Uncertainty-based Meta-Reinforcement Learning for Robust Radar Tracking

1st Julius Ott
Infineon Technologies AG
julius.ott@infineon.com

2nd Lorenzo Servadei
Infineon Technologies AG
lorenzo.servadei@infineon.com

3rd Gianfranco Mauro
Infineon Technologies AG
gianfranco.mauro@infineon.com

4th Thomas Stadelmayer
Infineon Technologies AG
thomas.stadelmayer@infineon.com

5th Avik Santra
Infineon Technologies AG
avik.santra@infineon.com

6th Robert Wille
Technichal University of Munich
robert.wille@tum.de

***Abstract*—Nowadays, Deep Learning (DL) methods often overcome the limitations of traditional signal processing approaches. Nevertheless, DL methods are barely applied in real-life applications. This is mainly due to limited robustness and distributional shift between training and test data. To this end, recent work has proposed uncertainty mechanisms to increase their reliability. Besides, meta-learning aims at improving the generalization capability of DL models. By taking advantage of that, this paper proposes an uncertainty-based Meta-Reinforcement Learning (Meta-RL) approach with Out-of-Distribution (OOD) detection. The presented method performs a given task in unseen environments and provides information about its complexity. This is done by determining first and second-order statistics on the estimated reward. Using information about its complexity, the proposed algorithm is able to point out when tracking is reliable. To evaluate the proposed method, we benchmark it on a radar-tracking dataset. There, we show that our method outperforms related Meta-RL approaches on unseen tracking scenarios in peak performance by 16% and the baseline by 35% while detecting OOD data with an F1-Score of 72%. This shows that our method is robust to environmental changes and reliably detects OOD scenarios.**

***Index Terms*—Radar Sensors, Reinforcement Learning, Meta Learning, Out-of Distribution Detection**

I. Introduction

Radar sensors are gaining momentum in the modern semiconductor industry. Various modulation types, independence of light conditions, low-cost and privacy-friendly features lead the radar technology to be successfully employable in applications such as people detection and object tracking [1]. To keep up the pace of advancements, attention directs to applications like multi-person tracking, which is critical in several areas such as automotive safety, medical services, or logistics [2], [3]. Multi-person tracking attempts to estimate the position of each target in a scene. For estimating the track positions, the Unscented Kalman Filter (UKF) is often used in radar tasks [4]. This method has the scope of a bayesian estimation of the target position incorporating a nonlinear dynamic movement model. The nonlinear transition is approximated by the unscented transform, described in [5]. By design, the UKF relies on hyperparameters. These tunable hyperparameters describe the dynamics of the sensor system and real-world scenarios. Particularly for radar data, occlusions, non-human disturbances, and limited resolution are significant challenges for robust tracking. Thus, the choice of optimal hyperparameters varies with the environment, and often their initial choice is suboptimal. Given the environment's variety of settings and noise, finding the best hyperparameters for any possible scenario is infeasible.

Recent work focused on estimating the underlying dynamics of a UKF with specific neural networks [6] or using Reinforcement Learning (RL) to provide the best hyperparameters for a given scene, as shown in [7], [8]. Although this approach is promising, it often lacks robustness, and the data distribution on inference time might be different from the one used for training. Furthermore, the UKF model might fail to track in overcrowded scenarios, given the capability of its underlying model.

In real-life applications such as automotive radar sensing, robustness also has to be critical, and then Deep Reinforcement Learning (DRL) is employed. However, neural networks are known to fail when the test data distribution is far off the training data distribution [9]. As a consequence, meta-learning is often used in literature to close the gap between training and test data distributions [10] and aims to adapt quickly to novel tasks. Although model-based meta-learning algorithms have obtained excellent results, they are limited to the network design stage [11]. For this reason, research has recently focused on model-agnostic meta-learning, which enables learning tasks independently of the machine learning model. This is granted through task-specific optimization and is widely applicable in domains such as few-shot classification and RL. In the case of Meta-RL, context variables are a promising way to incorporate task-specific information, as shown in [12]. In this method, the context variable is learned by a neural network from task-specific data. Although the method is performant, it requires computing a context variable using multiple data samples from each additional task. Storing this data during inference is inefficient. Hence, our proposed method uses domain information to formulate a context variable. Our approach does not need to store data: it computes the context variable from easy-to-obtain input distribution statistics, which we refer to as context prior. Furthermore, as shown in the experiments, our method leads to an improved domain generalization compared to other state-of-the-art approaches. Nevertheless, in several real-world applications (e.g., radar-tracking), improved domain generalization cannot address the limitations inherent to the task itself. For example, tracking operations in small, crowded scenarios with obstacles becomes increasingly tricky. In order to assess the reliability of our Meta-RL method, we develop an uncertainty mechanism via bootstrapped networks. The uncertainty mechanism is combined with the context prior that encodes information about the task difficulty. In this approach, scenes where tracking is prone to failure, are classified as OOD, thus assessing the particular reliability of the tracker in the current scenario.

As a summary, the contributions of the paper are the followings:

1) Meta-RL for domain generalization without additional memory footprint using context priors
2) Enhanced OOD detection with context priors that encode task difficulty

The remainder of the paper proceeds as follows: in Section II, we introduce the radar-tracking problem and how to tackle it with RL. Afterward, we explain the specific signal processing in Section III. In the same section, we show how the input data distribution is used to compute an informative context variable. At the end of this section, using the context variable, we propose a Meta-RL algorithm for environment generalization and detection of OOD scenarios. Finally, we evaluate our method on a multi-target radar tracking dataset against related Meta-RL methods in Section IV. Our proposed approach outperforms comparable Meta-RL approaches in terms of peak performance by 16% on the test scenarios and the baseline of fixed parameters by 35%. In the same way, it detects OOD scenarios with an F1-score of 72%. Thus, our approach is more robust to environmental changes and reliably detects OOD scenarios.

In Section V, we summarize our results and give an outlook on future work in Section.

II. Background and Motivation

In this section, we review the background and related work. In Section II-A, we first outline the principle of radar tracking. Afterward, we explain how RL can be used to optimize radar tracking. Additionally, we extend this concept by introducing the fundamentals of Meta-RL and Uncertainty-based RL.

A. Radar Tracking

Frequency Modulated Continuous Wave (FMCW) radars can estimate the range, Angle of Arrival (AoA), and velocity of targets. In the case of radar tracking, we use the range and AoA to determine the target position. The typical radar tracking pipeline can be divided into signal processing, detection, clustering, and tracking, as shown in [13]. A high level description is given in Figure 1. The signal processing stage elaborates the sensor data from each radar antenna to estimate the reflected signal's range and angle. The resulting image is a so-called Range-Angle Image (RAI). Afterward, the RAI is convolved with a window that determines the signal threshold based on the surrounding noise. Usually, a Constant False Alarm Rate (CFAR) algorithm [14] or a variation thereof defines the threshold. A clustering algorithm groups nearby detected signals, and the respective cluster means are input to the tracking stage. In this part of the pipeline, the track management determines whether to assign the measurement to a track, open a new track, discard the measurement or delete non-active tracks. Before updating the track, the measurement has to be filtered by the tracking filter based on the last position and an underlying movement model. The UKF is a commonly used tracking filter [15]. The presented tracking pipeline heavily

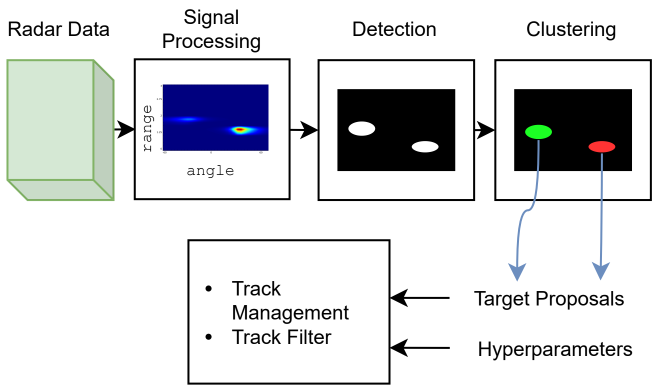


Fig. 1: High-level description of a Radar Tracking Pipeline.

relies on hyperparameters. Namely, the tracking performance depends on the gating threshold for assigning tracks and the covariance matrix of the measurement and state transition models. Typically, those hyperparameters are determined by an expert user with recorded data and ground truth positions evaluating the Normalized Estimation Error Squared (NEES). However, this approach is unlikely to perform well once the radar is deployed in a different environment. Thus, recent work proposed to use RL to tackle the combinatorial problem of finding the best set of parameters for any scenario [16].

B. Reinforcement Learning

In RL the problem is formalized as a Markov Decision Process (MDP) by (S, A, R, p, γ), where S is the state space as radar sensor input, A is the action space defined as hyperparameters, R is the reward as tracking performance shown in [16], p_π is the unknown transition probability between states following policy π and γ is the discount factor. Let $\tau = (s_t, a_t, r_t, s_{t+1})$ define the transition from state s at time step t to the next state s_{t+1} following action a_t with reward r_t. In traditional RL the goal is to maximize the sum of expected rewards

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim p_\pi[r(s_t,a_t)]}. \tag{1}$$

This can be achieved by value iteration methods [17]. There, we define a Q-Value $Q(s_t, a_t)$ for each state and action pair that estimates the expected reward. Afterward, for each state we select the action which maximizes the Q-Value. With the advancements of neural networks in the context of universal function approximators [18], methods like DQN [19] use neural networks for the Q-value approximation. However, value iteration methods become infeasible when the action space is continuous. Thus, the policy gradient theorem [20] provides a way to update the policy for continuous actions. The combination of Q-Learning and policy gradients forms the basis of actor-critic methods. In recent years, policy gradient-based algorithms like Proximal Policy Optimization (PPO) [21] or actor-critic methods like Soft Actor-Critic (SAC) [22] have shown great success. The particular choice of the algorithm depends on the task at hand. On-policy methods update the policy only with transitions from following the *same* policy, which is data inefficient. In contrast, off-policy methods store transitions in a replay buffer and update the policy from these. The current policy and the policy that collected the transitions can be unrelated, which is the core concept of off-policy methods. Since the policy update is possible with transitions from any policy, the collected data is reusable. Due to the data efficiency, off-policy methods are widely used in applications where data is scarce [23]. Despite advancements in RL, transferring RL agents to real-world problems remains challenging [24]. To address these limitations, [25] has shown how meta-learning can be used to transfer an RL agent to a real-world problem.

C. Meta Reinforcement Learning

State-of-the-art Machine Learning (ML) models usually require many data [26]. However, humans can transfer experiences to have a "educated guess" about new tasks based on their knowledge from related tasks, e.g., recognizing an animal from the zoo in the TV. Meta-learning has been introduced to support the training of ML models to imitate this human capability. To this end, each task is associated with a respective dataset $\mathcal{D}$. The meta-learner aims to increase the performance on unseen test tasks, which are different from the tasks it has been trained on. This procedure can be adapted to many machine learning fields, such as supervised learning [27] and RL [28]. In Meta-RL, the task difference can be induced by different reward functions or environments, e.g., several Maze puzzle environments [29]. Successful algorithms as Pearl [12], Reptile [30] or MAML [31] are designed to minimize the adaptation steps in new environments. As an example, adaptation is not feasible in a radar tracking application during inference due to the lack of rewards. Thus, in applications without reward signal during inference, domain generalization is key. Inspired by [32], where the target objective of MAML is adapted to improve domain generalization, we adapt the design of Pearl to an efficient context variable computation. In Pearl, these context variables are computed by a neural network from stored transitions of the test task. As a consequence, there is a need for context variables that can be efficiently computed from the input data and do not require reward information during inference. Although, the discussed methods increase the generalization, they do not guarantee the robustness in case the inference distribution differs from the training distribution.

D. Uncertainty-based Reinforcement Learning

In order to increase the robustness, highly uncertain predictions need to be classified as OOD in safety-critical applications, as shown in [33]. At the same time, the radar tracker is often limited by the resolution of the radar settings. Thus, there is a limit on the amount of people who can be tracked simultaneously in the scene. In addition, moving disturbances like curtains can be detected as false targets. Consequently, the algorithm needs to detect those scenarios and notify the user that tracking is impossible. Thus, we have a bayesian setting, where we estimate the posterior probability $p(\theta|x)$ on the network parameters θ given the data X. The posterior is proportional to a prior believe on the distributional function $p(\theta)$ and the likelihood $p(x|\theta)$. Further, the posterior probability determines the uncertainty on the prediction.
RL literature proposed several ways to estimate the uncertainty on the neural network parameters: dropout [34] or bootstrap DQN [35] and bootstrap DQN with random prior functions [36]. Dropout applies a random Bernoulli mask on the neural network weights to prevent co-adaptation. In [34], they argue that the dropout rate distribution approximates the posterior distribution. However, dropout is insufficient as a posterior estimation since the dropout rate is not dependent on the data. Hence, this method can not differentiate between data points it has seen once or multiple times, as pointed out in [36]. A more prominent way is the bootstrap method with random prior functions. The bootstrap DQN defines a neural network with K heads that estimate K Q-Values $Q_{i=1,\dots,K}(s_t, a_t)$. The update is computed for every single head with the Temporal Difference (TD) error, shown in Equation 2, with the reward r_t, discount factor γ, state s, action a and time step t.

$$\delta_i = Q_i(s_t, a_t) - (r_t + \gamma \max_a Q_i(s_{t+1}, a)) \tag{2}$$

For each state and action, the variation in the head prediction estimates the uncertainty about the given scene. In order to enhance this variation, a binary mask selects the active heads during training, such that the heads observe different data. In addition, randomized prior functions dictate the network behavior wherever is no training data. The random prior function can be estimated by a non-trainable, random initialized neural network that processes the input data [36].
Since radar tracking is a continuous action space problem, bootstrap DQN is not applicable. However, DQN is a critic-only method, referring to [37]. Hence, we can adapt the bootstrap mechanism with priors to actor-critic methods without loss of generality.

III. Approach

In this section we present an approach for robust radar tracking with meta-RL in combination with OOD detection. In Section III-A, we introduce the signal processing of the radar-based tracking chain and RAI generation. Afterward, we describe context priors in Section III-B and how they encode the complexity of the ongoing RL task. Furthermore, as described in Section III-C, the information is used for meta-RL, which bears the generalization on tasks of different complexity. Finally, the approach is completed by Section III-D, where, using uncertainty, the approach can detect OOD tasks where the tracking is likely to fail.

A. Radar Signal Processing

The received data at each time step is a three-dimensional array of shape (N_C, N_S, N_{rx}), dependent on the number of chirps N_C, the number of samples N_S per chirp, and the number of receiving antennas N_{rx}. Let the axis along the chirps denotes the slow time, and the axis along the number of samples the fast time. We subtract the mean in the fast time to prevent any transmit/receiving antenna leakage. Additionally, we subtract the mean in the slow-time as a high-pass filter to remove completely static target information, e.g., furniture. As the last step, we transform the array to the frequency domain with a Fast-Fourier Transformation (FFT). The range corresponds to frequency along the fast time. Let B be the bandwidth of the transmit signal, T_c the active chirp time, f_s the sampling frequency, and c_0 the speed of light. The maximum range is defined in Equation 3.

$$R_{max} = \frac{f_s}{2}\frac{c_0 T_c}{B} \tag{3}$$

The AoA can be estimated by the phase difference between the minimum two receiving antennas using Digital Beamforming (DBF), which depends on the antenna gain at a given AoA. Here we use the Capon beamformer described in [38]. This paper uses a FMCW radar with one transmit and three receiving antennas in a triangular alignment. Hence two antennas are used for the azimuth AoA estimation. By utilizing a bandwidth of 1 GHz, chirp time of 399 μs, and the sampling frequency of 2 MHz, the maximum detection range is approximately $5m$. As example, we show an RAI with two targets in Figure 2. In our method, the RAI is used as input to the meta-RL algorithm.

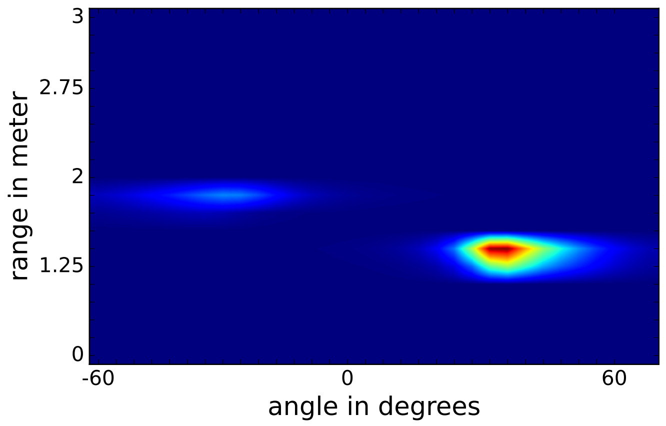
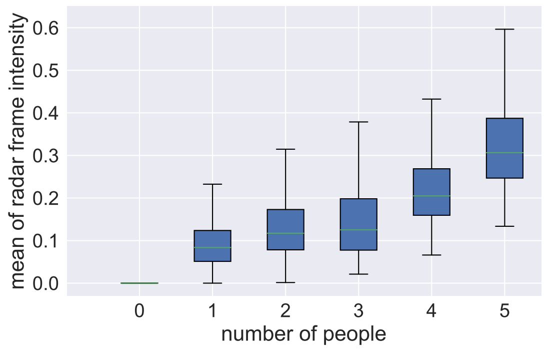
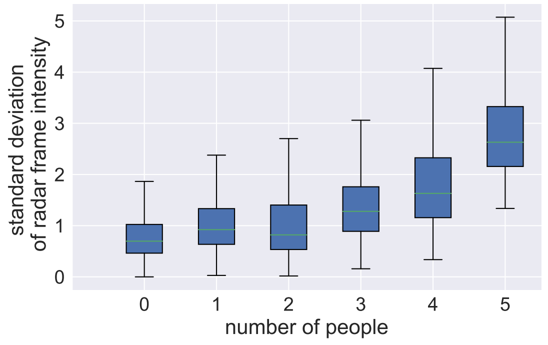


Fig. 2: Example of a Range-Angle Image with two targets. The color encodes energy intensity from high (red) to low (blue).



(a) Distribution of mean Range-Angle Image intensity values with respect to the number of targets.



(b) Standard deviation distribution of Range-Angle Image intensity values with respect to the number of targets.

Fig. 3: Comparison of RAI intensity distributions with respect to the number of targets.

B. Random Priors and Context Variables

In Section II we introduced random priors for uncertainty estimation and context variables for meta-learning. Both are random vectors computed from the input data and serve as additional input to the neural network. However, random priors are computed by a non-trainable and random initialized neural network. In contrast, context variables in [12] are Gaussian random variables computed from task-specific transitions following the context-conditioned policy. In that way, the context variable infers information about the task difficulty from the reward. For radar tracking, the reward is given by the tracking performance, which requires ground truth positions. This information is not available during inference time. However, the average intensity and the variance in the RAI are increasing with the number of people in the current scene, as shown in Figure 3a and Figure 3b. Since the RAI distribution is proportional to the intensity of the tracking targets, we can use the RAI distribution to encode the task difficulty. To this end we use the mean μ_{rai} and standard deviation σ_{rai} of the radar input to define a Gaussian context prior $\mathcal{N}(\mu_{rai}, \sigma_{rai})$.

C. Meta-RL with Context Prior

In this Meta-RL problem, we define different rooms as tasks since the goal is to optimize the tracking parameters independent of the room. In addition, we divide the different rooms into training and test tasks. The test tasks are not observed during training. Moreover, we report the evaluation reward as the average reward over the test tasks, while the Meta-RL algorithm is trained on the train tasks. A detailed description of the training and testing procedure is given in Figure 4. The reward is given in Equation 4, where $\hat{N}$ is the predicted number of targets, N is the true number of targets, $M = \min(\hat{N}, N)$, $\hat{\mu_k}$ is the predicted mean, $\hat{\Sigma_k}$ the predicted covariance and $p_k \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$, as described in [16].

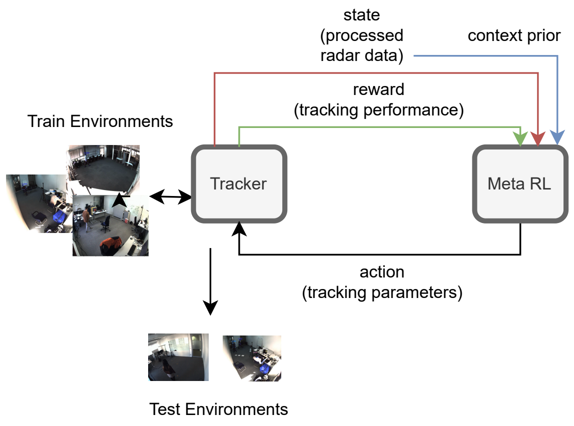


Fig. 4: Overview of the meta-training and evaluation procedure.

$$-R_t = \rho(\hat{N_t}, N_t) + \frac{1}{M}\sum_{k=0}^{M} 1 - p_k(P_k) \quad (4)$$

The first term ρ describes the relative error between the predicted and actual number of tracks. The second term evaluates the likelihood of missing the ground truth positions incorporating the variance of the UKF. For stability reasons, $p(\cdot)$ is clipped to 1.

As baseline RL algorithm we use SAC, a state-of-the-art off-policy actor-critic method from [22].

By design of the context prior, gradient calculation can be omitted and we can use the loss functions for the actor from [22], given in Equation 5 and the critic loss for the individual bootstrap heads in Equation 6.

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D}\left[D_{KL}\left(\pi(\cdot|\mathbf{s}_t)\middle\|\frac{exp(Q_\theta(\mathbf{s}_t,\cdot))}{Z_\theta(\mathbf{s}_t)}\right)\right] \quad (5)$$

$$J_Q^i(\theta) = \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{r},\mathbf{s}`)\sim D}\left[Q_\theta^i(\mathbf{s},\mathbf{a}) - (\mathbf{r}(\mathbf{s},\mathbf{a}) + \bar{Q}^i(\mathbf{s'a}))\right]^2 \quad (6)$$

Furthermore, the critic uses the bootstrap mechanism. To this end, we define a Base Network that computes an embedding vector x from the state action pairs. During training, we compute a binary mask in every epoch, determining whether the parameters for specific heads are updated. For every active head, we sample a context vector from the context prior distribution and add it to x. Afterward, every head predicts a Q-Value $Q^i(s,a)$. In detail, we describe the bootstrap critic algorithm in Algorithm 1. We apply the exact prior mechanism for the actor network that predicts the actions from the current state. A detailed description of the used networks is given in Figure 5.

For meta-learning, we update the parameters with the accumulated losses from every training task, as shown in [12].

Algorithm 1 Bootstrap Critic with Context Prior

```
Require: Bootstrap heads $\{H^i\}_{i=1,...,N}$, Buffer $B$, learning rate $\alpha$
 1: Initialize $BaseNetwork$ parameters $\theta$
 2: Initialize Head parameters $\delta_i$
 3: while not done do
 4:     compute active head mask: $m_i \sim \mathcal{B}(0,1)$
 5:     active Heads: $H_m^i$
 6:     $(s_t, a_t, r_t, s_{t+1})$ from $B$
 7:     $\mu = mean(state)$
 8:     $\sigma = std(state)$
 9:     context prior: $\mathcal{N}(\mu, \sigma)$
10:     for head in $H_m^i$ do
11:         $x = BaseNetwork(s_t, a_t)$
12:         $z_i \sim \mathcal{N}(\mu, \sigma)$
13:         $Q_i = head(x + z_i)$
14:         $\delta_i = \delta_i - \alpha\nabla_{\delta_i} J_Q^i$
15:     end for
16:     $\theta = \theta - \alpha\frac{1}{M}\sum_i^M \nabla_\theta J_Q^i$
17: end while
```

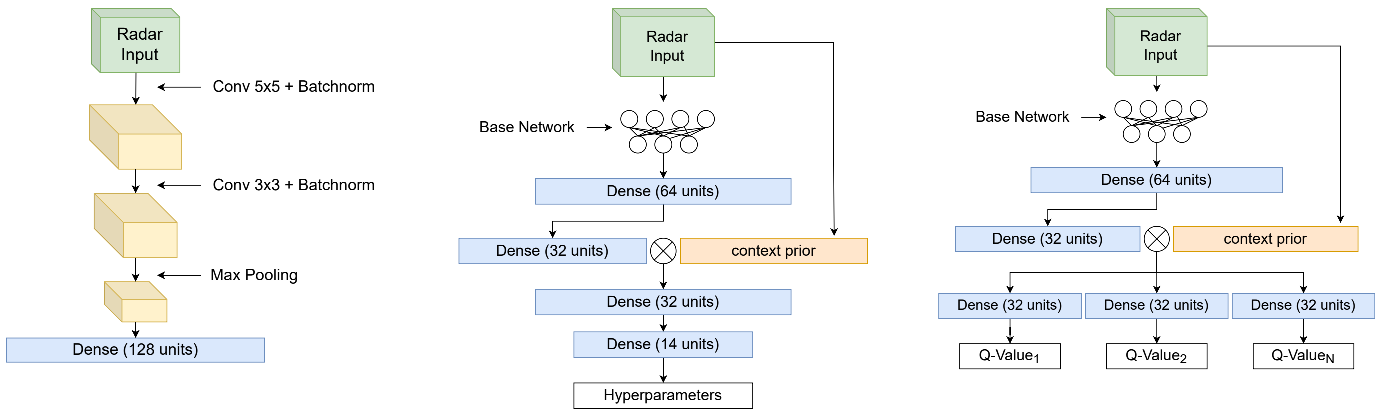
Algorithm 2 Meta-Training

```
Require: training tasks $\{T^i\}_{i=1,...,T}$
 1: Initialize parameters $\theta$, $\phi$
 2: Initialize buffer for each task $B_i$
 3: learning rates $\alpha_1$ , $\alpha_2$
 4: while not done do
 5:     for task in $T^i$ do
 6:         rollout $\pi(\theta|s)$ and store transitions in $B_i$
 7:         sample batch $b^i \sim B_i$
 8:         compute $\mathcal{L}^i_{boot-critic}(b^i)$
 9:         compute $\mathcal{L}^i_{actor}(b^i)$
10:     end for
11:     $\theta = \theta - \alpha_1\nabla_\theta\sum_{i=1}^N \mathcal{L}^i_{boot-critic}$
12:     $\phi = \phi - \alpha_2\nabla_\phi\sum_{i=1}^N \mathcal{L}^i_{actor}$
13: end while
```

D. *Out-of Distribution Detection*

In literature, OOD approaches aim to detect whether an environment has not been seen yet. The challenge in our setup is, for new scenarios, to generalize well and detect low-performance simultaneously.

We apply the bootstrap mechanism for the critic for OOD, as shown in [39]. The critic aims to predict the future reward,



(a) Base Network (b) Actor Network (c) Multi-Head Critic Network

Fig. 5: Neural Network designs for actor-critic RL with context prior

and the variation along multiple predictions is a measure of uncertainty. As we see from Figure 3, a huge variance in energy peaks between two detected targets in a radar image, is an example of increased task complexity. Thus, our proposed context prior emphasizes higher variation for a more difficult task. The training procedure of the bootstrap critic is shown in Algorithm 1. In order to detect OOD samples during inference, we define a threshold c as given in Equation 7,

$$c = \mu_{head} + \alpha\sigma_{head} \tag{7}$$

where μ_{head} is the prediction mean of all heads, σ_{head} is the standard deviation of the predictions and α is a hyperparameter that determines the size of the OOD detection interval.

IV. Results and Discussion

In this section, we present the detailed implementation settings for the experiments, starting with the used hardware and software tools, followed by the dataset specifications. The results for Meta-RL are presented in Section IV-B and the results for OOD detection are shown in Section IV-C.

A. Implementation Settings and Dataset

In the implementation, we used Tensorflow™- GPU v2.9.0 with CUDA® Toolkit v11.2.0 and cuDNN v8.1.0. As a processing unit, we used the Nvidia® Tesla® P40 GPU, Intel® Core i9-9900K CPU, and DIMM 16GB DDR4-3200 module of RAM. As a radar sensor for data recordings, we used three of Infineon Technologies XENSIV™ 60 GHz BGT60TR13C FMCW radar and three cameras for labeling purposes. The recorded dataset includes 200.000 radar frames from five different rooms, including human activities from zero to five people, divided into recordings of 350 frames. The radar data has been recorded with a frame rate of 10 Hz. We use Yolo-v5 [1], an object detection framework from Ultralytics®, with the three cameras to gather ground truth positions for the recordings.
For meta-training, we take recordings from three rooms and use the other two rooms for the evaluation. The number of frames per environment is evenly split for the number of targets.

[1]https://github.com/ultralytics/yolov5

B. Domain Generalization

The current policy obtains the Meta-RL results in two unknown test environments. We report the average evaluation reward over the test environments after each iteration. The optimal reward is 0, and a higher reward indicates better tracking performance. The reward formulation is given in Equation 4 and leverages the distance between predictions and ground truth and the number of false targets. As a baseline, we use the performance of the tracker using fixed hyperparameters determined by an expert user. In addition, we compare our method against MAML with the formulation from [32] and Reptile. In Figure 6, we show that our proposed method improves 35% over the baseline and 16% over the comparable Meta-RL methods. In addition, our proposed method explores more efficiently and is improving while MAML and Reptile saturate.

The effect of adaptive parameter selection, where the reward focuses on the distance and false targets, is depicted in Figure 7. There we see that our approach can estimate the correct number of tracks and is more accurate with a Root Mean Squared Error (RMSE) of 1.45 compared to the baseline (RMSE of 1.94) in estimating the target's angular position (x-position). To further gain insights on meta-learning, we scale the reward during training to observe the impact on the environment generalization. Since the reward signal is used to improve tracking, and the context prior is used for environment generalization, weighting the reward higher than the context prior should decrease the generalization capabilities. We report the peak performance of the Meta-RL approach in the test environments with different reward scaling factors during training n Table I. There we show that scaling the reward by two results in the highest average evaluation reward. Further we see that scaling the reward far off the context prior leads to worse evaluation performance, which supports our hypothesis.

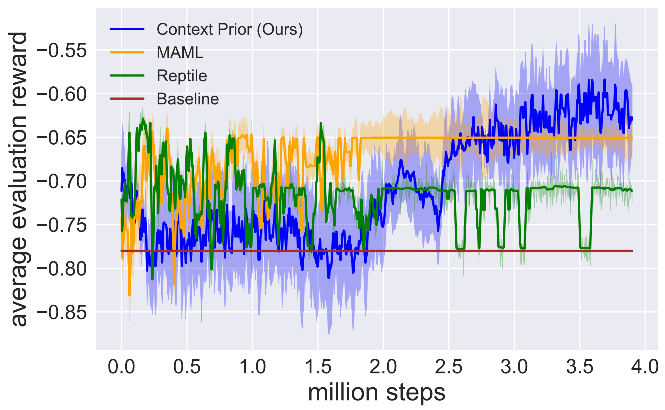


Fig. 6: Comparison of the context prior (blue) against MAML (orange) and Reptile (green). The context prior reaches higher peak performance while MAML and Reptile saturate to almost static parameters.

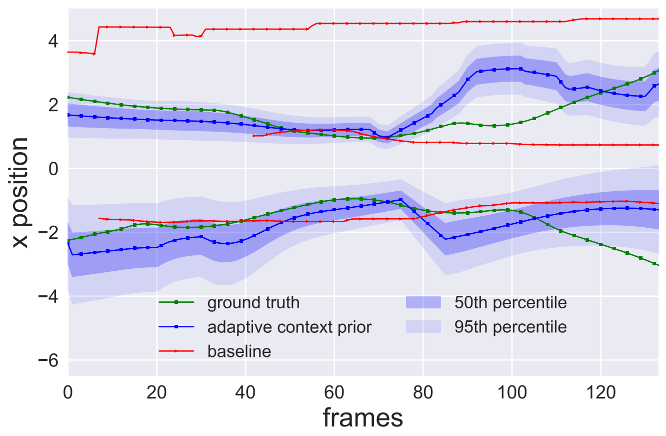


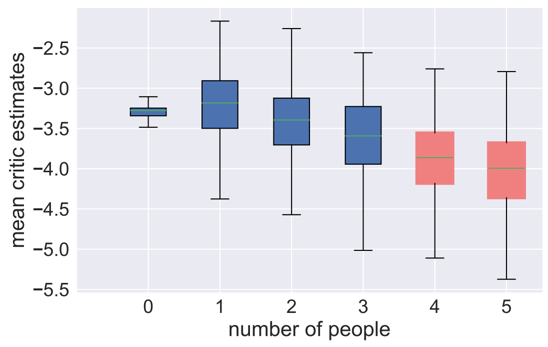
Fig. 7: Comparison of our adaptive approach with the baseline for estimating the angular direction (x-position in Cartesian coordinates). The adaptive approach predicts the correct number of tracks while resulting more accurate in estimating the position with an RMSE of 1.45 incorporating the predicted variance of the UKF prediction.

C. *Uncertainty-based OOD Detection*

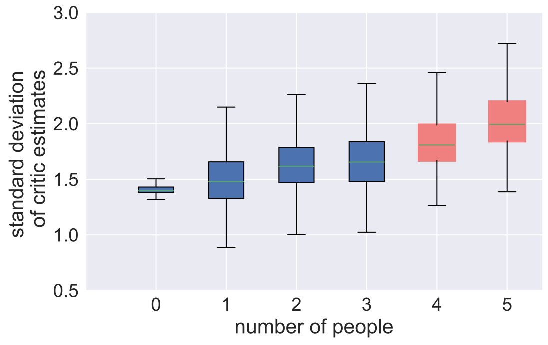
With our defined setup, the expected limit of the tracking system is up to three targets. Hence, we aim to predict scenes with more than three targets as OOD. In Figure 8 we show the distributions of the critic predictions. Both mean and standard deviation are proportional to the number of people in the scene. Since we have fewer OOD scenarios (four and five targets) than training scenarios (zero to three targets), we report the F1-Score, which considers data imbalance. With the threshold defined in Equation 7, α determines the size of the detection interval. Thus, choosing α very small focuses on accurately predicting all the OOD scenarios and vice versa, leveraging precision and recall. With $\alpha = 0.17$ we reach a maximum F1-Score of 72%.

Reward scale factor	Best Reward
1	-0.58
2	**-0.51**
5	-0.61
10	-0.68

TABLE I: Study about the effect of reward scaling on the average test evaluation reward.



(a) Mean Distribution of critic estimates on test environments.



(b) Standard deviation Distribution of critic estimates on test environments.

Fig. 8: Distributions of critic estimates on test environments. The OOD scenarios are highlighted in red.

V. Conclusion

This paper presents an approach that utilizes context priors for an uncertainty-based Meta-RL. This is used for domain generalization and OOD detection. In order to assess the performance of our contribution, we benchmark it on a radar-tracking dataset towards related Meta-RL algorithms. In a set of multi-target radar-tracking scenarios, the proposed method outperforms related Meta-RL approaches in peak performance by 16% and the baseline by 35% while detecting OOD data with an F1-Score of 72%. This shows that our method is more robust to environmental changes and well-addresses the OOD tracking scenarios. In future work, we want to expand the Meta-RL framework to different radar positions, radar devices, and internal settings (e.g., bandwidth, sampling frequency).

References

[1] L. Servadei, H. Sun, J. Ott, M. Stephan, S. Hazra, T. Stadelmayer, D. S. Lopera, R. Wille, and A. Santra, "Label-aware ranked loss for robust people counting using automotive in-cabin radar," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3883–3887.

[2] J. F. Tilly, S. Haag, O. Schumann, F. Weishaupt, B. Duraisamy, J. Dickmann, and M. Fritzsche, "Detection and tracking on automotive radar data with deep learning," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE, 2020, pp. 1–7.

[3] M. N. Woodland, S. B. Kelly, C. Abell, C. Lindsay, C. S. Joseph, and R. H. Giles, "Millimeter-wave radar patient motion tracking technologies for medical imaging applications," in *Terahertz, RF, Millimeter, and Submillimeter-Wave Technology and Applications XV*. SPIE, 2022, p. PC120000D.

[4] K. Ramachandra, *Kalman filtering techniques for radar tracking*. CRC Press, 2018.

[5] S. J. Julier, "The scaled unscented transformation," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 6. IEEE, 2002, pp. 4555–4559.

[6] G. Revach, N. Shlezinger, R. J. Van Sloun, and Y. C. Eldar, "Kalmannet: Data-driven kalman filtering," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3905–3909.

[7] N. Wang, Y. Gao, H. Zhao, and C. K. Ahn, "Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 3034–3045, 2020.

[8] H. Li, Y. Wu, and M. Chen, "Adaptive fault-tolerant tracking control for discrete-time multiagent systems via reinforcement learning algorithm," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1163–1174, 2020.

[9] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[10] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

[11] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *International Conference on Learning Representations*, 2018.

[12] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 5331–5340. [Online]. Available: https://proceedings.mlr.press/v97/rakelly19a.html

[13] A. Santra and S. Hazra, *Deep Learning Applications of Short-Range Radars*, ser. Artech House radar library. Artech House, 2020. [Online]. Available: https://books.google.de/books?id=DjxgzgEACAAJ

[14] H. Rohling, "Radar cfar thresholding in clutter and multiple target situations," *IEEE transactions on aerospace and electronic systems*, no. 4, pp. 608–621, 1983.

[15] J. Shen, Y. Liu, S. Wang, and Z. Sun, "Evaluation of unscented kalman filter and extended kalman filter for radar tracking data filtering," in *2014 European Modelling Symposium*. IEEE, 2014, pp. 190–194.

[16] M. Stephan, L. Servadei, J. Arjona-Medina, A. Santra, R. Wille, and G. Fischer, "Scene-adaptive radar tracking with deep reinforcement learning," *Machine Learning with Applications*, vol. 8, p. 100284, 03 2022.

[17] E. Pashenkova, I. Rish, and R. Dechter, "Value iteration and policy iteration algorithms for markov decision problem," in *AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning*. Citeseer, 1996.

[18] Y. Lu and J. Lu, "A universal approximation theorem of deep neural networks for expressing probability distributions," *Advances in neural information processing systems*, vol. 33, pp. 3094–3105, 2020.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[20] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[23] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Learning from demonstrations for real world reinforcement learning," *CoRR*, 2017.

[24] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[25] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[26] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[27] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," *arXiv preprint arXiv:1803.00676*, 2018.

[28] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[29] T. Nam, S.-H. Sun, K. Pertsch, S. J. Hwang, and J. J. Lim, "Skill-based meta-reinforcement learning," *arXiv preprint arXiv:2204.11828*, 2022.

[30] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[32] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[33] A. G. Pacheco, C. S. Sastry, T. Trappenberg, S. Oore, and R. A. Krohling, "On out-Of-Distribution detection Algorithms with Deep Neural Skin Cancer Classifiers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 732–733.

[34] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[35] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.

[36] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[37] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[38] P. Stoica, Z. Wang, and J. Li, "Robust capon beamforming," *IEEE Signal Processing Letters*, vol. 10, no. 6, pp. 172–175, 2003.

[39] A. Sedlmeier, T. Gabor, T. Phan, L. Belzner, and C. Linnhoff-Popien, "Uncertainty-based out-of-distribution classification in deep reinforcement learning," *arXiv preprint arXiv:2001.00496*, 2019.