Multi-object tracking (MOT) without dynamic models and hard association metrics

Christian Alexander Holz, Christian Bader, Matthias Drüppel

***Abstract*—In this paper, we develop Machine Learning (ML)-based methods for Multi Object Tracking (MOT) within the context of Advanced Driver Assistance Systems (ADAS). Given the increasing complexity and demand for precise and efficient object tracking systems in the automotive industry, this work focuses on the integration of ML techniques into established tracking methodologies. Key contributions encompass the creation and evaluation of three specialized neural networks: (i) the Prediction Network for predicting the trajectories of tracked objects, (ii) the Single Association Network (SANT) for associating incoming sensor objects with existing tracks sequential, (iii) and the Multi Association Network (MANTa) for associating multiple sensor objects with existing tracks within one timestep. We combine our ML methods with a traditional Kalman filter framework, offering a data driven approach to address MOT challenges while maintaining the modularity and interpretability of classical filter approaches. We asses both, the performance of all three models alone as well as their impact on the performance when they are integrated in the Kalman framework. Replacing single components allows us to get a clear evaluation of the impact of each ML model on the overall tracking system while maintaining the modularity of the system. The results reveal a modular, robust, and maintainable tracker, underscoring the potential of ML integration in AD Tracking Systems.**

***Index Terms*—Article submission, IEEE, IEEEtran, journal, LaTeX, paper, template, typesetting.**

I. Introduction

THE ongoing evolution of Advanced Driver Assistance Systems (ADAS) has brought the need for precise and reliable Multi Object Tracking (MOT) into the spotlight [1], [2]. In complex and dynamic environments, as encountered in urban traffic, it is crucial to simultaneously and accurately capture the positions and movements of multiple objects - a key challenge in computer vision (CV).

In the commonly used Tracking-by-Detection (TbD) paradigm, a tracker fuses detected sensor objects (SO) to create consistent object tracks over time. A key challenge within this paradigm is associating the incoming measured SO with their corresponding existing object tracks or initializing new object tracks.

Tracking frameworks form the heart of ADAS systems that are used in millions of vehicles around the globe. The vast majority of these frameworks rely on classical approaches such as the Kalman filter (KF) or its variants. 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. Furthermore the math and the theory itself is often clean and comprehendible. However, in the automotive industry these tracking systems need to be applied to a variety of different car models with different sensor sets, different installation heights of the sensors, different countries and always perform for a wide range of driving scenarios. This often leads to the increasing implementation of heuristics and parameters that tweak the tracking system for specific configurations and situations. But hand-engineered parameters and heuristics are hard to maintain and develop further from a software engineering point of view. Furthermore, performance of classical approaches can reach its limits in challenging situations.

In this work, we propose a data driven approach to tracking frameworks, which would allow the same system to be fine tuned for specific configurations relying only on data, thus increasing maintainability and adaptability. We do this preserving one of the biggest strengths of classical approaches: its modularity, by replacing only single tracking components with ML models.

In comparison to the Kalman filter, our Prediction Network is capable of predicting the state of individual objects without the need for a predefined state or observation model at runtime. Our Prediction Network holds the potential, particularly in terms of adaptability to various scenarios and the ability to effectively handle nonlinearities. Many conventional tracking systems rely on static methods for data association, often based on simple heuristics or fixed thresholds. In contrast, Single Association Network (SANT) employs machine learning to automate these processes and adapt more effectively to different scenarios. As a result, within the Tracking-by-Detection (TbD) MOT framework, SANT replaces the calculation of a distance metric and the Hungarian algorithm for the corresponding assignment. Furthermore, we integrate the Prediction Network and SANT 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 Advanced Driver Assistance Systems (ADAS), contributing to the further evolution of technologies for autonomous driving (AD).

II. Related work

A. Real time applications

ADAS are embedded real-time applications where it is crucial to predict the state of objects immediately after their

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. Drüppel is with the Center for Artificial Intelligence, Duale Hochschule Baden-Württemberg (DHBW), Stuttgart, Germany

detection. This rules out offline tracking methods as presented in [3] that process the entire video material at once in a batch process. Therefore, most recent approaches for tracking multiple objects rely on online methods that do not depend on future image information. Online methods use various features to estimate the similarity between the recognized objects and the existing tracks. This can be done on the basis of their predicted positions or even similarities in appearance [1], [2] [4]–[7] [8]–[12] .

B. Tracked object prediction

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], [13]. However, they reach their limits in more complex scenarios, particularly in the presence of non-linear motion patterns and interactions among multiple objects. In this work, we introduce a novel MOT approach that leverages Machine Learning (ML) to overcome these challenges. We specifically focus on the development and implementation of Neural Networks (NN), which can enable more precise and flexible data-driven object tracking.

C. Association

Another fundamental problem in TbD framework is data association. For this some approaches only consider the current state of the tracks, while others integrate temporal information such as the track history. This aggregate of temporal information can be done for example using attention mechanisms as used in [8], [9] or recurrent neural networks (RNNs) as in [4], [6]. The latter is what we are also pursuing in this work. Similar to the problem statement in Mertz et al [6], 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. Mertz et al [6] use a distance matrix based on the Euclidean distance measure as input for the developed association network, thus replacing an association algorithm such as the Hungarian Algorithm (HA) [14]. It can be assumed that the Euclidean distance measure was used as the basis for calculating the ground truth (GT) training data (distance matrices) and for the evaluation. However, this is not explicitly stated. When the Euclidean distance is used to calculate the training data, the network will only learn to follow this logic. This deprives the network of the opportunity to follow a different association logic. 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 using an undefined distance measure. This can increase the assignment of the time-based memory units, i.e. the history, and thus boost performance. For this purpose, the Long Short-Term Memory network (LSTM) layer is implemented, which enables the memory of information using hidden units over a time interval [15]. The goal of this work was therefore to develop an association network that is intended to solve the assignment of one or more sensor objects (SO) to an existing number of object tracks without a defined distance measure.

III. Overview of our proposed models

Our primary contribution is the development and evaluation of three NN that we labeled: (i) the Prediction Network, (ii) the Single Association Network (SANT), and (iii) the Multi Association Network (MANTa). Figure 1 provides an overview of our approach in which the association network can be represented by (ii) or (iii) and the prediction network is represented by (i). The input for the proposed Prediction Network (i) are the sensor objects (SO) in every time step. These objects consists of a fixed-dimensional state vector that contains information such as object position, orientation and dimension. The Prediction Network predicts all vectors to the next time step where they are used as input to the Single Association Network (SANT) to build target trajectories or object tracks.

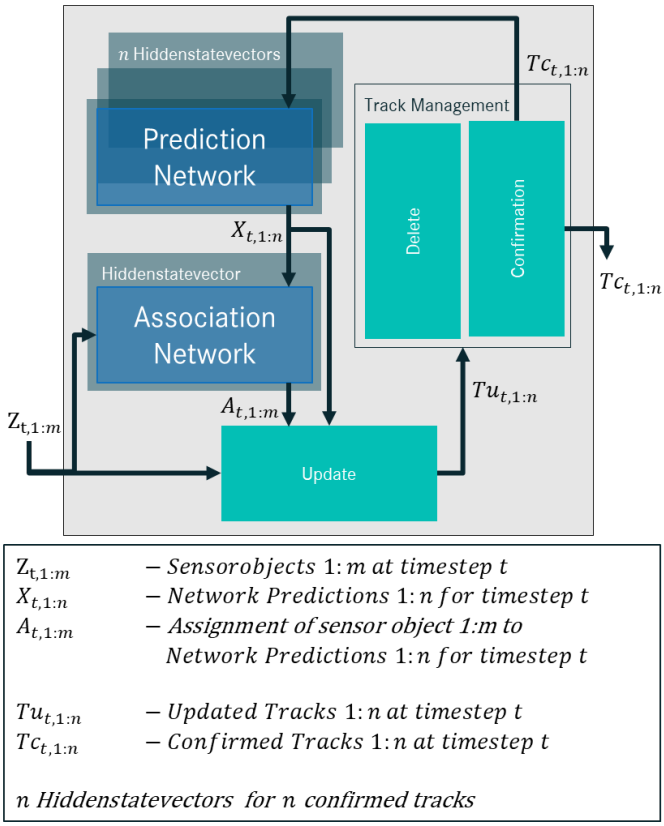

Fig. 1. Schematic representation of the two integrated networks in a tracking-by-detection (TbD) framework.

IV. Tracking with Prediction and Association Networks

We apply the tracking-by-detection (TbD) paradigm, in which a tracker fuses object fuses the object detections to generate object tracks that are consistent over time. [13], for example, provides a framework for analyzing tracking approaches that follow the TbD paradigm. This was used accordingly in this study.

To enable our models to incorporate temporal information, we use Long Short-Term Memory (LSTM) and bidirectional Long Short-Term Memory (BiLSTM) networks. Both for the

prediction and the association (SANT) of the sensor objects (SO) to the existing tracks.

A. Prediction Network

Most existing tracking methods associate incoming detections pairwise with object states predicted by a simple motion model, e.g. a constant velocity model, using a Kalman filter [1], [2], [13]. However, recent work has demonstrated that aggregating temporal information as well as contextual information can improve the tracking of multiple objects by utilizing higher order information in addition to pairwise similarities between detections [4], [6]–[8]. Our approach is to use the hidden states of the LSTM layer as an object-specific information storage. The Prediction Network was designed for an open-loop application, this means that the network predicts values for a future time step based on previously received data. For use in an online MOT process, the network is therefore able to make a prediction about the most likely next state values using the state values received for a given Statevector of a Sensorobject (SO).

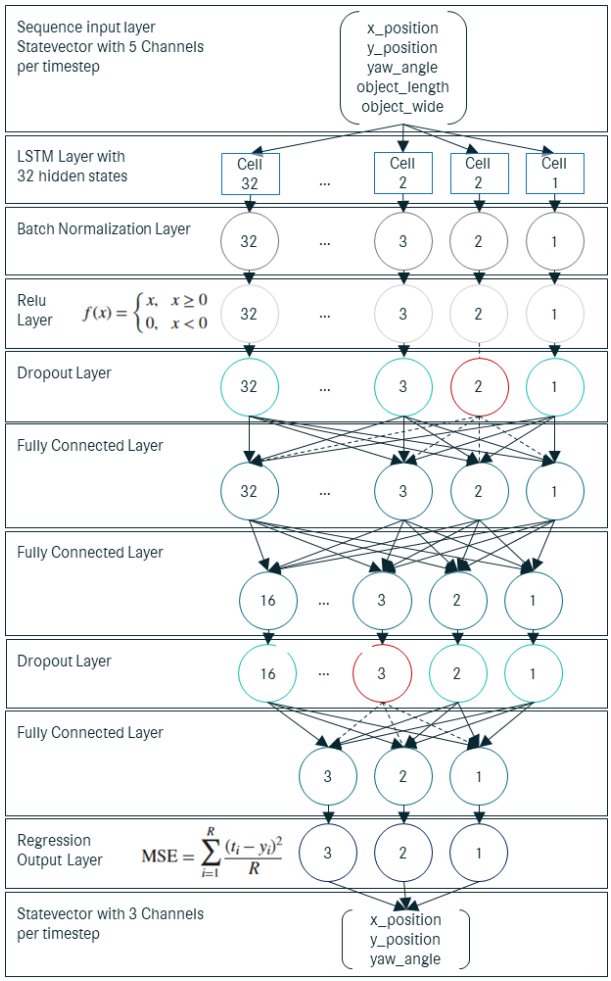

Fig. 2. Schematic representation of the generic prediction network structure.

a) Network architecture: The internal values (hidden states) of the LSTM layer are updated per time step based on the received measurement data. This updating process therefore provides a correction over the course of the sequence of each track. The number of hidden units (cells) corresponds to the amount of information that the layer remembers between time steps, as shown in Figure 2. The hidden states can contain information from all the previous time steps, regardless of the sequence length. The values of the hidden states of the LSTM layer are updated in each time step based on the received measurement data and the hidden states of the past timestep. These updates therefore results in an internal correction over the course of the sequence. In our prediction network, we have placed a batch normalisation layer directly after the LSTM layer. This means that the output of the LSTM layer is normalised before being passed on to the subsequent Relu layer. This helped speed up training and improve convergence by reducing internal covariate shifts [16]. A Relu layer performs a threshold operation to each element of the input, where any value less than zero is set to zero. The Relu function, also known as Rectified Linear Unit, is defined as follows:

$$\text{ReLU}(x) = \max(0, x) \tag{1}$$

At training time, a dropout layer randomly sets input elements to zero with a given probability. This operation effectively changes the underlying network architecture between iterations and helps prevent the network from overfitting [17]. The Fully Connected (FC) Layers multiplies the input by a weight matrix and then adds a bias vector. As shown in Figure 2 all neurons in a FC layer connect to all the neurons in the previous layer. This layer combines all of the local information learned by the previous layers. The last FC layer must be equal to the number of response variables in the Regression Output layer [18]. The regression layer computes the half-mean-squared-error loss. On the output of the regression layer at the end of the network (”regressionoutput”), the network calculates the loss through the mean square error (MSE) for the prediction for each state value. The MSE indicates the average of the squared difference between the model prediction and the target value, which we use as the measure of the quality of the prediction. For a single observation, the MSE is given by:

$$MSE = \sum_{i=1}^{R} \frac{(t_i - y_i)^2}{R} \tag{2}$$

where R is the number of predicted states, t_i is the ground truth value (from the Kitti tracks) and y_i is the prediction of the network for sample i. For our sequence-to-sequence regression network the loss function of the regression layer is half the mean square error of the predictions for each time step, normalized by the sequence length S, not by the number of the predicted Statevector R:

$$loss = \frac{1}{2S} \sum_{i=1}^{S} \sum_{j=1}^{R} (t_{ij} - y_{ij})^2. \tag{3}$$

During training, the average loss is calculated using the observations in the mini-batch, so S equals the mini-batch length. In relation to the KITTI data set (cars and vans), a Root Mean Square Error (RMSE) of 0.025 was achieved for the position prediction of all objects in the dataset. Using a

standard Kalman filter (KF), an RMSE of 0.066 was achieved on the same data set.

b) Data preprocessing: Ground truth (GT) data in the form of tracks of vehicles from the KITTI data set was used for our development. A GT Track contains the information of an object over time, starting with its appearance and ending with its exit from the sensor detection range. In order to achieve better generalization and to increase the chance that the training converges, the track state values at time t (predictors) and track state values at time t+1 (targets) were normalized following [19], such that the possible predictions and targets have a mean value of zero and a unit variance. The mean value μ and the standard deviation σ for each state variable were calculated for all tracks using the following equations:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \quad \text{and} \quad \sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} |A_i - \mu|^2} \tag{4}$$

where A_i the combined length of all tracks and N is the number of states. We also apply pre-padding as described in [20] where Reddy et al. show how padding influences the performance of neural networks in sequence-based tasks. Their study suggests that although both pre-padding and post-padding are feasible, the choice of padding technique can have a significant impact on the efficiency of the model, especially for LSTM networks where the sequence context is crucial. As [20] described, padding can lead to noise in the network, but one sequence adjustment per mini-batch is required for training LSTM networks. To minimise the effect, the sequences of the training data were therefore sorted by length before the mini-batch padding. Figure 3 clearly shows that the padding (turquoise area) is minimised by a sorted training data set per mini-batch. Without this modification, no convergence could be achieved in the training.

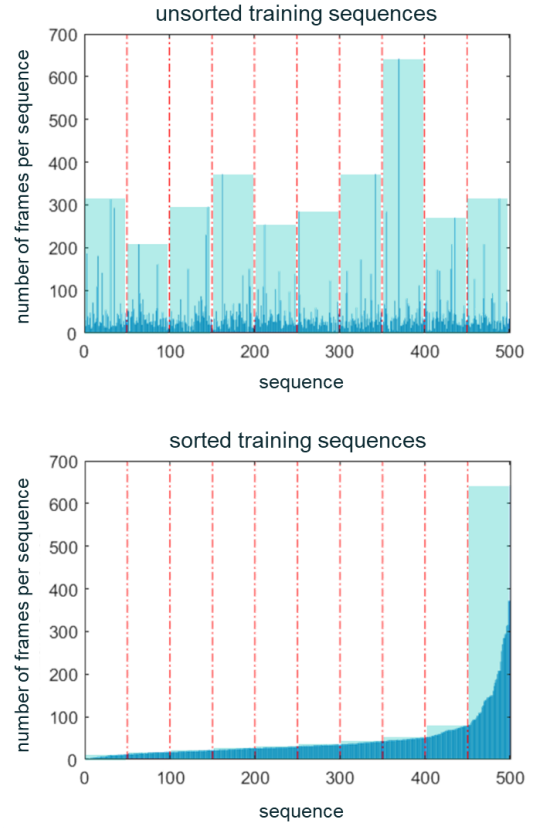

Fig. 3. Mini-batch padding effect, training data for LSTM networks.

B. Single Association Network (SANT)

SANT stellt einen datenbasierten Ansatz dar, welcher lernen kann, das kombinatorische Non Deterministic Polynomial Time (NP) hard Optimierungsproblem der Datenassoziation vollständig zu lösen. Im Vergleich zu den meisten Assoziationsverfahren [4] [6] [6] stellt SANT einen Ansatz dar, welcher als Inputdaten keine definierte Distanzmatrix erhält, sondern die bestehenden Tracks sowie jede neue Messung (SO) erhält. Damit entfällt die Definition eines Abstandsmaßes, womit dem Netzwerk die Freiheit gegeben wurde, einer eigenen Assoziationslogik zu folgen bzw. diese datenbasiert zu lernen. Somit ersetzt SANT die Berechnung eines Abstandsmaßes sowie einen Assoziationsalgorithmus wie z.B. den Hungarian Algorithm (HA).

a) Datenvorverarbeitung: Für die Entwicklung des Single Association Networks (SANT) wurde die Datenassoziation als zeitlich gegliedertes Problem (Sequenz-zu-Vektor) betrachtet. Das Datenassoziationsproblem beschreibt somit die Zuordnung von einem SO zu einem Set von Tracks. Die entsprechenden Tracks wurden aus dem KITTI Datensatz der gelabelten Kameraaufzeichnungen extrahiert. Allerdings

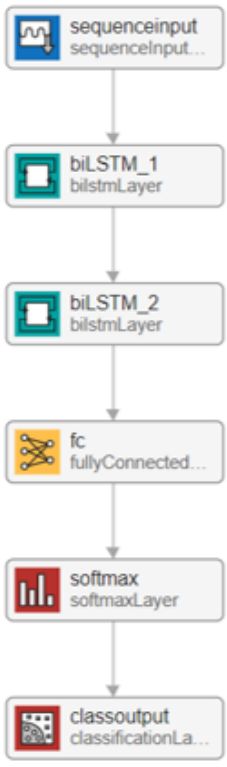

Fig. 4. Schematic representation of the generic network structure of SANT.

existieren für das betrachtete Problem der Assoziation keine realen GT Daten. Um sicherzustellen, dass die Zuordnung eines Sensor Objekts (SOs) zu einem Track eines Tracksets genau eine korrekte Lösung besitzt, wurde das SO in einem Zeitschritt aus dem bestehenden Trackset eines Zeitschritts erzeugt. Die Daten wurden entsprechend künstlich verrauscht, um aus den GT Daten realistische Sensordaten zu erzeugen. Das Datenformat wurde entsprechend gewählt, um eine indexbasierte Trackzuordnung für das Single Association Network (SANT) zu ermöglichen. Die Größe der Inputmatrix entspricht daher $m * n$. Mit $m = 2 * AnzahlZustandswerte$ und $n = Trackanzahl(t)$.

b) Netzwerkentwicklung: Das Netzwerk (Fig. 4) ist als Sequenz-to-Classification ausgelegt, wobei der sequenceinput die beschriebene Inputmatrix pro Zeitschritt darstellt.

BILSTM,... Die vollständig verknüpfte Schicht (FC) gibt über die Anzahl der der Ausgabewerte die Klassenanzahl n vor. Die n Klassenwerte werden in der Softmax Schicht durch Anwendung der Softmaxfunktion in eine eindeutige Wahrscheinlichkeitsverteilung berechnet (wie in Kapitel 2.2.1 beschrieben). Durch Anwendung der Cross-Entropie-Operation in der Ausgabeschicht (Classout) konnte SANT auf einen korrekten Zuordnungswert (GT Klassenindex) trainiert werden.

Die Cross-Entropie-Kostenfunktion (Kreuzentropie) berechnet den Cross-Entropie-Verlust zwischen Netzvorhersagen und Zielwerten für die eindeutige Zuordnungsaufgabe (für sich gegenseitig ausschließende Klassen). Dabei wird mittels One-Hot-Kodierung die Klasse binär in einem Vektor dargestellt und somit ein 1-zu-n Code generiert. Nach folgender Formel werden die Crossentropie-Verlustwerte für jeden Eingabewert Y_j und zugehörigen Zielwert (Targetvalue) T_j elementweise berechnet:

$$loss_j = -(T_j ln Y_j + (1 - T_j) ln(1 - Y_j)) \tag{5}$$

Um einen Skalar $loss$ zu erhalten werden alle Verlustwerte $loss_j$ aufsummiert und durch die Anzahl der Samples N geteilt. Optional können die Verlustwerte von definierten Samples mit dem Gewichtungsfaktor w_j gewichtet werden:

$$loss = \frac{1}{N} \sum loss_j w_j \tag{6}$$

Eine entsprechende Nutzung des Gewichtungsfaktors w_j kann bei Datensätzen mit unausgewogenen (imbalanced) Klassenverteilung hilfreich sein.

C. Multi Association Network (MANTa)

a) Datenvorverarbeitung:

b) Netzwerkentwicklung:

V. Experimental Evaluation

... KITTI-Car Benchmark.

VI. Conclusion

The conclusion goes here.

Acknowledgments

This should be a simple paragraph before the References to thank those individuals and institutions who have supported your work on this article.

Appendix
Proof of the Zonklar Equations

Proof of the First Zonklar Equation

Appendix goes here.

Proof of the Second Zonklar Equation

And here.

References

[1] Jenny Seidenschwarz, Guillem Brasó, Victor Serrano, Ismail Elezi, Laura Leal-Taixé, "Simple cues lead to a strong multi-object tracker," *Paper*, 2022.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, Ben Upcroft, "Simple online and realtime tracking," *Paper*, 2016.

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

[4] Anton Milan, Seyed Rezatofighi, Anthony Dick, Ian Reid, Konrad Schindler, "Online multi-target tracking using recurrent neural networks," *Paper*, 2016.

[5] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, Nenghai Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *Paper*, 2017.

[6] H. L. H. Z. C. Mertz, "Deepda: Lstm-based deep data association network for multi-targets tracking in clutter," *Paper*, 2019.

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

[8] W.-C. H. H. K. T.-Y. L. Y. C. R. Yu, "Soda: Multi-object tracking with soft data association," *Paper*, 2020.

[9] Q. C. W. O. H. L. X. W. B. L. N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *Paper*, 2017.

[10] T. M. A. K. L. L.-T. C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," *Paper*, 2022.

[11] P. C. H. Ling, "Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking," *ICCV*, 2019.

[12] Q. C. W. O. H. L. X. W. B. L. N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," *ICCV*, 2017.

[13] Ba-Tuong Vo, "code set for research use: Multi-sensor multi-target tracking," 2013. [Online]. Available: https://ba-tuong.vo-au.com/codes.html

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

[15] S. H. J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

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

[17] 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.

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

[19] I. G. Y. B. A. Courville, *Deep Learning*. MIT Press, 2019.

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