An Analysis of Discretization Methods for Communication Learning with Multi-Agent Reinforcement Learning

Astrid Vanneste
University of Antwerp - imec
IDLab - Faculty of Applied
Engineering
Antwerp, Belgium
astrid.vanneste@uantwerpen.be

Simon Vanneste
University of Antwerp - imec
IDLab - Faculty of Applied
Engineering
Antwerp, Belgium
simon.vanneste@uantwerpen.be

Kevin Mets
University of Antwerp - imec
IDLab - Department of Computer
Science
Antwerp, Belgium
kevin.mets@uantwerpen.be

Tom De Schepper
University of Antwerp - imec
IDLab - Department of Computer
Science
Antwerp, Belgium
tom.deschepper@uantwerpen.be

Siegfried Mercelis
University of Antwerp - imec
IDLab - Faculty of Applied
Engineering
Antwerp, Belgium
siegfried.mercelis@uantwerpen.be

Steven Latré
University of Antwerp - imec
IDLab - Department of Computer
Science
Antwerp, Belgium
steven.latre@uantwerpen.be

Peter Hellinckx
University of Antwerp - imec
IDLab - Faculty of Applied
Engineering
Antwerp, Belgium
peter.hellinckx@uantwerpen.be

ABSTRACT

Communication is crucial in multi-agent reinforcement learning when agents are not able to observe the full state of the environment. The most common approach to allow learned communication between agents is the use of a differentiable communication channel that allows gradients to flow between agents as a form of feedback. However, this is challenging when we want to use discrete messages to reduce the message size since gradients cannot flow through a discrete communication channel. Previous work proposed methods to deal with this problem. However, these methods are tested in different communication learning architectures and environments, making it hard to compare them. In this paper, we compare several state-of-the-art discretization methods as well as two methods that have not been used for communication learning before. We do this comparison in the context of communication learning using gradients from other agents and perform tests on several environments. Our results show that none of the methods is best in all environments. The best choice in discretization method greatly depends on the environment. However, the discretize regularize unit (DRU), straight through DRU and the straight through gumbel softmax show the most consistent results across all the tested environments. Therefore, these methods prove to be the best choice for general use while the straight through estimator and the gumbel softmax may provide better results in specific environments but fail completely in others.

KEYWORDS

Communication Learning, Multi-Agent, Reinforcement Learning

1 INTRODUCTION

Over the past several years, both single-agent reinforcement learning (RL) and multi-agent RL (MARL) have gained a lot of interest. In (MA)RL, the agents often have to deal with partial observability. The agents can only observe part of the global state, making it hard to choose appropriate actions. For example, an agent navigating through an environment cannot see certain parts of the environment. In MARL, this partial observability can often be alleviated by allowing the agents to share information with each other. By combining this information with their own observation, agents get a more complete view of the environment and can choose better actions [12, 18]. For example, multiple agents navigating through the same environment can share the information they can see with each other, resulting in a more complete view.

One of the subfields within MARL is research towards learned communication between agents. The most commonly used approach thus far is allowing gradients to flow between agents as a form of feedback on the received messages. However, in the case of discrete communication messages this raises a problem since gradients cannot flow through a discrete communication channel. Several different approaches have been proposed in the state of the art [2, 8, 10, 14] to discretize messages while allowing gradients to flow through the discretization unit. Each of these methods was tested using different communication learning approaches and applied on different environments, making a fair comparison very difficult.

Table 1: Comparison the state of the art and our work
(DRU - Discretize Regulize Unit, GS - Gumbel Softmax, STE - Straight Through Estimator, ST-DRU - Straight Through DRU, ST-GS - Straight Through GS)

	Message Type	Communication Learning Technique	Discretization Method
RIAL [2]	Discrete	DQN using team reward	Discrete policy
DIAL [2]	Discrete	Gradients from other agents	DRU
CommNet [16], A3C3 [15]	Continuous	Gradients from other agents	N/A
MADDPG [10, 14]	Continuous/Discrete	Gradients from the critic	GS
Freed et al. [3]	Discrete	Gradients from other agents	Randomized Encoder/Decoder
Jaques et al. [6]	Discrete	A3C using team reward augmented with social influence reward	Discrete policy
MACC [19]	Discrete	Counterfactual reasoning	Discrete policy
Lin et al. [8]	Discrete	Reconstruction loss	STE
This Work	**Discrete**	**Gradients from other agents**	**(ST)-DRU / STE / (ST)-GS**

Our contributions consist of two parts. First, we present an in-depth comparison of several discretization methods used in the state-of-the-art. In our comparison we focus on using these discretization methods to allow discrete communication when learning communication using the gradients of the receiving agents. We compare each of the approaches on several environments with increasing complexity as well as analyze their performance when the environment introduces errors to the communication messages. Secondly, we present two discretization methods (ST-DRU and ST-GS) that have not been used in communication learning before.

The remainder of this paper is structured as follows. Section 2 gives an overview of work related to our research. Section 3 contains some additional background information. Section 4 provides a detailed explanation of the discretization methods we compare in this paper. In Section 5, the different experiments that were done are explained along with their results. We discuss the results of our experiments further in Secion 6. In Section 7 we draw some conclusions from our experimental results.

2 RELATED WORK

In this section, we review state-of-the-art work relevant to our research. We give an overview of several communication learning methods, focusing on methods that learn discrete communication. Here, we see some alternative approaches for learning discrete communication beside using a differentiable communication channel as well as different discretization techniques used in the state of the art.

Foerster et al. [2] and Sukhbaatar et al. [16] proposed the first successful methods for learning inter-agent communication. Foerster et al. [2] proposed two novel approaches, Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL). Both RIAL and DIAL use discrete communication messages, but learn the communication policy in a different way. RIAL learns the communication policy the same way as learning the action policy, by using the team reward. However, the results clearly show that this is not sufficient in most environments. DIAL proved more successful by using gradients originating from the agents receiving the messages which provide feedback on the communication policy. Sukhbaatar et al. [16] proposed a different approach called

CommNet. Messages consist of the hidden state of the agents, resulting in continuous messages. Similar to DIAL, CommNet uses gradients that flow through the communication channel to train the communication.

A lot of the research that follows these works chooses to use continuous communication like Sukhbaatar et al. [16], avoiding the problem of discretizing the communication messages [15]. Other methods choose a different method to learn communication than using gradients through the communication channel. This also avoids the challenge of discretizing the messages. Jaques et al. [6] train the communication policy using the team reward augmented with a social influence reward. This additional reward is based on how much the message changes the action policy of the receiving agents. Vanneste et al. [19] use counterfactual reasoning to directly learn a communication protocol without the need of a differentiable communication channel. Freed et al. [3] use a randomized encoder at the sender to encode the continuous messages into discrete messages. At the receiver, a randomized decoder is used to approximate the original continuous message. They show that by using this technique they can consider the communication channel equivalent to a continuous channel with additive noise, allowing gradients to flow between the sender and receiver agent.

Lowe et al. [10] and Mordatch and Abbeel [14] propose Multi-Agent Deep Deterministic Policy Gradients (MADDPG). In their work they evaluate MADDPG on multiple different scenarios, including communication tasks. They do not use a differentiable communication channel to learn communication but they have to make sure the messages are differentiable to allow the MADDPG method to work properly since policies are learned using gradients that originate from the critic. They allow discrete communication by using a gumbel softmax. Lin et al. [8] use an autoencoder at the sender to compose a representation of the observation that will be used as communication message. To discretize these messages they use a straight through estimator in the autoencoder. Both Lowe et al. [10], Mordatch and Abbeel [14] and Lin et al. [8] have to use differentiable discretization techniques in their methods to allow discrete communication. However, they do not use the techniques in the same way we do in our work. Lowe et al. [10] and Mordatch and Abbeel [14] use the discretization method in a similar way as
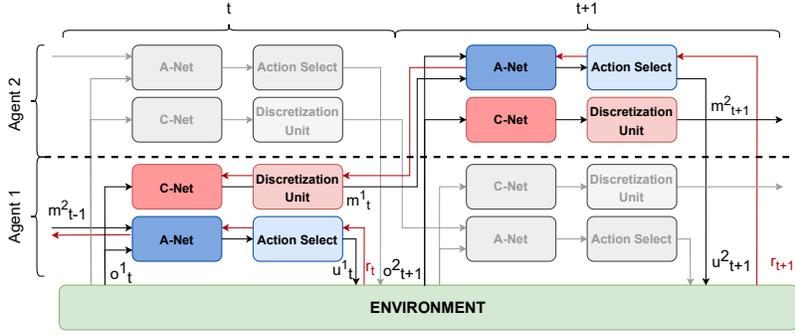
Figure 1: Architecture of DIAL

our work but in MADDPG the gradients that correct the communication originate from the critic instead of from other agents. Lin et al. [8] use the discretization method in a very different way since they train the communication policy entirely using the reconstruction loss of the autoencoder instead of the gradients from the other agents.

Summarized, in the state-of-the-art related to our research, we see that multiple discretization methods have been proposed. But, differences in communication learning approaches and the fact that each of these methods is tested on different environments makes comparing these different discretization methods very hard.

3 BACKGROUND

3.1 Deep Q-Networks (DQN)

In single agent RL [17], the agent chooses an action $u_t \in U$ based on the state $s_t \in S$ of the environment. As a result of this action, the environment will transition to a new state s_{t+1} and provide the agent with a reward $r_{t+1} \in R$. This reward is used to train the agent. Q-learning uses this reward to calculate a Q-value for each state action pair $Q(s, u)$. This Q-value represents a value for each state action pair, where a higher Q-value indicates a better action. Therefore, the policy of our agent can be defined by Equation 1.

$$\pi(s) = \underset{u}{argmax}\left(Q(s, u)\right) \tag{1}$$

Deep Q-learning [13] uses a neural network with parameters θ to represent the Q-function. The deep Q-network is optimized at iteration i by minimizing the loss in Equation 2.

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s_t, u_t, r_t, s_{t+1}}\left[\left(r_t + \gamma \max_{u_t} Q(s_{t+1}, u_{t+1}, \theta_i^-) - Q(s_t, u_t, \theta_i)\right)^2\right] \tag{2}$$

where γ is the discount factor and θ_i^- are the parameters of the target network. This target network will be updated after each training iteration according to Equation 3.

$$\theta_{i+1}^- \leftarrow \tau\theta_i + (1 - \tau)\theta_i^- \tag{3}$$

where τ is a weight that indicates how fast the target network should follow the parameters θ. In our work, the agent does not receive the full state s_t but only a limited observation $o_t \in O$ of this state. This increases the complexity since the observation might lack important information.

3.2 Differentiable Inter-Agent Learning (DIAL)

To allow for a fair comparison, we will use the same communication learning approach for each of the different discretization methods. We use DIAL as proposed by Foerster et al. [2] in our experiments since it is the most general and well known architecture to learn discrete communication using gradients from the other agents. The architecture of DIAL can be seen in Figure 1. We adapted the original DIAL architecture by separating the action and communication network. This allows us to keep the communication network small, making communication learning easier. In our experiments, we examine environments where the agents only need to share and encode part of the observation which allows us to make this adaptation. When the agents are expected to communicate about a strategy, splitting the action and communication network may no longer be possible.

Each agent consists of two networks, the A-Net and the C-Net. The A-Net produces Q-values to determine the action based on the observation and the incoming messages. The C-Net is responsible for calculating the messages based on the observation. It does not receive the incoming messages in our experiments because in these environments the communication policy does not need the incoming messages to determine the output message. Before the messages are broadcast to the other agents, the discretization unit applies one of the discretization techniques that we are comparing in this paper. To train the agents, we apply the team reward provided by the environment on the A-Net according to deep Q-learning. The gradients from the A-Net are propagated to the C-Net of all the agents that sent a message to that agent. This allows us to train the C-Net using the feedback of the agents receiving the messages.

4 METHODS

In this section, we describe the different discretization modules, that we will compare, in more detail. Table 2 provides an overiew of all the discretization methods and their differences. We show the difference between the function used to calculate the output of the discretization unit during training and during evaluation as well as the function that is used for the backward pass.

4.1 Discretize Regularize Unit (DRU)

In the DIAL method, Foerster et al. [2] propose a module called the Discretize Regularize Unit (DRU) to allow gradients to be used for

Table 2: Differences between the discretization methods where x is the input of the discretization unit (the output of the C-Net), $H(x)$ is the heaviside function, $n \sim \mathcal{N}(0, \sigma_G^2)$ is Gaussian noise, $i \in \{0, 1\}$, $g_i \sim G$ is Gumbel noise, $\pi_0 = \sigma(x)$, $\pi_1 = (1 - \sigma(x))$, τ is the softmax temperature and $\sigma(x)$ is the sigmoid function

	Training Output (forward pass)	Function used for backward pass	Evaluation Output (forward pass)
STE	$H(x)$	x	$H(x)$
DRU	$\sigma(x + n)$	$\sigma(x + n)$	$H(x)$
GS	$softmax\left(\frac{log(\sigma(x))+g_1}{\tau}, \frac{log(1-\sigma(x))+g_2}{\tau}\right)[0]$	$softmax\left(\frac{log(\sigma(x))+g_1}{\tau}, \frac{log(1-\sigma(x))+g_2}{\tau}\right)[0]$	$one\_hot(\underset{i}{argmax}[g_i + log\pi_i])[0]$
ST-DRU	$H(x + n)$	$\sigma(x + n)$	$H(x)$
ST-GS	$one\_hot(\underset{i}{argmax}[g_i + log\pi_i])[0]$	$softmax\left(\frac{log(\sigma(x))+g_1}{\tau}, \frac{log(1-\sigma(x))+g_2}{\tau}\right)[0]$	$one\_hot(\underset{i}{argmax}[g_i + log\pi_i])[0]$
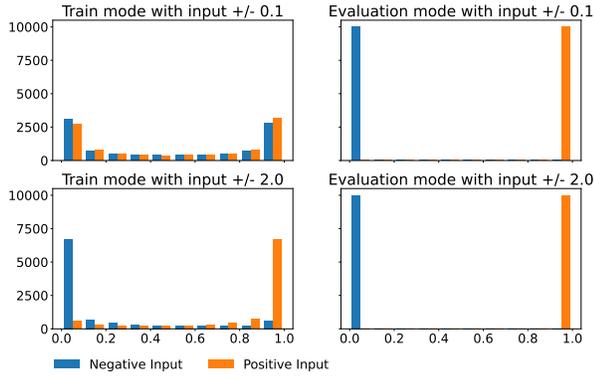
Figure 2: Histogram of the output of the DRU when calculating the output 10k times for different input values (positive and negative) for both train and evaluation mode

Figure 3: Histogram of the output of the GS when calculating the output 10k times for different input values (positive and negative) for both train and evaluation mode with temparature $\tau = 1.0$

training while learning discrete communication messages. The DRU has two modes, discretization and regularization. The discretization mode is used at execution time and discretizes the input into a single bit using Equation 4.

$$m = H(x) \tag{4}$$

where $H(x)$ is a heaviside function and x is the input of the discretization unit (output of the C-Net). This calculation cannot be used during training because the derivative of the heaviside function is the Dirac function which is zero everywhere except at $x = 0$, where the output is infinite. Therefore, the regularization mode is used during training. When using the regularization mode, the agents are allowed to communicate using continuous messages. However, the DRU tries to encourage the communication policy to generate messages that can easily be discretized at execution time. This is achieved by applying Equation 5.

$$m = \sigma(x + n) \tag{5}$$

where x is the input of the discretization unit (output of the C-Net), n is noise sampled from a Gaussian distribution with standard deviation σ_G and $\sigma(x)$ is the sigmoid function. The noise will affect the output of the DRU the most when the input is around zero since the sigmoid is the steepest there. The influence will be much smaller for inputs with high absolute values. The output in those

cases will also go towards zero and one, making it very similar to discrete, binary messages. This can be seen in Figure 2.

4.2 Straight Through Estimator (STE)

A straight through estimator (STE) [1, 20] performs a normal discretization, as in Equation 4, when calculating the output. However, when performing backpropagation, it uses the gradients of an identity function instead of the gradients of the discretization. The advantage of this technique is that the agent receiving the message will immediately receive binary numbers and can learn how to react to these messages while still being able to use the gradients from the receiving agents to train the communication network. For the STE, the output will always look like the DRU in evaluation mode, shown in Figure 2.

4.3 Gumbel Softmax (GS)

The Gumbel Softmax (GS) [5, 11] is a method to approximate a sample from a categorical distribution in a differentiable way. Normal sampling techniques are not differentiable and therefore not directly applicable in this context. The GS achieves this desirable property by using Gumbel noise and the gumbel-max trick[4]. Using the gumbel-max trick, we can sample from a categorical distribution
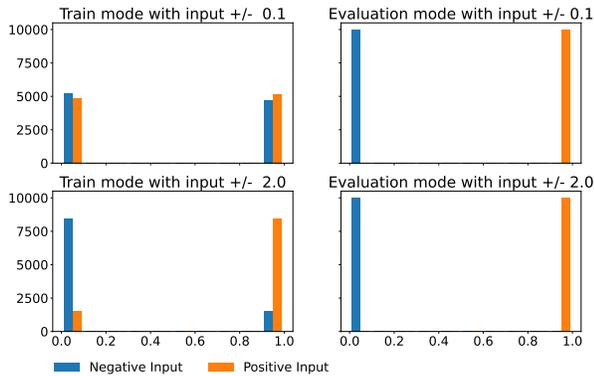
Figure 4: Histogram of the output of the ST-DRU when calculating the output 10k times for different input values (positive and negative) for both train and evaluation mode

Figure 5: Histogram of the output of the ST-GS when calculating the output 10k times for different input values (positive and negative) for both train and evaluation mode

with class probabilities π as described in Equation 6.

$$z = one\_hot(\underset{i}{argmax}[g_i + log\pi_i]) \qquad (6)$$

where $g_1, ...g_k$ are i.i.d. samples drawn from Gumbel(0,1), $i \in \{0, 1\}$, $\pi_0 = \sigma(x)$, $\pi_1 = (1 - \sigma(x))$, x is the input of the discretization unit (output of the C-Net) and $\sigma(x)$ is the sigmoid function. To make this differentiable, we have to approximate the *one_hot* and *argmax* functions with a *softmax* function. Since we need two probabilities to obtain a categorical distribution for both states of a bit we will obtain the output message by using Equation 7.

$$m = softmax\left(\frac{log(\sigma(x)) + g_1}{\tau}, \frac{log(1 - \sigma(x)) + g_2}{\tau}\right)[0] \qquad (7)$$

where x is the input of the discretization unit (output of the C-Net), τ is the softmax temperature and $\sigma(x)$ is the sigmoid function. A low temperature will result in an output that closely matches the output of Equation 6. The higher the temperature the more the output will approach a uniform distribution. Figure 3 shows the behaviour of the GS for different inputs in training and evaluation mode with temparature $\tau = 1.0$.

4.4 ST-DRU

We propose a novel discretization method (ST-DRU) that combines the DRU and STE methods. During execution, we discretize the input messages in the same way as mentioned in the DRU and STE methods, shown in Equation 4. During training, we will use a different function for the forward and backward pass. In the forward pass, we add gaussian noise and apply the same discretization, resulting in Equation 8.

$$m = H(x + n) \qquad (8)$$

where x is the input of the discretization unit (output of the C-Net) and n is noise sampled from a Gaussian distribution with standard deviation σ_G. However, during backpropagation we use the gradients of Equation 5 instead. The advantage of this approach over the original DRU can be seen in Figure 4. The agents receiving the messages will receive binary messages from the start of training. The DRU uses continuous messages during training. Even though
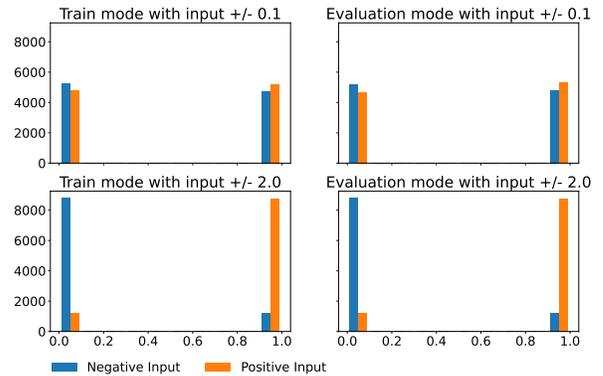
the agents are encouraged by the DRU to produce outputs with a high absolute value, it will take a while before the output messages will resemble binary messages. The ST-DRU will also encourage the agent to produce outputs that can easily be discretized. But, the receiving agents will immediately receive discrete messages, allowing them to learn to interpret them more quickly.

4.5 ST-GS

Similarly to the ST-DRU, we also test a straight through version of the GS as proposed by Jang et al. [5]. Here, we will use the sampling technique described in Equation 6 to calculate the output, while using the gradients of the softmax approximation in Equation 7 to train the communication network. Similarly to the DRU, the GS produces continuous messages during training. When evaluating the agents the GS will discretize the messages. If the sending agent is not producing outputs with a high enough absolute value, the difference between the messages at training time and at evaluation time will be very large. This prevents the receiving agents from correctly interpreting the message and choosing the appropriate actions. The ST-GS on the other hand will produce discretized messages at evaluation time and at training time, as can be seen in Figure 5. This way, we make sure that the receiving agent knows how to correctly interpret discrete messages.

5 EXPERIMENTS

In this section, we explain each of our experiments and analyze the results. For each experiment we show the average performance over five different runs for each of the discretization methods. The hyperparameters and network architectures are identical for each of the discretization methods since our hyperparameter search showed that the best hyperparameters and network architecture were not influenced by the choice in discretization method. All of our experiments are run using the RLlib framework [7].

5.1 Matrix Environment

The Matrix environment is inspired by the Matrix Communication Games presented by Lowe et al. [9]. In the Matrix environment N
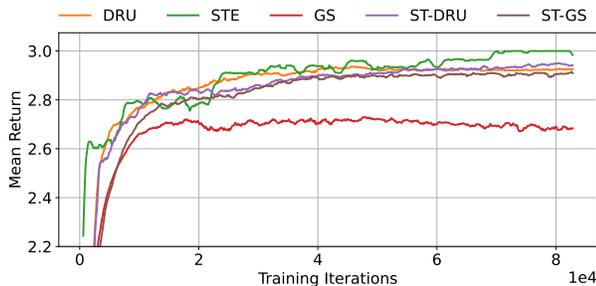
Figure 6: Results in the Simple Matrix environment

Figure 7: Results in the Complex Matrix environment

Table 3: Reward matrices for the Matrix environment ($N = 2$ and any value for M). The agents have two possible actions, indicating they got the same number (S) or a different number (D)

(a) The agents received the same number

	S	D
S	2, 2	1, 1
D	1, 1	0, 0

(b) The agents received a different number

	S	D
S	0, 0	1, 1
D	1, 1	2, 2

Table 4: Different configurations of the Matrix environment

	N	M
Simple Matrix Environment	3	4
Complex Matrix Environment	5	256

agents receive a natural number in $[0, M − 1]$. The values for N and M can be chosen independent of each other. The agents are allowed to broadcast one message to the other agents before they have to indicate whether all agents received the same number or not. The odds of the agents receiving the same number are 50% regardless of the value of N and M. The minimum number of bits required to be able to represent each possible input number can easily be determined by applying a base two logarithm on M. The team reward in this environment is equal to the number of agents that correctly determined whether all agents got the same number or not. Therefore, the maximum reward is equal to N. Table 3 shows the reward matrices that correspond with a Matrix environment with $N = 2$ and any value for M.

We examine the results for two different configurations of this environment which can be seen in Table 4. In each of these experiments, we show the evaluation reward of our agents, measured by performing 100 evaluation episodes after each 100 training iterations. During the evaluation episodes, the agents do not explore and the discretization methods are applied in evaluation mode. In this environment, all of the agents are identical. Therefore, we can use parameter sharing between the agents, which improves their performance significantly as shown in the results of Foerster et al. [2].
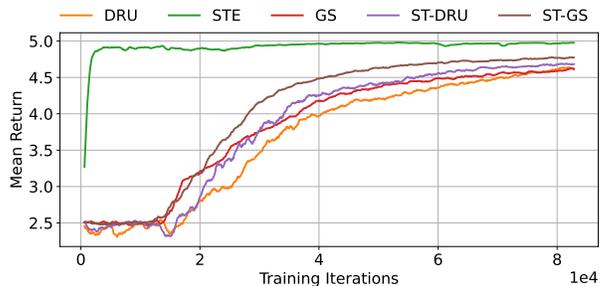
5.1.1 Simple Matrix Environment. In Figure 6 and Table 5, the results for the different discretization methods are shown. The maximum reward the agents can achieve in this scenario is a reward of 3. We can see that most of the methods are able to achieve a reward very close to this maximum except for the GS. The ST-GS does not have the same issue as the GS to achieve the maximum reward. We also see that the STE method is faster at the beginning of the training but this difference disappears rather quickly. Due to the limited complexity of this environment, the differences between the methods are still small.

5.1.2 Complex Matrix Environment. The Complex Matrix environment has more agents ($N = 5$) as well as more possible input numbers ($M = 256$). The agents need a message consisting of a full byte to be able to encode each of the possible input numbers. Figure 7 and Table 5 show the results of this experiment. The maximum reward in this configuration is 5. We see that the difference between the methods is larger than in the Simple Matrix environment due to the added complexity. We see that the STE method is the only one that is able to reach the maximum reward in this training period. It reaches a reward close to the maximum reward after only 5k training iterations. The other methods only start improving after 15k training iterations and take over 60k training iterations to reach their maximal performance. We can also see that the adapted versions of the DRU and GS which include the STE technique also perform better than the version without the STE technique. The ST-DRU has an average reward that is 0.072 higher than the DRU and the ST-GS has an average reward that is 0.176 higher than the GS during the final 10% of training iterations. The communication amplitude in Figure 8 provides an explanation for the training speed of the STE method. The communication amplitude is the mean absolute value of the input of the discretization unit. We can see a clear difference between the STE and the other methods. The communication amplitude of the STE stays below 0.5 while the communication amplitude of the DRU and GS approaches 1.8 and the communiation amplitude of the ST-DRU and the ST-GS exceeds 2.0. This is caused by the noise that is included in all of the discretization methods except for the STE. For a low communication amplitude the output of each of these discretization methods is still very random. The output during training will be determined by the noise instead of by the sign of the input which is done during evaluation. This encourages the agent to produce outputs with a higher communication amplitude. However, this puts a delay on
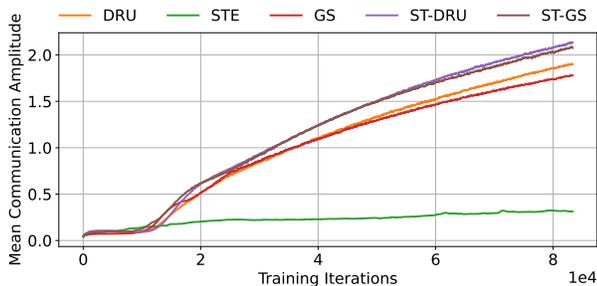
Figure 8: Communication amplitude in the Complex Matrix environment smoothed with an exponentially weighted moving average function with $\alpha = 5 \cdot 10^{-5}$
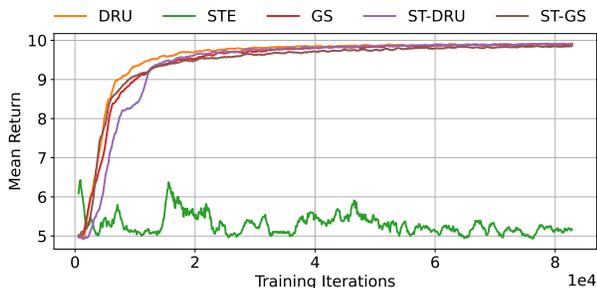
Figure 10: Results in the Matrix environment with a 50% chance that a bit error will occur at a random location in the message

the speed at which the agents can discover communication protocols. In Figure 8 we see that the communication amplitude starts rising more quickly at around 15k training iterations. Once the communication amplitude starts rising we can see in Figure 7 that the rewards that the agents receive also starts rising, indicating that they are starting to learn how to communicate with each other.

5.2 Speaker Listener Environment

As a more complex environment, we use the speaker listener scenario from the particle environment by OpenAI [10, 14]. This is one of the environments that was used to evaluate MADDPG [10, 14]. In this environment there are two agents and three landmarks. One of the agents, the speaker, observes which landmark is the target during this episode. The speaker then has to communicate this information to the other agent, the listener. Next, the listener has to navigate to the target landmark. Both agents are rewarded using a team reward that is composed based on the distance of the listener to the target landmark. Contrary to the Matrix environment, the agents are not the same in this environment. The speaker will only consist of a communication policy while the listener will only consist of an action policy. In this experiment, we show the evaluation reward of our agents, measured by performing 10 evaluation episodes after each 50 training iterations. During the evaluation episodes, the agents do not explore and the discretization methods are applied in evaluation mode.
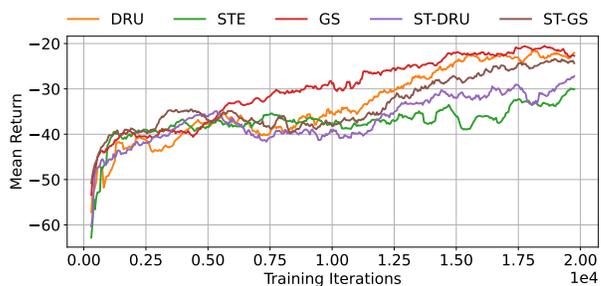
Figure 9: Results in the Speaker Listener environment

Figure 9 and Table 5 show the results in the speaker listener environment. We see that the STE is no longer performing as good as in our earlier experiments. It results in the worst result while the GS and DRU provide the best results. The delay that is caused by the noise in the DRU, GS, ST-DRU and ST-GS is no longer the determining factor in the training speed. The exploration that is done in the communication policy by the DRU, GS, ST-DRU and ST-GS due to the noise appears to have a beneficial result.

5.3 Error Correction

In addition to comparing the different discretization methods in ideal circumstances, we also want to make this comparison in a situation with more uncertainty. Therefore, we perform some additional experiments on the Matrix environment as discussed in Section 5.1. However, instead of perfect communication circumstances without any errors as done before, we flip a certain amount of random bits with a certain probability. This causes the receiver to receive different information than intended by the sender. Depending on the maximum amount of bits that can be flipped, the agents need more message bits to be able to counteract the errors that are introduced. We use a simple Matrix environment with $N = 10$ and $M = 2$. In this experiment, we show the evaluation reward of our agents, measured by performing 100 evaluation episodes after each 100 training iterations. During the evaluation episodes, the agents do not explore and the discretization methods are applied in evaluation mode. In this environment, all of the agents are identical. Therefore, we can use parameter sharing between the agents, which improves their performance significantly as shown in the results of Foerster et al. [2].

We perform a test where there is 50% chance that an error will occur. Normally, the agents would be able to represent both possible incoming numbers using a single bit. However, if they have to be able to correct the errors that get introduced, they require three bits. The results of this experiment can be seen in Figure 10 and Table 5. We see that the STE is not able to correct the errors that occur. Therefore, it is not able to achieve good results. However, the other discretization methods are able to detect and correct the errors. Our hypothesis is that these methods are more robust to errors due to the noise that is used within these discretization methods.

To see how the agents are able to correct the introduced errors, we examine which message the agents choose for which incoming

Table 5: The average return and the standard deviation for each of the experiments during the final 10% of training iterations

Experiment Name	DRU	STE	GS	ST-DRU	ST-GS
Simple Matrix	2.924 ± 0.152	2.999 ± 0.005	2.685 ± 0.275	2.944 ± 0.112	2.906 ± 0.123
Complex Matrix	4.600 ± 0.118	4.972 ± 0.018	4.588 ± 0.048	4.672 ± 0.134	4.764 ± 0.047
Speaker Listener	-22.330 ± 4.514	-32.224 ± 6.833	-21.533 ± 4.758	-29.966 ± 9.430	-24.211 ± 5.177
Error Correction	9.904 ± 0.051	5.140 ± 0.413	9.892 ± 0.051	9.906 ± 0.048	9.842 ± 0.059
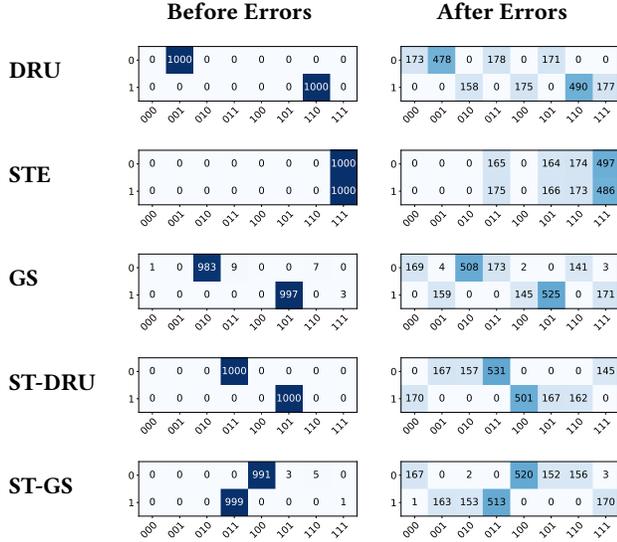
Figure 11: The communication protocol for the Matrix environment ($N = 10$, $M = 2$) for each of the discretization units before and after introducing errors. On the y-axis the different possible input numbers are displayed. On the x-axis the different possible output messages are displayed.

number. Figure 11 shows the different communication policies for the different discretization methods. We can see the output message depending on which input number was given to the agent before and after the errors are introduced. We see that the agents with the DRU, GS, ST-DRU or ST-GS have chosen messages where the possible messages after the introduction of errors do not overlap between the possible input numbers. This way the agents make sure that the messages are still comprehensible, even if errors occur. For the GS we see that there are 9 messages that overlap with the output messages for a different input number after error introduction. The same thing can be observed for the ST-GS in 6 cases. We see that when we use the STE, the agents are not able to find this communication protocol. Even before the errors are introduced, the messages for both possible inputs are the same. This indicates that the agents did not find a useful communication protocol.

6 DISCUSSION

In our experiments, we compared different discretization techniques in different environments where the agents need to learn a communication protocol to achieve the goal. In this section, we discuss some general trends that we saw accross the experiments. Table 5

shows how each of the different methods performed in each experiment. It shows the average return and standard deviation during the last 10% of training iterations. In our results, we saw only small differences in a simple environment. However, the differences become a lot more apparent when using a more complex environment. The STE method performs very well in both the Simple and Complex Matrix environment, while performing worst in the speaker listener and not being able to achieve the goal in the error correction task. This makes the use of the STE as a standard method not recommended, especially in environments where perfect communication cannot be guaranteed. Similarly, the GS either performs the best among the tested methods or the worst. The ST-GS has a more consistent performance than the regular GS. The DRU and the ST-DRU perform very similar except in the speaker listener environment. There, the DRU clearly outperforms the ST-DRU.

Overall, we can state that in most cases, either the DRU, ST-DRU or ST-GS should be used to discretize communication. These methods provide consistent results across the experiments while the STE and GS might achieve a higher return or be faster in some cases but fail dramatically in others.

7 CONCLUSION

In this paper we compared several discretization methods in different environments with different complexities and challenges. We focus on the situation where these discretization methods are used to discretize communication messages between agents that are learning to communicate with each other while acting in an environment.

The results showed that the choice of discretization method can have a big impact on the performance. Across all of the experiments, the DRU, ST-DRU and ST-GS performed best. The STE performs a lot better in the matrix environment in terms of speed and return. However, in the speaker listener environment, the STE performs the worst and in the error correction task, it fails to learn a communication protocol. The GS performs best among all of the methods in the evironments where the STE fails, but performs the worst in the Matrix environment. The DRU, ST-DRU and ST-GS show more consistent results, close to the best result, making them a better standard choice. However, sometimes it may prove useful to perform additional experiments to establish the best discretization method.

REFERENCES

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432 [cs.LG]

[2] 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.

[3] Benjamin Freed, Guillaume Sartoretti, Jiaheng Hu, and Howie Choset. 2020. Communication Learning via Backpropagation in Discrete Channels with Unknown Noise. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (Apr. 2020), 7160–7168. https://doi.org/10.1609/aaai.v34i05.6205

[4] Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. US Government Printing Office.

[5] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. arXiv:1611.01144 [stat.ML]

[6] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z. Leibo, and Nando De Freitas. 2019. Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3040–3049. https://proceedings.mlr.press/v97/jaques19a.html

[7] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[8] Toru Lin, Minyoung Huh, Chris Stauffer, Ser-Nam Lim, and Phillip Isola. 2021. Learning to Ground Multi-Agent Communication with Autoencoders. arXiv:2110.15349 [cs.LG]

[9] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. 2019. On the Pitfalls of Measuring Emergent Communication. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) *(AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 693–701.

[10] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2020. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv:1706.02275 [cs.LG]

[11] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. arXiv:1611.00712 [cs.LG]

[12] Francisco S. Melo, Matthijs T. J. Spaan, and Stefan J. Witwicki. 2012. Query-POMDP: POMDP-Based Communication in Multiagent Systems. In *Multi-Agent Systems*, Massimo Cossentino, Michael Kaisers, Karl Tuyls, and Gerhard Weiss (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 189–204.

[13] 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), 529–533.

[14] Igor Mordatch and Pieter Abbeel. 2018. Emergence of Grounded Compositional Language in Multi-Agent Populations. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17007

[15] David Simões, Nuno Lau, and Luís Paulo Reis. 2020. Multi-agent actor centralized-critic with communication. *Neurocomputing* 390 (2020), 40–56. https://doi.org/10.1016/j.neucom.2020.01.079

[16] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf

[17] Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[18] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[19] Simon Vanneste, Astrid Vanneste, Kevin Mets, Ali Anwar, Siegfried Mercelis, Steven Latré, and Peter Hellinckx. 2021. Learning to Communicate Using Counterfactual Reasoning. arXiv:2006.07200 [cs.LG]

[20] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. 2019. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets. arXiv:1903.05662 [cs.LG]