Hierarchical Multi-Agent Deep Reinforcement Learning to Develop Long-Term Coordination

Marie Ossenkopf
Distributed Systems Group,
University of Kassel
Kassel, Hessen, Germany
mos@vs.uni-kassel.de

Mackenzie Jorgensen
Dept. of Computing Sciences,
Villanova University
Villanova, PA, USA
mjorgen1@villanova.edu

Kurt Geihs
Distributed Systems Group,
University of Kassel
Kassel, Hessen, Germany
geihs@uni-kassel.de

ABSTRACT

Multi-agent systems need to communicate to coordinate a shared task. We show that a recurrent neural network (RNN) can learn a communication protocol for coordination, even if the actions to coordinate lie outside of the communication range. We also show that a single RNN is unable to do this if there is an independent action sequence necessary before the coordinated action can be executed. We propose a hierarchical deep reinforcement learning model for multi-agent systems that separates the communication and coordination task from the action picking through a hierarchical policy. As a testbed, we propose the Dungeon Lever Game and we extend the Differentiable Inter-Agent Learning (DIAL) framework [3]. First we prove that the agents need a hierarchical policy to learn communication and actions and, second, we present results from our successful model of the Dungeon Lever Game.

KEYWORDS

Hierarchical Learning, Multi-Agent Systems, Agent Communication, Deep Reinforcement Learning

ACM Reference Format:
Marie Ossenkopf, Mackenzie Jorgensen, and Kurt Geihs. 2019. Hierarchical Multi-Agent Deep Reinforcement Learning to Develop Long-Term Coordination. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3297280.3297371

1 INTRODUCTION

In recent decades, autonomous agents have repeatedly beaten human experts or at least achieved human-level skills at complex games [11]. These include Atari games [11], Starcraft [12], and Go [16]. This is no small feat for AI as deep reinforcement learning continues progressing.

The world of gaming and autonomous agents employs multi-agents as well as single agents. Although deep reinforcement learning algorithms are successful in many single-agent scenarios, as the problems become more complex and multiple agents in a single-environment must interact, the algorithms do not scale well [9].

To succeed in shared tasks, multi-agent systems must communicate and coordinate. Through reinforcement learning, these systems can learn a communication protocol enabling cooperative actions. In Foerster et al.'s work, a deep RNN allowed for agents to learn to communicate in order to solve a number of riddles [3]. Similar work on multi-agent communication includes the CommNet framework [17], utilizing a deep feed-forward network architecture to solve multiple games. CommNet is unable though to keep a memory over several action steps because it has no recurrent structure. The communication used in the research of multi-agent teams working together to play StarCraft games focuses more on memory [13]. Still, the communication in CommNet and Starcraft is permanently available and the network decides on an action directly after communicating. However, humans are able to coordinate a whole sequence of actions, even if they can no longer continue to talk throughout the execution. With our approach, we pursued a solution for multi-agents to act similarly to the human behavior of coordinating a sequence of actions after communication has ended. This has not yet been solved in multi-agent reinforcement learning.

To model this situation, we propose a toy example inspired by a typical dungeon riddle called the Dungeon Lever Game. To open a door, two levers on opposite sides of a dungeon must be pulled at the exact same time. Communication is only possible as long as the two players did not start walking towards their lever. In this paper, we argue that it is possible to let a RNN develop a communication protocol and coordinate an action after an interrupting subtask if and only if the learning of the communication task is separated from the interrupting subtask (in this case the walking part). In order to do this, we implemented a deep RNN that separates the communication from the other actions in a hierarchical fashion.

For our model, we chose DIAL [3]. DIAL is especially useful for developing a communication protocol because the error can be backpropagated through the communication channels. This provides the agents with feedback on how helpful their messages were to the other agent. Furthermore, DIAL allows the agents to store past communication in the cumulated internal state.

When testing our Dungeon Lever Game with the original DIAL, the communication feedback faced the vanishing gradient problem, making it extremely difficult for the agents to learn communication and action protocol at the same time [5]. The agents cannot solve the game because solving the action-picking task weakens the communication learning when only a single RNN learns. There are many real-world tasks in which it is vital that the agents are able to coordinate actions outside of the communication range.

Through our findings, we argue that agents who need to coordinate a task that will occur after an interrupting subtask face the same problem as single agents who face a high temporal difference in task horizons. To solve this problem, we suggest a hierarchical network when facing a task where the communication and the actions to coordinate are separated by a sequence of independent actions. Hierarchical learning includes two RNNs where an internal critic rewards the lower network for following the goals upon which the upper network chose its course. Hierarchical multi-agent deep reinforcement learning provides a solution to the issue of deep reinforcement learning algorithms scaling to more complex problems [9]. We propose a hierarchical version of DIAL to separate the communication learning from independent subtasks. Further on, we used imitation learning [15] to counter the problem of sparse reward in some of the experiments elaborated in Section 4.

The main contributions that we outline in this paper are:

- We developed a hierarchical deep reinforcement learning model that learns how to solve the Dungeon Lever Game through separation of subtasks. The model reveals that agents are able to coordinate their actions even after they are out of communication range.
- We show that the model is only able to learn communication and coordination if hierarchical learning is used because otherwise the gradients get blurred by the interrupting subtask.
- We present experiments including the Dungeon Lever Game and multiple derivations that allow us to identify the cause for the lack of success of a single RNN.

The remainder of this paper is organized as follows: Section 2 begins with the official formulation of deep Q-Learning and the vanishing gradient problem; Section 3 discusses the related work; Section 4 explains our toy example and describes the experiments we ran to test our hypotheses; Section 5 reveals our findings; Section 6 highlights our conclusions and future work.

2 PROBLEM FORMULATION

Reinforcement Learning An agent in a reinforcement learning setting can fully or partially observe its current state $s_t \in S$ at every time step t, is able to choose an action $a_t \in A$ through a policy $\pi(s) = a$, observe a reward r_t at time step t, and move to the next state s_{t+1} [4]. The agent's goal in its environment is to maximize the expected discounted reward: $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$, where γ is the discount rate.

Q-Learning Q-Learning is part of the value function approach to reinforcement learning. The Q-function expresses the current knowledge of the expected reward. It maps the expected future reward to every state-action pair: $Q^y(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a]$.

Deep Q-Networks (DQNs) DQNs approximate the Q-function using a non-linear neural network which is based on Q-learning. The network represents $Q(s,a;\theta)$ by the parameter θ. The network (θ in this case) learns to reduce the loss function:

$$L(\theta) = \mathbb{E}[(r + \gamma \max_{\mathbf{a'}} Q(s', a'; \theta') - Q(s,a;\theta))^2] \tag{1}$$

where s' is the next state, a' is the next action, and θ' is the target net. This is achieved through backpropagation and gradient descent[12].

Deep Recurrent Q-Learning While Deep Q-Networks work best in situations with full observability, sometimes the environment is only partially observable for the learning agent. Deep recurrent Q-networks pose the solution for single agents in not-fully observable settings [4]. In these environments, the s_t is hidden, so the agent only knows an observation o_t that correlates with s_t. In the architecture, Q is approximated by a RNN which has the ability to keep an internal state and collect observations over a time sequence. To model this, the network's hidden state h_{t-1} results in $Q(o_t, h_{t-1}, a)$; in this instance, we leave out θ's reliance on Q.

Vanishing Gradient Problem RNNs face the dilemma, that the net needs to allow gradients decay over the time steps to let new input influence the internal state. As the total gradient is the sum of long-term and short-term influences, the short-term influences can dominate, making the learning of long-term influence difficult to impossible [2]. In a fully connected RNN, the backpropagated error flow is scaled by the following equation, when the node l_0 which error we want to receive is y time steps in front of the considered node l_y at time step t. In the following expression, f is the activation function, net is the forward propagated input, and w is the corresponding weight [5]:

$$\prod_{m=1}^{y} f'_{l_m}(net_{l_m}(t-m)) w_{l_m l_{m-1}} \tag{2}$$

The error flow disappears if

$$p(m, l_m, l_{m-1}) := |f'_{l_m}(net_{l_m}(t-m)) w_{l_m l_{m-1}}| < 1.0 \tag{3}$$

for all m the Eq. (2)'s maximum product exponentially decreases with y.

Long Short-Term Memory Network (LSTM) A long known solution to the vanishing gradient problem is the LSTM. LSTMs are specifically created to remember information over an extended period of time [6]. The internal state is only changed by the LSTM's gates adding or removing information from the state. The LSTM gates themselves are sigmoid neural network layers with pointwise multiplication operations. There are three gates which have control over the internal state. The forget gate decides what information needs removing from the internal state. The input gate decides what information should be added to or updated with the internal state. The output gate decides what filtered version of the cell state should be released.

3 STATE OF THE ART

We will discuss the current research findings in the areas of multi-agent communication learning, hierarchical learning, and imitation learning. The work in multi-agent communication and hierarchical learning aided us in shaping our architecture, while imitation learning improved the number of correct examples for some of our experiments to find the shortcoming of the non-hierarchical RNN.

3.1 Multi-Agent Communication Learning

Communication is a crucial aspect of every person's life, whether a message is delivered through body language, voice, or text. As multi-agent systems become more important and must complete exponentially harder tasks, communication is necessary to succeed. The goal of multi-agent communication is to achieve a maximum shared utility. Research on AI communication dates back to the 20th century. Related work includes research by Werner and Dyer who

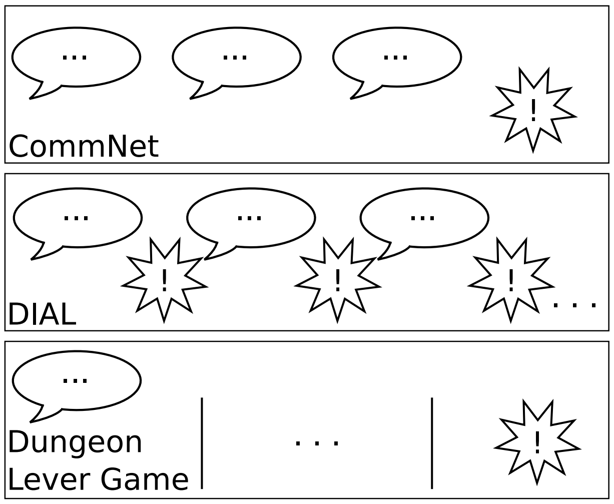

Figure 1: Difference in the sequence of communication and coordinated actions of CommNet, DIAL, and the Dungeon Lever Game we propose. The quotation bubbles symbolize communication between agents, the stars with bang signs symbolize the action taken, and ". . ." without a quotation bubble symbolizes unrelated actions.

conducted predator and prey agent experiments with communication [18]. They found that when the prey agents communicated, all prey agents benefited because of their coordination. There was a shift in 1994 when Saunders and Pollack argued that RNNs allowing for several communication channels were superior to the previous agent communication research with discrete communication [14]. Communication research debates continue to today on whether discrete communication (similar to how natural language works) or continuous communication (which calculates gradients and signals, accelerating learning) is more beneficial for multi-agent systems.

More recent multi-agent communication work includes DIAL and Reinforced Inter-Agent Learning (RIAL) [3]. These use deep reinforcement learning algorithms to learn multi-agent communication. RIAL is a recurrent network with independent deep Q-learning for communication and action-picking using decentralized learning, while DIAL uses centralized learning. The agents in DIAL can share messages without limitation during training but, during testing, the communication is limited. DIAL allows for both discrete and continuous message communication. DIAL proved more effective than RIAL, but both succeeded in solving the complex riddles. DIAL was the model for Jorge et al.'s successful communication sequence of dialogue remembered by both agents to solve the Guess Who game [7]. Furthering the idea of dialogue, Lewis et al. developed a model that learns to negotiate and reason with other agents [10]. These approaches highlight that longer communication sequences can be kept in the agent's memory. The common feature with these architectures is that an action is taken directly after the communication happens; thus, the agents do not have a period of time without communication before the action is coordinated. However, in a real-world application, robots that lose connection to each other would still need to coordinate.

In a different architecture, CommNet uses a deep feed-forward network with a continuous vector accessible through the communication channel [17]. CommNet's controller is the neural network which connects inputs to all agents' actions. It gives a solution that works for multiple games through continuous communication in homogeneous multi-agent systems.

Another architecture works with heterogeneous multi-agent systems, resulting in human-level coordination of tasks while playing StarCraft [13]. The Bidirectionally-Coordinated Nets allow for dynamic communication that prioritizes communication between certain agents who must act together in a sequence of actions. The framework can work with many agents because the state of the game space is the same across all agents. After the agents are trained, they successfully avoid collisions and learn to collaborate between agent-types, while also learning the skills or tactics of hit-and-run, cover attacks, and focus fire. These skills and behaviors mimic the top human players at a high-level team combat game, highlighting the accuracy of the Bidirectionally-Coordinated Nets architecture.

The frameworks described here only work for continuous communication or when actions are taken immediately after communication. The difference is depicted in Figure 1. Thus, the architectures do not teach agents to memorize communication over a series of actions in order to correctly act in their environment. Our project offers a solution for agents recalling and using past communication (since they can only communicate for a limited time) to pick actions in order to solve a task with coordination.

3.2 Hierarchical Learning

Hierarchical reinforcement learning (HRL) tests well with agents that must learn how their actions affect the overall goals of their environment with a limited amount of feedback [8]. Instead of making a single framework learn multiple tasks, which can lead to poor results, HRL splits the learning-workload into a sequence of separate sub-tasks. The meta controller and controllers decide what sub-tasks to complete at different stages [9]. This optimizes the learning of the specific tasks for the agents in their environment.

However, multi-agent systems do not scale well with HRL, thus, Kumar et al. proposed Federated Control with Reinforcement Learning (FCRL). FCRL integrates multi-agent deep reinforcement learning with HRL. In FCRL the upper policy decides when an agent should talk to another agent, which is then pursued by the lower policy. In our framework, we allow the communication (in the upper policy) to decide which goals should be pursued by the lower policy to separate the communication learning from the other subtasks. Hierarchical reinforcement learning aids our agents in learning how to communicate and how to pick actions for the greatest reward, by dividing the learning goals.

3.3 Imitation Learning

Imitation learning is often utilized in robots when teaching a robot a certain motion or set of actions. It has been successful in robots learning human-like motions [15]. The use of imitation learning for agents learning a "perfect behavior" benefits the autonomous agent's greatly, by them receiving a fully optimized reward or reaching a goal. This correct behavior provides the agents with a model of what actions should be taken at a certain point. In complex environments with a sequence of actions, imitation learning speeds up the learning of agents. Imitation learning also solves the problem of sparse reward in some of the experiments described below.

4 EXPERIMENTS

We used the DIAL and RIAL frameworks provided by Foerster et al. [3]. Our games and model are written in Lua [1]. The model is a LSTM with discretize/regularize units as communication channels to allow for inter-agent backpropagation. We tested if the state of the art algorithms for multi-agent systems could learn a communication protocol, when the actions that need coordination were in the future and outside of the communication range. We chose a simple dungeon riddle as a toy example.

4.1 Dungeon Lever Game

The Dungeon Lever Game includes two agents each of which has a lever in its environment. Each lever has a specified position in the environment called the lever position. The goal of the agents is to pull their respective levers at the same time and as quickly as possible. To be successful, the agents must communicate since their lever positions differ, resulting in an unequal amount of time required to reach the lever. Thus, they need to know when the other agent will arrive at their lever to pull it jointly. The crucial point about the communication is it can only happen as long as the agents stay in the same position at the starting point. Thereby, we enforce the agents to learn a policy, where they communicate their own lever position and take actions after that, which allow for them to each pull their lever simultaneously.

[1]https://github.com/mjorgen1/learning-to-communicate

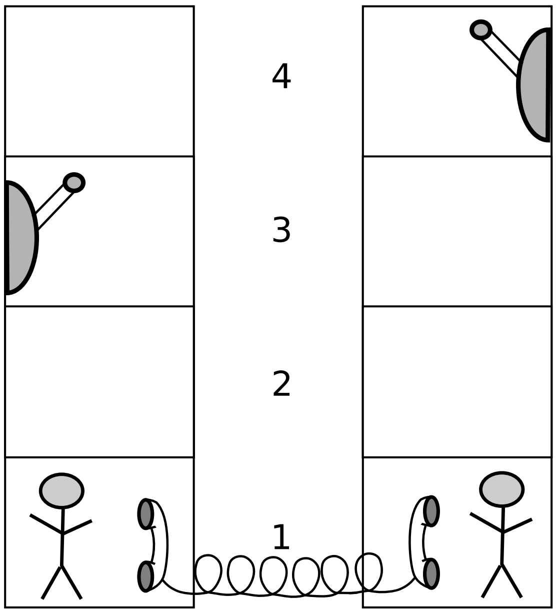

Figure 2: Sketch of the Dungeon Lever Game: Communication is only possible on the first tile. The levers must be pulled at the same time step. One walking action takes an agent one tile forward.

The agents have the option to stay in the same position, move forward one step, or pull the lever. On every step, the agents receive their current position and their lever position as an input state. Since coordination is the task's main objective, we chose a completely shared reward [1]. We use a shared reward because no matter what one agent chooses to do, it cannot be rewarded if its action does not fit the other agent's action. We first assigned the reward independent from the time the agents took to solve the task, resulting in the agents only learning to wait the maximum possible time and pull once they reached that time. Through this approach, the agents avoided communication and coordination. We, therefore, decreased the reward over time to encourage a time-efficient solution and communication.

The training and test datasets are produced in the same way: the agents start the game on the first tile and take actions with their current policy until at least one agent pulls the lever. The error is the difference between received reward and the expected reward given by the Q-network. In training, this is done until the batch size is reached, then the batch normalized error is backpropagated.

Game Elements

- The state of the world consists of the states of the agents:
$$S = (s_1, s_2)$$
- The state of one agent consists of its own position and the position of its respective lever:
$$s_a = x_a, x_{La}$$
- The observation of time t consists of the agent's state and the communication of the other agent one time step before:
$$o_{a,t} = (s_{a,t}, comm_{2,t-1})$$
- The deterministic policy takes the current observation and the last hidden state to derive an action u, a communication, and a new hidden state h:
$$\pi(o_{a,t}, h_{a,t-1}) = u_{a,t}, comm_{a,t}, h_{a,t}$$
- The action set u consists of three elements:
$$u = \{stay, forward, pull\}$$
- Since the game is fully deterministic, the transition function is defined like this:
$$T(x_a, u_a = forward, x'_a = x_a + 1) = 1$$
$$T(x_a, u_a = stay, x'_a = x_a) = 1$$
- The game is a partially observable Markov Decision Process as the reward depends on the combined actions and states of the two agents:
$$R(u_1, u_2) = \begin{cases} 1 & u_1 = u_2 = \text{pull} \wedge x_1 = x_{L1} \wedge x_2 = x_{L2} \\ 0 & \text{otherwise} \end{cases}$$
- The agents' objective is to maximize the expected reward over time: $\pi_{opt} = \arg maxE\left(\sum R_{a,t}\right)$

4.2 Hypotheses

We initially began using an LSTM with DIAL, but the network was unsuccessful in solving the Dungeon Lever Game. To find the reason why the given model and algorithm were unable to find a solution, we tested four hypotheses:

(1) The model cannot map one of the underlying tasks.
(2) The model cannot map the combined task as a whole.
(3) The algorithm cannot converge due to sparse reward.
(4) The algorithm cannot converge due to vanishing gradients.

4.2.1 Hypothesis Test 1. To test the first hypothesis, we tested the following derivations of the Dungeon Lever Game:

(a) No limit to communication. If there is no limit on the communication range and the agents can communicate through the whole episode, the game strongly resembles the examples already solved by state of the art algorithms [13, 17].
(b) Single-agent memory test. To test the ability of the LSTM to memorize information for a sufficient time, one agent observes its lever positions only at the first time step and needs to learn to pull it quickly.
(c) Long-term communication test. In this derivation of the Dungeon Lever Game, the agents do not need to walk towards the lever, but the correct motion sequence is carried out by the game itself. Still, the agents can only communicate in the first time step and need to learn how long to wait and when to pull. This game tests the possibility of learning a communication protocol that coordinates actions carried out in the future but without an interrupting action sequence.

4.2.2 Hypothesis Test 2. To test the second hypothesis, we used RIAL instead of DIAL combined with imitation learning: we provided positive examples of communication actions together with the respective action sequence for walking and pulling. Therefore, the algorithm does not need to establish a protocol itself but can follow the demonstrated protocol and map it.

4.2.3 Hypothesis Test 3. To test the third hypothesis, we calculated the chances for rewards and provided the algorithm with enough positive examples through imitation learning, without providing a protocol.

4.2.4 Hypothesis Test 4. To test the fourth hypothesis, we compared the gradients that reach the communication in the Dungeon Lever Game with the ones in (c) the long-term communication test.

The results are elaborated in Section 5.

4.3 Hierarchical DIAL

To tackle the problem of vanishing gradients and the limitations of the state of the art algorithms, we decided to follow the h-DQN model as proposed by Kulkarni et al. [8]. The resulting architecture is in Figure 3.

We divided the Dungeon Lever Game in an upper and a lower policy; the first policy is learned by the meta controller, and the second is learned by the controller and rewarded by the internal critic. The meta controller receives the same reward as in the Dungeon Lever Game but it does not need to decide on the walking action sequence. After communicating, the controller sends the goal—the time step at which the agent is supposed to pull its lever—down to the controller. The controller itself does not care about cooperating with the other agent but is rewarded for pulling its lever at the time step specified by the meta controller. Hence, the meta controller must choose the goal in unison with the other agent to reach multi-agent coordination.

We reduce the learning task for the meta controller to a task already proven learnable by many algorithms. The action that decides over the success of the team is taken directly after the communication and without an interrupting action sequence. The action sequence is outsourced to the lower policy. The error signal does not need to be propagated through several time steps in the RNN to reach the communication, hence a communication protocol can be developed.

5 RESULTS

In this section, we will elaborate the results of the experiments that we carried out to test the hypotheses explained in Section 4. We find the reason why a single RNN is unable to learn a communication protocol when the communication and the coordination task are interrupted by an independent action sequence.

5.1 Hypothesis 1

In Table 1, the size of the networks that solve the three derived games are either labeled the fastest or the smallest network. These results highlight that the subtasks themselves are not challenging. Hence, it is neither a problem to map nor to learn the subtasks with the given algorithm and model.

5.2 Hypothesis 2

The Dungeon Lever Game can be solved by a single RNN when using RIAL combined with imitation learning. We provided positive examples of communication actions together with the respective action sequence for walking and pulling. Following the demonstrated protocol, the algorithm can then establish a coordinating policy, even though the communication and the action to coordinate are interrupted by the walking subtask. Consequently, a single RNN can map the policy necessary to solve the Dungeon Lever Game; although, it cannot develop a communication protocol itself.

5.3 Hypothesis 3

A typical problem in reinforcement learning lies in the sparse reward. The Dungeon Lever Game indeed faces the problem of sparse reward as the chance of receiving the reward with completely random actions is roughly 1%. To cope with this problem, we used imitation learning on DIAL, by presenting the correct action but not specifying the communication, to provide enough positive examples. This still did not result in evolving a communication protocol but the network learned a naive solution as a result. The agents hoped that the other agent had the same lever position as themselves and always pulled the lever as soon as they reached their lever position. The communication did not mirror their lever positions in any way. Thus, the learning progress in the Dungeon Lever Game is not primarily hindered by the sparse reward.

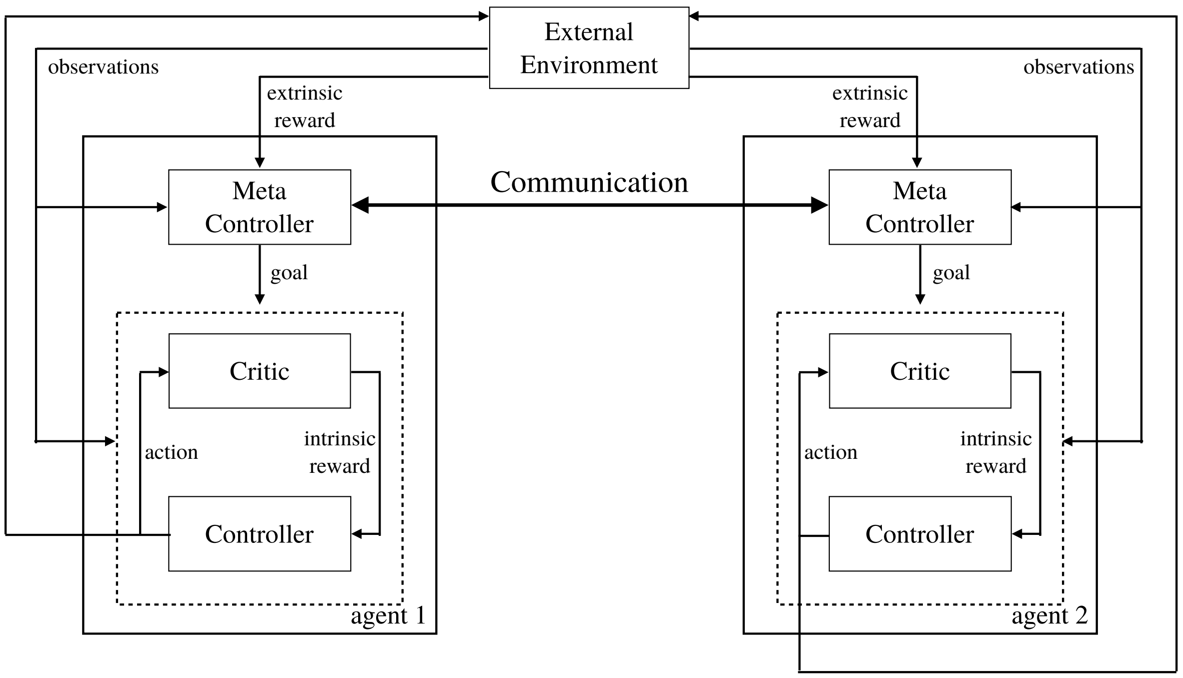

Figure 3: Overview of the architecture of the hierarchical DIAL: The communication works over continuous messages exchanged through a channel that enables backpropagation of gradients towards and from the other agent.

Table 1: Smallest and fastest replicable net each achieving at least 95% on the test set in the three derived games, batch size: 32 trials per epoch.

Game	Case	Average Performance	Average Epoch	RNN Size	RNN Layer
a) No limit to communication	Fastest	0.995	2180	512	1
a) No limit to communication	Smallest Net	0.995	2600	256	1
b) Single-agent memory test	Fastest	0.982	300	512	2
b) Single-agent memory test	Smallest Net	0.961	1370	64	1
c) Long-term communication test	Fastest	0.999	900	512	1
c) Long-term communication test	Smallest Net	0.959	6320	64	1

5.4 Hypothesis 4

The last hypothesis proved the reason why the algorithm could not develop a communication protocol and solve the game with a single RNN. We looked at the gradients that reached the communication in c), the long-term communication test. The communicated values and their respective gradients can be seen in Figure 4. They were taken amidst learning (after 1600 epochs and 80% performance on the test set), where the different preferences for communicating the possible lever positions are recognizable. In Figure 4, the sections a), b), and c) reveal that the algorithm already starts to converge to distinct messages for the three positions. The messages that will be developed in this example are shown in Table 2.

Parts d) and e), in Figure 4, show the communication in trials where the agents did not receive the reward and it becomes apparent that the wrong action comes from a misunderstood communication. Section d) resembles more of a 2 than a 3 and the second agent makes the mistake of pulling too early; accordingly, the agent also understood a 2 instead of a 3. On the other hand, in part e), the second agent waits too long, because it understood a 4 when a 3 was communicated. The gradients of these miscommunications point in the direction that would be understood as a 3 and, thereby ensure that the algorithm will converge to distinct and clear messages.

The same patterns in communication and the gradients should be visible in the Dungeon Lever Game if it was to develop a communication protocol. In our tests, it became obvious that the gradients of the Dungeon Lever Game do not allow communication learning.

Table 2: Converged messages in long-term communication test.

lever position	comm-bit 1	comm-bit 2
2	-	-
3	-	+
4	+	+

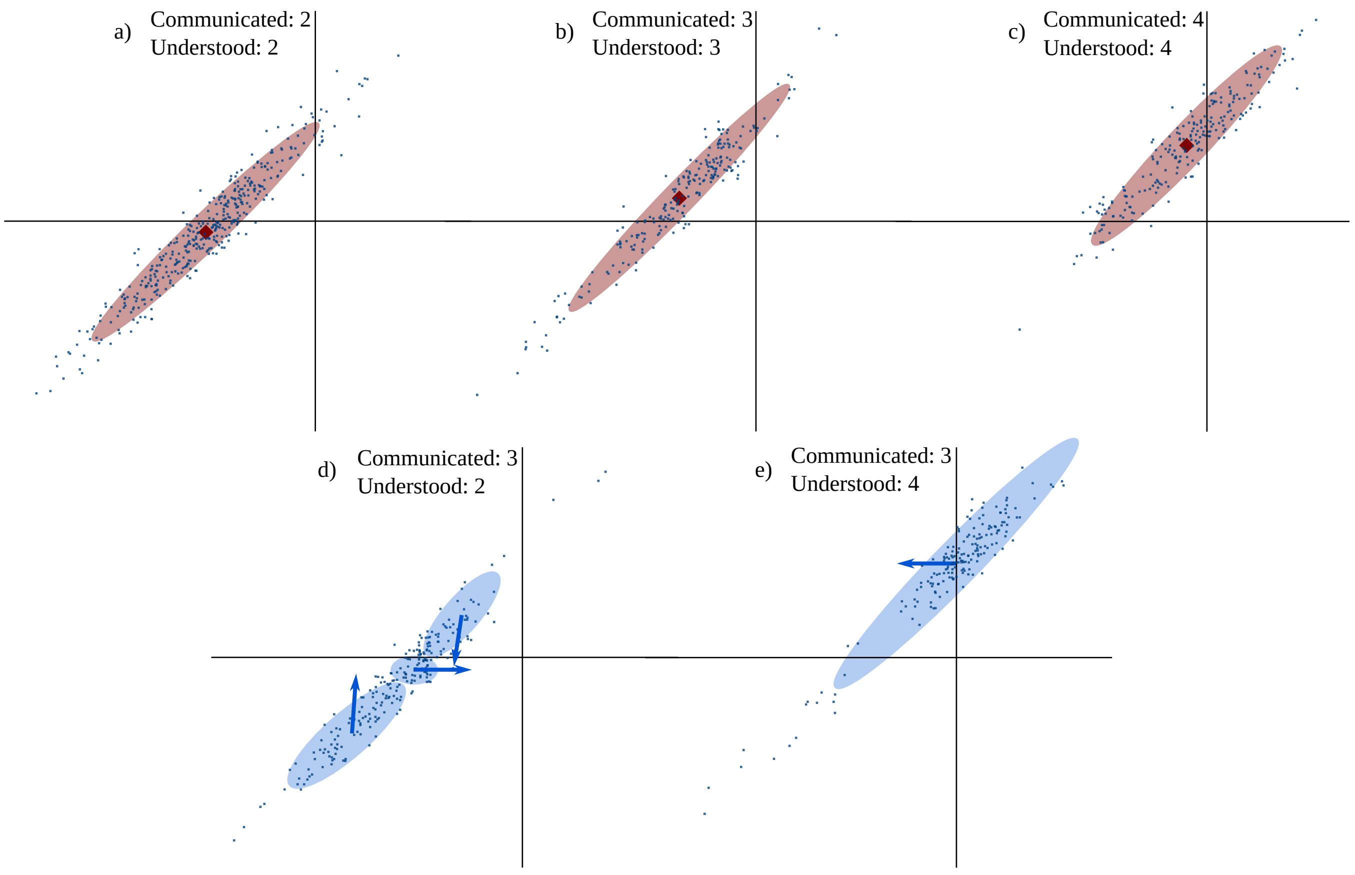

Figure 4: The communication instances sent from one agent to the other one, grouped by the first agent's intention and the second agent's reaction. Every blue dot is a one two-channel message. The red ellipses show the standard deviation. The two dimensions represent the two communication bits. Parts a), b) and c) show instances of successful communication: the second agent pulled the lever at the time that fits the first agent's lever position. Part d) shows instances where the first agent needed to communicate the lever position 3 but the second agent pulled too early, because it understood a 2. Part e) shows instances where the second agent waited too long because it understood a 4. The arrows show the average direction of the gradients that reach the communication channel for messages in the respective blue ellipses.

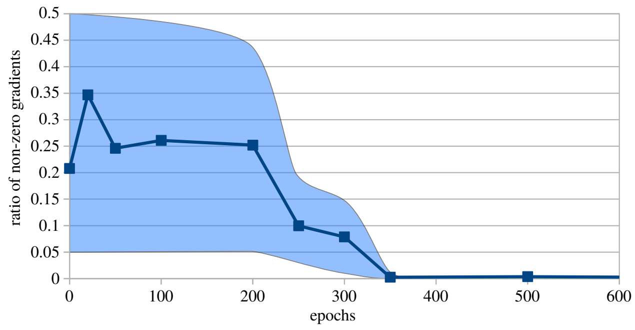

Figure 5: The proportion of gradients with standard deviations reaching the communication through backpropagation, which is non-zero in the Dungeon Lever Game.

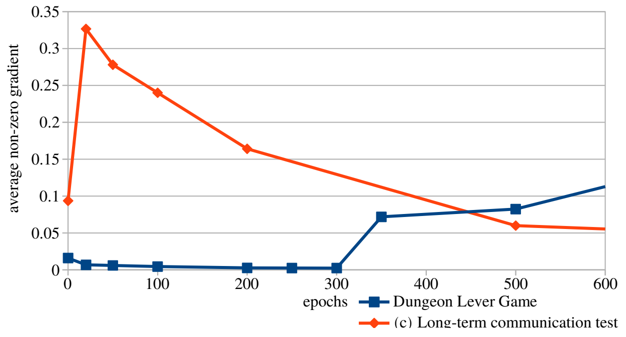

Figure 6: The average non-zero gradient reaching the communication through backpropagation. Comparison of Dungeon Lever Game and the long-term communication test.

Figure 5 shows that a great partition of the gradients is initially zero when reaching the communication channel. Figure 6 reveals that even the gradients that are non-zero are ten times smaller than the initial gradients in the long-term communication test and they shrink quickly. After approximately 350 epochs, almost all gradients fell to zero, making it impossible to adapt the communication in any direction. The rise of the average non-zero gradient is due to the fact that there was only one non-zero gradient left in 3000 trials, distorting the data. These vanishing gradients arise through the way the LSTM works. To store important information for a longer time, without exponential decay in the gradients and still take in new information, the LSTM forgets parts of the internal state on purpose and only lets some of the input influence the internal state. Because the necessary actions in the walking sequence are not influenced by the communication, the LSTM forgets the communicated information during the walking or does not even take it into the internal state in the beginning. In contrast, in the long-term communication test, no actions are taken that are not related to the communicated information. Therefore, the LSTM values the information as important and does not make the decision to forget it.

5.5 Hierarchical Protocol Result

We divided the task of the Dungeon Lever Game into two subtasks that resembled tasks that have been solved before easily. As we expected, neither the upper nor the lower policy is difficult to learn. The only difficulty in combining them is to find the right exploration and learning rate for the upper policy to enable the lower policy to converge before the upper policy starts exploiting. The performance of the policies is shown in Table 3.

Table 3: Performance of the hierarchical policy, batch size: 32 trials per epoch.

Policy	Average Performance	Average Epoch
Upper	1.0	1080
Lower	0.999	780
Combined	0.996	1510

6 CONCLUSION

We conveyed that a single RNN is able to learn to coordinate actions that lie outside of the communication range when no other actions need to be taken in between. When a sequence of independent actions separates the communication from the actions to coordinate, the gradients become too blurred to enable the development of a communication protocol. In this paper, we propose a hierarchical version of the DIAL framework. The hierarchical policy solution splits up the learning tasks, protecting the communication gradients from interference by the learning of the action sequence. Thus, the task can be split into two subtasks that are proven learnable.

The contribution of this paper is one step on the long path of enabling learning algorithms, to adapt to difficult collaboration tasks, by learning an appropriate communication policy. To understand what makes an algorithm capable of this, we need to stepwise toughen the task and then focus on improving the algorithm again. The hierarchical model we proposed can be improved by adding to the hierarchy and number of controllers. A third controller could decide when the agents talk and when they start walking. This would enable the algorithm to also learn the optimal length of communication to solve a given task. In our current task design, the agents only need to tell one another their lever position, without awaiting a confirmation. However, to strengthen the task, we want to make it a necessity to use a two-way handshake, to reach more complex agreements. We plan to continue improving our framework's hierarchy and communication scalability to let agents learn how to agree without human example.

ACKNOWLEDGMENTS

The authors would like to thank the DAAD RISE program for funding Mackenzie Jorgensen to work in the Distributed Systems Lab in Kassel and Joannis Assael for his help with the DIAL framework.

REFERENCES

[1] Adrian K Agogino and Kagan Tumer. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems* 17, 2 (2008), 320–338.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[3] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems.* 2137–2145.

[4] Matthew Hausknecht and Peter Stone. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *2015 AAAI Fall Symposium Series.*

[5] Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 02 (1998), 107–116.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Emilio Jorge, Mikael Kågebäck, Fredrik D Johansson, and Emil Gustavsson. 2016. Learning to play guess who? and inventing a grounded language as a consequence. (11 2016).

[8] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems.* 3675–3683.

[9] Saurabh Kumar, Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. 2017. Federated Control with Hierarchical Multi-Agent Deep Reinforcement Learning. (2017).

[10] Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or No Deal? End-to-End Learning of Negotiation Dialogues. (2017), 2443–2453.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. (12 2013).

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[13] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. (03 2017).

[14] Gregory Morris Saunders. 1994. *The evolution of communication in adaptive agents.* Ph.D. Dissertation. The Ohio State University.

[15] Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* 3, 6 (1999), 233–242.

[16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[17] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems.* 2244–2252.

[18] GM Werner and MG Dyer. 1991. Evolution of communication in artificial systems. *Artificial Life II, Addison-Wesley, Redwood City, CA* (1991), 659–682.