soft data association," *Paper*, 2020.

[9] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," *ArXiv*, vol. abs/2110.06864, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:238744032

[10] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision*, vol. 129, pp. 3069 – 3087, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221562313

[11] 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.

[12] A. Kampker, M. Sefati, A. S. A. Rachman, K. Kreisköther, and P. Campoy, "Towards multi-object detection and tracking in urban scenario under uncertainties." in *VEHITS*, 2018, pp. 156–167.

[13] H. Wu, W. Han, C. Wen, X. Li, and C. Wang, "3d multi-object tracking in point clouds based on prediction confidence-guided data association," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5668–5677, 2022.

[14] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, 2017. [Online]. Available: https://www.mdpi.com/2075-1702/5/1/6

[15] H. Winner, S. Hakuli, F. Lotz, C. Singer, and C. Stiller, *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2024.

[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, 1955.

[17] B.-T. Vo, "code set for research use: Multi-sensor multi-target tracking," *Code*, 2013. [Online]. Available: https://ba-tuong.vo-au.com/codes.html

[18] B. Ristic, S. Arulampalam, and N. Gordon, *Particle filters for tracking applications*. Artech House, 2004.

[19] S. J. Julier and J. K. Uhlmann, "The unscented kalman filter for nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[20] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*. IEEE, 2000, pp. 153–158.

[21] J. Dezert and Y. Bar-Shalom, "Joint probabilistic data association for autonomous navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 4, pp. 1275–1286, 1993.

[22] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *Paper*, 2017.

[23] N. Wojke, A. Bewley, and D. Paulus, "Deepsort: Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[24] Roberto Henschel and Laura Leal-Taixé and Daniel Cremers and Bodo Rosenhahn, "Fusion of head and full-body detectors for multi-object tracking," *Paper*, 2017.

[25] J. Fitz, "Datenassoziation für multi-objekt-verfolgung mittels deep learning," *Paper*, 2020.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *NIPS*, 2017.

[27] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *Paper*, 2017.

[28] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *ArXiv*, vol. abs/2005.12872, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:218889832

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[30] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[31] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[32] X. Gao, Z. Wang, X. Wang, S. Zhang, S. Zhuang, and H. Wang, "Dettrack: An algorithm for multiple object tracking by improving occlusion object detection," *Electronics*, vol. 13, no. 1, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/1/91

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2019.

[34] D. M. Reddy and N. V. S. Reddy, "Effect of padding on lstms and cnns," *Paper*, 2019.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ICML*, 2015.

[36] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2019. [Online]. Available: https://arxiv.org/abs/1803.08375

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 2014.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *AISTATS*, 2010.