Estimating Uncertainties of Recurrent Neural Networks in Application to Multitarget Tracking

Daniel Pollithy, **Marcel Reith-Braun**, **Florian Pfaff**, and **Uwe D. Hanebeck**

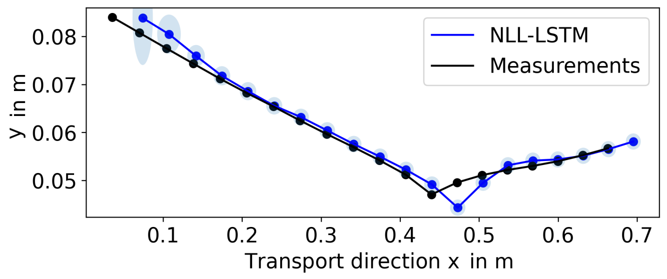
***Abstract*— In multitarget tracking, finding an association between the new measurements and the known targets is a crucial challenge. By considering both the uncertainties of all the predictions and measurements, the most likely association can be determined. While Kalman filters inherently provide the predicted uncertainties, they require a predefined model. In contrast, neural networks offer data-driven possibilities, but provide only deterministic predictions. We therefore compare two common approaches for uncertainty estimation in neural networks applied to LSTMs using our multitarget tracking benchmark for optical belt sorting. As a result, we show that the estimation of measurement uncertainties improves the tracking results of LSTMs, posing them as a viable alternative to manual motion modeling.**

I. Introduction

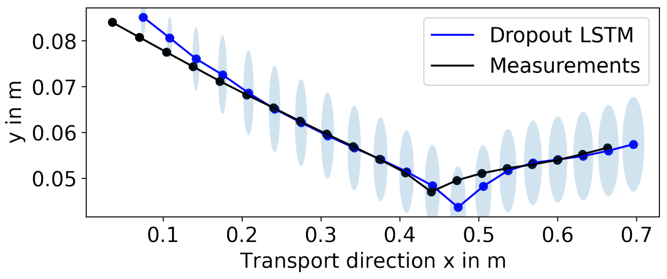
Multitarget tracking is a challenging task. In the past five decades, a variety of solutions have been proposed and implemented in different industrial areas, including air traffic control, oceanography, robotics, and biomedical research [1]. They share the necessity to estimate the number and state of subjects within their domain. In the example of a vehicle assistance system, this could be the state of the road users driving in the vicinity, which is important to regulate the vehicle's own velocity.

The level of difficulty of a tracking application depends upon the number of targets, the ability to estimate their states and to predict how the latter are going to evolve. A common approach to estimate the states of targets in multitarget tracking applications is to find a one-to-one correspondence between the measurements and known objects, which we also refer to as association, and then employ a Kalman filter. For every target, the Kalman filter predicts the state in the next time step, which in turn is used to determine the association in the subsequent one. For this method to work reliably, the a priori knowledge about the targets' motions has to be formulated, the uncertainty about this motion model must be subsumed with an appropriate noise term, and other parameters have to be optimized. For every type of target, the modeling process needs to be repeated. If a tracking task involves frequently changing types of targets, this becomes laborious and also hard to automate. Therefore, we investigate how this adaptation to new targets can be automated using data-driven methods.

Daniel Pollithy, Marcel Reith-Braun, Florian Pfaff, and Uwe D. Hanebeck are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany. daniel.pollithy@student.kit.edu, marcel.reith-braun@kit.edu, pfaff@kit.edu, uwe.hanebeck@kit.edu



(a) Estimation of heteroscedastic aleatoric uncertainties as a function of the input data. The large uncertainty of the first time step describes the difficulty of predicting the second measurement if only the first one is given.



(b) We use Monte Carlo dropout to estimate the epistemic uncertainty of predictions. The increasing uncertainties of this method are helpful to track the unusual motion of the particle.

Fig. 1: This figure displays the tracks on a top-down view onto the belt of a sorting machine. The tracks belong to the same particle moving along the transport direction from left to right. It is a challenging out-of-model example because the collision with the wall does not correspond to the regular behavior. Variances correspond to calibrated 99.9% confidence regions.

Optical belt sorting is an example of an application that could benefit from such a data-driven approach. In [2], a multitarget tracking algorithm is employed in which a Kalman filter is used to track the particles' states. Since the main purpose of the tracking is to derive a prediction that is used to perform the actual sorting via bursts of compressed air, this is referred to as predictive tracking. To make this approach even more versatile, the algorithm is to be extended to sort heterogeneous materials whose types are changing day by day.

In this work, we compare the currently used Kalman filter with a data-driven model based on neuronal networks that can adapt to new materials using a training phase and avoid the manual adjustments to the motion models and hyperparameters.

We started by using long short-term memory (LSTM) [3] for predictive tracking, inspired by their usage as a learned motion model in the area of multiobject tracking (MOT). However, LSTMs only yield point estimates and do not provide uncertainties that are useful for the association process of our multitarget tracker.

The novelty of this work is the application and evaluation of two approaches to uncertainty estimation for LSTMs in the context of multitarget tracking. The uncertainties are to be used to improve the associations. Furthermore, we investigate the obtained uncertainties to gain insights into the underlying mechanisms of the algorithms employed.

The paper is organized as follows. After an explanation of multitarget tracking and its application to optical belt-sorting in Sec. II, we go into uncertainty estimation for neural networks and calibration of uncertainties. In Sec. III, we show how the existing approach for estimating aleatoric uncertainties can be adapted to recurrent neural networks (RNNs). Sec. IV explains our different models. The results of the evaluation in Sec. V are discussed in Sec. VI and a conclusion is provided in the Sec. VII.

II. Background & Related Work

In the following subsections, basic knowledge about Bayesian filtering with the Kalman filter (KF) and time series predictions with RNNs is assumed.

A. Multitarget Tracking (MTT)

The goal of multitarget tracking (also known as multiobject tracking) is to estimate the number and states of all targets at time step t given sensor inputs from the time steps 0 to t. The multitarget tracker used for the bulk material sorting task proposed in [2] is used as the basis for our work. Sensor data is preprocessed by extracting the centroids of all particles that are visible in the current image. These centroids are the input to the multitarget tracker, which runs in an infinite loop. It consists of several components described in [4].

A central component is the observation-to-track association. For trackers that assume that each target only gives rise to a single measurement, the association of the measurements with the tracks is determined that maximizes some measure of compatibility. For example, one may choose the association that maximizes the likelihood of association, which includes both the uncertainty in the measurement and the prediction. For simplicity, we assume for now that all known particles are visible and no new particles enter the observable area (some additional considerations are required if this is not the case). Then, the association can be described by a permutation τ.

If all densities and likelihoods involved are Gaussians, the association that maximizes the likelihood is identical to the one that minimizes the sum of the squared Mahalanobis distances $d(\mathbf{H}\underline{\hat{x}}_t^i, \underline{\hat{y}}_t^{\tau(i)}, \mathbf{\Sigma}_t^{i,\tau(i)})^2$, in which $\underline{\hat{x}}_t^i$ denotes the predicted state of track i at time step t, $\mathbf{H}$ is the measurement matrix of the linear measurement model, and $\underline{\hat{y}}_t^{\tau(i)}$ denotes the centroid of the measurement that is associated with track i.

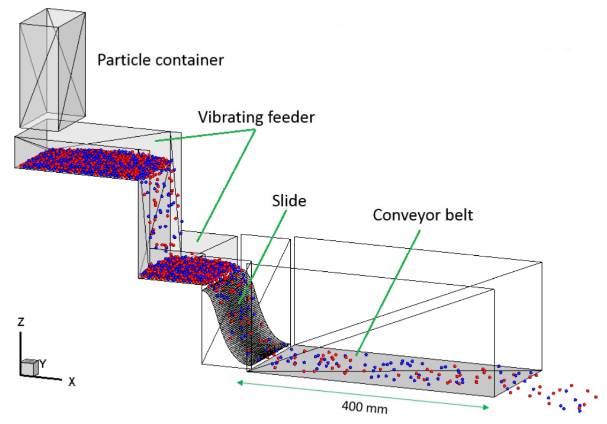


Fig. 2: Model of our experimental belt sorting machine [5] used for the DEM simulation. Multitarget tracking is applied only to the particles on the conveyor belt. In actual sorting scenarios, a camera is mounted above this part of the machine.

Based on these distances, the optimal association for this criterion can be found in $O(n^3)$ using a solver for linear assignment problems. In practical implementations, an extra step is frequently added. In the gating step, certain possible association decisions are ruled out before using the solver, which helps to speed up the process.

Based on the association decision, tracks are initialized, maintained and deleted in the track management component. The association decisions and the result of the track management are then used by the filtering and prediction component. In this, each track that was measured is updated with its corresponding new measurement and a prediction for the time step $t+1$ is generated. This is then fed into the association component to determine the association decision at the next time step.

B. Uncertainty Estimation for Neural Networks

A neural network $f^{\underline{\omega}}(\underline{u}) \rightarrow \underline{y}$, which maps inputs $\underline{u}$ to outputs $\underline{y}$, approximates an unknown, arbitrary, and continuous function with its parameters $\underline{\omega}$. During training, the parameters $\underline{\omega}$ are estimated based on the set of training data in form of the inputs $\mathcal{U} = \{\underline{u}_1, \ldots, \underline{u}_N\}$ and outputs (labels) $\mathcal{Y} = \{\underline{y}_1, \ldots, \underline{y}_N\}$, which constitute N pairs of samples from the unknown function. We assume all samples in $\mathcal{U}$ (and thus also $\mathcal{Y}$) are i.i.d.

1) Characterization of Stochastic Uncertainties: In stochastic system modeling, uncertainties in the estimates are caused either by system noise, measurement noise, or the inaccuracies of the models. All of these shall be modeled as stochastic uncertainties. In the neural network literature, they are sometimes characterized as aleatoric or epistemic, depending on how they are modeled by the neural network and thus on the properties they may have [6].

Uncertainties in neural networks that capture noise inherent in the data, i.e., stemming from measurement noise, are often referred to as aleatoric uncertainties [7]. They are said to have the property that, similar to measurement noise, they can usually not be reduced drastically without changing the sensor. Kendall and Gal [6] further distinguish two kinds of aleatoric uncertainty. It can be homoscedastic, which means that it is independent of the input data, or heteroscedastic. In the latter case, the noise can be thought of as a function of the input data. In our example, measurements that stem from different bulk material particles may have different measurement uncertainties due to, e.g., different positions on the belt.

The second source of uncertainty is the system model. Errors can arise both from the stochastic nature of the underlying system and the deviations between the real system and the system model. In the context of neural networks, uncertainties capturing these error sources are also referred to as epistemic uncertainties [6]. A higher modeling power of the network (obtained by, e.g., using more neurons and layers) can help to reduce such uncertainties, but can also lead to overfitting. With a suitable network architecture, increasing the training data can help to reduce the error in the model and thus the uncertainties [6].

2) Estimating Heteroscedastic Aleatoric Uncertainties: When regarding a regression problem, the measurement uncertainty is often modeled as an additive Gaussian noise placed on the model's output. Likewise, in modeling aleatoric uncertainties for neural networks, the likelihood $p(\underline{y}_i|\underline{u}_i, \underline{\omega})$ is often assumed to be Gaussian-distributed with the mean given by the output of the neural network and σ_i^2 describing its variance. Thus, σ_i^2 is a direct measure for the network's aleatoric uncertainties.

Along with the function to generate the outputs $f^{\underline{\omega}}$, our neuronal network learns a function $\sigma^{\underline{\omega}}$ that yields an uncertainty for any input. For brevity, we shall write $\sigma_i := \sigma^{\underline{\omega}}(\underline{\hat{u}}_i)$. To learn the correct parameters, we minimize the negative log-likelihood (NLL)

$$L(\underline{\omega}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\,\sigma_i^2} ||\underline{\hat{y}}_i - f^{\underline{\omega}}(\underline{\hat{u}}_i)||^2 + \frac{1}{2} \log \sigma_i^2 \quad (1)$$

of the data samples to obtain an estimator for the expectation value and the variance of the posterior [6]. Note that in this case, (1) serves as a loss function for the neural network. When fixing σ_i to the same constant value for all predictions, the loss reduces to the well-known MSE loss.

3) Bayesian Neural Networks: To estimate the epistemic uncertainty, we make use of Bayesian neural networks (BNNs) [8]. Whereas weights in ordinary neural networks can be interpreted as point estimates given by maximum likelihood estimation (MLE) or maximum a posteriori estimation (MAP), BNNs place distributions over their weights, i.e., treat them as random variables. A standard Gaussian distribution is the most common assumption for the prior of the weights.

To model a BNN, we then have to define $p(\underline{\omega}|\mathcal{U}, \mathcal{Y})$, describing the joint distribution of all weights given the training dataset and $p(\underline{y}|\underline{u}, \underline{\omega})$, describing the distribution of the output given the input and the network's weights. For regression, the latter is (similar to modeling aleatoric uncertainties) usually assumed to be Gaussian with its mean given by the model's output. Then, given a new input $\underline{\hat{u}}$ that is not in the training dataset, we can get the distribution of the output $\underline{y}$

$$p(\underline{y}|\underline{\hat{u}}, \mathcal{X}, \mathcal{Y}) = \int_{\Omega_{\underline{\omega}}} p(\underline{y}|\underline{\hat{u}}, \underline{\omega})\, p(\underline{\omega}|\mathcal{U}, \mathcal{Y})\, d\underline{\omega} \quad (2)$$

by marginalizing out the weights $\underline{\omega}$ by integrating over the sample space $\Omega_{\underline{\omega}}$ of the weights. Usually, the posterior of the weights $p(\underline{\omega}|\mathcal{U}, \mathcal{Y})$ is intractable and must therefore be approximated, e.g., using variational inference [9].

4) MC Dropout: Monte Carlo dropout is a method to perform approximate variational inference in neural networks. In variational inference, the true posterior $p(\underline{\omega}|\mathcal{U}, \mathcal{Y})$ is approximated by a surrogate model $q(\cdot)$.

Gal et al. [9] state that the use of the neural network regularization technique *dropout* during training can be seen as a variational approximation to the posterior of a BNN when using standard MSE loss and L_2 regularization. Thus, after training a neural network with dropout, we obtain a surrogate model $q(\cdot)$.

Substituting $p(\underline{\omega}|\mathcal{U}, \mathcal{Y})$ in (2) with the surrogate model and using Monte Carlo integration with T Monte Carlo runs (referred to as dropout during inference [9], denoted $f^{\underline{\omega}}_{\text{Drop}}$), the first moment of the predictive posterior $q(\underline{y}|\underline{\hat{u}})$ is approximated simply by their average, i.e.,

$$\mathrm{E}_{q(\underline{y}|\underline{\hat{u}})}(\underline{\boldsymbol{y}}) \approx \frac{1}{T} \sum_{t=1}^{T} \underline{f}^{\underline{\omega}}_{\text{Drop}}(\underline{\hat{u}}) \,. \quad (3)$$

In contrast to the standard application of dropout, in which it is only used during the training, it is also applied during the application of the network. Gal et al. [9] refer to this approximation as MC (Monte Carlo) dropout.

The variance of the output is obtained by calculating the sample variance of the Monte Carlo runs (epistemic part) and integrating the inverse model precision (aleatoric part)

$$\begin{aligned}\mathrm{Cov}_{q(\underline{y}|\underline{\hat{u}})}(\underline{\boldsymbol{y}}) \approx \tau^{-1}\mathbf{I}_D + \sum_{t=1}^{T} \underline{f}^{\underline{\omega}}_{\text{Drop}}(\underline{\hat{u}})^\top \underline{f}^{\underline{\omega}}_{\text{Drop}}(\underline{\hat{u}}) \\ - \mathrm{E}_{q(\underline{y}|\underline{\hat{u}})}(\underline{\boldsymbol{y}})^\top \mathrm{E}_{q(\underline{y}|\underline{\hat{u}})}(\underline{\boldsymbol{y}})\,, \quad (4)\end{aligned}$$

in which τ is the inverse variance (precision) of the Gaussian likelihood $p(\underline{y}|\underline{\hat{u}}, \underline{\omega})$. It is assumed to be fixed for all examples and can be calculated from the dropout probability $(1-p)$, the weight decay hyperparameter λ, the number of training samples N and the prior length-scale l according to $\tau = (p\,l^2)(2N\lambda)^{-1}$ [9].

Under the assumptions of [9], MC dropout is said to be applicable to any deterministic neural network that is trained with the mean squared error loss function, dropout, and L_2 regularization.

5) Variational Recurrent Neural Networks: Gal et al. [10] state that if the dropout masks are held constant throughout the Monte Carlo runs during all time steps of an RNN, then the use of MC dropout is also a method for obtaining the expectation value of the posterior distribution when applying the RNN to generate predictions. There exist a few other

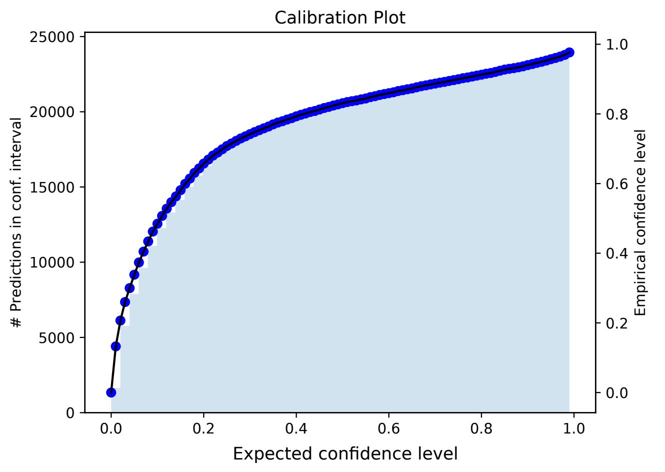


Fig. 3: Reliability curve of the NLL-LSTM evaluated on the DEM dataset before calibration. The closer the curve is to the diagonal, the better calibrated it is. The calibration curve for a given expected confidence level is calculated from the cumulative histogram depicted in light blue.

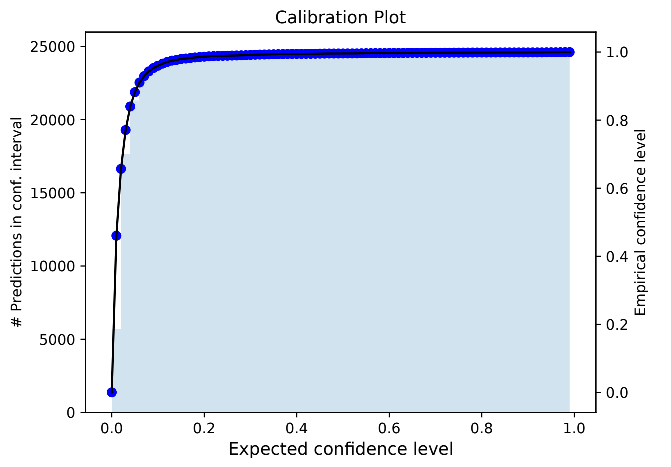


Fig. 4: Reliability curve of the predictions made by the dropout LSTM with a sample size of 250 evaluated on the DEM dataset. The model clearly overestimates the uncertainty. For example, over 90% of all realization in the held out dataset are within the 0.1 confidence region.

approaches to model epistemic uncertainties of RNNs, for example, Bayesian recurrent neural networks, which make use of a scheme called Bayes-by-Backprop [11].

C. *Uncertainty Calibration & Calibration Methods*

Kuleshov et al. [12] point out that the covariances of the uncertainties provided by neural networks using MC dropout are not aligned with the actual prediction errors. For example, a 90% confidence region for the predicted positions might contain only half of the points, which effectively makes it an empirical 50% confidence region. The mapping from expected confidence levels to empirically observed confidence levels is called reliability curve. A model is said to return calibrated uncertainties if its calibration curve is a straight line from $[0, 0]$ to $[1, 1]$. The authors of [10] have found that the dropout rate used in MC dropout has an important influence on how well the uncertainty of a BNN is calibrated. Therefore, they proposed a novel loss function called *ConcreteDropout* that can be used to adapt the dropout rate in a way that improves the calibration.

To achieve calibrated uncertainties, [12] proposed a simple regression scheme that can be fitted on held out data to obtain a mapping from empirical to predicted confidence levels. It makes use of monotonic regression in order to account for the monotonicity of the calibration curve. [12] also compared their method to ConcreteDropout and to tuning the dropout rate by line search. They report that the uncertainties provided by their approach are more reliable than those of the two other methods.

III. Estimating Aleatoric Uncertainties for RNNs

In this section, we describe how an RNN can be trained to capture aleatoric uncertainties by considering the negative log-likelihood. For this, we adapt the approach presented by Kendall and Gal [6] to a probabilistic model of an RNN.

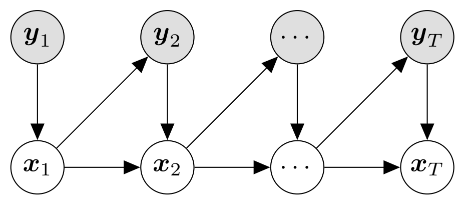


Fig. 5: Probabilistic graphical model representation of a recurrent neural network used for modeling $(\boldsymbol{y}_t)_{t=1}^T$.

A. *Probabilistic Model of RNNs*

In terms of a generative model, the forward pass of an RNN comprises two different functions: $\underline{\hat{x}}_t = \underline{f}_{\mathrm{x}}(\underline{\hat{x}}_{t-1}, \underline{\hat{y}}_t)$ maps the old RNN's hidden state $\underline{\hat{x}}_{t-1}$ with a given input to the new state $\underline{\hat{x}}_t$. $\underline{f}_{\mathrm{y}}(\underline{\hat{x}}_t)$ provides a value for the measurement at the next time step. For our interpretation as a graphical model in Fig. 5, we convert the generative model into a probabilistic one. The state transition probability density function is then given by $p(\underline{x}_t|\underline{x}_{t-1}, \underline{y}_t) = \delta(\underline{x}_t - \underline{f}_{\mathrm{x}}(\underline{x}_{t-1}, \underline{y}_t))$. For the first time step, the prior hidden state is set to the zero vector: $p(\underline{x}_1|\underline{x}_0, \underline{y}_1) = \delta(\underline{x}_1 - \underline{f}_{\mathrm{x}}(\underline{0}, \underline{y}_1))$. Moreover, we assume a Gaussian likelihood for $p(\underline{y}_t|\underline{x}_{t-1})$. A similar approach for discrete random variables can be found in [13].

B. *Negative Log-Likelihood Loss*

Starting from an MLE/MAP objective, we model the joint probability of a single track $p(\underline{y}_{1:T}) = p(\underline{y}_1, \underline{y}_2, \ldots, \underline{y}_T)$. $(\underline{y}_t)_{t=1}^T$ are the realizations from $(\boldsymbol{y}_t)_{t=1}^T$. We assume $p(\underline{y}_1)$ to be uniformly distributed on the input domain as a noninformative prior about the initial position of tracks.

$$p(\underline{y}_{1:T}) = p(\underline{y}_1) \int_{\Omega_{\underline{x}_1}} \cdots \int_{\Omega_{\underline{x}_{T-1}}} \prod_{t=1}^{T-1} p(\underline{x}_t | \underline{x}_{t-1}, \underline{y}_t) \\ p(\underline{y}_{t+1} | \underline{x}_t) \, d\underline{x}_{T-1} \cdots d\underline{x}_1 \quad (5)$$

is calculated according to our model in Fig. 5 by marginalizing over the latent variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}$. (5) can be simplified by applying the sieve property of the Dirac-distributed transition density $p(\underline{x}_t | \underline{x}_{t-1}, \underline{y}_t)$. This can be seen as using $\underline{f}_{\mathsf{x}}(\cdot)$ repeatedly (t times) until we reach the very beginning of the chain, where the state is defined to be $\underline{0}$. Thus, (5) becomes

$$p(\underline{y}_{1:T}) = p(\underline{y}_1) \prod_{t=2}^{T} p(\underline{y}_t | \underbrace{\underline{f}_{\mathsf{y}}(\cdots \underline{f}_{\mathsf{x}}(\underline{0}, \underline{y}_1), \cdots, \underline{y}_{t-1})}_{\underline{x}_{t-1}}) \, . \quad (6)$$

Considering the likelihood of all N tracks, we take the log and neglect constant terms to get the loss

$$L = \frac{1}{2N} \sum_{i=1}^{N} \frac{1}{T_i - 1} \sum_{t=2}^{T_i} \frac{1}{\sigma_{i,t}^2} ||\underline{\hat{y}}_t^i - \underline{f}_{\mathsf{y}}(\underline{\hat{x}}_{t-1}^i)||^2 + \log \, \sigma_{i,t}^2 \, . \quad (7)$$

This equation is equivalent to the mean over a negative log-likelihood loss (see Sec. II-B.2) for every time step. It therefore coincides with an RNN trained with standard RNN training methods with the objective to minimize the negative log-likelihood. As in (1), for a fixed value of $\sigma_{i,t}^2$, the standard MSE loss for RNNs can be used.

IV. Predictive Tracking with LSTM

Our measurement space is two-dimensional and comprises the position along (x) and orthogonal to the transport direction (y). We train two different LSTMs to predict the next position of a particle $(\mathsf{x}_t, \mathsf{y}_t)$ along with a diagonal covariance matrix $\mathbf{\Sigma}_t = \mathrm{diag}(\sigma_{\mathsf{x},t}^2, \sigma_{\mathsf{y},t}^2)$ based on the past measurements of the corresponding track. In Sec. IV-B and Sec. IV-C, we describe these models, which we call NLL-LSTM and dropout LSTM. The first one captures the aleatoric uncertainties, whereas the second model subsumes the epistemic uncertainty. We consider the individual models separately to measure their influence on the data association and compare them with the standard LSTM (Sec. IV-D) used as a baseline.

We use the calibrated uncertainties (see Sec. IV-E) as the basis for calculating the Mahalanobis distances for the association component of our tracker. The tracker works as described in Sec. II-A and the utilization of the LSTM's uncertainties works analogously to the use of the predicted covariance matrices obtained by a Kalman filter. To our knowledge, no one has previously estimated the uncertainties of LSTMs and used them for association decisions in multitarget tracking.

A. LSTM Architecture & Training

All of our trained RNNs share the following settings[1]:

- Network topology: LSTM with two layers and 64 and 32 hidden units, respectively. Final dense layer without an activation function. The number of output units depends on the uncertainty estimation method.
- L_2 regularization with $\lambda = 10^{-5}$.
- Optimizer: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$).
- The initial learning rate is 0.01. It is decayed every 200 epochs. 1000 training epochs in total.
- Backpropagation through time is used with the same type of loss function for every time step.

B. NLL-LSTM

The loss function for this LSTM is the negative log-likelihood plus an L_2 regularization term. Its final dense layer has four outputs: The predicted x- and y-coordinate at time step $t+1$ and the two corresponding variances. The variances need no supervision as they are learned implicitly using the NLL.

As in [6], we train the network to predict the log variance $s_{i,t} = \log \sigma_{i,t}^2$ for every time step. The logarithm is used to avoid a division by zero and to mitigate the problem of invalid negative variances. With this change, the loss in (7) becomes

$$L \propto \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i - 1} \sum_{t=2}^{T_i} \exp(-s_{i,t}) ||\underline{\hat{y}}_t^i - \underline{f}_{\mathsf{y}}(\underline{\hat{x}}_{t-1}^i)||^2 + s_{i,t}.$$

C. Dropout LSTM

In contrast to the NLL-LSTM, we do not describe how MC dropout can be derived for RNNs, but instead refer to the detailed description in [9]. With the output of every time step being Gaussian-distributed, MC dropout (3) provides an estimator for the expected value of the posterior of an RNN. It follows from [9] that the variance of the posterior can be approximated as in (4).

Our dropout LSTM has two units in its last layer that output an estimate for the next position. The loss function is the MSE. The prior length scale l is set to 10^{-2}. The dropout probability and the number of Monte Carlo runs are determined by performing a grid search with the aim to maximize the predictive log-likelihood [10]. Dropout rates between 1% and 0.1% yielded the highest predictive log-likelihoods. We chose to work with a rate of 0.2%.

D. Standard LSTM

An LSTM trained with the MSE loss that does not consider uncertainties serves as a baseline. A scaled identity matrix is used as the covariance matrix for calculating the Mahalanobis distance for the association decision. A line search is used to find a scaling factor that yields a good association accuracy.

E. Calibration

We now go into the post-processing step that calibrates the uncertainties for use in the association component. First, we generate the calibration curve for a held out dataset, which is explained in the next paragraph. Second, we use the monotonic regression procedure proposed in [12] to find a function $R : [0, 1] \to [0, 1]$ that maps the observed empirical

[1]Our implementation can be found at `https://github.com/KIT-ISAS/TrackSort_Neural_Public/tree/MFI2020`.

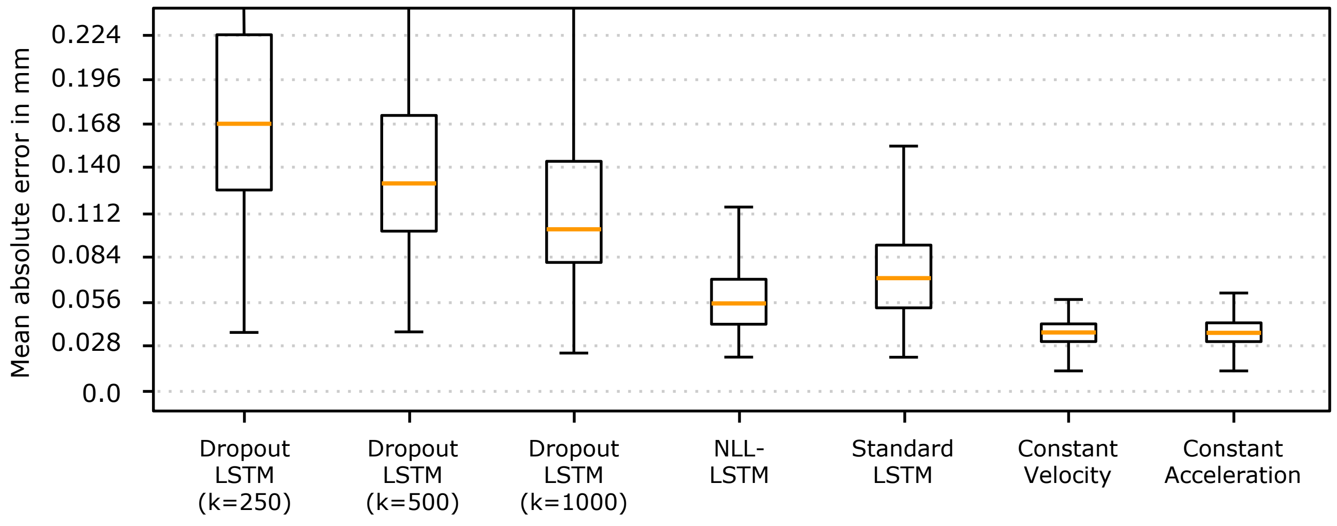


Fig. 6: Evaluation results for data-driven models and classical models used in combination with a Kalman filter on the peppercorn dataset. The box plots show the distances between the predictions and corresponding ground truths.

confidence levels to the expected ones. Last, this function is used to rescale the covariance ellipsoids. The uncertainties are scaled solely based on their magnitude and irrespective of which particle and time step the uncertainty was provided. In practice, this is generally an oversimplification, but it significantly reduces the amount of data required.

To determine the calibration curves in our two-dimensional case, we first calculate the squared Mahalanobis distance $d(f_{\mathrm{y}}(\underline{\hat{x}}^i_{t-1}), \underline{\hat{y}}^i_t, \mathbf{\Sigma}^i_t)^2$ to obtain the uncertainty-aware sum of the squared errors for every prediction. For Gaussian-distributed predictions with calibrated uncertainties, the squared Mahalanobis distances are χ^2-distributed with two degrees of freedom. Using the inverse CDF $F^{-1}_{\chi^2}(p_{\mathrm{exp}})$ of the χ^2-distribution, we can calculate a confidence region[2] in which a share of p_{exp} should be on average. Afterward, we calculate the corresponding empirical confidence level $p_{\mathrm{emp}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{d(f_{\mathrm{y}}(\underline{\hat{x}}^i_{t-1}), \underline{\hat{y}}^i_t, \mathbf{\Sigma}^i_t)^2 \leq F^{-1}(p_{\mathrm{exp}})\}$ as the fraction of the number of tuples whose squared Mahalanobis distances lie within the mentioned confidence interval. Repeating this procedure for various confidence levels p_{exp} provides us with the calibration curve. An example can be found in Fig. 3.

In our calibration, we do not change the shapes of the covariance ellipsoids but only rescale them with a rescaling factor r, i.e., $\mathbf{\Sigma}^{\mathrm{cal}}_i = r\mathbf{\Sigma}_i$. For this, we first have to chose a p_{des} for which we want the expected confidence level to be equal to the empirical one. Afterward, we calculate r as the ratio $r = F^{-1}_{\chi^2}(R(p_{\mathrm{des}}))/F^{-1}_{\chi^2}(p_{\mathrm{des}})$ using the mapping R derived from the calibration curve. Note that in our calibration, r is the same constant for all predictions, thus preserving the ratios between all covariances.

[2]In our setting, the described region is effectively a credible region. However, for reasons of consistency with the literature on calibration (see Sec. II-C), we refer to it as a confidence region.

V. Evaluation

A. Datasets

We use two slightly different datasets to evaluate the predictions: A real one, which captures the movement of peppercorns, and an artificial one comprising simulated cylinders. Both involve particles that fall onto the begining of a conveyor belt, are then accelerated while moving to the end, and finally disappear at the edge of the belt, which is shown on the right in Fig. 2.

1) Peppercorn Dataset: This dataset contains the tracks of a few thousand peppercorns that were captured on our experimental belt sorting machine. The true association of the measurements with the tracks was generated by running the existing tracker and correcting false associations manually.

2) DEM Dataset: The second dataset used in this evaluation was generated by the discrete element method (DEM), which uses a three-dimensional model of our experimental belt sorter [14]. The dataset contains the ground truth positions of simulated cylinders at high frame rates. Due to the artificial nature of the dataset, there are no measurement errors. Further, the true associations are available, which can be used to verify the results. As in [5], the frame rate is downsampled from 10,000 Hz to a realistic frame rate of 100 Hz. In the time the particles are on the belt, 20 to 48 measurements are obtained per particle. Measurements of up to 150 different particles are obtained in the individual time steps.

B. Evaluation of the Position Prediction

As a first step, we compare the accuracy of the predicted positions of all LSTM-based approaches with a Kalman filter with constant velocity or constant acceleration model in Fig. 6. To determine the prediction accuracy, we calculate the deviations between the measurements and the predicted measurements. The box plots show the absolute prediction errors averaged for each track in the peppercorn dataset when using the ground truth associations.

It can be observed that the magnitudes of the errors are comparable and the majority of all predictions deviate less than $0.3\,\mathrm{mm}$ from the true value. The accuracy of the dropout LSTM is dependent on the number of samples k. This matches the observations in [9]. Even sampling 1000 forward passes results in predictions that are inferior to those of the constant acceleration model. Note that the dropout LSTM yields errors at the level of the standard LSTM if MC dropout is not used.

Both the standard LSTM and the NLL-LSTM do not perform better than the Kalman filters. This clearly shows the difficulty to model physical phenomena without additional world knowledge. However, the NLL-LSTM achieves significantly better results than the standard LSTM.

Kendall and Gal [6] interpret the negative log-likelihood loss in (1) as a learned loss attenuation. This is a possible explanation for the NLL-LSTM's increased accuracy as the loss for every time step gets scaled according to the predicted uncertainty. Therefore, the network can continue to minimize its loss, even though the predictions of the first time step have a far larger mean squared error than the later time steps. This is due to the decreasing aleatoric uncertainty in the dataset.

C. MTT Evaluation

To evaluate whether the estimated uncertainties are beneficial for the data association, we compare the number of errors made by the different methods when applied to the task of tracking the particles of the DEM dataset. All experiments were done with a confidence level of $p_{\text{des}} = 99.9\%$ for the calibration.

In [15], two metrics to quantify the number of errors in the association in the bulk material sorting scenario were introduced. In the optimal case, all measurements of a specific particle are assigned to the same track and only these measurements are assigned to the track.

An error of the first kind occurs when measurements of multiple tracks were associated with one track. The number of such errors is denoted e_1. The number of errors of the second kind (e_2) provides the number of particles whose measurements were assigned to more than one track. In this work, the harmonic mean $e_{\text{mean}} = \frac{2\hat{e}_1\hat{e}_2}{\hat{e}_1+\hat{e}_2}$ between the proportions of errors of the first kind ($\hat{e}_1 = e_1/n_{\text{tracks}}$) and errors of the second kind ($\hat{e}_2 = e_2/n_{\text{particles}}$) is used to rank the results.

Since noise-free measurements are unrealistic in real-world scenarios, we added some artificial white Gaussian noise to the measurements. In Fig. 7, the e_{mean} of the tracker is shown for different noise levels. The NLL-LSTM provides lower numbers of errors compared with the standard LSTM. While the NLL-LSTM was trained only once, a line search was employed for the standard LSTM to determine the best scaling factor for the identity matrix used as the covariance matrix in the calculation of the squared Mahalanobis distance. Since both, the covariance matrix determined using the line search and the uncertainties of the NLL-LSTM lead to improvements in association accuracy, we believe that modeling aleatoric uncertainty in RNNs does indeed help to improve multitarget tracking. The error level of the dropout LSTM is inferior to the other methods. We discuss this in Sec. VI.

TABLE I: Comparison of additive noise and aleatoric uncertainty evaluated using the peppercorn dataset. The uncertainty area of a prediction is approximated by the size of the bounding rectangle around the covariance ellipse with size $4\sigma_x\sigma_y$. The positive correlation between additive noise and mean aleatoric uncertainty area suggests that the NLL-LSTM adapts its uncertainty predictions to the measurement noise.

Additive noise standard deviation in mm	Mean aleatoric uncertainty area in mm^2
0	0.1743
0.1	0.1843
0.2	0.2157
0.3	0.3073
0.4	0.5130
0.5	1.0424

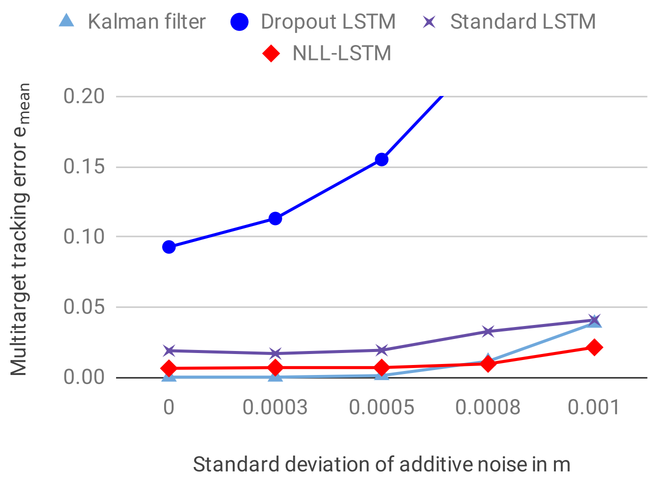


Fig. 7: Comparison of the LSTM models and a Kalman filter with a constant velocity motion model. Gaussian-distributed noise with the standard deviations shown on the abscissa was added. The sample size of the dropout LSTM is 250.

D. Reliability of the Uncertainty

The reliability diagram in Fig. 3 shows that the uncalibrated NLL-LSTM is closer to the diagonal describing the correct calibration than the curve of the uncalibrated dropout LSTM, which is shown in Fig. 4. For the latter, an expected confidence level of 0.2 already contains more than 98% of the predictions of the held out data. As the curves show, calibration is necessary for both approaches.

E. Aleatoric Uncertainty

In this subsection, we analyze the aleatoric uncertainties of the NLL-LSTM. The learned aleatoric uncertainties are heteroscedastic since they are dependent on the network's input, which can be observed in Fig. 8. The figure also implies a heavy influence of the number of measurements on the predicted aleatoric uncertainty. In order to investigate whether the magnitude of the aleatoric uncertainty is not only affected by the time step, we calculate the mean of the sizes of the bounding rectangles around all covariance ellipses for increasing standard deviations of additive noise. The results are depicted in Table I. The mean area of the variances are positively correlated with the additive noise terms. This suggests that the magnitude of the predicted uncertainties is not just a learned bias in the network.

VI. Discussion

The absolute error of the dropout LSTM's predictions show how much this method relies on large sample sizes. Fig. 6 suggests that sample sizes above 1000 could potentially yield more accurate results. However, since an enormous amount of computation may be required to obtain acceptable results, this method is not suitable for our task and we cannot report advantageous effects on the tracking when using reasonable sample sizes.

In contrast, the estimates of aleatoric uncertainties, trained via a negative log-likelihood loss, deliver good results on our benchmark. The decreasing uncertainty over the number of observations of a track suggests that the model is capable of subsuming the noise inherent in the data. In particular, the model yields a high uncertainty for the very first prediction, which is the expected result because little is known about the particle when it was observed only once.

Our uncertainty calibration method rescales the estimated variances of different time steps with the same factor. In practice, this is an oversimplification that could be amended by conditioning the predictions onto their time step, which may be a subject for future research.

VII. Conclusion

In this work, we evaluated Monte Carlo dropout and training with negative log-likelihoods as methods for predicting uncertainties for recurrent neural networks. The uncertainties were employed in a multitarget tracking task with the goal of improving the associations. Quantitative comparisons of the different methods show promising results for the estimation of aleatoric uncertainties and that data-driven models can yield accuracies in the same order of magnitude as Kalman filters with standard motion models.

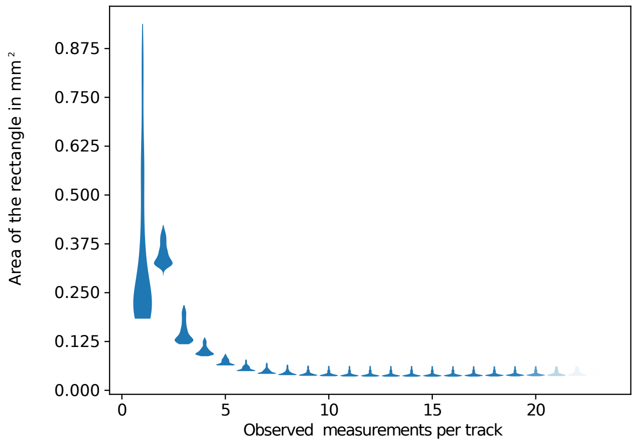


Fig. 8: A Negative correlation (r=-0.59) between the number of observations of a track and the size of the predicted covariance's bounding rectangle can be observed when applying the NLL-LSTM to the peppercorn dataset.

Acknowledgment

The IGF project 20354 N of the research association Forschungs-Gesellschaft Verfahrens-Technik e.V. (GVT) was supported via the AiF in a program to promote the Industrial Community Research and Development (IGF) by the Federal Ministry for Economic Affairs and Energy on the basis of a resolution of the German Bundestag.

References

[1] M. Mallick, B. Vo, T. Kirubarajan, and S. Arulampalam, "Introduction to the Issue on Multitarget Tracking," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 373–375, 2013.
[2] F. Pfaff, M. Baum, B. Noack, U. D. Hanebeck, R. Gruna, T. Längle, and J. Beyerer, "TrackSort: Predictive Tracking for Sorting Uncooperative Bulk Materials," in *Proceedings of the 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2015)*, San Diego, California, USA, Sep. 2015.
[3] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.
[4] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, 1999.
[5] F. Pfaff, C. Pieper, G. Maier, B. Noack, H. Kruggel-Emden, R. Gruna, U. D. Hanebeck, S. Wirtz, V. Scherer, T. Längle, and J. Beyerer, "Simulation-Based Evaluation of Predictive Tracking for Sorting Bulk Materials," in *Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2016)*, Baden-Baden, Germany, Sep. 2016.
[6] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" in *Advances in Neural Information Processing Systems (NIPS)*, 2017.
[7] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in neural information processing systems*, 2017, pp. 3581–3590.
[8] R. M. Neal, "Bayesian Learning for Neural Networks," Ph.D. dissertation, Dept. of Computer Science, University of Toronto, 1994. [Online]. Available: https://www.cs.utoronto.ca/~radford/bnn.book.html
[9] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *International Conference on Machine Learning*, Jun. 2016, pp. 1050–1059.
[10] ——, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 1027–1035, event-place: Barcelona, Spain.
[11] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian Recurrent Neural Networks," *ArXiv*, vol. abs/1704.02798, 2017.
[12] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate Uncertainties for Deep Learning Using Calibrated Regression," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, Jul. 2018, pp. 2796–2804.
[13] Y. J. Choe, J. Shin, and N. Spencer, "Probabilistic interpretations of recurrent neural networks," 2017.
[14] C. Pieper, G. Maier, F. Pfaff, H. Kruggel-Emden, S. Wirtz, R. Gruna, B. Noack, V. Scherer, T. Längle, J. Beyerer, and U. D. Hanebeck, "Numerical Modeling of an Automated Optical Belt Sorter Using the Discrete Element Method," *Powder Technology*, Jul. 2016.
[15] F. Pfaff, G. Kurz, C. Pieper, G. Maier, B. Noack, H. Kruggel-Emden, R. Gruna, U. D. Hanebeck, S. Wirtz, V. Scherer, T. Längle, and J. Beyerer, "Improving Multitarget Tracking Using Orientation Estimates for Sorting Bulk Materials," in *Proceedings of the 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017)*, Daegu, Republic of Korea, Nov. 2017.