AUTHOR(S): HUSSEIN, A., ELYAN, E. and JAYNE, C.

TITLE: Deep imitation learning with memory for robocup soccer simulation.

YEAR: 2018

Publisher citation: HUSSEIN, A., ELYAN, E. and JAYNE, C. 2018. Deep imitation learning with memory for robocup soccer simulation. In Pimenidis, E. and Jayne, C. (eds.) Communications in computers and information science, 893: engineering applications of neural networks; proceedings of the 19th international engineering applications of neural networks (EANN 2018), 3-5 September 2018, Bristol, UK. Cham: Springer [online], pages 31-43. Available from: https://doi.org/10.1007/978-3-319-98204-5_3

OpenAIR citation: HUSSEIN, A., ELYAN, E. and JAYNE, C. 2018. Deep imitation learning with memory for robocup soccer simulation. In Pimenidis, E. and Jayne, C. (eds.) Communications in computers and information science, 893: engineering applications of neural networks; proceedings of the 19th international engineering applications of neural networks (EANN 2018), 3-5 September 2018, Bristol, UK. Cham: Springer, pages 31-43. Held on OpenAIR [online], Available from: https://openair.rgu.ac.uk/

Deep Imitation learning with memory for Robocup Soccer Simulation

Ahmed Hussein[1], Eyad Elyan[1], Chrisina Jayne[2]

[1] School of Computing, Robert Gordon University, Garthdee Road Aberdeen, UK AB10 7QB
[2] School of Engineering, Computing and Mathematics, Oxford Brookes University, Oxford OX3 0BP

Abstract. Imitation learning is a field that is rapidly gaining attention due to its relevance to many autonomous agent applications. Providing demonstrations of effective behaviour to teach the agent is useful in real world challenges such as sparse rewards and dynamic environments. However, most imitation learning approaches don't retain a memory of previous actions and treat the demonstrations as independent and identically distributed samples. This neglects the temporal dependency between low-level actions that are performed in sequence to achieve the desired behaviour. This paper proposes an imitation learning method to learn sequences of actions by utilizing memory in deep neural networks. Long short-term memory networks are utilized to capture the temporal dependencies in a teacher's demonstrations. This way, past states and actions provide context for performing following actions. The network is trained using raw low-level features and directly maps the input to low-level parametrized actions in real-time. This minimizes the need for task specific knowledge to be manually employed in the learning process compared to related approaches. The proposed methods are evaluated on a benchmark soccer simulator and compared to supervised learning and data-aggregation approaches. The results show that utilizing memory while learning significantly improves the performance and generalization of the agent and can provide a stationary policy than can produce robust predictions at any point in the sequence.

1 Introduction

Recent years have seen a rise in demand for autonomous intelligent agents. Imitation learning [7] is a promising approach for teaching agents intelligent behaviour by providing demonstrations performed by an expert. Providing demonstrations by performing a task is substantially easier than articulating how the task should be performed and explicitly programming the agents. Moreover, learning from demonstrations is suitable for realistic training scenarios which impose restrictions on learning from experience such as sparse rewards and dynamic environments. Imitation learning approaches commonly deal with demonstrations as discrete instances of state and action pairs. Although most autonomous applications involve performing sequences of actions to achieve a goal, most learning

methods process instances separately as independent and identically distributed (i.i.d.) samples. These methods rely on the hypothesis that the observed state contains enough information to make an accurate decision; and that performing a series of accurate independent decisions will accumulate to effective behaviour. This hypothesis overlooks the dependencies between actions which can be key in planning long trajectories of actions. This is especially sensitive in imitation learning as the teacher might inherently be relying on memory, without this information being presented to the learning agent. Even if an accurate decision can be made from the current state alone, the teacher might choose a different course of action based on previous experience. If this additional information is not presented to the agent, it won't be able to learn from the demonstrated behaviour [6]. Moreover, even if sampling the training data is dependent on previous actions such as data aggregation methods, the learning algorithm doesn't take temporal relationships between these observations into account. Using memory of past events as context, allows the policy to learn different reactions to similar observations in different point along the trajectory [11]. It is therefore necessary to represent training demonstrations as sequences and learn to reproduce dependent action trajectories.

Recurrent neural networks have shown great success in learning from sequences [2,12]. They capture temporal dependencies by having looping connections so the nodes consider previously processed samples along with new input to produce a decision. However, most RNN applications involve processing the sequence in its entirety before producing a decision or generating an output sequence [19]; which is not suitable for real time autonomous agents. Some applications such as handwritten text generation utilize RNNs to generate a sequence one step at a time [3]. However, these sequences are generated in isolation from other factors while autonomous agents are required to react mid trajectory to dynamic environments. For that, imitation learning requires new RNN based methods that can learn from long sequences of dependent actions and react based on real time observations of the environment.

This paper proposes a novel approach which includes representing demonstrations as sequences of dependent state-action pairs and using a long-short-term-memory network (LSTM) to learn a policy. The LSTM network learns a mapping between states and actions while taking into consideration memory of previous events and actions and the temporal dependencies between these instances. This approach is demonstrated on the "robocup soccer simulator" [9]; a multi-agent soccer simulator. The multi-agent setting provides a dynamic environment for which generating static sequences is not suitable as the policy is required to react to the other agents' actions. This makes the simulator a popular benchmark for intelligent agents. Unlike most machine learning methods, the proposed LSTM network learns from raw low-level sensory data, without the need for engineered feature extraction. Similarly, the policy performs low-level parametrized actions that making a decision as well as predicting continuous values for the actuators simultaneously. Performing sequences of these low level actions makes up the desired behaviour without manually engineering high level strategies. To evaluate

the proposed LSTM approach its performance is compared to the hand-crafted teacher policy, and policies learned via neural networks without memory (MLP). To evaluate the generalization ability of RNNs in imitation learning, the proposed approach is further compared to a data aggregation method [16] conducted on the MLP agents.

The remainder of the paper is organised as follows: Section 2 reviews related work in the literature. Section 3 describes the proposed methods. Section 4 details the experimental setup and the produced results. Finally, Section 5 concludes the paper and provides directions for future research.

2 Related Work

In this section we present related work that utilize deep reinforcement learning and describe different methods proposed in the literature to combine learning from demonstrations and experience.

Recurrent neural networks can be used to generate sequences by considering the past generated samples. Such an approach is used in [1] to generate continuous handwriting. An extension to this approach is also proposed in [1] that allows the generated sequence to be conditioned on a sequence of input text characters. Clearly, such approaches can be very relevant to imitation learning if the actions can be formulated as a generated sequence conditioned on a sequence of observed states. Similarly sequence to sequence learning [19] has been gaining a lot of attention recently. However, for most applications, the entire input sequence is analysed before generating the out sequence, while autonomous agents are required to act in real time to every sensory input.

An LSTM based system is proposed in [11] to learn how to perform surgical procedures by controlling a robotic arm. The network is trained on demonstrations by a human expert. Although the static setting of the surgery allows for policies that replicate manually designed trajectories, this supervised learning approach provides better generalization.

The robocup simulator is a popular benchmark for intelligent learning methods as it shares many characteristics with real world applications. A cooperative defensive task is learned in [15] using demonstrations provided by two human players simultaneously. Several classifiers are used to learn from the demonstrations and the results are favourably compared to human performance and simple hand-coded agents. However, this approach employs high level strategies as the decision to be learned by the agent, such as "approach the ball" or "block attacker's path" which in turn need to be translated into low level actions through manual programming. High level actions also enable learning the task through evolutionary algorithms [14] as the solution space becomes smaller. Similarly, in [8,18,10] reinforcement learning is used to learn high level actions. It is noteworthy that each paper employs a different set of macro-actions; so each new macro-action has to be manually designed. Deep reinforcement learning is used in [5] to learn an offensive task from raw sensory data. The reinforcement learning policy is used to predict low-level parametrized actions and thus doesn't

require manual policy design. However, the organic reward in this task (scoring a goal) is very sparse and requires performing long trajectories of low-level actions to reach this state. As reinforcement learning exploration fails to reach the environment's reward, this approach employs a manually engineered reward function that guides the agent to perform desired behaviours. This engineering requires substantial task knowledge and limits the general application of this approach.

3 Method

This section presents the proposed method for training agents to play soccer via deep imitation learning with memory. A recurrent neural network is trained solely from demonstrations and doesn't require any explicit tailored engineering. The proposed method minimizes the need for expert knowledge by utilizing the low level sensory features and learning a mapping to atomic parametrized actions. We start by describing the process of data collection and representation. Demonstrations are provided by a teacher that performs the task for a number of rounds. For each round the teacher attempts to score a goal; the round ends with a successful or unsuccessful attempt. A plethora of hand-crafted agents exist for the 'robocup soccer simulator' and can serve as the teacher to provide examples of effective behaviour. Existing agents are also available to control the opponent to provide a realistic setting for the demonstrations.

Each round is represented as a sequence of state-action pairs. For each frame t the state of the environment x_t is captured along with the action taken by the teacher y_t and are added to the sequence $S_i = x, y$. The state x_t represents low level information about the agent's surroundings and is captured from the agent's point of view using its simulated sensors. So all the information about the field and the objects and players in it are captured relative to the agent's position and status. The action y_t is chosen from a set of the low level parametric actions available to the agent. That is, the agent decides what move to perform from its list of actuators as well as one or more continuous values that serve as parameters for the selected actuator. Such atomic actions performed in a sequence construct a higher level behaviour that is usually identified and modelled manually in other studies. The captured sequences are used to construct the training dataset $D = S_1, S_2..S_n$ is used to train recurrent neural network.

The training set is used to train a deep recurrent neural network. The network consists of 3 stacked LSTM layers containing 100, 50 and 6 nodes respectively, followed by a reshaping layer to present the 6 output values for all samples in the batch to the loss function. The loss function calculates the error for the predictions of the entire batch rather than the final prediction only. This is because unlike most RNN applications we are interested in producing accurate predictions at each frame rather than optimizing one prediction after reading the entire sequence. The LSTM layers are used to extract high level temporal features from the raw input and the context provided by the networks memory. The LSTM layers utilize hyperbolic tangent (tanh) activation functions. The output

nodes in the final layer correspond to parametrized actions and are used to predict continuous values for the 6 possible parameters for the agent's actuators. The output layer utilizes linear activations and a mean square error loss function is used, therefore the network behaves as a multivariate regressor. The actuator with the highest predicted parameter value is selected for execution by the agent. This method allows for prediction values for multiple parameters simultaneously. In many applications the output of the nodes in the final layer is not produced until the end of the sequence and is only fed into the next time step without being output as the network's prediction. In contrast, the proposed network does not read the entire sequence before producing a decision or generating an output sequence. Instead, at each instance of the input sequence the network predicts an output. Thus generating the output sequence step by step with the input, at each step utilizing all the information available up to this instance. By representing the demonstrations as sequences, this approach provides context for most of the samples facing the agent.

However, this makes the prediction dependent on the position of the sample in the sequence. For example if the agent starts performing the trained policy mid episode, the current frame will be treated as if it is at the beginning of the sequence even though it is not. To ensure the stationarity of the agent's policy, we train another network on a modified version of the training set D in which all the sequences $S_1, S_2..S_n$ are augmented into one list of samples. This list is subsequently segmented into segments of uniform length that serve as the new training sequences to be fed into the LSTM network. Figure 1 illustrates the segmentation of the artificial sequences. This arbitrary creation of sequences presents different states in different parts of the training sequences while maintaining a temporal dependency between the consecutive instances in a sequence. This approach is not expected to outperform training on fully structured sequences given that complete sequences are always presented to the agent during testing. However, it demonstrates that the proposed approach does not depend on reproducing entire training sequences and that utilizing memory in imitation learning is beneficial even if the beginning and end of the sequence are unknown.

Table 1 highlights the differences between the proposed method and other intelligent methods used for "Robocup". Most methods rely on manually engineering features and high-level actions which require significant task specific knowledge and engineering which does not allow for a general learning process. [5] uses deep learning to alleviate the need for engineering features and directly map raw features to low-level actions. However, designing dense reward functions to guide the agent require similar effort and produce similar results to manually engineering the policy. A change in the setting such as the number of players on the field requires designing new reward functions. This is in contrast with organic reward functions which are directly provided by the rules of the game. The proposed approach is general and only receives knowledge about the task from the demonstrations. Providing new demonstrations for changes in a task is considerably easier than designing reward functions or low-level policies to execute the high-level decisions made by the policy.

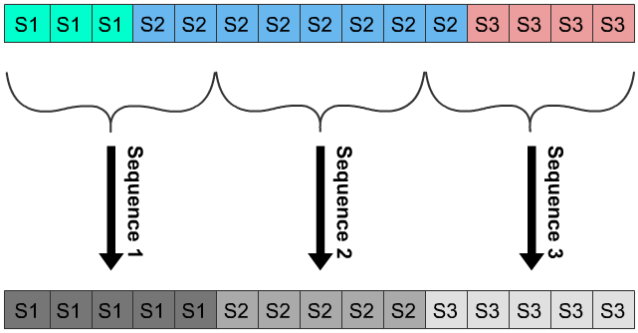

Fig. 1. Re-segmenting the demonstrated sequences into arbitrary sequences

Table 1. A comparison of machine learning approaches for rocbocup

Method	**Learning**	**Features**	**Actions**	**Rewards**
Jain et al.[8]	Reinforcement learning	Selected	High-level	Engineered
Raza et al.[15]	Supervised (various)	Engineered	High-level	N/A
Stone et al.[18]	Reinforcement learning	Engineered	High-level	Organic
Masson et al.[10]	Reinforcement learning	Selected	High-level parametrized	Engineered
Hausknecht et al.[5]	Reinforcement learning	Raw	Low-level parametrized	Heavily-engineered
Ours	Supervised (LSTM)	Raw	Low-level parametrized	N/A

The proposed LSTM network is compared to a multi-layer perceptron that doesn't have a memory and treats all frames as independent and identically distributed samples. In this case the sequences are augmented to create one training set, from which batches of samples are drawn. Keeping the sequence of samples without utilizing memory can be detrimental to training as the samples in training batches will be too similar and lack diversity. Therefore, when training the MLP, the entire dataset is shuffled before sampling the training batches to ensure that they contain diverse samples from a variety of situations. This is similar to the replay buffer approach used in [13] which is a key factor in the success of deep reinforcement learning. The architecture of the MLP consists of 3 fully connected layers containing 100, 50 and 6 nodes respectively. The first 2 layers utilize rectifier activation functions and the output layer uses a linear activation function.

Moreover, to evaluate the generalization of the proposed LSTM approach, it is compared to a data aggregation approach that is applied during MLP training. DAGGER [16] is a seminal data aggregation method that aims to enhance gen-

eralization in imitation learning by providing additional training samples based on the agent's initially trained policy. The agent is allowed to perform the task using the policy trained using the MLP. For each frame the teacher provides the optimal action for the state observed by the agent and a new training dataset is collected. The agent stochastically chooses to perform the teacher's instruction or the action predicted based on its current policy. The new samples are added to the original training set and used to train a new agent. The new samples show states that are likely to be visited by the agent according to its trained policy and thus improving generalization. This process is repeated iteratively and the new set of training instances are aggregated into the final training set.

4 Experiments

4.1 robocup

The robocup soccer simulator [9] is a 2D simulator that allows for full soccer matches between 11 player teams. The simulator is a popular benchmark for artificial intelligence as it contains a number of real characteristics and challenges found in real applications such as a dynamic multi-agent environment and relative sensory information. A challenge closely related to this study is the fact that playing soccer requires performing long sequences of actions that depend on previous actions as well as actions from other agents. Because soccer is a familiar activity, this application provides extensive evaluation of the agents' performance; not only according to the well-established rules of soccer but also qualitatively analysing the agents' behaviour through 2D visualization. Implementation of the proposed methods is available at `https://github.com/ahmedsalaheldin/RoboCupLSTM`

4.2 Half-Field-Offence

This study is conducted on a simplified sub problem of soccer simulator called Half-field-offence (HFO) [4]. Over the years, researchers have used simplified versions of the game of soccer to create intelligent autonomous agents in a more controlled setting [17]. As the name suggests HFO takes place in half the soccer field and is only concerned with the task of offence. The round is initiated with the offensive player and the ball randomly placed in the half-field. The objective of the offence is to score in the opponents goal while the defence tries to intercept the ball. The round ends if a goal is scored or the defence captures the ball or the ball goes out of the half-field bounds or if a time-limit is reached. Full details of the available features and actions can be found in [4]. In our experiments we use demonstrations provided by the offensive agent to teach an intelligent agent to play the offensive role. A simple hand crafted agent is used as the teacher to facilitate communication with the server. More complex higher performing hand crafted agents can be used to provide demonstrations in the future. Figure 2 shows the visualization of the HFO environment.

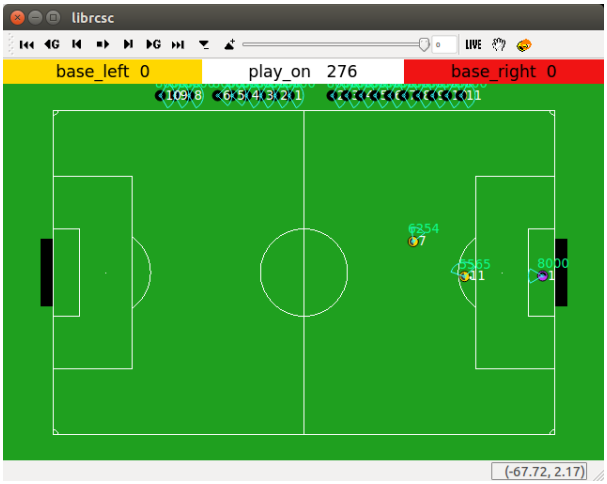

Fig. 2. Illustration of the Half-field-offence environment

4.3 Experimental Setup

The experimental evaluation compares between 4 learning methods. Firstly the proposed LSTM trained on the captured sequences. This method referred to as "LSTM episodic" as each episode or round of HFO makes up a training sequence. Secondly, "LSTM segmented", where the LSTM model is trained on the uniformly segmented sequences. The third method "MLP" trains a supervised multi-layer perception on the training set. And finally "MLP shuffled" is similar to "MLP" but shuffles the dataset before training the model. Moreover, data aggregation of 3 iterations is applied to the MLP based approaches. All models are trained on the same collected demonstrations consisting of 20000 samples. Data aggregation adds a further 5000 samples for each iteration. The sequence length used for training "LSTM segmented" is 80 samples. The models are trained offline for 1500 epochs and the trained networks are saved to be later used by the agent in real time.

We use a client that is decoupled from the learned models to connect to the simulator server so that the same client can be used to execute any learned policy. The client communicates with the simulator to receive the raw sensory data observed by the agent at each frame and send the decisions of the neural network to control the agent's actions. The models are evaluated on 1000 rounds of HFO. Each round can end in one of 4 outcomes. Firstly, a goal is scored, which is the best possible outcome, followed by the defence capturing the ball, then the ball going out of bounds and finally running out of time represents the poorest behaviour by the agent. Table 2 shows the percentage of each outcome achieved by the teacher used to provide the demonstrations; in 1000 rounds of playing.

Table 2. HFO results for the hand crafted teacher

Method	Goal	Defence	Bounds	Time
Teacher	44.37%	51.43%	4.19%	0%

4.4 Results

Firstly, the results comparing the proposed LSTM approach to imitation learning without memory are presented. Figure 3 shows the results for the 4 trained models "LSTM episodic", "LSTM segmented", "MLP" and "MLP shuffled". The results are shown for 1000 rounds of testing. The graph shows the percentage of rounds that resulted in the 4 possible outcomes: goal scored, captured by the defence, ball out of bounds and out of time. The percentage of goals scored is the most important measure as scoring is the primary objective of the task, however the other measures show the rest of the picture. The proposed method "LSTM episodic" has resulted in the highest percentage of goals, similar to the teacher's performance and outperforms networks without memory with statistical significance. "LSTM segmented" comes in second place also outperforming the MLP methods with statistical significance, demonstrating that utilizing memory is the contributing factor in the effectiveness of the learned policy, even if the beginning and end of the sequence are unknown.

The results also show that shuffling the training set resulted in significantly more goals. This corroborates the hypothesis that using dependent sequences of samples to train models without memory can be detrimental as the training batches lack diversity. The remaining measures show the robustness of utilized imitation learning methods with small percentages of unwanted outcomes ("out of bounds" and "out of time") especially the LSTM based methods. Qualitative analysis of the performance shows that running out of time is usually the result of the agent getting stuck and constantly performing the same action. As can be expected the teacher never produced this outcome and it is considered the poorest behaviour displayed by the imitating agents. Being stuck is an indication of ambiguity in the agent, and the extremely low percentage of this behaviour in the LSTM agents demonstrates that utilizing memory significantly improves the generalization of the learned policy in addition to its effectiveness.

Following, the results for data aggregation are presented. Figure 4 shows the results for using data aggregation on the MLP network, with and without shuffling the data. With "MLP shuffle", the entire data set is shuffled each iteration, so the training batches can contain samples from all the aggregated datasets. The graph shows the percentage of goals scored in 1000 rounds for supervised learning (using the original training set), and three iterations of data aggregation. The graph shows that for "MLP shuffle", there is no significant improvement in the percentage of scored goals. Without shuffling, we can see an improvement in scoring for the first two DAGGER iterations but this pattern doesn't hold for the third iteration. For both methods, the graph shows no consistent improvement in scoring with increasing the DAGGER iterations and in all cases doesn't reach the performance of the LSTM approaches.

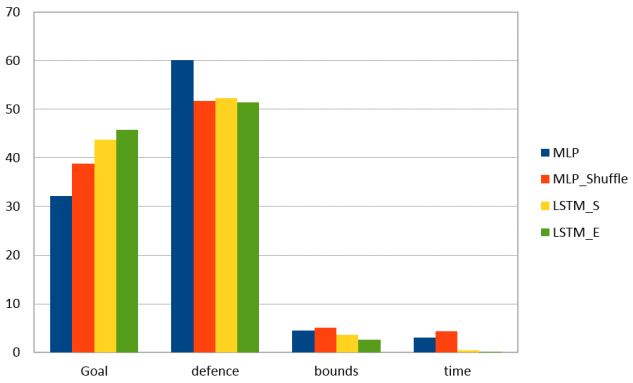

Fig. 3. Results for robocup half field offence. The outcomes presented in the graph are: Goal: The offensive agent scored a goal, Defence: The ball was captured by the defence, Bounds: The ball went out of bounds, Time: A time limit was reached before any of the other outcomes

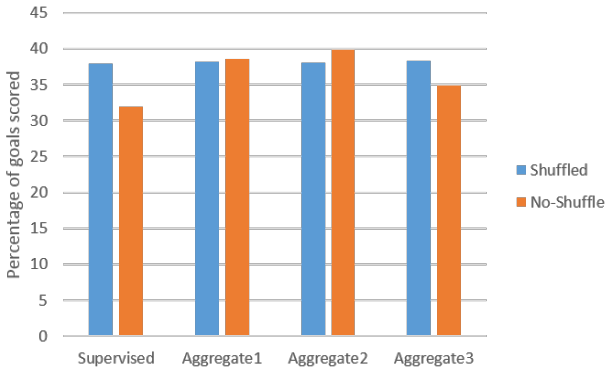

Fig. 4. Scoring percentage for MLP with data aggregation

Tables 3 and 4 shows the complete results for data aggregation with multilayer perceptrons, with and without shuffling respectively. The results show that aggregating new samples does not necessarily decrease the percentage of undesirable outcomes. For both approaches, there does not appear a pattern for decreasing the "out of time" percentages with increasing iterations of data aggregation. In "MLP no-shuffle" where the scoring rate was substantially improved with the first iteration of data aggregation, we observe that this improvement is accompanied by a huge rise in the percentage of "out of time" rounds. This emphasises that data aggregation in this study doesn't provide a consistent improvement in the agent's performance. Although data aggregation utilizes more information, by sampling demonstrations from likely states, it fails to improve the generalization of the agent compared to the proposed LSTM approach.

Table 3. MLP data aggregation results with shuffling

Method	Goal	Defence	Bounds	Time
Supervised	38.83%	51.74%	5.02%	4.40%
Aggregate 1	38.17%	55.12%	4.46%	2.23%
Aggregate 2	38.01%	53.09%	5.68%	3.20%
Aggregate 3	38.31%	53.14%	2.98%	5.56%

Table 4. MLP data aggregation results without shuffling

Method	Goal	Defence	Bounds	Time
Supervised	32.19%	60.12%	4.56%	3.11%
Aggregate 1	38.59%	49.08%	4.27%	7.94%
Aggregate 2	39.81%	54.86%	3.78%	1.53%
Aggregate 3	34.89%	59.89%	3.67%	1.53%

5 Conclusion and Future Work

This paper proposes a novel imitation learning approach for learning from sequences in a dynamic environment. A demonstration is represented as an ordered sequence of state-action pairs. The states are represented by a feature vector of low level sensory information from the agent's perspective. The actions available to the agent are low level parametrized actions. A deep Long-short-term-memory network is used to learn a policy that retains a memory of past experiences and learns from the entire demonstrated trajectory of actions. The trained model uses memory to provide context to improve generalization and predicts an action at every frame in real-time. Results on a multi-agent soccer simulator show that learning from sequences using memory networks can significantly outperform learning from i.i.d. samples and reach comparable performance to the teacher. Using the memory to provide context when learning from sequences outperforms data aggregation methods for improving generalization and is much faster to train. Moreover, it is also shown that the proposed LSTM method can be stationary by training on sequences that are arbitrarily segmented from the demonstrations without a significant drop in performance. Experiments using multilayer perceptions show that if the model has no memory when learning from sequences, shuffling the training data can result in a significant improvement in performance as the samples in the training batches become more diverse. In the next step, we aim to use a number of high performing agents from the robocup competitions to provide the demonstrations and include more agents in the game.

References

1. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
2. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. pp. 6645–6649. IEEE (2013)
3. Graves, A., et al.: Supervised sequence labelling with recurrent neural networks, vol. 385. Springer (2012)
4. Hausknecht, M., Mupparaju, P., Subramanian, S., Kalyanakrishnan, S., Stone, P.: Half field offense: An environment for multiagent learning and ad hoc teamwork. In: AAMAS Adaptive Learning Agents (ALA) Workshop (2016)
5. Hausknecht, M., Stone, P.: Deep reinforcement learning in parameterized action space. arXiv preprint arXiv:1511.04143 (2015)
6. Hussein, A., Elyan, E., Gaber, M.M., Jayne, C.: Deep imitation learning for 3d navigation tasks. Neural computing and applications pp. 1–16 (2017)
7. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: A survey of learning methods. ACM Comput. Surv. 50(2), 21:1–21:35 (Apr 2017), http://doi.acm.org/10.1145/3054912
8. Jain, D., Shah, M., Garg, B.K.: Watchdogs 2d soccer simulation
9. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: The robot world cup initiative. In: Proceedings of the First International Conference on Autonomous Agents. pp. 340–347. AGENTS '97, ACM, New York, NY, USA (1997), http://doi.acm.org/10.1145/267658.267738
10. Masson, W., Ranchod, P., Konidaris, G.: Reinforcement learning with parameterized actions. arXiv preprint arXiv:1509.01644 (2015)
11. Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., Schmidhuber, J.: A system for robotic heart surgery that learns to tie knots using recurrent neural networks. Advanced Robotics 22(13-14), 1521–1537 (2008)
12. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech. vol. 2, p. 3 (2010)
13. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
14. Pietro, A.D., While, L., Barone, L.: Learning in robocup keepaway using evolutionary algorithms. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. pp. 1065–1072. Morgan Kaufmann Publishers Inc. (2002)
15. Raza, S., Haider, S., Williams, M.A.: Teaching coordinated strategies to soccer robots via imitation. In: Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on. pp. 1434–1439. IEEE (2012)
16. Ross, S., Gordon, G.J., Bagnell, J.A.: A reduction of imitation learning and structured prediction to no-regret online learning. arXiv preprint arXiv:1011.0686 (2010)
17. Stone, P., Kuhlmann, G., Taylor, M.E., Liu, Y.: Keepaway Soccer: From Machine Learning Testbed to Benchmark, pp. 93–105. Springer Berlin Heidelberg, Berlin, Heidelberg (2006), https://doi.org/10.1007/11780519_9
18. Stone, P., Sutton, R.S., Kuhlmann, G.: Reinforcement learning for robocup soccer keepaway. Adaptive Behavior 13(3), 165–188 (2005)
19. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)