Vision and Communication in Multi Agent Reinforcement Learning

Bachelor´s Thesis in Artificial Intelligence
7,5 ECTS

July 1, 2022

Author:
Martijn Makaske
6778933

Supervisor:
Wijnand van Woerkom
Second Supervisor:
Henry Prakken

Bachelor Kunstmatige Intelligentie
Faculty of Humanities
Utrecht University

Abstract

Communication is an important element for improving performance in multi agent reinforcement learning. The study of Singh et al. (2019) proposed an algorithm called IC3Net which is capable of learning agents when to communicate in different scenarios. This research aims to confirm the results from Singh et al. (2019) and gain a better understanding about the relation between vision, communication and performance. This is done by installing the algorithm and replicating experiments from Singh et al. (2019). In addition, new experiments are conducted to investigate the influence of vision. These experiments make a comparison between trained models with poor and good vision. When replicating, the trained models show similar performance and communication. Furthermore, results show that models trained with better vision communicate less. Likewise, results show models with better vision achieve better performance. Interestingly, agents with sufficient vision choose to stop communication. These results confirm the results from Singh et al. (2019) and thus support their conclusion that IC3Net is able to learn when to communicate at scale. This research concludes that an increase in vision leads to less communication as agents are provided with another way of observing their environment. This also proves agents only communicate when necessary. Lastly, this research also concludes that an increase in vision results in an increase in performance, depending on the nature of the task.

Table of contents

1 Introduction

Communication plays a crucial role in both human and computer intelligence. It has enabled humankind to develop a high level of intelligence, which is unmatched on earth. Through communication humans are able to learn from the mistakes of others. Although human perception is limited to only a small area around them, people can still be aware of what is happening at the other side of the world. In the last decades, this key element for intelligence has also found its way into computer science. Particularly in multi agent systems with partially observable environments, comparable to our own world, communication has resulted in great improvement in intelligence, just as with humans. Through sharing their internal states, agents improve both their performance and learning rate.

Since the rise of machine learning in artificial intelligence, the subfield reinforcement learning has also gotten lots of attention. The increased amount of research in this field has made it one of the most promising methods for modelling intelligent behavior. With high performance in a wide range of applications (Li, 2019), reinforcement learning has lived up to this promise and shown remarkable results. It is now being used to solve complex problems by companies across all fields (Oliveira et al., 1999). These techniques enable computers to reach humanlike performance in increasingly more complex games, such as chess (Campbell et al., 2002) and more recently go (Silver et al., 2016) and show the true potential of reinforcement learning. However, a large amount of these success stories were single agent environments, with a clear goal. When reinforcement learning is applied to help solve real world problems, problem solving often requires cooperation between a large number of agents, which is known as multi agent learning. The number of multi agent systems and applications has been growing significantly (Lazaridou et al., 2016; Mordatch & Abbeel, 2017), stretching from networks in a manufacturing plant to optimizing traffic flow (Walraven et al., 2016).

Multi agent reinforcement learning, or MARL, is a group of autonomous agents, learning in a common environment (Buşoniu et al. 2010). Just as in the real world, most environments are partially observable for the agent, so the agents are often forced to cooperate to solve the problem. Studies with communicating agents have shown significantly increased performance over agents which do not communicate with each other (Sukhbaatar, 2016). MARL can be split up in three scenarios: cooperative, competitive and a mixture of these two. The first of these is studied most intensively, and often learns a solution just as humans learned to cooperate. Nonetheless, most real life scenarios can better be described as mixed scenarios, where humans pursue individual goals alongside a shared goal. This points out the importance of knowing when to communicate. It is clear that communication can improve performance, but it can also have a negative effect on the capabilities of agents. For instance, when playing a hide and seek game, it is important that the hiding agent does not communicate its location to other agents. Meanwhile, when in a partially observable environment, the searching agents should communicate to search a bigger area in a shorter amount of time.

This research aims to develop a better understanding about communication in multi agent systems by further exploring an algorithm proposed by Singh et al. (2019). Besides communication, the research also focusses on the influence of vision in MARL.

Before the proposed algorithm of Singh et al. (2019) can be further analyzed, it is important to verify that the algorithm is correctly implemented on the machine which performs the analysis. After installing the correct dependencies and setting up the algorithm, two experiments from the article of Singh et al. (2019) can be replicated. After that, this research looks to extend the understanding about the influence of vision, by conducting different experiments than in the paper. The aim is to gain a better understanding about the influence of vision on both performance and communication. In contrast to the experiments already conducted by Singh et al. (2019), different values for vision are used, thereby providing a more realistic scenario. First, two experiments with a difference in vision are conducted in the same predator-prey environment. The communication in both trained models can then be compared. Second, again two experiments with a difference in vision are conducted, but this time in the traffic junction environment. Now a comparison can be made between the performance of both models, while also looking at the learned communication. In both experiments, the resulting behavior, and possible differences are also studied. In summary, this research aims to answer the following questions:

1. Can we replicate the experiments from Singh et al. (2019) using the implementation they provide? Essentially asking, is the algorithm correctly installed on the machine?
2. How does increasing the vision in the predator-prey environment affect the communication and behavior of the model? This gives useful insight in the relation between vision and communication.
3. How does increasing the vision in the traffic junction environment affect the performance and behavior of the model? This clarifies the influence of vision to the agents' performance.

All trained models will be analyzed using different measures such as success rate, communication, average steps taken and rewards.

It is hypothesized that replicating experiments from Singh et al. (2019) gives similar results, thereby confirming their findings. It is also expected that an increase in vision leads to less communication, because agents are able to collect more information on their own and are therefore less reliant on the information of others. Lastly, it is expected that an increase in vision results in an increase in performance. This is expected in particular in the traffic junction environment, since the direct surroundings of an agent are much more important here rather than the situation further away.

This article is divided into several sections. Firstly, 2 Related work looks at earlier research done in this field and other approaches for communication in MARL. Secondly, 3 Model and Environments explains the algorithm IC3Net and describes the environments used in the experiments. Following, the exact experiments conducted are explained in detail in 4 Experiments. This section gives a clear overview of all conducted experiments, defines the exact parameters and explains certain choices made in the process. Thereafter, 5 Results analyzes the results obtained from training the models in the given experiments. The section provides a description of the exact results, tables for comparison and graphs giving an overview of important results. With the results, we can interpret them and look at possible limitations in 6 Interpretation and Discussion. 7 Conclusions and Future Work, closes the article by drawing conclusions from the results and suggesting possible future research.

2 Related work

An early approach for multi-agent reinforcement learning is done by Tan (1993), by using an independent controller for each agent. With an independent controller, agents train their own policies without communication with other agents. This research shows that cooperative agents can outperform independent agents, although be it at the expense of slower learning. However, as most research, this approach focused only on cooperative tasks and does not scale to different scenarios.

The same holds for more recent and performing work. For instance, the research by Omidshafeie et al. (2017) presents a decentralized algorithm, using deep recurrent neural networks, capable of solving cooperative tasks. Recurrent neural networks are often used when solving partially observable reinforcement learning problems and have proven to outperform conventional methods (Schäfer et al., 2005). The work of Foerster et al. (2016) also uses recurrent neural networks and proposes different approaches for learning a communication protocol. Although this research does successfully introduce learning agents to communicate, it is still not easily scalable and is restricted to only cooperative scenarios.

The research by Sukhbaatar et al. (2016) is again bound to cooperative tasks, but proposes a model using multiple communication cycles to improve performance. Comparable to the work of Foerster et al. (2016), the agents create their own language, rather than a pre-determined language. The models uses full communication by agents sending their complete hidden states and thereby revealing everything about the agent. This enables the model to achieve remarkable performance in cooperative scenarios, but limits the model from performing well in other scenarios. Furthermore, the model also uses global rewards, hence suffering from the credit assignment problem. This means the model has a problem with determining which actions of agents led to a certain outcome (Arumugam et al., 2021). Therefore it cannot effectively reward agents with good actions and punish agents with bad actions.

Singh et al. (2016) try to solve these problems by proposing a new, but similar algorithm, called IC3Net. In contrast to other algorithms (Sukhbaatar et al., 2016), agents in IC3Net can choose whether to communicate through a gating action. The credit assignment issue is solved by using a different control model, enabling individual rewards. The agents are able to learn when to communicate in all scenarios and therefore achieve better performance than other baselines, such as CommNet from Sukhbaatar et al. (2016). In their experiments, primarily blind, or agents with poor vision are used, to encourage communication.

3 Model and Environments

This section goes into further detail about the internal workings of the algorithm and the environments. IC3Net uses an Individualized Controlled Continuous Communication Model, which will first be explained. Thereafter, important parameters for the model are described. Lastly, the environments in which the agents perform tasks will be discussed.

Model

Earlier proposed algorithms often use an independent controller model, where each agent is controlled by an individual LSTM (Jiang et al. 2018; Sukhbaatar et al., 2016). LSTM stands for Long Short-Term Memory and is currently one of the most powerful classifiers used in machine learning (Staudemeyer et al., 2019). A LSTM network is a kind of RNN, such that it has feedback connections. However, what makes a LSTM special is the ability to remember data for an unknown period of time. This makes a LSTM very suitable for processing data on a time series, such as observations in an environment over multiple time steps. In these independent controller models, the hidden and cell states of an LSTM control the policy of the agent. All LSTM models use the same parameters, such that there is no difference between agents.

CommNet from Sukhbaatar et al. (2016) enables communication by interconnecting the individual networks from each agent, forming one big neural network. This forced the agents to always communicate and therefore could only perform in cooperative scenarios. Moreover, it also required an unified loss function, hence causing it to suffer from the credit assignment issue. IC3Net uses a similar idea, but does not connect all individual controllers. Instead, it uses a discrete gating action, which determines whether the agent communicates its internal state or not. Like CommNet, IC3Net uses continuous communication, which means that multiple communication cycles are used each time step before deciding the action.

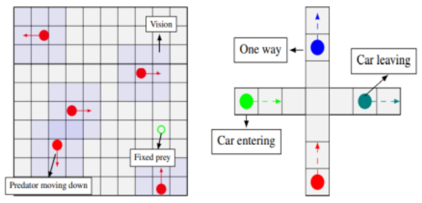
Furthermore, IC3Net uses multiple big networks each controlling an individual agent. Each of those big networks consist of multiple LSTM networks, which separately process an observation. This allows to train agents individually and maximize its own rewards. This solves two problems faced by earlier research. First, it enables the model to also be used in mixed and competitive scenarios. Second, it solves the credit assignment issue by giving agents their own reward.

Environments

The first environment is the predator-prey environment. This environment can be seen on the left in Figure 1. In this environment, a group of agents has to find a stationary prey. Once they find the prey, the predators stay in the same location as the prey to get a positive reward. This task can be completed in three scenarios, all with different reward structures. For instance, in competitive mode the agent gets a higher reward when it is the only agent who captured the prey. For this reason, the agent might decide not to communicate its location when it found the prey. Vision plays an important role, since it influences when the predator can see the prey. When the agent has a vision of zero, it is practically blind and can only see the prey when it is at the same location.

The second environment is a traffic junction, which can be seen on the right in Figure 1. In this environment, cars have to navigate a traffic junction without colliding. The cars have two actions each time step, namely gas and brake. To encourage the cars to move, a small negative reward is given for each time step they are in the traffic junction. A collision occurs when two cars are at the same location, which results in a large negative reward. This environment is only possible in cooperative mode. To encourage communication, the experiments from Singh et al. (2019) always use a vision of zero. Moreover, to eliminate the time to train the gating action, they fixed the communication action to one. This restriction is further explored in our experiments.

Figure 1
Two different environments



Note. Figure taken from Singh et al. (2019)

4 Experiments

Now, the experiments used will be introduced in detail. The experiments reproduced from the paper are explained first, thereafter the new experiments regarding communication and performance. The setup of the program required minor changes to the algorithm. These changes and other details about the setup can be found in the Appendix.

In total six different experiments are conducted. Two experiments to replicate the results from Singh et al. (2019), two experiments to look at the influence of vision on communication and finally two experiments to look at the influence of vision on performance. The experiments are conducted in this order, since some can be seen as complements of others. All experiments include a large number of parameters. The most important parameters are discussed below, however, for the exact parameters and commands used, please refer to the Appendix. When training the models, there is also a look for possible interesting behavior of the agents. The exact experiments will now be discussed in more detail.

Replication

The goal of the first two experiments is to replicate the results from Singh et al. (2019) and become familiar with the workings of the algorithm. The experiments take place in the predator-prey environment in cooperative and mixed mode. The paper included a large number of detailed results of the predator-prey environment in these modes, therefore making them suitable for a comparison. To match the properties of the experiments in the paper, the model trains for a 1000 epochs on 16 processes. Because these settings are already hardware intensive, the experiments run on easy difficulty, to reduce the training time. After training, the converged values can be compared. These values include: average steps and communication action.

Influence of Vision on Communication

Thereafter, two experiments are conducted to gain a better understanding about the influence of vision on communication and performance. Again, these experiments are conducted in the predator-prey environment, because this gives agents the choice to communicate. Since the results of these experiments are not compared to results from the paper, the parameters can be chosen more freely. For this reason, both experiments run for a 1000 epochs on eight processes. Eight processes suits our machine better and reduces the training time, while a 1000 epochs is enough to let the models converge. The main difference between the models is the difference in vision. One of the models trains with blind agents, i.e. a vision of zero, while the other model trains agents with a vision of two. Both experiments take place in mixed mode. This mode allows for a good comparison in communication, while cooperative and competitive would converge to always and never communicate respectively, regardless of their vision. In contrast to the experiments earlier, the size of the environment is set to 10, accordingly the maximum number of steps is set to 40. This is done to prevent the agents with good vision from gaining an unfair advantage. Since in a small environment, they would observe the entire environment in only a few steps. Taking into account the training time, all other parameters are set to easy. When training is finished, converged values can be compared, which include: average steps, success rate and communication action.

Influence of Vision on Performance
The last two experiments focus on the relation between vision and performance. Contrary to the experiments earlier, these experiments take place in the traffic junction environment. As with the second two experiments, these experiments run for a 1000 epochs on eight processes and with a large difference in the vision of the agents. The way communication works in the traffic junction environment is different from the predator-prey environment. The first experiment trains blind agents and forces them to communicate, similar to the experiments from the paper, but with slightly different parameters due to the change in number of epochs and processes. In the second experiment, this constraint is removed and the agents have a choice to communicate. This experiment trains agents with a vision of two. Again, the experiments run on easy difficulty. After training, the communication in the second experiment is analyzed together with the success rate of both models.

5 Results

In this section, the results are presented. First the results meant to replicate the experiments from Singh et al. (2019) are presented. Second, the results regarding communication and performance are analyzed. Last, there is quick look at interesting behavior that arises from some of these experiments.

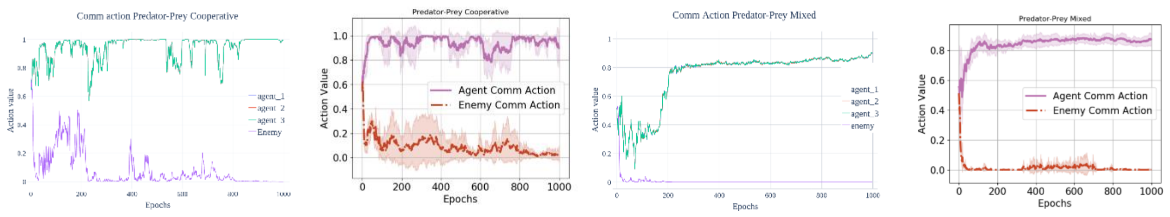
Replication
Table 1 shows a comparison between the results of the experiments and the results from Singh et al. (2019). These results clearly show a similarity in performance of both models, however looking at the small standard deviation, it cannot be said that the results match exactly. Looking at the communication of both models, a similar pattern can be seen. Figure 2 shows the development of communication in the experiments and the paper of Singh et al. (2019) for both cooperative and mixed scenarios. In both modes, the graphs show similar patterns and converge to the same values. It stands out our own models show much more extreme spikes than the models from Singh et al. (2019). In particular in cooperative mode, the action value fluctuates much more intense. In mixed, it takes the model longer to increase its communication, but after 200 epochs, it shows a similar increase.

Table 1
Comparison of the performance

	Singh et al. (2019)	Experiment
PP Mixed (Avg. Steps)	8.90±0.02	9.23
PP Cooperative (Avg. Reward)	1.57±0.008	1.52

Note. The results from Singh et al. (2019) include standard deviation. Note that different modes use different units of measurements.

Figure 2
Comparison of communication in predator-prey



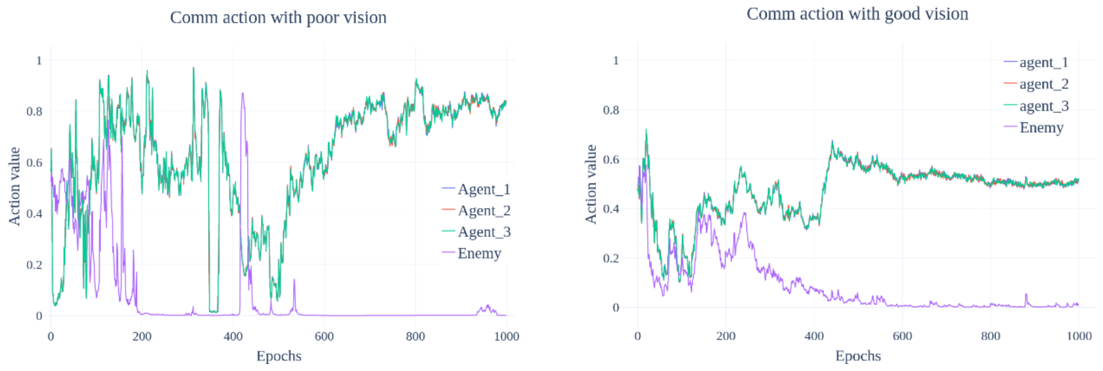
Note. The two left graphs show the results in cooperative mode, while the right two graphs show results in mixed mode. In both cases, the left graph shows the results from the experiment and the right graph shows the results from Singh et al. (2019).

Communication and Performance

Now the influence of vision on communication can be analyzed. Figure 3 shows the communication action for both a model with poor vision and a model with good vision in the predator-prey environment. Both models show a lot of volatility in the beginning, similar to the communication in mixed mode in Figure 2. Nonetheless, after about 500 epochs, both models start converging. The model with poor vision converges to an action value of around 0.84, while the model with good vision converges to an action value of around 0.52. This clearly shows that a model with poor vision converges to a significantly higher communication value than a model with good vision. For an increase in vision of two, the communication action value decreased 38%.

Figure 3

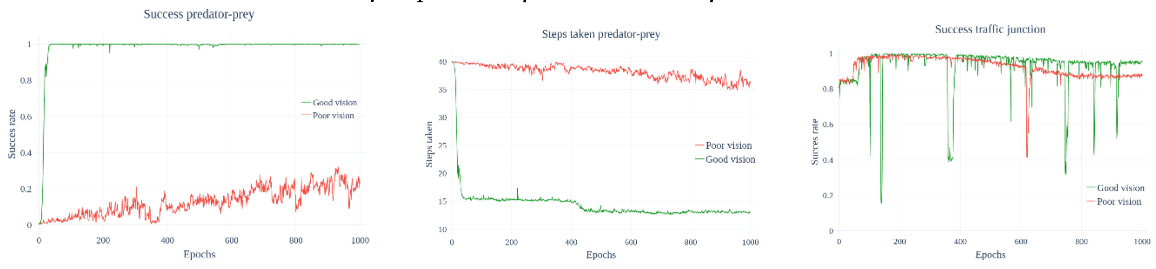
Comparison of communication for poor and good vision in mixed scenario



Now, the difference in performance can be analyzed. Figure 4 shows the success rate and number of steps taken of both predator-prey and traffic junction models. The left graph shows a significant difference in success rate between poor and good vision. The model with good vision converges quickly to a success rate of 1, on the other hand, the model with poor vision ends up with a success rate of only 0.23, but does not seem to be fully converged yet. The graph in the middle shows a comparable pattern. Again the model with good vision converges quickly to 12.97 steps, but the model with poor vision ends up with 36.53 steps and is still decreasing. The right graph shows the success rate for the traffic junction environment. Here, the models with poor and good vision converge to 0.87 and 0.95 respectively. Interestingly, in this environment, the difference is much smaller. Vision seems to not have such a big influence on the performance in this environment. Nevertheless, the same pattern as earlier can be seen here, where a better vision results in better performance. In the beginning, the models achieve similar performance, both achieving almost always full success. However, they both show a decrease in performance after epoch 200, where the difficulty starts increasing, but the model with poor vision is much more affected by this. It is important to note that the model with good vision did not learn to communicate. The graph for its communication can be found in the Appendix.

Figure 4

The success rate and number of steps taken for both cases of vision

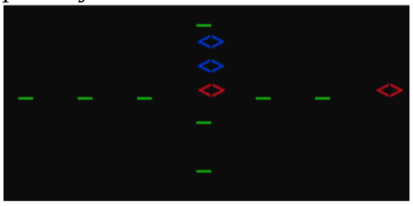


Behavior

Both of the traffic junction environments show very interesting behavior. There are two patterns that stand out. First, the agents seem to brake much more often closer to the junction. Often, one car is directly behind another car when entering the environment. On the straight road, the second car stays directly behind the first car, however when they approach the traffic junction, the second car decides to break to create distance between itself and the car in front. This behavior is comparable to the cautious behavior humans show when approaching a junction. The second pattern is even more similar to human behavior. When two cars both approach the junction and are on course to collide with each other, one of the cars has to decide to press the brake. The car on the vertical road is always the car to do this. This can be seen in Figure 5, where the car in the middle comes from the left and the cars just above the junction both brake to give way. Without any communication, the cars have developed the rule to give the horizontal road priority. This is exactly the same as the rule in the real world, namely cars coming from the right have priority. This again shows the intelligent behavior these agents can develop.

Figure 5

Cars on the vertical road giving priority



Note. Red diamond indicates a driving car, while a blue diamond represents a car that presses the brake

6 Interpretation and Discussion

This research replicated experiments from Singh et al. (2019) and found very similar results. As an extension to their research, experiments with different amounts of vision are conducted. The results of those show an increase in performance and a decrease in communication, when vision increases. These results will now be interpreted in more detail.

Interpretation

First, when looking at the results from replicating the experiments, two differences stand out. There is a slight difference in success between our results and the results from Singh et al. (2019), and perhaps more eye-catching, the graphs show much more fluctuations in communication. The first of those two presents a difference of 3.6 and 3.2 percent. The results from Singh et al. (2019) are an average of 5 runs and show a very small standard deviation. Because of this, it is not possible to conclude the results match exactly. However, the difference is so small that the results can be interpreted as a successful reproduction of experiments. The small difference could be caused by a slight difference in parameters. This limitation will be discussed in more detail later. The large fluctuations can be attributed to not taking the average over multiple runs. The graphs in Figure 2. from Singh et al. (2019) also show the extreme values, which are comparable to the spikes in our experiments. In spite of the line being much smoother, both experiments share similar patterns of peaks and valleys. Although, in the beginning of mixed mode there is a clear difference in trends. It takes the experiments much longer to start using more communication. Unlike in cooperative mode, this difference cannot be attributed to different extreme values, since the extreme values from Singh et al. (2019) do not show this pattern. It would be interesting to see if this pattern can be seen over multiple runs, or whether it is an exception.

Second, there is a clear gap in the agents' communication between models with poor and good vision. As the results from Singh et al. (2019) show, in a competitive scenario, when the agents reach the prey, they stop communicating. Since there is no benefit to an agent in mixed scenario, the agent likely stops communicating. This is based on the conclusion from Singh et al. (2019), that agents learn to communicate only when necessary. This could be a cause for the difference in communication. Because agents with a good vision observe the prey earlier, they can also stop communication earlier. The agent with good vision observes the prey at least two cells earlier, and therefore can stop at least two steps earlier with communication. However, in most cases the agent observes the prey much earlier than two steps and can stop communicating faster. This shows the large influence a small difference in vision can have on communication.

Third, in Figure 4, it clearly shows that the model with poor vision performs significantly worse after a 1000 epochs, but also did not converge yet. This suggests that at least, a worse vision slows down the learning rate of a model significantly. On the other hand, this makes it harder to interpret results about the performance of the model. It could be that after sufficient training time the model achieves similar performance as the model with good vision. This seems unlikely since searching blind is much harder than searching with knowledge of your surroundings.

Fourth and last, it stands out that the difference in performance in traffic junction is much smaller compared to predator-prey. The results from Singh et al. (2019) show that the performance of blind agents was already remarkable and so there was not much room for improvement. Furthermore, in the traffic junction environment, an agents' knowledge of their direct surroundings is one of the most important aspects of success. Just as in the real world, knowing there is a car in front of you or behind, has a large impact on whether you have to brake or push the throttle.

These results show that with an increase in vision, more realistic scenarios arise. The agents decided that vision alone was enough to navigate through a traffic junction, very similar to how an actual driver navigates. It would be interesting to see whether the agent still prefers not to communicate when it has worse vision. Moreover, the results point out that there is an important balance between vision and communication, which greatly differs in different tasks. When applied to the real world, communication with all other agents would sometimes not be possible and vision would still act as the primary observation.

Limitations

Although the experiments provide realistic scenarios, this research is limited by two important factors. These factors withhold this research from better investigating more complex environments and generalizing over more parameters. These limitations will now be discussed.

The first limitation are the exact parameters used by Singh et al. (2019). To replicate their experiments successfully, it is important to be sure that the exact same parameters are used. Otherwise, comparing results would give a wrong impression about the algorithm. In this research, the results from Singh et al. (2019) are used to confirm that the algorithm is correctly implement, to then further analyze the behavior of agents in new experiments. As presented earlier, when trying to replicate, the results differ slightly, which could be attributed to this limitation. Although the GitHub page of this algorithm does include exact commands for running different kinds of experiments, these commands do not produce the experiments used in their research. A lot of parameters used, are described in Section 4 and specified in the Appendix, but there is no precise overview of parameters for each experiment. By looking through the code of the algorithm and collecting parameters from their paper, two experiments to replicate their result are constructed. The models still require about 30 parameters all influencing the results, for some of which this research assumed their default value. Because of this, there could be differences in the exact parameters used.

Second, this research is limited by the hardware used. Training reinforcement learning models is very time consuming, in particular when the model uses large neural networks. IC3Net has the ability to use multiple processes or threads, which increases how much agents learn in one epoch. This decreases the time in which agents can learn intelligent behavior greatly, but also requires a lot of CPU intensive work. This research uses an Intel i5 processor with four cores and eight logical processes. This means that training the models on eight processes is optimal regarding time and learning. In the research of Singh et al. (2019), 16 processes are used to achieve maximum performance, which forced the replication experiments to use the same number of processes. For all other experiments, eight processes are used to reduce training time. Training all models still takes a lot of time, with most models training for 32 hours. Due to this extensive training process, the experiments are limited to small environments and easy difficulties. This also meant that for each experiment only one run is used, thus resulting in single results, rather than averaged results. Concluding, this research is able to answer the research questions for simple experiments, but for more complex experiments, more capable hardware is needed.

7 Conclusions and Future Work

The aim of this research was to further analyze IC3Net, a proposed algorithm by Singh et al. (2019). It tried to confirm their results by replicating their experiments and looking into the relation between vision, communication and performance. As seen earlier, similar performance and communication is shown when replicating the experiments from Singh et al. (2019). This is in line with the hypothesis and confirms the results from Singh et al. (2019), stating that IC3Net is able to learn agents to communicate at scale in all different scenarios. The results show significantly less communication when training with better vision in the predator-prey environment, concluding that an increase in vision results in a decrease in communication, which confirms the hypothesis. The results also show a significantly higher success rate when training with better vision in both the mixed predator-prey and cooperative traffic junction environment. Thereby, the predator-prey environment also sees a large difference in number of steps taken, favoring better vision. This leads to the conclusion that an increase in vision results in higher performance in multiple scenarios and environments, thus accepting the hypothesis. Besides this, the research has also shown that IC3Net does not see communication as a necessity, when provided with sufficient vision.

Similar to the research from Singh et al. (2019), this research can help with further development of communication in multi agent reinforcement learning. In the future, research with more capable hardware to confirm the results in more complex environments is suggested. In addition, more research looking into local communication could make a great contribution. Lastly, further research might explore environments with limited resources to communicate.

References

- @IMIMyzj (2020, 10 september). *Maybe visdom.line() has a bug · Issue #755 · fossasia/visdom.* GitHub. Viewed at 21 06 2022, from https://github.com/fossasia/visdom/issues/755

- Arumugam, D., Henderson, P., & Bacon, P. L. (2021). An information-theoretic perspective on credit assignment in reinforcement learning. *arXiv preprint arXiv:2103.06224*. https://doi.org/10.48550/arXiv.2103.06224

- Buşoniu, L., Babuška, R., & Schutter, B. D. (2010). Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183-221. https://doi.org/10.1007/978-3-642-14435-6_7

- Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). Deep blue. *Artificial intelligence, 134*(1-2), 57-83. https://doi.org/10.1016/S0004-3702(01)00129-1
  Foerster, J., Assael, I. A., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, *29*. https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html

- Gommers, R. (2018, 14 july). NEP 23 — *Backwards compatibility and deprecation policy — NumPy Enhancement Proposals*. Numpy.org. viewed at 29 06 2022, from https://numpy.org/neps/nep-0023-backwards-compatibility.html#:%7E:text=NumPy%20is%20also%20actively%20maintained,user%20experience%20for%20new%20users.

- Jiang, J., & Lu, Z. (2018). Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems, 31*. https://proceedings.neurips.cc/paper/2018/hash/6a8018b3a00b69c008601b8becae392b-Abstract.html

- Lazaridou, A., Peysakhovich, A., & Baroni, M. (2016). Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*. https://doi.org/10.48550/arXiv.1612.07182

- Li, Y. (2019). Reinforcement learning applications. *arXiv preprint arXiv:1908.06973*. https://doi.org/10.48550/arXiv.1908.06973

- Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, *30*. https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html

- Oliveira, E., Fischer, K., & Stepankova, O. (1999). Multi-agent systems: which research for which applications. *Robotics and Autonomous Systems,* 27(1–2), 91–106. https://doi.org/10.1016/s0921-8890(98)00085-2

- Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017, July). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning* (pp. 2681-2690). PMLR. http://proceedings.mlr.press/v70/omidshafiei17a.html

- Schäfer, A. M., & Udluft, S. (2005). Solving partially observable reinforcement learning problems with recurrent neural networks. In *Workshop Proc. of the European Conf. on Machine Learning* (pp. 71-81). https://openreview.net/pdf?id=41eqUzY4Ja8

- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961

- Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*. https://doi.org/10.48550/arXiv.1909.09586

- Sukhbaatar, S., & Fergus, R. (2016). Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, *29*. https://proceedings.neurips.cc/paper/2016/hash/55b1927fdafef39c48e5b73b5d61ea60-Abstract.html

- Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330-337). https://books.google.nl/books?hl=nl&lr=&id=TrqjBQAAQBAJ&oi=fnd&pg=PA330&dq=%E2%80%A2%09Tan,+M.+(1993).+Multi-agent+reinforcement+learning:+Independent+vs.+cooperative+agents.+In+Proceedings+of+the+tenth+international+conference+on+machine+learning+(pp.+330-337).+&ots=v3WcWU7ZZk&sig=leFECNBbb44HoaVFyk7NHHtD_68&redir_esc=y#v=onepage&q&f=false

- Walraven, E., Spaan, M. T., & Bakker, B. (2016). Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, *52*, 203–212. https://doi.org/10.1016/j.engappai.2016.01.001

- 16.6. multiprocessing — Process-based “threading” interface — Python 2.7.2 documentation. (2013, 8 september). python.readthedocs.io. Viewed at 21 june 2022, from https://python.readthedocs.io/en/v2.7.2/library/multiprocessing.html

8 Appendix

Local implementation:
Before conducting any experiments, the program needs to be correctly installed on a local computer. Although the code is easily accessible through GitHub, there are a few challenges worth mentioning before the algorithm runs correctly. These challenges will now be briefly discussed.

One of the primary challenges is installing the correct dependencies, despite of the exact packages and versions being specified. Older versions of packages are hard to find and raise issues when installing, thus a combination of recent and old versions is used. First, Numpy's most recent version is used, since Numpy is mostly backwards compatible (Gommers, 2018) and this does not seem to influence the code. Second, the algorithm requires Torch version 0.4.0, but this version is not widely available, so version 0.4.1 is used. Third, Visdom version 0.1.4 has been tried, but comes with a few bugs, so the recent version of Visdom is installed. Nonetheless, this version also comes with bugs. In particular one stands out, for raising an error over not corresponding matrix shapes. This bug is well known and a solution is suggested on GitHub (@IMIMyzj, 2020), where three lines of code raising the error are eliminated. This solution is used, by commenting out line 1687 to 1689 in file __init__.py. All other packages are installed without any problems.

Another challenge is to get multiprocessing to work correctly. When training a model, the number of processes can be specified to speed up learning and increase the performance. The code for the program was written on a Linux machine, where starting new processes works slightly different than on Windows (16.6. multiprocessing — Process-based "threading" interface, 2013). On Linux, when starting a new process, it is forked, which means this process inherits the memory of the process that started it. In contrast, on Windows, a new process is spawned, which means it inherits no memory and start entirely on its own. Because of this, when multiprocessing on Windows, the code: *if __name__ == '__main__':* , needs to be included in the main file. All code needs to be inside the if statements, except imports and default settings for the program. This line of code is added on line 21 of the *main.py* file and everything below that is put in the body of the if-statement. This ensures that multiprocessing can be used in the experiments.

Experiment commands
This section provides the exact commands used for each experiment. For all parameters not in the command, the default value is assumed.

Replication of predator-prey in mixed mode:
```
python main.py --env_name predator_prey --nagents 3 --nprocesses 16 --num_epochs
325 --hid_size 128 --detach_gap 10 --lrate 0.001 --dim 5 --max_steps 20 --ic3net
--vision 0 --recurrent --plot --enemy_comm --mode cooperative
```

Replication of predator-prey in cooperative mode:
```
python main.py --env_name predator_prey --nagents 3 --nprocesses 16 --num_epochs
200 --hid_size 128 --detach_gap 10 --lrate 0.001 --dim 5 --max_steps 20 --ic3net
--vision 0 --recurrent --plot --enemy_comm --mode mixed
```

Predator-prey in a medium environment with good vision:
```
python main.py --env_name predator_prey --nagents 3 --nprocesses 8 --num_epochs
450 --hid_size 128 --detach_gap 10 --lrate 0.001 --dim 10 --max_steps 40 --ic3net
--vision 2 --recurrent --plot --enemy_comm --mode mixed
```

Predator-prey in a medium environment with poor vision:
```
python main.py --env_name predator_prey --nagents 3 --nprocesses 8 --num_epochs
1000 --hid_size 128 --detach_gap 10 --lrate 0.001 --dim 10 --max_steps 40 --
ic3net --vision 0 --recurrent --plot --enemy_comm --mode mixed
```

Traffic junction in a small environment with good vision:

```
python main.py --env_name traffic_junction --nagents 5 --nprocesses 8 --
num_epochs 1000 --hid_size 128 --detach_gap 10 --lrate 0.003 --dim 6 --max_steps
20 --ic3net --vision 2 --recurrent --add_rate_min 0.1 --add_rate_max 0.3 --
curr_start 200 --curr_end 800 --difficulty easy --plot
```

Note that in this experiment lines 126 and 127 are commented out to make sure agents are not forced to communicate.

Traffic junction in a small environment with poor vision:

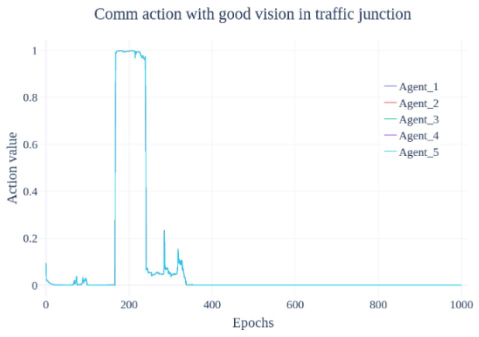
```
python main.py --env_name traffic_junction --nagents 5 --nprocesses 8 --
num_epochs 1000 --hid_size 128 --detach_gap 10 --lrate 0.003 --dim 6 --max_steps
20 --ic3net --vision 0 --recurrent --add_rate_min 0.1 --add_rate_max 0.3 --
curr_start 200 --curr_end 800 --difficulty easy --plot
```

Communication without enforcing it

Figure 6 shows the communication in the traffic junction environment when constant communication is not enforced. It is clear the agents try communication for a short time, but converge quickly to no communication.

Figure 6

Communication without being forced


Comm action with good vision in traffic junction
Action value
Epochs
Agent_1
Agent_2
Agent_3
Agent_4
Agent_5