Object Localization Algorithm Based on Meta-Reinforcement Learning

Han Yan
Xi'an Technological University
School of Computer Science & Engineering
Xi'an, China
E-mail: 18713877573@163.com

Hong Jiang
Xi'an Technological University
School of Computer Science & Engineering
Xi'an, China
E-mail: 249479898@qq.com

***Abstract*—When the target localization algorithm based on reinforcement learning is trained on few-sample data sets, the accuracy of target localization is low due to the low degree of fitting. Therefore, on the basis of deep reinforcement learning target localization algorithm, this paper proposes a target localization algorithm based on meta-reinforcement learning. Firstly, during the initial training of the model, the meta-parameters were classified and stored according to the similarity of the training tasks. Then, for the new target location task, the task feature extraction was carried out and the meta parameters with the highest similarity were matched as the initial parameters of the model training. The model dynamically updated the meta parameter pool to ensure that the optimal meta parameters of multiple different types of features were saved in the meta parameter pool, so as to improve the generalization ability and recognition accuracy of multiple types of target location tasks. Experimental results show that in a variety of single target localization tasks, compared with the original reinforcement learning target localization algorithm, under the same data set size, the model converges under a small number of training steps with the meta-parameters in the matching meta-parameter pool as the initial training parameters. Moreover, the training speed of the meta-reinforcement learning method based on MAML-RL is increased by 28.2% for random initial parameters, and that of the meta-reinforcement learning method based on this paper is increased by 34.9%, indicating that the proposed algorithm effectively improves the training speed, generalization performance and localization accuracy of object detection.**

Keywords-Meta-reinforcement Learning; Meta-Parameter; Target; Generalization Ability; Deep Reinforcement Learning

I. INTRODUCTION

Humans can quickly find a new object in the field of vision without too much complicated process, because humans have mastered the ability to learn quickly. This is very difficult for computers, especially for the object localization process, which often requires a large number of datasets and computational costs for training new tasks. This leads to a low degree of model fit and accuracy of target localization for real-world few-sample data. It is particularly important to improve the convergence speed of the model for new tasks by storing and learning the model's historical experience.

With the introduction of reinforcement learning, the accuracy of target positioning has been improved to a certain extent [1], and the typical algorithms are RAM [2] and UR-DRQN positioning models [3]. This kind of algorithm regards the process of target localization as the process of Agent constantly interacting with the task environment, getting positive and negative rewards to update the model parameters, and finally locating the target. Such algorithms need to train agents according to different task objectives and require a large amount of labeled data, so they will have the problems of low fitting degree and slow convergence speed when facing few-sample tasks [4].

Meta reinforcement learning is an important field of machine learning research. It is a method that enables agents to learn and converge quickly when facing new tasks by training a global optimal parameter as the initial parameters of model

retraining. Among them, MAML-RL [5] is the most classical method in meta-reinforcement learning. Its core idea is to optimize the model through multiple kinds of tasks to train an initial parameter, so that the model can quickly converge on a new task with only a small number of samples or a few gradient updates. The existing target localization algorithms based on meta-reinforcement learning improve the generalization ability of the model by training the optimal parameters of the task, but with the improvement of the task type, its learning ability will decrease. In view of this, this paper sets up a memory storage module to save the training parameters of historical tasks according to similarity, and conducts meta-learning operations on them. It can significantly improve the convergence speed of the model for new tasks and alleviate the problem of reduced learning ability of the model.

II. RELATE WORKS

Meta-reinforcement learning has achieved great success on many complex and high-dimensional tasks [6]. Although reinforcement learning provides a new solution for object localization algorithms, it mainly focuses on the localization efficiency under a certain task, rather than generalization in multiple scenarios and rapid adaptation to few-shot tasks. The meta-reinforcement learning method effectively learns new tasks through agent learning in reinforcement learning environment t [7-8]. Existing meta-reinforcement learning methods mainly focus on model-free methods [9-10]. These algorithms tend to have more complex training pipelines than non-meta reinforcement learning methods, making it difficult to apply to real-world applications. Moreover, the existing model-free methods [11] tend to ignore the attenuation of learning ability for new tasks, which reduces their training efficiency when the types of tasks increase.

Some meta-reinforcement learning designers improve the learning ability of the model by designing the architecture of the model or designing new optimization algorithms and update rules. The typical meta-reinforcement learning algorithm MAML-RL [5] obtains a set of initial parameters of the model through training, so that the model can maximize the performance of a new task by only one or a few gradient updates on a small number of samples. On this basis, a storage and replay memory pool is designed to classify and update the meta-reinforcement learning parameters according to tasks, so that the parameters in the memory pool have the maximum generalization performance within the range of task types. In addition, our method allows the model to match the appropriate historical memory according to the new task, and allows the model to automatically adapt to the leap of task types with large differences, thereby reducing the amount of data required for the model to learn new tasks.

III. MODEL

In the target localization algorithm of reinforcement learning, a large number of data sets are usually required for training. However, in real life, there are many kinds of tasks with few samples, for which the bottleneck of localization accuracy is easy to be reached [12]. In object localization algorithms based on meta-reinforcement learning, a set of initial parameters that can converge quickly on new tasks is trained by learning the commonality of task types. This paper combines meta-reinforcement learning on the target localization framework based on reinforcement learning, and learns the optimal parameters of tasks with high similarity by setting a storage mechanism.

Figure 1 shows the framework diagram of the proposed algorithm model, which is mainly composed of three parts: the target localization module, the feature mapping module and the meta-parameter pool module. In the feature mapping module, the improved VGG-16 [13] network extracts the features of the task, and classifies and maps them into the corresponding feature space. The model first uses the Training Data set to train in the reinforcement learning target localization model, records the convergence parameters and loss gradient of each task type, and updates the gradient of the meta-parameters after the training of each task type. The updated model parameters are stored in the meta-parameter pool module according to the mapping area of the feature mapping module. The parameters in the meta-parameter pool are updated by using the meta-parameter update function, and the updated

parameter a $Meat_{\theta}^{i}$ shows the global optimum in the feature region i.

The purpose of meta-reinforcement learning is to make the model learn the commonality under multiple task types, and then master a learning ability to quickly converge under new tasks. For few-shot data in reality, the proposed model uses the feature mapping module to match the meta-parameters in the meta-parameter pool as the initial training parameters. The meta-parameters preserve the historical exploration experience of the model, and the gradient correction of the few sample data can have better positioning accuracy for the new target.

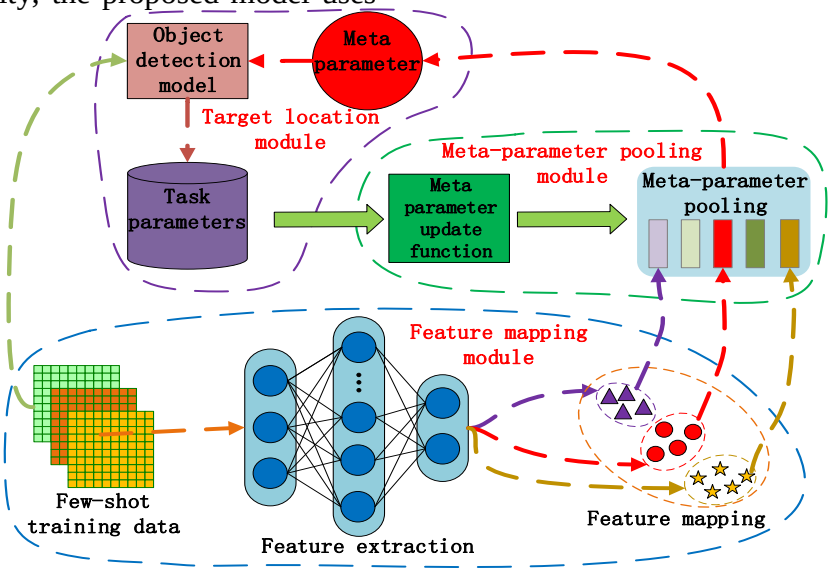


Figure 1. Process of target localization in meta-reinforcement learning

A. *Target positioning module*

As shown in Figure 2, this paper uses the reinforcement learning target localization model with joint action network and regression network as the task training model of the model. It mainly consists of three parts: feature extraction network, action network and regression network. The feature extraction network is the improved GAP-VGG16 network. The model matches the task to the feature space corresponding to the meta-parameter pool according to the feature values extracted by the feature extraction network, and stores the updated parameters after completing a batch of training. At the same time, the feature vector extracted by the feature is fused with the memory vector and sent to the action network. The action network is responsible for taking adjustment actions according to the current environment state, until the stop action is generated, the regression network is used for regression operation, and the final positioning result is output.

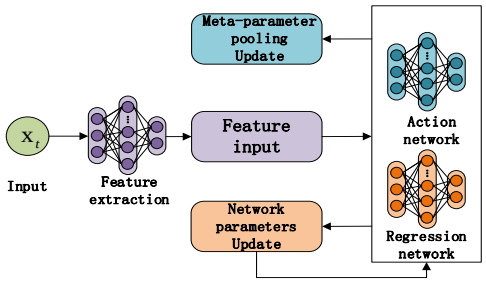


Figure 2. Object localization model

For the few-shot object localization task, the model extracts a few samples from the task for feature extraction, and matches the meta parameters with the largest similarity from the meta parameter pool according to the feature value type extracted by the task as the initial parameters of the few-shot object localization task for retraining. The parameters of the retrained model are updated in the corresponding meta-parameter pool to retain the newly learned task memory. Through the method of this paper, the model has a

certain ability to learn the task at the beginning, avoiding the agent to explore in a completely unfamiliar environment. The model regards each image input as a reinforcement learning environment, and selects the exploration action according to the fusion state of the input and historical exploration. The model gives feedback according to the pre-designed reward function to judge the quality of the model action selection, and updates the network parameters of the model through the process of circulation to improve the accuracy of target localization. The detailed design of the model states, actions, and rewards is as follows.

B. State

The process of human searching for the target is not only related to the current visual field, but also involves the memory of the past historical exploration in the brain. Human beings realize the accurate recognition of the target by combining the brain memory with the current visual field. The state S of this paper is the procedural simulation of this process, which is represented by a tuple $s_t = (o_t, h_t)$ related to time t, which represents the fusion information of o_t and h_t, and the agent makes the next action selection according to this fusion state s_t.

C. Actions

The action taken by the Agent acts on the adjustment of the candidate box, which is divided into horizontal movement (left and right), vertical movement (up and down), scale transformation (horizontal expansion, horizontal reduction, vertical expansion, vertical reduction), and stop action). Each action is adjusted discretized according to the multiple of. Among them, the output termination action indicates that the target is in the field of view of the agent. The specific classification of actions is shown in Figure 3.

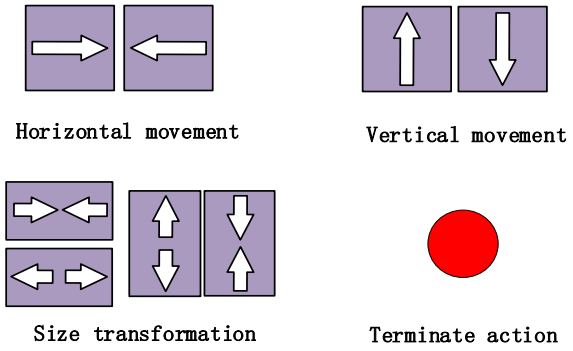


Figure 3. Action diagram

The model selects the actions of the agent through the DQN network, and uses $\varepsilon-greedy$ [15] strategy to make the agent explore new actions [16], so as to ensure that the agent takes the optimal action under long-term exploration. $\varepsilon-greedy$ strategy is shown in (1).

$$\pi(a|s) \leftarrow \begin{cases} 1-\varepsilon+\dfrac{\varepsilon}{|A(s)|}, if & \arg\max_a Q(s,a) \\ \dfrac{\varepsilon}{|A(s)|}, if & \arg\max_a Q(s,a) \end{cases} \quad (1)$$

Where s is the current state the Agent is in, and a is the action taken by the Agent based on the current state. A(s) is the set of actions that the Agent can choose at states, and |A(s)| denotes the number of actions that can be chosen. $\varepsilon \in [0,1]$ Is the exploration factor, and $\pi(a|s)$ is a policy, which represents the probability distribution of possible actions taken by the agent at a given state s. For the state at a given time in the policy, the agent selects the action corresponding to the output with the maximum probability to adjust the attention field.

D. Rewards

The good or bad of the action taken by the agent can be intuitively seen through the reward function, and in this paper, the reward function is set by the IOU change of the attention field after the state change s after the agent takes the action. As shown in (2), where b is denoted as the visible area of the Agent and g is the real labeled area of the target object.

$$IOU(b,g)=\frac{area(b\cap g)}{area(b\cup g)} \quad (2)$$

At each time step, after taking an action, the agent will obtain a new visual area. By calculating IOU between this area and the real area, the reward value of the agent's state change after taking an action can be obtained, which is defined by the reward function shown in (3).

$$R_a(a,s\rightarrow s')=sign(IOU(b',g)-IOU(b,g)) \quad (3)$$

This function indicates that after the agent's state has changed. If the value of IOU increases upward, it means that the agent has obtained positive feedback, and the model will store this state and action tuple (s,a,r,s',b,g) as experience in the experience pool, which is used as a reference for the agent to explore the target position. On the contrary, if the IOU decreases after the state change, it indicates that the action is poor and negative feedback is obtained. For the determination of stop action, when the IOU value rises above 0.6 after the agent makes an action, it is determined that the target is in the field of view of the agent, and the stop action is taken, and the regression network is selected to take a smaller step to fine-tune the field of view frame. The reward function for the stop action is given in (4).

$$R_t(s\rightarrow s')=\begin{cases}+\eta & if\ IOU(b,g)\geq\tau\\ -\eta & otherwise\end{cases} \quad (4)$$

In order to ensure the training efficiency, when the IOU of agent reaches 40 steps, it is determined that the exploration fails, and the regression network is not used for fine-tuning. (7) is used to update the parameters of the two networks.

E. Action Network Structure

The action network consists of two parallel fully connected networks with the same structure and different parameters. One produces the "predicted value" and the other produces the "target value". In the training phase, the "target value" is calculated to assist the learning of network parameters. In the testing phase, the "target value" is not calculated, but when the fusion information ht is received, a random action is selected with the help of the ε-greedy strategy with probability ε, as shown in Figure 4:

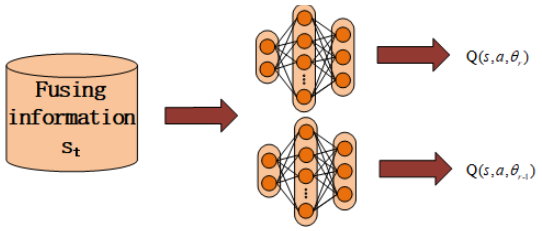


Figure 4. Structure of the location network

F. Regression network structure

The regression network is a fully connected network $fc(128*128*4)$. When the termination action is generated during the learning of an epoch, and the IoU between the visible area and the real marked area is greater than 0.6, the network will fine-tune the coordinates of the current visible area to obtain the offset that needs to be adjusted in the corresponding direction of the bounding box $(\nabla X,\nabla Y,\nabla W,\nabla H)$, as shown in Figure 5:

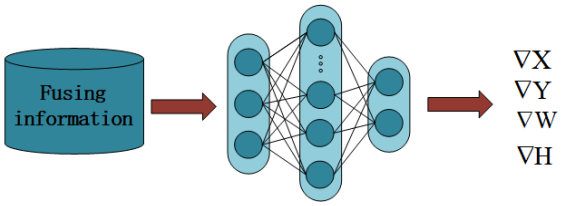


Figure 5. Structure of the regression network

G. Feature space mapping

The tasks of the training set are mapped to different positions in the feature space by means of feature mapping for the generation of meta-parameters. Due to the problem of parameter redundancy and computational complexity in the original VGG-16 network, in this paper, the original fully connected layer is replaced by the global average pooling layer, and GAP-VGG16 is constructed as the feature extraction network, as shown in Figure 6. The feature types of each task can be obtained through feature extraction, and the number of feature types is controlled by specifying the range of mapping (set to 10 in this paper), and each feature range corresponds to the storage space in a meta-parameter pool.

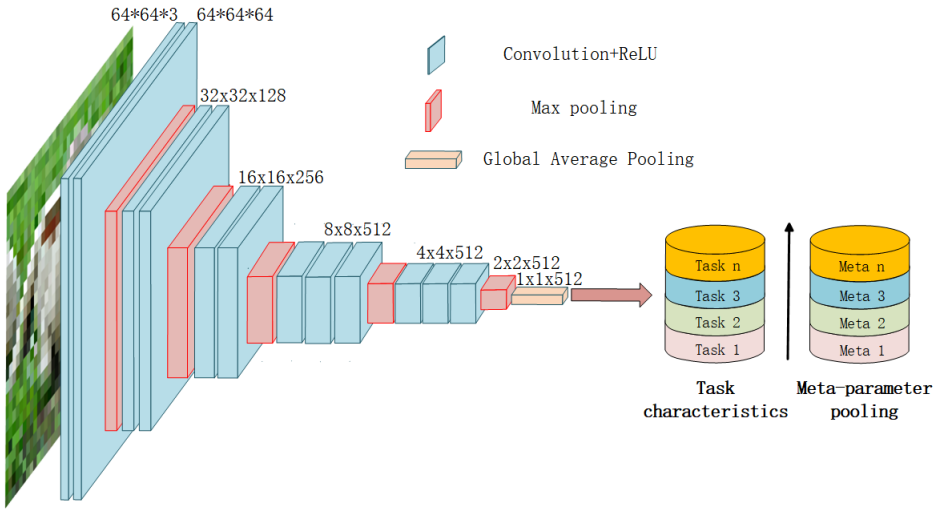


Figure 6. Feature network structure

The feature mapping result of few-shot task is vector form a_i . For each meta-parameter $Meta_\theta$ obtained by training, there is a feature vector b_i corresponding to it, and the corresponding meta-parameter $Meta_\theta$ is matched by measuring the difference $d(a_i,b_i)$ between vectors a_i and b_i .In this paper, the Euclidean distance between two vectors is used to judge the mapping space region, as well as the degree of similarity between tasks. The calculation formula is given in (5).

$$d(a_i,b_i)=\sqrt{\sum_{i=1}^{n}(a_i-b_i)} \tag{5}$$

In this paper, the feature vector corresponding to the task of the first meta-parameter of model training is used as the benchmark vector, as the label of 1 meta-parameter stored in the meta-parameter pool position, the difference value κ is set, and the multiple of the difference value κ is set to select the meta-parameters in the model meta-parameter pool, that is, the feature extraction of a few-sample training task is carried out. The similarity between the extracted feature vector and the feature vector label corresponding to the first position of the meta-parameter pool is calculated, and the matched meta-parameter position is obtained by dividing it with κ and adding 1. As shown in (6).

$$meta_{\theta_i}=\frac{d(a_i,b_1)}{\kappa}+1 \tag{6}$$

Here, $Meta_{\theta i}$ represents the initial meta-parameters matched during retraining of the i^{th} few-shot dataset, and b_1 represents the feature vector corresponding to the meta-parameter at the first position of the meta-parameter pool.

In order to avoid forgetting the historical tasks of the meta-parameters, this paper uses (7) to update the meta-parameters, and retains the previous memory at each update of the meta-parameters. As shown in (7).

$$Meta_{\theta n}=Meta_{\theta n}(1-n\lambda)+\lambda(\theta_1+\theta_2+....\theta_i) \tag{7}$$

IV. MODEL TRAINING

The whole training process of the model includes the training of the parameters θ_t of the inner recurrent network in the meta-reinforcement learning process, and the update of the meta-parameter $Meta_\theta$ in the meta-parameter pool. In the outer loop, the model updates the meta-parameter pool according to the feature mapping region of the task. The same network architecture is used to update the action network and the regression network in the inner loop.

A. Meta-parameter pool training

The meta-parameter pool O stores the meta-parameters of N kinds of tasks and updates them. For the new training task, the region Oi in the meta-parameter pool is matched according to the way of feature mapping, and the corresponding

meta-parameters Meta_{θ} are selected as the initial parameters for retraining. The retrained meta-parameters allow the retention of the previous memory, and the trained meta-parameter pool maintains the optimal loss value for tasks of the same task type. The update process of the meta-parameter pool is shown in Figure 7.

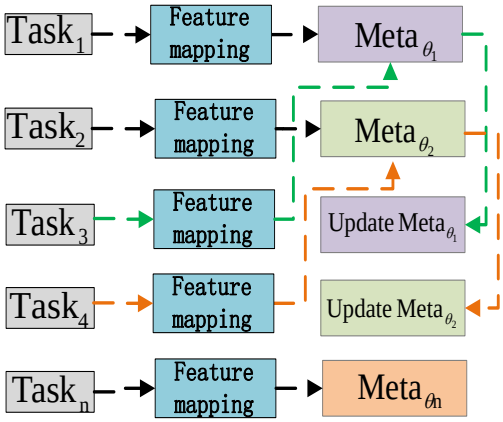


Figure 7. Training process of meta-parameter pooling

The meta-parameters in the meta-parameter pool are updated by continuous learning, and (7) is used to preserve the historical memory when the agent is updated. Where θ_i represents the meta-parameter after the i[th] update and represents the learning coefficient, which is used to prevent the model distortion caused by too large parameter changes.

B. Training of target localization parameters

The parameters of the target localization model include the parameters of the action network and the regression network, namely $\theta = (\theta_a, \theta_g)$. The historical experience of the action network is represented by the tuple (s, a, r, s', b, g), as shown in Figure 3. There are multiple exploration tasks in the same environment, and each exploration task will generate an MDP sequence. Expressed as the strategy of the agent, the loss function of each task Ti is shown in (8).

$$\theta_i^* = \theta - \alpha \nabla L_{Ti}(f_\theta) \tag{8}$$

C. Loss function

The comprehensive loss of the target localization model includes the loss of the action network and the loss of the regression network and (7) is the loss function. The weighted sum of the losses of the action network and the regression network is used as the comprehensive loss of the target localization model. The mean square error loss function is used for the action network and the $\mathrm{smoothL}_1$ loss function is used for the regression network. The overall loss function is defined as in (9) - (11).

$$L(s, a, t) = L_{action} + \lambda L_{reg} \tag{9}$$

Among them,

$$L_{\mathrm{action}} = \frac{1}{N_{\mathrm{action}}} \sum [(y_i - Q(s, a; \theta_i))^2] \tag{10}$$

$$L_{\mathrm{reg}} = \frac{1}{N_{reg}} S(t_i - t_i^*) \tag{11}$$

Where N_{action} and N_{reg} are the number of execution steps of the action network and the number of execution steps of the regression network, and their losses are averaged as the loss of the exploration action and the regression action. $Q(s, a; \theta_i)$ is the predicted value derived from the "prediction branch" of the action network, and y_i is the target value derived from the "target branch" of the action network. In the regression network loss L_{reg}, $t_j = \{t_x, t_y, t_h, t_w\}$ denotes a vector of dimension size 4. Here, $\sum L_{Ti}(f_\Phi)$ and t_y are the center coordinates obtained by the regression network, respectively, and t_h, t_w are their corresponding heights and widths. t_x^* and t_y^* are the center coordinates of the true labeled regions of the regression network, respectively, and t_h^*, t_w^* are the corresponding heights and widths. Where S is the $\mathrm{smoothL}_1$ function, see (12).

$$soomthL_1(x) = \begin{cases} 0.5x^2, if \ |x| < 1 \\ x - 0.5, otherwise \end{cases} \tag{12}$$

a is the loss balance coefficient, which is used to balance the loss of the action network and the regression network, and s stands for the regression loss.

D. Meta-reinforcement Learning parameter training

Each RL goal localization task T_i contains an initial state distribution L_{T_i} and transition distribution $q_i(x_{t+1}|x_t,a_t)$, and the loss L_{T_i} corresponds to the negative reward function R. In this paper, we treat each object localization task as a Markov Decision Process (MDP) with an initial state S0, which allows the agent to retrace historical exploration trajectories and perform learning across tasks T_i the model is defined with respect to task T_i and loss $L_{Ti}(f_\varphi)$ as shown in (13).

$$L_{Ti}(f_\varphi) = -E_{x_t,a_t\sim f_\varphi,\ q_{Ti}}[\sum_{t=1}^{H} R_i(x_i,a_t)] \quad (13)$$

For each of the k tasks Task1-k in the meta-parameter pool O_i, the model generates H exploration actions (x1,a1,... xH,aH) and the corresponding loss $L_{Ti}(f)$, K accumulated losses $\sum L_{Ti}(f_\Phi)$ generated for the model are used for the meta-parameter $Meat_\theta(T_{1\text{-}k})$ corresponding to the task type in the meta-parameter pool O_i region, see (14).

$$Meta_\theta = \theta\text{-}\beta\nabla_\theta \sum_{T_{1-k}\sim p(T)} L_{T_i}(f_{\theta_i'}) \quad (14)$$

Using meta-parameters when training on a new task leads to convergence in fewer exploration steps. In this paper, meta-reinforcement learning includes two parts *InnerLoop* and *outerLoop*, and the agent uses network random parameters as initialization parameters in each task pair. In *InnerLoop*, the agent continuously explores the target to generate an *MDP* -sequence, which converges to θ_i^* through multiple training gradient updates. The outer loop of s is classified according to the feature types of the training tasks, and the corresponding meta-gradients of s are cumulatively updated to obtain the meta-parameters of multiple task types, that is, the meta-parameter $Meta_\theta^i$ maximizes the sum reward of multiple task rewards. The trained meta-parameter $Meta_\theta^i$ is stored in the meta-parameter pool module, waiting for matching and updating during retraining. Figure 8 shows the parameter training process of meta-reinforcement learning.

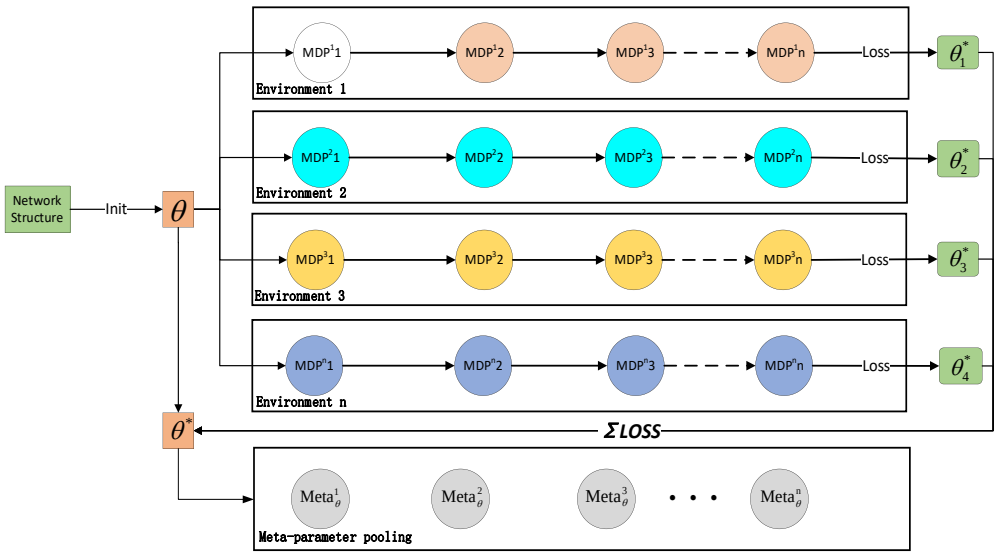


Figure 8. Parameter training process of meta-reinforcement learning

V. EXPERIMENT AND RESULT ANALYSIS

A. Experimental platform and parameter setting

In this paper, Torch deep learning platform is used to train model parameters with each kind of reinforcement learning object recognition task in the joint data set of VOC2007 +VOC 2012 as a task type, and the training of the object localization model is tested with the Test data set of VOC2007. In order to test the generalization performance and convergence efficiency of the proposed model for new tasks, six samples of different task types in VOC 2007+VOC 2012 dataset are selected for retraining. For the other 14 kinds of task datasets used for training update model meta-parameter pool. The experiment measured the convergence speed of the target localization model with different initial parameters, the localization precision (ap), the return rate (recall) of the trained model, and the number of required positioning steps.

B. Meta reinforcement learning Loss comparison plot

In this section, the random initial parameter θ_{random} , the MAML-RL based meta-parameter θ_{maml} and the adaptive meta-parameter θ_{meta} of this paper are compared in the convergence speed of retraining with few samples on VOC 2007+VOC 2012 datasets, respectively. In this paper, only one type of target is located in the target location part. As shown in Figure 9, the Loss value of the model with random parameter θ_{random} is 2.8583 in the initial training, and converges to 0.2 after 2000 training times. Using the meta-parameter θ_{maml} based on MAML-RL, the Loss value of the initial training of the model is 2.0525, indicating that the model has a certain learning ability, and converges to around 0.2 after 1000 training times. For the method in this paper, the model uses adaptive meta-parameters to have a low Loss value (1.3347) for few sample data in the initial case, indicating that the model automatically matches meta-parameters adapted to the new task as initial parameters, and the model reaches convergence in a few steps (500). In addition, with the same number of training steps, the few-shot task overall fits more than r and m through retraining. It shows that the proposed method has good generalization and learning ability for few-shot data training.

C. Results of target positioning

In this paper, six categories cat, bicycle, aeroplane, cow, tvmonitor and DOGin VOC 2007+VOC 2012 dataset are selected for testing with the meta-parameters based on the proposed method as initial parameters. By testing the test samples after the same training batch, it is found that the proposed method can make the target localization model have better recognition accuracy for the target in a few steps, and part of the samples can capture the target position in one action step. Figure 10 shows the test results of the test sample, where Iteration represents the number of steps explored by the model, and the blue wireframe represents the prediction of the model on the target field of view.

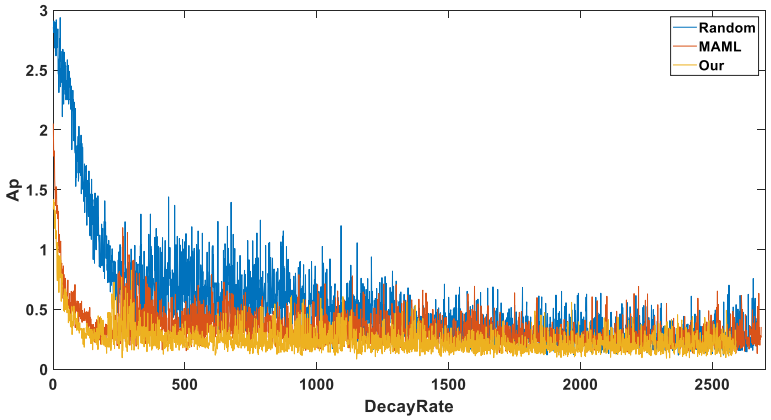
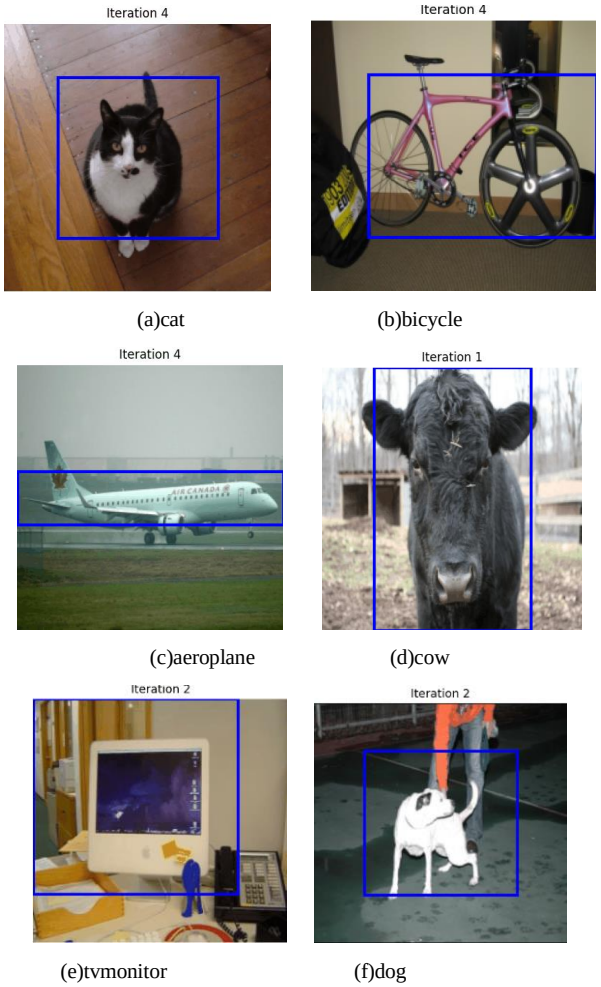


Figure 9. Comparison of training loss functions



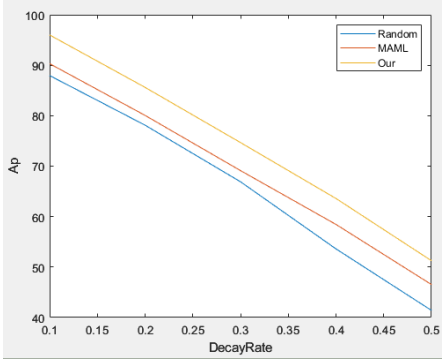
(a)cat (b)bicycle

(c)aeroplane (d)cow

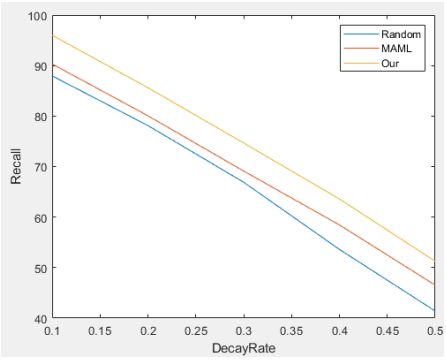
(e)tvmonitor (f)dog

Figure 10. Results of ta

Figure 11 shows the precision rate (Ap) and Recall rate (Recall) of the model under the test data set and the proposed method under the VOC 2007 test set for the new tasks bird, motorbike, diningtable and train categories. The initial discount rate of the model is set to 0.5 and decreases to 0.1 in steps of 0.1, which shows the probability that the agent chooses the optimal action according to the model. It can be seen from Figure a and Figure b that the accuracy and recall of the model decrease when the learning rate increases from 0.1 to 0.5, and the accuracy and recall of the model for different tasks reach the highest when the learning rate is 0.1.



(a) Accuracy Ap



(b) Recall Rate

Figure 11. Comparison of precision and recall

VI. CONCLUSIONS

In order to overcome the shortcomings of low generalization ability caused by insufficient data in few-sample data sets and forgetting of meta-reinforcement learning, this paper proposes a meta-reinforcement learning target localization algorithm based on meta-reinforcement learning parameter playback. Firstly, the model uses MAML method to train various tasks to obtain local optimal parameters. Then, a meta-parameter pooling method is used to store and playback the task meta-parameters, and the optimal parameters for few-sample data training are retrained by feature matching to improve the generalization ability, training speed and target positioning accuracy of the model.

In the model training phase, the positioning model and meta-parameter pool are trained in stages to improve the positioning accuracy of the model, and the data utilization efficiency is improved by sharing the data in the meta-parameter pool. The experimental results show that the number of samples required for model

training and the computational cost are effectively reduced by using the memory and replay method of the meta-parameter pool. The experiments on the target positioning data set verify the effectiveness and generalization of the method for practical problems. However, the test results in the process of multi-type object detection are not ideal, and there is a large room for improvement. For multi-target detection, target detection is realized by using yuan reinforcement learning way and the transfer between multiple targets is in-depth study in this part.

REFERENCES

[1] Mathe S, Pirinen A, Sminchisescu C. Reinforcement learning for visual object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2894-2902.

[2] Zhou W, Lai J, Liao Y, et al. Meta-reinforcement learning based few-shot speech reconstruction for non-intrusive speech quality assessment [J]. Applied Intelligence, 2023, 53(11):14146-14161.

[3] Yao Hongge, Zhang Wei, Yang Haoqi et al. Joint return target depth of intensive study [J]. Journal of automation, 2023, 49 (5):1089-1098. The DOI: 10.16383 / j.a as c200045.

[4] Snell J, Swersky K, Zemel R. Prototypical networks for few-shot learning [J]. Advances in neural information processing systems, 2017, 30.

[5] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks [C]//International conference on machine learning. PMLR, 2017:1126-1135.

[6] Gupta A, Mendonca R, Liu Y X, et al. Meta-reinforcement learning of structured exploration strategies [J]. Advances in neural information processing systems, 2018, 31.

[7] Thrun S, Pratt L. Learning to learn: Introduction and overview [M]//Learning to learn. Boston, MA: Springer US, 1998:3-17.

[8] Ajay, Anurag, et al. "Distributionally adaptive meta reinforcement learning." Advances in Neural Information Processing Systems 35 (2022):25856-25869.

[9] Duan Y, Schulman J, Chen X, et al. Rl $^ 2$: Fast reinforcement learning via slow reinforcement learning [J]. arXiv preprint arXiv:1611.02779, 2016.

[10] Al-Shedivat M, Bansal T, Burda Y, et al. Continuous adaptation via meta-learning in nonstationary and competitive environments [J]. arXiv preprint arXiv:1710.03641, 2017.

[11] Fakoor R, Chaudhari P, Soatto S, et al. Meta-q-learning [J]. arXiv preprint arXiv:1910.00125, 2019.

[12] Wang Y, Yao Q, Kwok J T, et al. Generalizing from a few examples: A survey on few-shot learning [J]. ACM computing surveys (csur), 2020, 53(3):1-34.

[13] Schoettler G, Nair A, Ojea J A, et al. Meta-reinforcement learning for robotic industrial insertion tasks [C]//2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020: 9728-9735.

[14] Garcia F, Thomas P S. A meta-MDP approach to exploration for lifelong reinforcement learning [J]. Advances in Neural Information Processing Systems, 2019, 32.

[15] Sutton R S, Barto A G. Reinforcement learning: An introduction [M]. MIT press, 2018.

[16] Yu T, Quillen D, He Z, et al. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning [C]//Conference on robot learning. PMLR, 2020:1094-1100.