Communication

Not peer-reviewed version

Adaptive Modeling Multi-Agent Learning System for Video Behavior Clustering Recognition

Xingyu Qian , Aximu Yuemaier , Wen-Chi Yang , Xiaogang Chen * , Shunfen Li , Weibang Dai , Zhitang Song

Posted Date: 9 May 2023

doi: 10.20944/preprints202305.0600.v1

Keywords: Continuous Learning; Multi-agent System; Prediction; Adaptability

Article

Adaptive Modeling Multi-Agent Learning System for Video Behavior Clustering Recognition

Xingyu Qian [1], Aximu Yuemaier [2], Wen-Chi Yang [3], Xiaogang Chen [1,*], Shunfen Li [1], Weibang Dai [1] and Zhitang Song [1]

1 Chinese Acad Sci, Shanghai Institute of Microsystem and Information Technology, Shanghai 200050, China
2 Shanghaitech University, Shanghai 201210, China
3 NeuHelium Co., Ltd., Shanghai 200050, China
* Correspondence: chenxg@mail.sim.ac.cn

Featured Application: This work combines a decentralized multi-agent system working form, using a similar approach to associative memory, focusing on the dynamic association of motion processes, to achieve clustered recognition of motion behavior by designing a multi-layer structured multi-agent system.

Abstract: In this work, we propose a self-supervised multi-agent system that meets the online learning of clustering tasks for video behavior recognition spatio-temporal tasks. Encoding visual behavioral actions as discrete temporal sequence(DTS). Real-time clustering recognition task in a multi-agent system for continuous model building, training, and correction. Finally, we implemented a fully decentralized multi-agent system and completed its feasibility verification in a surveillance video application scenario on vehicle path clustering.

Keywords: continuous learning; multi-agent system; prediction; adaptability

1. Introduction

Multi-agent systems are extremely adaptable to deal with complex problems, especially after combining reinforcement learning algorithms, there has been more research progress [1,2]. Independent training approaches such as IQL can lead to difficult learning convergence because of the shared dynamic instability environment factor, while centralized execution, such as VDN integrates the value function of each agent, and the performance can be improved even more by ignoring individual characteristics. And most of the multi-agent systems use the centralized training decentralized execution (CTED) approach, such as MADDPG [3] and QMIX [4]. However, for the centralized training approach, the multi-agent system always uses the system goal to evaluate the computational tasks and training processes of the agents, and even in some cases, the learning is performed in a simulator or laboratory [5], in which additional state information is available and the agents can communicate freely through global data sharing.

In contrast, the decentralized execution approach uses a strategy of decentralizing the computational tasks, where each agent performs its learning process with local data and the environment. And the decentralized approach also reduces the complexity of interaction with the increased number of agents and avoids partial observability challenges. Therefore, it is always necessary to find an effective decentralized execution strategy for both the execution process and the training process.

In addition, the process of changing the visual information of the video is always continuous and connected in the temporal task. If the frame is divided into different regions, the change of different regions over time shows a necessary regularity. The associated change process of different regions can be summarized in higher dimensions, so that the frame's change can be summarized in higher dimensions by focusing on the dynamic situation of many small regions. With this law, if the changes of different regional pictures are encoded, the temporal encoding sequence that are different but temporally related can be obtained. For a such discrete-temporal sequence to carry out real-time clustering recognition behavior, then the clustering recognition results of neighboring regions at the same level can form a new sequence. By analogy, a multi-layer structure of clustering recognition structure can be established, to cover the global video frame picture, and realize the learning from the summary of small change patterns to the global behavior recognition.

Based on the above design concepts, by using traditional methods rather than the Neural Network or RL, this paper designs a framework termed as memory-like adaptive modeling multi-agent learning system with the following main contributions:

1. The design system handles video-level behavior recognition, encoding video information as discrete temporal sequence as the system's way of processing data. The design uses a multi-layer structure, and the clustering results of Agents form a new DTS at higher-level agents, forming a self-organizing structure pattern in the bottom-up direction.
2. The design agents all use prediction as the evaluation criterion, making the agents only concerned with local performance. The top-level agent also only cares about its own execution, but the top-level agent also represents the performance of the system, and its clustering result is expressed as the recognition result of the video behavior.
3. Designing dynamic model-building method in helping agent comparison calculations to achieve pattern recognition or behavioral clustering, and the ability to maintain the matched models based on matching degree fusion sequence characteristics to achieve continuous learning with online updates.

We use video vehicle behavior recognition captured from streetscape as a validation application scenario. The validation process uses the original video without the use of manual feature extraction. The vehicle type and driving behavior are identified and verified using direct input from the original video, without the application of manual feature extraction. Validation results show that, for the scenario of image behavior recognition, our scheme can perform reasonable clustering of different behaviors. And it can dynamically adjust the node network data when new scenarios arise, thus adapting to new situations in a timely manner.

2. Relate Work

Mainstream multi-agent systems use reinforcement learning for policy optimization. And, most use centralized training decentralized execution (CTED) approaches, such as MADDPG, QMIX, where the learning process is centralized and relies on labeled datasets to proceed with offline learning. For the agent evaluation problem such as the credit assignment, Google's Sunehag et al. proposed the DeepMind [6] approach with the main purpose of solving the "lazy agent", making each agent a learning goal of its own, and the overall contribution of the team is the sum of individual contributions. Iqbal et al. introduced the Attention mechanism [7] with the aim that agents need to pay more attention to those agents that should be paid attention to while learning, allowing the agents' attention to be differentiated. Qian et al. [8] also used a similar attention structure as MAA to solve the problem of excessive learning complexity as the number of agents increases, and introduced genetic algorithms for course learning.

In video processing, optical flow method is one of the very common ways in video processing, such as FlowNet [9] uses CNN to solve the optical flow estimation problem. Two-Stream [10] uses two classifiers, one trained specifically for RGB graphs and one specifically for optical flow graphs, and then the results of both are subjected to a fushion process. However, the optical flow method is computationally demanding and slow due to high density computation. We chose to use the background-based modeling method to convert video frames using filtering to discrete sequence coding in a 1/0 binary coding manner. Although it leads to a loss of accuracy and detail, it can focus on the changing pattern of states.

As for self-supervised learning methods, they are broadly classified into three types: Context-based, Temporal-based, and Contrastive-based. For context-based methods, for example, Jigsaw [11] generates losses by predicting the relative positions of parts of the picture to learn a representational model with semantics. For temporal-based methods, for example, TCN [12] compares each frame and performs self-supervised constraints by constructing similar (position) and dissimilar (negative) samples. For contrastive-based methods, e.g., DIM [13] collects global features (final output of the encoder) and local features (features of the middle level of the encoder) of the image to identify whether the classification global and local features of the model come from the same image. Self-supervised learning is more often used to train convolutional neural networks (CNNs) and graph neural networks (GNNs), while SSL-Lanes [14] performed a self-supervised learning paradigm in motion prediction systems.

3. Discrete Temporal Sequence and Empirical Models

The vision behavior occurs with the change of RGB information, for example, a hand wave causes the change of the related screen area within a specified time. If using binary data (0 means no change, 1 means a change) to represent the change in a certain area of the screen. Such as the top half of Figure 1 shows, the vehicle passed by the four screens is represented as 1 but cannot distinguish the different driving situations. By subdividing into four regions, as shown in the bottom half of Figure 1, the data of four regions under the same screen are encoded separately, and the same scene can distinguish different situations of vehicles. The four regions are coded in chronological order, and the four coding sequences of 0000, 0011, 1100, and 0110 can be obtained to represent the change of the four regions over time respectively. In this case, by refining and covering the whole visual image region, visual information can be encoded as a DTS, and feature information extraction can be realized. And for similar intersections or similar complex motion situations, the structure shown in Figure 1 is insufficient, and more refinements are needed.

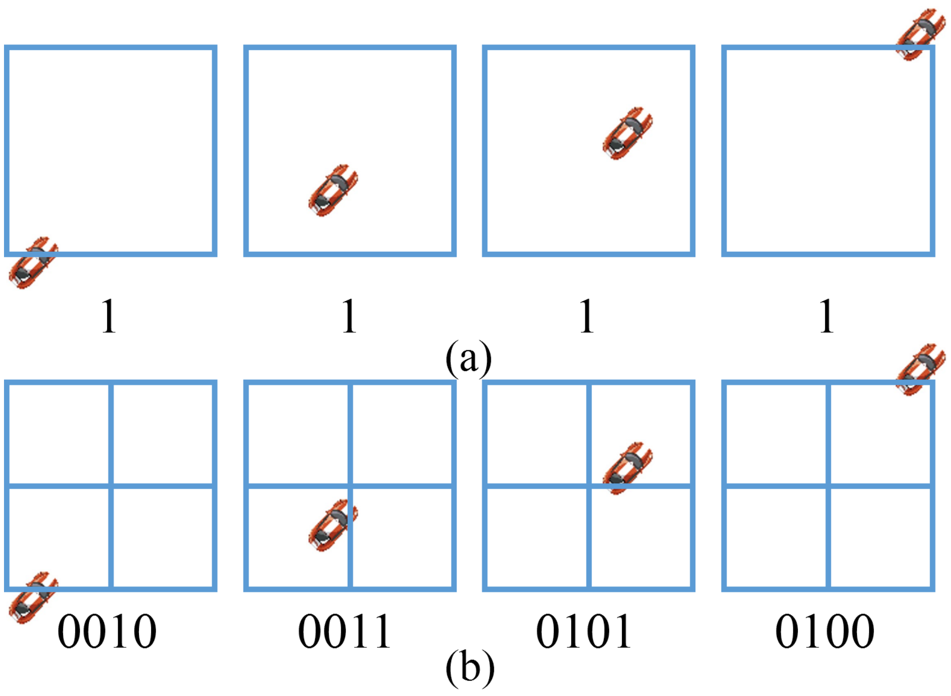

Figure 1. Encoding the DTS.

In a similar way to associative memory, we design agents to serially model the input discrete temporal sequence, and the model itself is a discrete sequence. The model is used to match the new input discrete time series using the model to achieve the clustering capability of the discrete temporal sequence. The results generated by multi-agents, i.e., the ID of the selected models, are combined into new a DTS at a higher-level agent in the multi-layer structure. As shown in Figure 2, data such as "a", "b", "c", "d", etc., input from different frames of an agent are gradually combined into a DTS "abcd...". The agent selects the appropriate model through a matching process, and outputs the model number as the clustering result, using multiple models that are saved locally. When "1", "2", "2", and "1" are output at different frames, they are combined into a new DTS "1221..." at the next agent which represents the activity of the agent. And for the model built and used by the agents using the memory-like method, we call the empirical model (EM). EM is the result of clustering the input DTS by an agent. By matching with EM, merging operation can be executed the feature of DTS and EM that matched, which updates the learning model, enabling online learning.

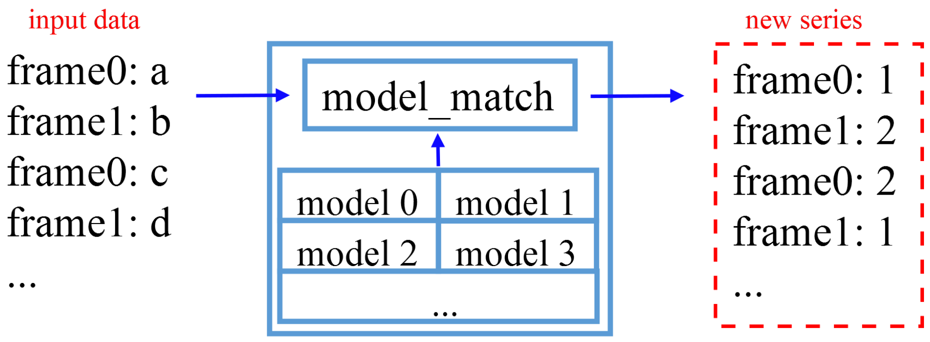

Figure 2. Application of DTS and EMs.

4. Multi-Agent Clustering System

The computational process of discrete temporal sequence matching and the fusion update and maintenance of the model is done in each agent, using a multi-layer structure to combine multi-agent systems. The system is not a fully connected neural network. In the multi-layer structure, a pyramidal structure is used between layers, with correlated region correspondence between different layers. The communication between agents is performed with only two kinds of data transfer. One is the clustering result of the agent passed from the bottom-up direction, and the other is the prediction as feedback, which is in the opposite direction of the clustering result. The longest common sequence (LCS) similarity comparison algorithm performs the sequence comparison computation operation for arbitrary length DTS to EM, applied in the matching computation process, model feature fusion, and prediction. The model-matching degree is evaluated using the model weights and the LCS which is obtained by dynamic programming. For the model upgrading operation, according to the model matching degree, it will be decided to update the selected EM in which the LCS is used to modify the weights of EM units to set an effect of strong and weak memory, or create a new one.

The DTS matching computational process and model operation such as merging and modeling are implemented in each agent. Using agent design with fully homogeneous functional modules, the multi-agent system is combined using a hierarchical structure. As shown in Figure 3, this is the structure of the three agents, where agent1 and agent3 represent only the interface part. S_in and D_out interfaces are used for data input and clustering result output of the agent. And Fb_in and P_out interfaces receive feedback and output prediction results in the opposite direction of data transfer. The History Encoder module is used to encode the data that is input over time into DTS, the Model Library module is used for local saving of EMs and the Compare Core module adopts the received feedback on the DTS data and the history selection retained by the History State to perform matching calculations using EMs. The Model_op module performs model feature merging or new model generation operations based on the matching performance at the end of the behavior to achieve a self-supervised learning effect.

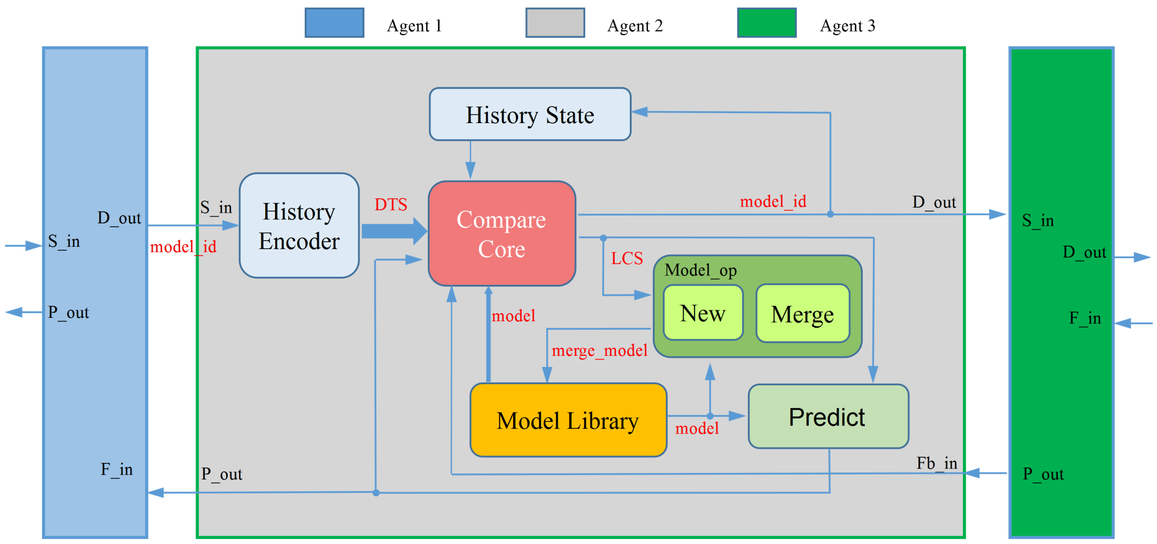

Figure 3. Agent Structure Diagram.

Assuming the existence of empirical model $M = \{m_1, \ldots, m_n\}$, and DTS $\boldsymbol{\Delta} = \{\delta_1, \ldots, \delta_n\}$, the specific algorithm formula of the dynamic planning table using dynamic programming to find the longest subsequence of the EM with DTS is as follows.

$$T_{ij} = \begin{cases} T_{i(j-1)} + 1, & \delta_i = m_j \\ T_{(i-1)j}, & \delta_i \neq m_j \end{cases} \tag{1}$$

where, T: the planning table generated by dynamic programming, δ_i: a certain sequence member, m_j : a certain model member. I and j denote pointers to S and M, respectively.

Moreover, in a multi-agent design, if the agents are all designed to care only about the local performance, it is like using a single agent's learning method ignoring the systematic relationship. In this paper, as higher-layer agents can learn the model-matching pattern of lower-layer agents through DTS and bottom-up approach, the prediction performance is fed back to the corresponding lower-layer agents, thus helping agents to provide priority suggestions in the next model-matching operation. For the model matching process, the agent can select the accurate model at the beginning of the DTS when the models maintained by the agent are widely different. When some models' fragments are similar, the agent needs more details of the DTS to make the right choice and prediction. Feedback from the higher-layer agent can assist in making faster choices among several similar models, thus improving data stability, and improving the learning convergence of the system. Even if the feedback is incorrect, the agent can discard the feedback based on the current performance.

5. Experiment and Results

We divide the video screen roughly into 16*16 total of 256 grid points according to the grid, as shown in Figure 4a, and it can get 256 DTS in total. Figure 4b–e shows the designed four-layer agent structure for the multi-agent system, where each rectangular box represents an agent in the corresponding position, so the data of the image and the agent transmission between different layers in this scenario are the four boxes on the left corresponding to the one on the right. Thus, the first layer uses 64 agents for receiving the 256 DTS that divide the image, and each agent receives the data corresponding to four regions. And so on, the second to fourth layers use 16, 4, and 1 agents, where the fourth layer agent's clustering effect represents the result of the whole system.

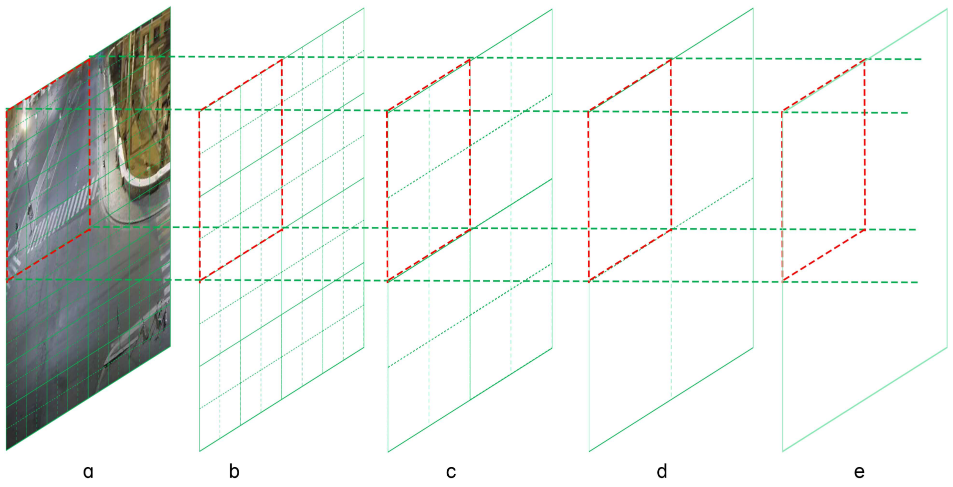

Figure 4. Grid division diagram.

As shown in Figure 5, the test has been performed a total of 500 times for three driving behaviors, such as straight ahead and right turn, for the vehicle videos. Figure 5a shows the correct rate of clustering for all agents (85 agents) in the test. Figure 5b,c shows the interaction of several agents in two layers during the two inputs. Figure 5d–g shows the correct rate of clustering of agents in the first to fourth layers in the test, respectively. The horizontal coordinate indicates the number of tests, and the vertical coordinate indicates the proportion of agents clustered correctly. The first 32 times (the orange) are the system start-up phase, where two types of behavioral sequential input are performed to help initial model building, followed by (the green) two types of behavioral random tests. Starting from the 200th time (the blue), the third type is gradually added during the random test until the end of the test.

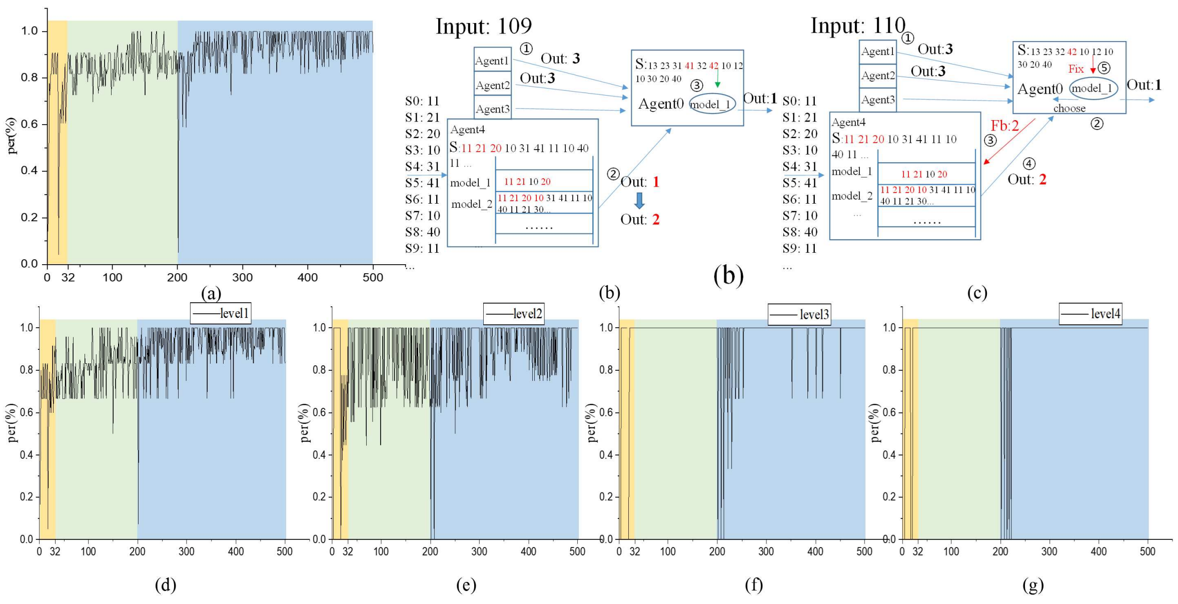

Figure 5. clustering result.

As shown in the figure, in the initial stage and for some time since the 200th test, the absence of the correlation model led to poor clustering results. Subsequently, the clustering effect gradually improves with the tests. This effect is obtained both for Figure 5a and Figure 5d–g, corroborating the hypothesis of continuous learning which is the ability to perform online modeling learning and continuously updated with new changes.

Figure 5b,c shows the continuous learning process of inter-agent collaboration. This is the agent communication between agent1-4 located in the first layer and agent0 in the second layer, for the 109th and 110th test processes, respectively. The two tests are of the same type, the difference being the presence of additional feedback behavior in the input110 test. Because the behavioral process makes there a delay, agent4 data arrives later than agent1 and agent2. In the 109th test, agent4 will select model_1 before model_2. Agent4 selected the best-matched model that directly benefited the feedback. agent0 can also get a more stable sequence, to continue to update the model features. The results of the hierarchical clustering in Figure 5d–g also show that the higher layer of the test process can obtain more stable clustering results, and the lower be more volatility. It is beneficial to have a multi-layer design structure of agents in that one agent at a higher layer is chosen to communicate with four agents at a lower layer. Even if there is an agent clustering error, the agent at the higher layer is more likely to make the correct choice based on the results of many agents. With such a structure, it is shown that higher layers always perform more consistently, except for new-type tests.

6. Conclusions

Facing the dynamically changing environment, it has been a challenge for multi-agent systems to design the system structure and task objectives to improve the response to change adaptability. Facing the diverse needs of environmental changes, we propose an adaptive multi-agent clustering recognition system that uses time series to achieve recognition by clustering predictive behaviors. The agent internally completes the clustering prediction activity on the input, and the system achieves the diversity requirement by designing the connection structure with different combinations of agents with different functions. Finally, the feasibility test experiments are conducted for video behavior recognition scenarios, and the experiments yield correct recognition results, while the synergistic effect between agents and the form of connection combinations can achieve convergence in the results, which shows the feasibility of the system to meet the diversity requirements. However, there are still many factors that can be explored. For example, in the selection of the number of layers of the system, we also conducted a test session for 5-layer and even 6-layer structures designed. The increase in the number of agents at the bottom layer allows the system to obtain finer classification results but also makes the convergence rate of the correct clustering rate of all agents slower. Another example is the design of the connection structure of the agents for self-organization to establish connection relations, so that the multi-layer structure can be implemented more flexibly according to the state of neighboring agents, etc. Finally, the feasibility test experiments on video behavior recognition scenarios are conducted to verify the synergistic effect between agents and the convergence of results achieved in the face of new changes, which shows the feasibility of the system to meet the diversity requirements.

Author Contributions: Conceptualization, X.C., X.Q., S.L.; methodology, X.C. X.Q., A.Y.; software, X.Q. and A.Y.; validation, X.Q.; formal analysis, X.C.; investigation, W.D. and W.Y.; resources, A.Y., X.Q.; data curation, A.Y. and X.Q.; writing—original draft preparation, X.Q.; writing—review and editing, X.C. and W.Y.; visualization, X.C. and X.Q.; supervision, W.Y.; project administration, Z.S.; funding acquisition, Z.S.. And All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Strategic Priority Research Program of OF the Chinese Academy of Sciences grant number XDB44010200.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We provide the code and dataset at: https://github.com/qian-git/MAMMALS.

Acknowledgments: I want to thank Mr. Longfei Liang for guidance in my research. He shared a lot of support for the research project.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DTS	Discrete Temporal Sequence
CTED	Centralized Training Decentralized Execution
EM	Empirical Model
LCS	Longest Common Sequence

References

1. Hernandez-Leal, P.e.a. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems* **2019**, *33*, 750–797.
2. Gronauer, S.e.a. Multi-Agent Deep Reinforcement Learning: A Survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943.
3. Lowe, R.e.a. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. Proceedings of the 31st ICNIPS; Curran Associates Inc.: Red Hook, NY, USA, 2019; Vol. 575, *NIPS'17*, pp. 350–354.
4. Rashid, T.e.a. QMIX:Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. Proceedings of the 35th ICML. PMLR, 2018, Vol. 80, *Proceedings of Machine Learning Research*, pp. 4295–4304.
5. Foerster, J.N.; et al.. Counterfactual Multi-Agent Policy Gradients. Proceedings of the Thirty-Second AAAI. AAAI Press, 2018, AAAI'18/IAAI'18/EAAI'18.
6. Sunehag, P.e.a. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. Proceedings of the 17th AAMAS; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, 2018; AAMAS '18, p. 2085–2087.
7. Iqbal, S.e.a. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. Proceedings of the 36th ICML. PMLR, 2019, Vol. 97, *Proceedings of Machine Learning Research*, pp. 2961–2970.
8. et al., Q.L. Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning. *Learning* **2020**.
9. Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazirbas, C.; Golkov, V.; Smagt, P.v.d.; Cremers, D.; Brox, T. FlowNet: Learning Optical Flow with Convolutional Networks. 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2758–2766.
10. Feichtenhofer, C.; Pinz, A.; Zisserman, A. Convolutional Two-Stream Network Fusion for Video Action Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1933–1941.
11. Doersch, C.e.a. Unsupervised Visual Representation Learning by Context Prediction. 2015 IEEE ICCV, 2015, pp. 1422–1430.
12. Sermanet, P.e.a. Time-Contrastive Networks: Self-Supervised Learning from Video. 2018 IEEE ICRA, 2018, pp. 1134–1141.
13. Hjelm, D.e.a. Learning deep representations by mutual information estimation and maximization. ICLR 2019, 2019, pp. 1422–1430.
14. Bhattacharyya, P.e.a. SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving. *arXiv e-prints* **2022**, p. arXiv:2206.14116, [arXiv:cs.CV/2206.14116].