# Discrete Communication Learning via Backpropagation on Bandwidth-Limited Communication Networks

Benjamin Freed
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213

**Abstract**

Efficient inter-agent communication is an important requirement for both cooperative multi-agent robotics tasks, as well as distributed computing. In both of these domains, the rate at which information can be transferred between robots or computing nodes is often much slower than internal information transfer rates. Techniques exist for learning communication protocols in such bandwidth-limited applications, however these approaches tend to converge very slowly, requiring large amounts of computational power and/or data. To address this problem, we introduce a discrete communication learning approach, which utilizes a stochastic message encoding/decoding procedure to make discrete communication channels mathematically equivalent to an analog channel with additive noise. Gradients can thus be backpropagated through the communication channel, enabling rapid, efficient and robust communication learning. By virtue of using discrete messages, this approach is naturally incorporates bandwidth limitations and is ideally suited to real-world (digital) communication networks. To enable communicating agents to further decrease the amount of information they transmit, we introduce a variable-length message code that provides agents with a means to modulate the number of bits they send to their neighbors. This message-length modulation, combined with a novel a message-length penalty objective, encourages agents to send short messages when possible, enabling agents to minimize their communication requirements while still effectively solving their task. We evaluate these contributions on both a partially observable reinforcement learning task involving robot navigation, as well as a supervised learning task with graph neural networks to model distributed computing. We find that in both tasks, our discrete differentiable communication approach enables communication learning with convergence rates comparable to approaches that allow the transmission of real-valued messages, which have been shown to converge much faster than typical discrete messaging approaches. Additionally, we find that in the supervised learning task, our approach for encouraging limited communication enables comparable levels of validation accuracy to be achieved, with up to a of factor of 34.0 fewer bits exchanged, compared to an approach in which nodes communicate 32-bit precision messages, indicating that our approach provides an effective means to rapidly learn efficient communications in a distributed computing setting.

## 1 Introduction

The research described by this thesis was inspired by the observation that, in both the natural and artificial world, large-scale systems consisting of many simple agents demonstrate complex, robust, and highly coordinated behavior, enabled by effective inter-agent communication. Examples in biology include bee and ant colonies, in which insects are able to demonstrate sophisticated teamwork and cooperation (Jackson and Ratnieks 2006). Behaviors in these insect colonies are so tightly coupled that, in some ways, the entire colony behaves as one organism, capable of much more than the sum of its parts. In the human-made world, the advent of the internet has drastically changed the functioning of many aspects of life, such as business, education, politics, news media, communication, transportation, etc. These changes can largely be attributed to the increased interconnectivity of computers, and the individuals or groups who use them, enabled by the internet.

Simultaneously, the advent of deep learning has changed on a fundamental level the way many of our technologies perform their intended functions. For example, whereas before the advent of deep learning, most image recognition and natural language processing systems utilized hand-crafted feature extractors, now many rely soleley on vast quantities of data to automatically identify the relevant patterns present there (Garcia-Garcia et al. 2017).

In the field of robotics and artificial intelligence, we see the trends of ubiquitous interconnectivity of computers and increased use of deep learning as presenting a unique opportunity for large-scale robotic and other AI systems to capitalize on these trend to more closely mimic the impressive coordination of many-individual biological systems. As the size of our robotic systems increase, so will the need for efficient and effective inter-robot communication, ideally integrating seamlessly with pre-existing communications infrastructure. Simultaneously, as the complexity of tasks robots perform increases, so will the need for control algorithms that automatically learn satisfactory behaviors, rather than requiring them to be hard-coded by human engineers.

Reinforcement learning (RL) offers an effective approach by which robots can automatically learn complex and sophisticated behaviors. RL has recently been successfully applied to complex multi-agent control problems such as DOTA, Starcraft II, and virtual capture the flag (Jaderberg et al. 2019; OpenAI 2018). When RL is applied to the learn control policies for groups of multiple agents concurrently, it is referred to as multi-agent RL (MARL). Many MARL approaches seek to learn decentralized policies, meaning each agent selects actions independently from all other agents, conditioned only on information available to that particular agent (Bernstein et al. 2002; Gupta, Egorov, and Kochenderfer 2017; Sartoretti et al. 2019; Foerster et al. 2017). Such decentralized approaches offer major scalability and parallelizability advantages over centralized planning approaches. Computational complexity of decentralized approaches scales linearly with the team size (as opposed to exponentially, as for typical centralized planners), and action selection for each agent can occur completely in parallel (Busoniu, Babuška, and De Schutter 2010).

Decentralized approaches pay for these scalability and parallelizability benefits with potentially sub-optimal policies or lower degrees of agent coordination, as one might expect given the seeming importance of communication for coordination. Particularly in partially observable environments, a decentralized approach in which agents make decisions based solely on their own limited observations, cannot make as informed decisions as a centralized planner, which conditions all actions on all available information.

In this thesis, we consider MARL problems in which agents have the ability to communicate with each other. This communication ability offers agents a way to selectively exchange information, potentially allowing them to make more informed decisions, while still achieving the same scalability and parallelizability advantages of a purely decentralized approach. However, the addition of communication abilities also increases the difficulty of the learning problem, as agents now have to make decisions not only about what actions to select, but also what information to send, how to encode it, and how to interpret messages received from other agents. This thesis focuses on developing efficient communicaiton-learning techniques that integrate into a MARL framework, and are readily applicable to real-world robotics problems.

While MARL offers compelling motivation for the study of communication learning, the importance of efficient and effective communication is more generally present in distributed computing, of which distributed multi-robot control can be seen as a particular case. Distributed computing refers to a system architecture in which computation is divided among many computing nodes, which are linked by a communication network and run processes in parallel. Communication between computing nodes is crucial for effective distributed computing; because computers do not share memory with one another, the passing of messages through the communication network is the only means by which computers can coordinate their computations in real-time. However, efficient communication is paramount, as the rate of data transfer between nodes is much slower than what would be possible if nodes shared memory (dis 2013).

In this thesis, we study a form of distributed computing that we call neural distributed computing, where the processes run by individual computing nodes are described by neural networks, with regular exchange of information between them in the form of latent vectors passed between neural networks. We choose to study this form of distributed computing because of the increased importance that deep learning has begun to play in our AI systems. However, traditional deep learning systems have not been very concerned with the amount of information transfer required within them, or the fact that virtually all real-world communication networks support only discrete communication. The discrete nature of the communication network poses additional challenges for the training of a distibuted neural computing system, as discretization of real-valued neural network outputs is typically non-differentiable, precluding the use of the highly efficient gradient-based optimization techniques that have been developed for the training of these neural networks (Rumelhart, Hinton, and Williams 1986; Kingma and Ba 2014). For this reason, we study the the problem of distributed neural computing on a bandwidth-limited discrete communication network.

In this thesis, we present a novel approach to differentiable communication learning that utilizes a stochastic message-encoding scheme to make discrete (and therefore non-differentiable) communication channels behave mathematically like a differentiable, analog communication channel. We can use this technique to obtain unbiased, low-variance gradient estimates through a discrete communication channel, enabling efficient neural network parameter optimization, given a differentiable objective function. We first demonstrate our technique in a RL domain, on a simple but illustrative two-agent partially observable robot navigation task. Here, agents must exchange information at every time-step to navigate to randomly selected goal locations, using only discrete messages exchanged between them.

To demonstrate the effectiveness of our approach on a distributed computing task in which computational nodes share only limited information, we apply our communication learning approach to a supervised learning problem involving graph node classification. Here, we modify a pre-existing graph neural network (GNN) architecture known as a graph attention network, to support only discrete communication between nodes. Here, we show that architectures utilizing our differntiable discrete communicatin approach are able to achieve performance comparable to or in excess of architectures that allow fully real-valued vector messages to be exchanged. Furthermore, we demonstrate that our approach is extremely robust to large variations in message discretization fineness and message length. In our reinforcement-learning task, we show that agents using our communication-learning approach learn much faster, and achieve much higher reward, when compared to an approach in which communication behavior is not optimized through gradient backpropagation, but instead through RL, in a manner that is analagous to how the standard behavioral policy is optimized. Agents also achieve a learning speed and final reward comparable to agents that are able to exchange real-valued messages, even when binary messages are used (representing the maximum possible discretization coarseness). Agents continue to demonstrate learning speed and performance comparable to the real-valued message approach as message length and discretization fineness are decreased, up until the point that agents are no longer physically capable of exchanging enough information to optimally solve the task.

In the supervised learning task, our graph attention network that utilizes discrete-message passing, trained to optimize a node classification objective on the Cora bibliographic dataset (Sen et al. 2008), actually achieves validation accuracy on a held-out dataset in excess of a similar architecture that differs only in that it is able to exchange real-valued messages, containing much more information than the discrete messages of our modified GAT.

In many situations, it is desirable for agents to communicate as little as possible during the execution of a task; for example, resources may be consumed during communication, or the communication network may need to be left as free as possible for other uses. To enable learning of such minimal communication behaviors, we present a technique for encouraging agents to

communicate messages containing as few bits as possible, while still effectively solving the task. We first introduce a variable-length coding scheme, which gives agents a means by which they can modulate the information content of their messages. We then introduce a message-length penalty, that closely matches a per-message-bit cost, while remaining differentiable. We add this penalty term to the original supervised learning objective used for our GNN experiment, and observe the impact it has on both the average amount agents communicate with one another, and the validation accuracy achieved by the network, for various penalty-term weightings. We find that information exchange can be decreased by as much as a factor of 34.0, without substantially decreasing validation accuracy. This indicates both that our approach is an effective means for minimizing the communication requirements of a neural-network-based distributed AI system, and that in real-world tasks, great potential exists for decreasing communication requirements compared to standard differentiable communication-learning approaches.

## 2 Background

The problem of communication learning has been addressed within two distinct subfields: MARL, and graph neural networks. Here we give a brief overview of the progress made on communication learning whithin these two areas, and their relationship to one another.

### 2.1 Multi-agent Reinforcement Learning with Communication

Many past approaches to MARL with inter-agent communication fall into one of two general categories: 1) approaches in which communication is treated as a differentiable process, allowing communication behavior to be optimized via backpropagation (differentiable approaches) (Foerster et al. 2016; Sukhbaatar, Szlam, and Fergus 2016; Mordatch and Abbeel 2017; Paulos et al. 2019), and 2) approaches in which messages are treated as an extension to the action space, and communication behavior is optimized via standard RL (reinforced communication learning, RCL) (Foerster et al. 2016; Lowe et al. 2017; Freed, Sartoretti, and Choset 2020).

RCL approaches tend to be more general than differentiable approaches (Lowe et al. 2017). This is because the communication channel, and all downstream processing of messages selected by agents, is considered to be part of the environment, and is therefore treated as a black box, meaning that no assumptions are made about what influence a particular message may have on other agents or future state transitions. RCL approaches therefore naturally handle unknown channel noise (Lowe et al. 2017). Additionally RCL naturally handles discrete communication channels because RCL does not require backpropagation through the communication channel, so message selection can be non-differentiable.

The downside of RCL's generality is that it typically requires significantly more learning updates to converge to a satisfactory policy, compared to differentiable approaches (Foerster et al. 2016). In RCL, agents are not explicitly provided with knowledge of how their message selection impacts the behavioral policy of other agents, and must instead deduce this influence through repeated trial and error. On the other hand, differentiable approaches allow one to explicitly compute the derivative of recipient agents' behavioral policy or action-value function with respect to the sending agent's communication policy parameters. This allows differentiable approaches to converge to better policies after fewer learning updates compared to RCL.

One additional advantage to differentiable approaches is that because the objective function used to optimize communication in general does not directly depend on communication behavior, it is possible to train communication behavior via imitation learning, with no explicit expert demonstrations of communication. This is useful when one can generate expert action demonstrations but not expert communication demonstrations, e.g., when a communication-unaware centralized expert is present (Paulos et al. 2019).

Several differentiable approaches in the past have allowed agents to exchange real-valued messages(Foerster et al. 2017; Sukhbaatar, Szlam, and Fergus 2016). Differentiable approaches that consider real-valued communication signals cannot naturally handle discrete communication channels, as a discrete channel is non-differentiable. Some recent approaches have circumvented this problem by using biased gradient estimators (Foerster et al. 2016; Mordatch and Abbeel 2017; Lowe et al. 2017); however, biased estimators typically require additional tuning parameters or more complex training techniques such as annealing (Foerster et al. 2016; Mordatch and Abbeel 2017; Jang, Gu, and Poole 2016). Moreover, biased gradient estimators lack the convergence guarantees of unbiased ones. Additionally, to the best of our knowledge, existing differentiable approaches cannot function with unknown channel noise, because the channel then represents an unknown stochastic function whose derivatives are consequently unknown. The fact that differentiable approaches are restricted to channels with no noise or known noise models limits their applicability to real-world robotic systems.

In the more general context of learning-based coding, several recent works have learned a source- or channel-coding method (Kim et al. 2018; Farsad, Rao, and Goldsmith 2018), because a learned coding scheme can in some cases be more optimal or more computationally efficient than a hand-crafted one. Typically in these prior works, a coding scheme is learned that minimizes a reconstruction metric that penalizes loss of information. There is a connection between these works and ours, in that one can think of our agents as learning a source-coding scheme that optimizes the reward signal, which implicitly penalizes loss of useful information, because agents must compress information available to them into a fixed-length message in a way that maximizes reward. We believe that this emphasis on useful information allows our agents to be even more selective with what information they choose to send than if a generic supervised loss function was used.

## 2.2 Graph Neural Networks

GNNs are a neural network architecture that can be seen to utilize a form of inter-node communication, and perform a learned distributed computation. GNNs generalize the notion of a convolutional neural network, which typically works on 1- or 2-dimensional grids, to arbitrarily graph structures. A very general GNN formulation is the message-passing GNN (MPGNN) (Gilmer et al. 2017). In this architecture, the forward pass through the network has two phases, a message-passing phase and a readout phase. Nodes initially each receive an input, which they each use to compute their initial node feature vectors. During the message-passing phase, each node computes a message to be sent to each of its neighbors. Each node then receives the messages sent to it by its neighbors, and passes them through its *message function*. These transformed messages are then aggregated according to a permutationally invariant function, such as summation. This aggregated message, along with the that node's current node feature vector, are passed through the *update function*, to compute the node feature vector at the next timestep.

After $T$ rounds of message passing, the network transitions to the readout phase, in which a feature vector is computed according to the *readout function*, which takes the set of all node feature vectors as input. Unless explicitly stated otherwise, all functions described above are learned neural networks.

Because the all the processes by which the MPGNN architecture computes and processes messages, updates the internal state of nodes, and computes final outputs are all extremely general, we feel that the MPGNN architecture offers a very general framework for distributed neural computing. One additional component that must be added in order for a MPGNN to be considered distributed neural computing, is that there should be a one-to-one correspondence between nodes in the MPGNN, and computing nodes, *i.e.*, the computations executed by exactly one node (message computation, message processing, node feature vector updates) should all be performed on exactly one computing node. Furthermore, the messages passed during an MPGNN's computation should be sent only via the communication network that links the computing nodes.

# 3 Theory

In this section we explain our approach to differentiable communication learning with a discrete communication channel. Throughout this thesis, we assume centralized training, meaning that all training data (agent/node observations and ground-truth communication signals) can be thought of as being sent to a centralized server where learning updates are computed for each agent, as is done in many MARL approaches, such as (Kraemer and Banerjee 2016; Foerster et al. 2017; 2016). We feel that this is a reasonable assumption, because often training takes place in a setting in which more state information is available than what will be available to agents at execution time. However, we also assume decentralized execution, meaning that at execution time (i.e., after the training has finished/converged), agents do not need to transfer data among themselves or with a centralized server, other than via the communication channels available to them within the environment.

## 3.1 Differentiable Communication in a Discrete Channel

Consider a multi-agent RL problem with $K$ agents. Assume that, at a given timestep, a particular agent (Alice) is able to communicate to another agent (Bob) via a noise-free discrete communication channel with capacity $C$ (in bits). For now, we neglect details associated with the learning algorithm used to train the agents, and assume only that agents are trained via some form of gradient descent to minimize a differentiable loss function. This assumption is sufficiently general to encapsulate most prominent learning algorithms, such as supervised, semi-supervised, and unsupervised learning, as well as model-free RL algorithms such as Q-learning and policy gradient. Additionally, we do not specify what aspects of the environment determine which agents can communicate, as this is problem-specific. Here we are primarily concerned with how gradients are backpropagated through the channel.

Because the communication channel is discrete, any message $m^{a \to b}$ sent through the channel from Alice (a) to Bob (b) is taken from a finite set of possible messages, *i.e.*, $m \in M$, where $M = \{m_1, ..., m_{|M|}\}$, and $|M| \leq 2^C$. If we wish to perform differentiable communication learning in such an environment, we need to compute the derivative of the input of Bob with respect to the output of Alice. However, the derivative of the channel output with respect to the input is not well-defined, making typical backpropagation impossible.

Existing approaches use a biased gradient estimator to circumvent this problem, such as the Gumbel-Softmax estimator (Jang, Gu, and Poole 2016). Such biased gradient estimator approaches typically require real-valued channels to be simulated during training, as opposed to the discrete channel that will be available during execution, as well as additional tuning parameters, such as temperature and annealing rate in the case of Gumbel-Softmax estimators. We propose an alternative, more streamlined approach, described below, in which Alice generates a real-valued communication signal $z$, which is encoded into a discrete message by a stochastic quantization procedure (stochastic encoder). The resulting discrete message $m^{a \to b}$ is sent through the communication channel and is received by Bob, who then uses this discrete message to compute an approximation of the original real-valued signal generated by Alice, $\hat{z}$, using a stochastic dequantization procedure (stochastic decoder) (fig. 1). We show that the encoder/channel/decoder system is mathematically equivalent to an analog communication channel with additive noise, allowing the partial derivative of Bob's reconstruction with respect to Alice's communication signal to be computed. Once again, because we assume centralized training, Bob's learning gradient is available to Alice for the computation of her learning update.

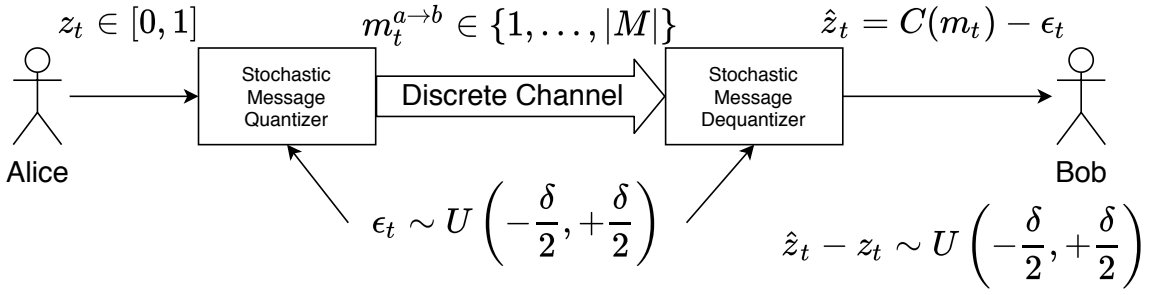


Figure 1: Stochastic encoder/channel/decoder: The real-valued output $z$ from agent Alice is converted into a discrete message $m$ by the stochastic encoder, which additionally takes random noise variable $\epsilon$. The message $m$ is then sent through the channel to agent Bob, who decodes it into $\hat{z}$ (a reconstruction of $z$), using the stochastic decoder, which also takes $\epsilon$ as an input. Under the simple reparameterization $\hat{z} = z+e$, $\hat{z}$ becomes a continuous, differentiable function of $z$, allowing gradients to be naturally backpropagated through the discrete communication channel.

**Stochastic Encoder** The purpose of the stochastic encoder is to convert a real-valued output from Alice into a discrete message that can be sent through the channel to Bob. One simple approach would be to quantize $z$ directly; however, the derivative of this operation then vanishes everywhere except at the boundaries of the quantization intervals, where it is undefined, precluding the use of this simple idea in a differentiable communication setting.

Instead, we propose the following non-deterministic quantization method (Fig. 1) [1]:

1. Alice's real-valued communication output $z_t$ is perturbed by noise $\epsilon_t \sim U(-\frac{\delta}{2}, \frac{\delta}{2})$, where $\delta = \frac{1}{|M|-1}$, *i.e.,* $z'_t = z_t + \epsilon_t$.
2. $z'_t$ is quantized according to a uniform quantization procedure, in which $z'_t$ is assigned to one of $|M|$ evenly spaced quantization intervals, ranging from $\frac{-\delta}{2}$ to $1 + \frac{\delta}{2}$. $m_t^{a\to b}$ is taken to be the index of the quantization interval.
3. $m_t^{a\to b}$ is sent through the discrete communication channel to Bob.
4. Bob creates a reconstruction of $z_t$, denoted $\hat{z}_t$, by taking the center of the quantization interval corresponding to $m_t^{a\to b}$, and subtracting $\epsilon_t$ from it.

Note that the integer representation of the message is arbitrary, as the message can be converted to any form (e.g., binary) for transmission through the communication channel. For messages of length $> 1$, the above procedure can be applied, element-wise, to every element in $z$, producing a vector message $m = [m[1], ..., m[L]]$, where $m[i] \in M$.

**Stochastic Decoder** The goal of the stochastic decoder is to reconstruct $z$ from $m^{a\to b}$, denoted $\hat{z}$, such that $\hat{z}$ is as a continuous, differentiable function of $z$, and a noise source that is independent of $z$.

Upon receiving $m^{a\to b}$, Bob computes the reconstruction as $\hat{z} = C(m) - \epsilon$, where $C(m)$ denotes the center of the quantization interval corresponding $m$, $C(m) = \delta(m + \frac{1}{2})$ (fig. 2). In short, the reconstruction is given by:

$$\hat{z} = \delta\left(m - \frac{1}{2}\right) - \epsilon. \tag{1}$$

We prove in the supplementary material that using the particular encoding/decoding procedure detailed above, $\hat{z}$ can be reparameterized according to $\hat{z} = z + e$, where $e \sim U(-\frac{\delta}{2}, +\frac{\delta}{2})$. Under this reparameterization, the partial derivative of $\hat{z}$ with respect to $z$ is precisely 1. The encoder/channel/decoder system therefore becomes mathematically equivalent to an analogous situation in which we send real-valued $z$ directly through an analog communication channel, that introduces independent additive noise to the signal, but is nonetheless differentiable.

Note that Bob requires knowledge of the value of $\epsilon$ that Alice used to encode $m^{a\to b}$, which we do not send through the channel. This is not a problem, because $\epsilon$ does not depend upon $z$ and we can therefore assume the agents agree ahead of time upon a sequence of $\epsilon$ values to use for each timestep. Practically, this would most likely amount to all agents using the same pseudorandom number generator with the same seed. To generalize the complete encoding/decoding procedure to cases in which $z$ is a vector rather than a scalar, one can apply the above procedure element-wise with a distinct value of $\epsilon$ to each component of $z$.

It can also be shown that, somewhat counter-intuitively, since Alice and Bob agree ahead of time on the sequence of random perturbations to use for the stochastic quantization/dequantization, this introduction of noise does not decrease the maximum

[1] For simplicity, assume $z$ is scalar and $z \in [0, 1]$, however the encoding process can be easily generalized to vectors of arbitrary length. Additionally, we assume the set of possible messages are the integers 1 to $|M|$, *i.e.*, $M = \{1, ..., |M|\}$.

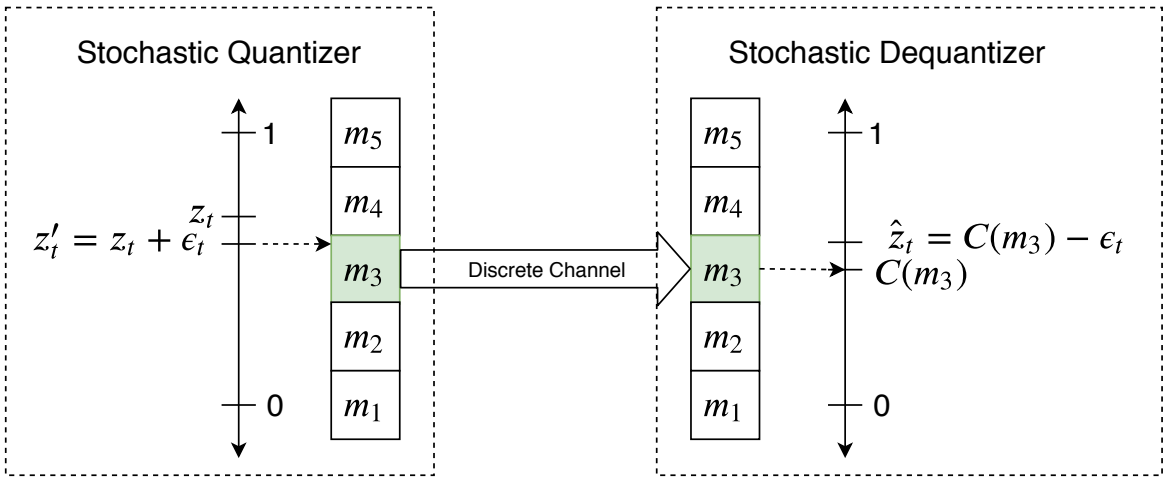


Figure 2: stochastic encoder and decoder: The real-valued output $z$ from agent Alice is first perturbed by random noise $\epsilon$, which is sampled from a zero-centered uniform distribution with width equal to the quantization interval to form $z'$. $z'$ is then quantized into one of the $|M|$ quantization intervals, the index of which is taken to be the discrete message $m$. The message $m$ is finally sent through the channel to agent Bob, who uses it to compute a reconstruction of $z$ by subtracting $\epsilon$ from the center of the quantization interval corresponding to $m$.

amount of information agents can send through the channel: the deleterious effect of this noise can in a sense be "undone" by the receiver. This property can easily be seen in the context of an illustrative example in which Alice is able to communicate with Bob at the maximum possible rate (the channel capacity). More specifically, if the channel supports the transmission of one of $|M|$ possible discrete values at a given time step, and Alice observes one of $|M|$ possible observations, $x \in \{1, ..., |M|\}$, then Alice can use her stochastic message quantizer to select one of $|M|$ possible messages in such a way that allows Bob to reproduce, error-free, the original value of $x$. For example, Alice can output $z_t = \frac{x_t}{|M|-1}$, which Bob can reconstruct as $\hat{z}_t = z_t + e_t$, with $e_t \sim U\left(-\frac{1}{2(|M|-1)}, \frac{1}{2(|M|-1)}\right)$, and map this to a reconstruction of $x_t$ according to $\hat{x}_t = x_t + (|M| - 1)e_t$. It can be seen from this example that, because the error in $\hat{x}_t$ is less than $\frac{1}{2}$ with probability 1, Bob may always deduce the original value of $x_t$, by simply mapping $\hat{x}$ to the closest allowed value of $x_t$. On the other hand, if Bob and Alice had not agreed ahead of time on a sequence of random numbers to use in their encoding, the error distribution would be wider, and unique reconstruction of $x_t$ would no longer be possible. In other words, information would be lost.

### 3.2 Limiting Agent Communication

To further encourage agents to utilize their communication channel efficiently, we provide agents with a means to regulate the length of the messages they send, and derive a principled message-length penalty term that acts as an upper bound on message length, while remaining differentiable.

**Variable-Length Message Coding** So far, we have taken a message to be a vector of integers, $m = [m_1, ..., m_L]$, where $m_i \in M = \{1, ..., |M|\}$. However, typical communication channels are digital, and thus a mapping from integer-vector messages to bits (a code) is required. To enable agents to adjust the number of message bits they send during a task, we adopt a variable-length code. This variable length code (which we fix *a priori*), maps successively larger integers in $M$ to successively longer bitstrings. Rather than representing the choice of message length as an additional set of actions available to the agents, the variable-length code allows us to incorporate this choice implicitly into agents' standard communication behavior. The variable-length code can be thought of as providing a means for agents to perform data compression on their messages. Whereas in typical data compression, the frequencies of source symbols are fixed, and a code is chosen to maximally compress the the source data, in our case we take the code to be fixed, and allow agents to modulate the frequencies of source symbols (in this case, discrete messages). In the ideal case, agents would choose to use messages with shorter binary representations more frequently, and use longer messages infrequently (only when needed). In our experiments, we adopt a coding scheme that maps $m = 0$ to the empty string (a message is not sent), $m = 1$ to 0, $m = 2$ to 1, $m = 3$ to 00, $m = 4$ mapping to 01, and so on (Table 1).

### 3.3 Message-Length Penalization

To encourage agents to exchange the minimum amount of information necessary in order to solve the task, we would like to introduce a penalty term to the original loss function that imposes a fixed cost proportional to the number of message bits sent

| | | |
|---|---|---|
| 1 | → | empty |
| 2 | → | 0 |
| 3 | → | 1 |
| 4 | → | 00 |
| 5 | → | 01 |
| 6 | → | 10 |
| ⋮ | | ⋮ |

Table 1: The variable length code that we adopt maps successively larger integer messages to successively longer bitstrings.

by each agent. However, such a penalty term would not be differentiable. Therefore, we introduce a surrogate message-length penalty term, given by

$$\ell(z_t^e) = \sum_{k=1}^{L} \log_2 \left( |M| z_t^e[k] + 1 \right), \tag{2}$$

where $z_t^e[k]$ is the $k$th element of the real-valued communication vector that is used to compute the message from agent $i$ to agent $j$ at time $t$.

The choice of this $\ell$ is based on the fact that $\ell$ can be shown to be an upper bound for expected number of bits in $m_{b,t}^e$ (that is, the binary message that is actually sent along edge $e$ at time $t$), given $z_t^e$.

*Proof.*

The expected value of discrete (integer) message $m_t^e$ that will be converted to binary and sent along edge $e$ at time $t$ given $z_t^e$ is

$$\mathbb{E}\big[m_t^e \big| z_t^e\big] = |M| z_t^e, \tag{3}$$

For our code,

$$\text{length}(m_{b,t}^e) \leq \log_2(m_t^e + 1). \tag{4}$$

Taking the conditional expectation of 4 of the length of the binary message given $z_t^e$, we have that

$$\mathbb{E}\big[\text{length}(m_{b,t}^e) \big| z_t^e\big] \leq \mathbb{E}\big[\log_2(m_t^e + 1) \big| z_t^e\big]. \tag{5}$$

Applying Jensen's inequality, we arrive at

$$\mathbb{E}\big[\text{length}(m_{b,t}^e) \big| z_t^e\big] \leq \log_2 \left( \mathbb{E}\big[m_t^e \big| z_t^e\big] + 1 \right) \tag{6}$$

$$\leq \log_2 \left( |M| z_t^e + 1 \right). \tag{7}$$

This penalty term can be thought of as a log-regularizer on $z_t^e$, encouraging it to be as small, and messages to therefore be short.

The above penalty term is multiplied by scalar weight $\lambda$, and added to the original objective function. The weight $\lambda$ can be used to regulate the amount agents communicate, and balance this with decreases in accuracy that may be suffered as a result of reduced information-sharing between nodes. The full objective function therefore becomes

$$L = L_0 + \lambda \frac{1}{|E|TL} \sum_{e \in E} \sum_{t=1}^{T} \ell(z_t^e), \tag{8}$$

where $E$ denotes the set of all edges present in the communication network, and $T$ represents the total number of timesteps for which messages are passed along those edges.

While it may appear that our proposed coding scheme is overly optimistic, as it assumes no additional overhead bits are necessary for message synchronization or error correction, our message penalty term is actually applicable to a wide range of possible codes. It is in fact equally suited to any code with codewords that differ in length from ours by multiplicative or additive constants, because multiplicative constants can be absorbed into the penalty weight, and additive constants simply impart a constant offset to the objective function and therefore do not change the optimal solution. This property allows our approach to function with a wide range of possible codes.

## 4 Experiments

We evaluate the performance of our proposed differentiable discrete communication learning approach in two example domains: an RL task, as well as a much larger-scale supervised learning task, using GNNs.

### 4.1 Reinforcement Learning

We compare the performance of our differentiable discrete communication learning approach to a differentiable approach with real-valued messages, and a reinforced communication learning approach, on a simple but illustrative example robot navigation task. In the example task, two simulated robots are initialized on random grid cells within a 10 x 10 grid world. At each timestep, both robots can move in either of the 4 cardinal directions (up, down, left or right), or remain stationary. Each robot is assigned a randomly selected goal grid cell, for which it receives positive reward for navigating toward. However, neither agent is able to directly observe its own goal location. Instead, each agent observes only its location, as well as the goal location of the other robot. Both locations and goals are encoded as separate 10 x 10 one-hot matrices, which are concatenated to form the input to each agents' policy network, a 2 x 10 x 10 tensor.

The actor-critic algorithm is used as the RL algorithm for all experiments, with separate actor and critic networks, composed of a convolutional stack followed by two fully connected layers. The same architecture is used for both the differentiable and RCL approaches we test, except for differences to the message inputs and outputs necessitated by the differences between the two approaches. Agents' policy networks are feed-forward neural networks with no persistent state, meaning they cannot remember past messages or observations, and must therefore communicate at every timestep. Because the critic network is used only during training, we choose to give it full state observability in all experiments. During training, the policy networks are given only information they are will have during execution time. In both tasks, we consider 2 agents that are each able to send the other 40 bits of information at each timestep. In both the differentiable and RCL approaches we test, policy parameters are shared across agents, and each agents' action policies are represented as a categorical distribution over discrete actions. In our RCL approach, messages between agents are strings of 40 bits, and the communication policy of each agent (from which a message is sampled and sent to the other agent at each timestep) is represented as a collection of independent Bernoulli distributions representing the probability of each corresponding message bit being a 1. In our differentiable approach, at each timestep, each agent outputs a real-valued vector containing 10 elements. Each element of this communication signal is then discretized independently into one of 16 discrete message elements, meaning each element can be thought of as a 4-bit word, allowing for a total of $16^{10} = 2^{40}$ different possible messages, equivalent to 40 bits.

We utilize a shared reward structure, in which agents both receive the same total reward, computed as a sum of both agents' individual rewards. At each timestep, agents are rewarded proportionately to how much closer to their goals they become during that timestep, with an additional penalty for every timestep the agents have not reached their goal, and a penalty for each timestep agents choose not to move while not at their goal location to encourage exploration, similar to (Sartoretti et al. 2018a; 2018b).

We found that our proposed discrete differentiable communication learning approach drastically outperformed RCL approaches, and was able to solve the task nearly as rapidly as a real-valued differentiable approach that places no limit whatsoever on information transfer rate through the channel (Fig.3).

To assess the affect of message quantizaiton fineness, we additionally compare the performance of our discrete differentiable approach on the robot navigation problem with various numbers of quantization levels and message lengths. Agents have no memory and no knowledge of the location of the other agent, and must therefore encode the other agent's goal location in the message, requiring $\log_2(100) = 6.6439$ bits of information. We first test 10 element messages with real-valued messages, and 16, 8, 4, and 2 quantization levels per message element. In all these cases, agents can transmit enough information to solve the task optimally. We then compare 5- and 2-element messages, each with 2 quantization level per element, in which case not enough information is transmitted to optimally solve the task (*i.e.*, fewer than $6.6439$ bits). These systematic experiments allow us to assess how efficiently agents make use of their available message bits.

We found our approach to be robust to large variations in message length and discretization fineness. Final performance in the case of 10-length messages with 16, 8, and 4 quantization levels per message element is comparable to that of real-valued messages (Fig. 4). In the case of 10- and 5-length messages with 2 quantization levels, final performance is slightly worse; however, the fact that performance did not deteriorate until the available channel capacity was very close to the theoretical minimum required to solve the task optimally (6.6439 bits) indicates that agents were able to make efficient use of the channel available to them. In all cases, training was stable despite the noise introduced through the stochastic quantization procedure, and learning efficiency remained high. Performance in the case of 2-length messages with 2 quantization levels was far inferior to the other cases tested, as one would expect given not enough information is exchanged to solve the task optimally.

### 4.2 Graph Neural Network Supervised Learning

A graph is a natural model for a network of communicating computational nodes. Nodes which are able to communicate directly with one another, without sending data through an intermediate node, can be thought of as adjacent in the graph. By extension, a GNN is a natural, and very general, structure for decentralized neural computation. Here we assume that at timestep 0, every computational node receives some data as input. Then, after some fixed number of message passing rounds, each node should output a desired value, that will be trained via gradient descent. We assume that the communication channels between computational nodes are bandwidth-limited, and therefore we wish to minimize the number of total message bits sent through them while still solving the task to a satisfactory level. While typical GNN models have no way to directly modulate the amount of information nodes communicate amongst themselves during computation, our model of discrete differentiable communication provides us with a means to limit this communication.

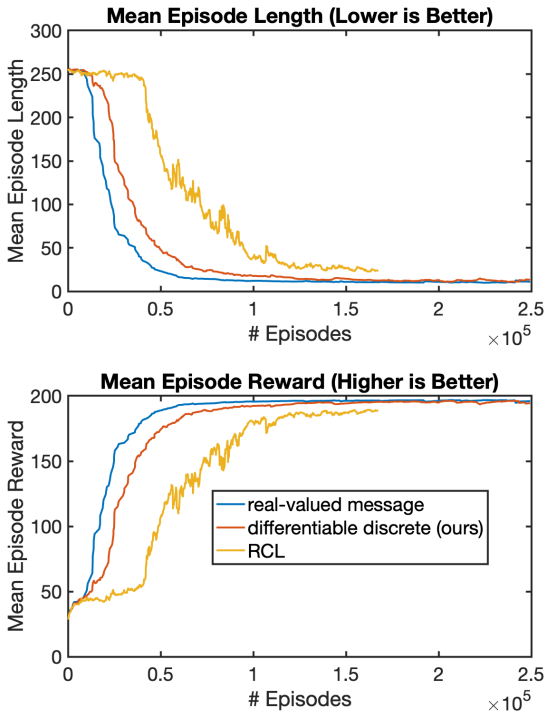


Figure 3: Mean episode length (top) and mean episode reward (bottom) vs. number of episodes completed during training, for a differentiable communication learning approach that uses real-valued messages, our differentiable discrete approach, and an RCL approach. Our differentiable discrete approach displayed convergence speed and final performance comparable to the real-valued messages approach, and converged much faster and to a better final performance compared to the RCL approach.

To obtain a GNN model to feature discrete communication between nodes, we adapted a recent GNN model, the graph attention network (GATs) (Veličković et al. 2017), to discretize messages using our discrete differentiable approach. We then use this architecture to model the popular The Cora bibliographic data set (Sen et al. 2008), which contains 2708 machine learning papers divided into one of seven classes, each associated with a 1433-dimensional feature vector, and 5429 links between papers. In our experiment, we imagine each paper's feature vector is input to one particular node in a network of computational nodes, and each node may only communicate with nodes that share links in the dataset.

To assess the effect of message discretization fineness on the stability and speed of training, and accuracy of the trained model, we train the same GAT architecture with real-valued messages, as well discrete messages, with 16, 8, 4, and 2 possible message values per message element. We plot the training loss and validation accuracy vs. epoch for each of these models (Fig 5), and also report the final validation accuracy achieved for each model (Table 2). Here, our findings were similar to the analogous experiment in the RL domain: training was very robust to varying levels of message discretization, and remained stable even in the case of binary messages, despite the large amount of noise that was introduced to the nodes' real-valued message reconstructions. Additionally, we found that validation accuracy of the models that exchanged discrete messages remained comparable to the model that was able to exchange real-valued messages, and in fact exceeded it slightly in all cases, as can be seen in Table 2. The reason for this improvement in performance with message discretization is likely that the limit on message information content acts as a form of regularization, and helps prevent overfitting. Indeed, the authors of the original GAT paper found it beneficial to introduce regularization techniques such as neighbor dropout, in which nodes ignore randomly selected neighbors at a given timestep. We chose not to implement this form of regularization, as we felt that incorporating other forms of message channel noise could obscure the effects of message discretization on the model. The authors of the GAT paper report a slightly higher test accuracy than our models achieved (83.0%), indicating that our results could likely be improved by incorporating additional forms of regularization. We leave the exploration of regularization techniques to future work.

To assess the effect of message-length penalization on both validation accuracy and quantity of information exchanged between nodes, we again train a GAT with real-valued messages, as well as discrete messages, for various discretization finenesses and message penalty term weightings. We assume the GAT that uses real-valued messages approximates these with 32-bit precision vectors. We additionally evaluate the performance of a GAT that exchanges messages of length 1 (*i.e.*, one 32-bit scalar

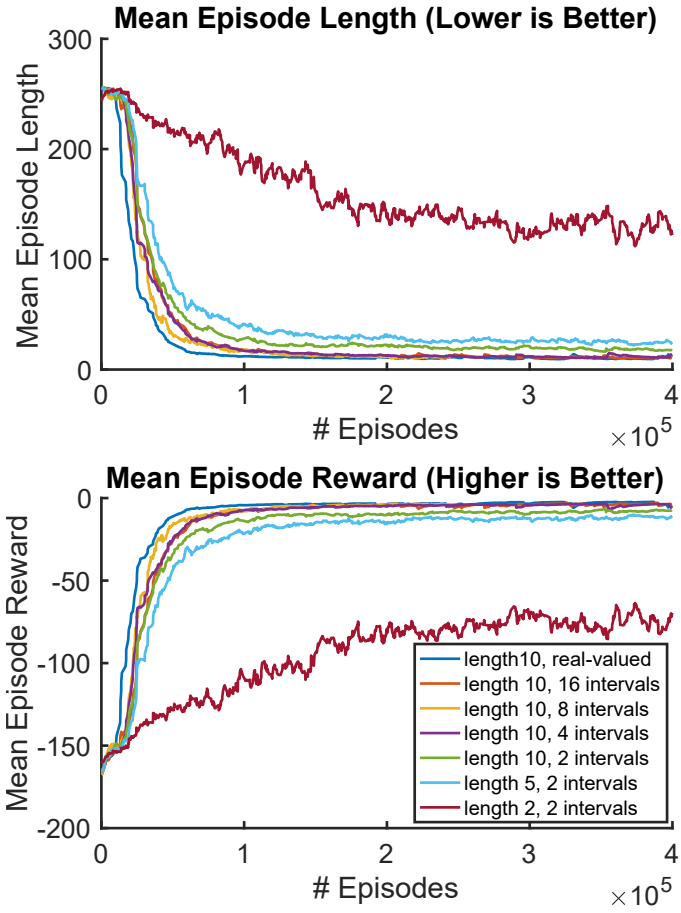


Figure 4: Mean episode length (top) and mean episode reward (bottom) achieved by our differentiable discrete communicaiton learning approach on the RL task, for various message lengths and discretization fineness. Convergence speeds and final performance were comparable for all but the shortest and most coarse messages, when too little information could be exchanged to optimally solve the task.

message), in order to assess the performance of this GAT when its messages are restricted to use a similar amount of information to those that send discrete messages. We report the final average number of bits sent down each edge during execution on the validation dataset for the two real-valued message GATs, as well as the discrete message GATs, for both 16 and 8 possible message values per message element, and message-length penalty weights of $0$, $10^{-6}$, $10^{-5}$, $10^{-4}$, $10^{-3}$, $10^{-2}$, and $10^{-1}$ 3. We additionally display the same data on a plot of validation accuracy vs. mean message bits communicated, so that the trade-off between these two performance metrics can be better visualized (Fig. 6). As can be seen from the table, the number of message bits used decreases significantly with increases in $\lambda$, indicating our message-length penalty approach is an effective way to limit agent communication. Additionally, validation accuracy remains comparable, even as average message bits exchanged decreases substantially, up until the most extreme value of $\lambda$ tested, and average message bits have been decreased by over a factor of 64 from their unconstrained values. This finding indicates that our discrete communication learning approach is able to make efficient use of the commutation channel, and that far fewer message bits need be exchanged to solve the task than one might expect, given the amount of information that is typically exchanged in GNNs. Finally, and perhaps most importantly, when our differentiable discrete communication approach is compared to the real-valued message approach, it can be seen that any discrete-message GAT that achieves comparable validation accuracy to the real-valued message GAT, does so while exchanging drastically fewer bits, up to a factor of 34.0. Additionally, when the real-valued message GAT is made to operate under similar constraints as a discrete-message GAT, as is the case when its message length is set to 1, its validation accuracy falls dramatically, and is comparable to a discrete-message GAT that exchanges over 90-fold fewer bits. This observation indicates that performance similar to our differentiable discrete communication approach under similar bandwidth constraints cannot be achieved from a network that utilizes 32-bit messages, simply by shortening the length of the message.

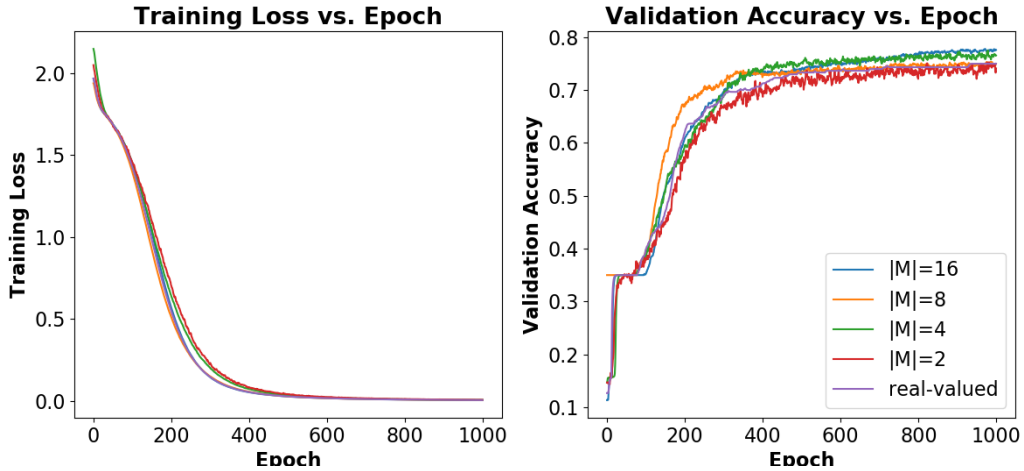


Figure 5: Training loss (left) and validation accuracy (right) vs. training epoch for various levels of discretization fineness on our supervised learning GNN task. Convergence speeds were comparable for all approaches, including one in which real-valued messages were used. Final validation accuracies were also comparable, and were in fact slightly higher in all cases in which discrete messages were used.

| $\lvert M \rvert$ | Validation Accuracy |
|---|---|
| real-valued | 0.7500 |
| 16 | 0.7767 |
| 8 | 0.7533 |
| 4 | 0.7667 |
| 2 | 0.7533 |

Table 2: Final validation accuracy of the GNN model trained using supervised learning and our differentiable discrete communication approach, as well as a a GNN that used real-valued messages. Validation accuracy was slightly higher for all cases that used discrete messages.

## 5 Conclusions

In this thesis, we presented a novel stochastic message discretization procedure that allows differentiable communication learning to be effectively applied to distributed computing tasks on networks that support only discrete communication. This form of communication learning is made possible by the fact that, with our proposed technique, the encoder/discrete channel/decoder system becomes mathematically equivalent to an analog channel with additive noise, through which derivatives can easily be backpropagated. We test this approach on both a RL task to simulate communicating robots, as well as a supervised learning task involving GNNs, to simulate distributed computation with a distributed network of communicating computational nodes. In both the RL and supervised learning domains, we find this approach to be robust to choice of message quantization fineness and message length, even for very coarse message discretizations such as binary messages. Furthermore, we demonstrated that our approach achieves performance comparable to (and in the case of the supervised learning task, in excess of) that of a real-valued message communication approach, where no limits are placed on message information content.

Additionally, we introduced a variable-length message code, which maps larger message values to longer binary messages, providing agents with a means to modulate the number of bits they send to their neighbors at a given time-step. We then derived a differentiable message-length penalty term that acts as a surrogate for a non-differentiable fixed cost per bit. When this penalty is added to the existing objective function and combined with the variable length code, we obtain an effective means for encouraging agents to minimize the quantity of information they communicate while still effectively solving the task. In our GNN experiment, we show that nodes are able to achieve a comparable level of validation accuracy to the approach using 32-bit precision messages, while communicating 34.0-fold less information.

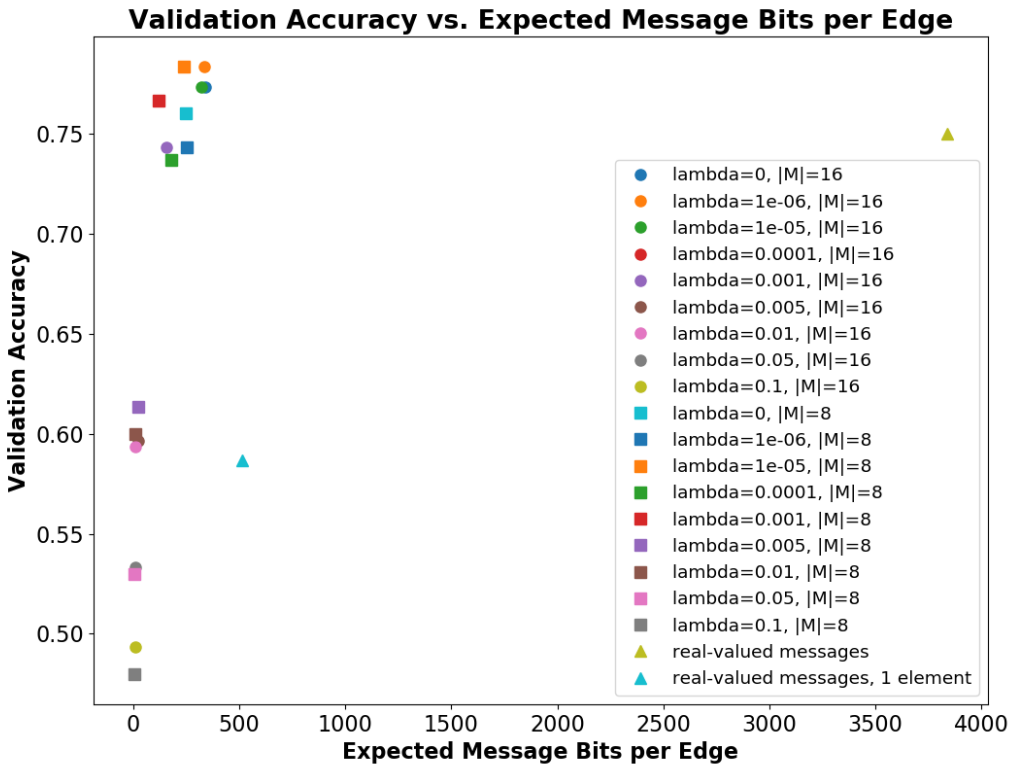


Figure 6: Validation accuracy vs. expected message bits exchanged during execution on the validation set for a GNN using our differentiable discrete communication approach with various message penalty weightings ($\lambda$) and message discretization fineness, as well as a GNN using 32-bit precision messages of equal length to those used in the discrete communication, and a GNN using 32-bit precision messages of length 1. In all cases where the GNN using 32-bit messages achieved comparable validation accuracy to the GNN using differentiable discrete communication, the model using differentiable discrete communication was able to use an order of magnitude or more fewer communication bits.

| | $\|M\| = 16$ | | $\|M\| = 8$ | |
|---|---|---|---|---|
| $\lambda$ | Validation Accuracy | Average Message Bits | Validation Accuracy | Average Message Bits |
| 0 | 0.7733 | 337.40 | 0.7600 | 247.48 |
| 1e-6 | 0.7833 | 334.27 | 0.7433 | 253.33 |
| 1e-5 | 0.7733 | 321.59 | 0.7833 | 240.22 |
| 1e-4 | 0.7433 | 250.28 | 0.7367 | 179.03 |
| 1e-3 | 0.7433 | 156.04 | 0.7667 | 120.62 |
| 5e-3 | 0.5967 | 24.01 | 0.6133 | 21.61 |
| 1e-2 | 0.5933 | 7.16 | 0.6000 | 7.06 |
| 5e-2 | 0.5333 | 7.25 | 0.5300 | 2.34 |
| 1e-1 | 0.4933 | 6.83 | 0.4800 | 2.16 |
| Real-valued | Message Length=8 | | Message Length=1 | |
| | 0.7500 | 3840 | 0.5867 | 512 |

Table 3: Validation accuracy and expected message bits exchanged during execution on the validation set for a GNN using our differentiable discrete communication approach with various message penalty weightings ($\lambda$) and message discretization fineness, as well as a GNN using 32-bit precision messages of equal length to those used in the discrete communication, and a GNN using 32-bit precision messages of length 1. In all cases where the GNN using 32-bit messages achieved comparable validation accuracy to the GNN using differentiable discrete communication, the model using differentiable discrete communication was able to use an order of magnitude or more fewer communication bits.

## References


Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of markov decision processes. *Mathematics of operations research* 27(4):819–840.

Busoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-agent reinforcement learning: An overview. *Innovations in Multi-Agent Systems and Applications-1* 310:183–221.

2013. References. In Buyya, R.; Vecchiola, C.; and Selvi, S. T., eds., *Mastering Cloud Computing*. Boston: Morgan Kaufmann. 29 – 68.

Farsad, N.; Rao, M.; and Goldsmith, A. 2018. Deep learning for joint source-channel coding of text. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2326–2330. IEEE.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. *CoRR* abs/1605.06676.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017. Counterfactual multi-agent policy gradients. *CoRR* abs/1705.08926.

Freed, B.; Sartoretti, G.; and Choset, H. 2020. Simultaneous policy and discrete communication learning for multi-agent cooperation. *IEEE Robotics and Automation Letters* 5(2):2498–2505.

Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; and Garcia-Rodriguez, J. 2017. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. J. 2017. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS Workshops*.

Jackson, D. E., and Ratnieks, F. L. 2006. Communication in ants. *Current biology* 16(15):R570–R574.

Jaderberg, M.; Czarnecki, W. M.; Dunning, I.; Marris, L.; Lever, G.; Castañeda, A. G.; Beattie, C.; Rabinowitz, N. C.; Morcos, A. S.; Ruderman, A.; et al. 2019. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science* 364(6443):859–865.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kim, H.; Jiang, Y.; Rana, R.; Kannan, S.; Oh, S.; and Viswanath, P. 2018. Communication algorithms via deep learning. *arXiv preprint arXiv:1805.09317*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kraemer, L., and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190:82–94.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR* abs/1706.02275.

Mordatch, I., and Abbeel, P. 2017. Emergence of grounded compositional language in multi-agent populations. *CoRR* abs/1703.04908.

OpenAI. 2018. Openai five.

Paulos, J.; Chen, S. W.; Shishika, D.; and Kumar, V. S. A. 2019. Decentralization of multiagent policies by learning what to communicate. *2019 International Conference on Robotics and Automation (ICRA)* 7990–7996.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature* 323(6088):533–536.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. K. S.; Koenig, S.; and Choset, H. 2018a. PRIMAL: pathfinding via reinforcement and imitation multi-agent learning. *CoRR* abs/1809.03531.

Sartoretti, G.; Wu, Y.; Paivine, W.; Kumar, T. K. S.; Koenig, S.; and Choset, H. 2018b. Distributed reinforcement learning for multi-robot decentralized collective construction. In *DARS 2018 - International Symposium on Distributed Autonomous Robotic Systems*, 35–49.

Sartoretti, G.; Paivine, W.; Shi, Y.; Wu, Y.; and Choset, H. 2019. Distributed learning of decentralized control policies for articulated mobile robots. *CoRR* abs/1901.08537.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3):93–93.

Sukhbaatar, S.; Szlam, A.; and Fergus, R. 2016. Learning multiagent communication with backpropagation. *CoRR* abs/1605.07736.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

# 6 Supplementary Material

**Claim:** Using the proposed randomized encoder/decoder detailed in sec. 3.1, the reconstruction error $e = \hat{z} - z$ is distributed according to $P_{\mathbf{e}|\mathbf{z}}(e|z) = U\left(-\frac{\delta}{2}, +\frac{\delta}{2}\right)$.

*Proof.* The reconstruction error distribution $P_{\mathbf{e}|\mathbf{z}}$ can be expressed as:

$$P_{\mathbf{e}|\mathbf{z}}(e|z) = \sum_{m \in M} \int_{-\infty}^{+\infty} P_{\mathbf{e}|\mathbf{z},\mathbf{m},\boldsymbol{\epsilon}}(e|z,m,\epsilon) P_{\mathbf{m}|\mathbf{z},\boldsymbol{\epsilon}\epsilon}(m|z,\epsilon) P_{\boldsymbol{\epsilon}}(\epsilon) d\epsilon. \tag{9}$$

Define $L(m)$ and $U(m)$ to represent the upper and lower bounds of the quantization interval represented by $m$, such that message $m$ is selected when $L(m) \leq z + \epsilon < U(m)$. Define $C(m)$ to be the center of the quantization interval corresponding to message $m$.

For a given $z$, there are only two possible values $m$ can take, which we denote $m_a$ and $m_b$. We define $m_a$ to be message that is selected when $\epsilon = -\frac{\delta}{2}$, and $m_b$ to be the message that is selected when $\epsilon = +\frac{\delta}{2}$. We therefore only have to consider those two values of $m$ in the summation in eq. 9:

$$P_{\mathbf{e}|\mathbf{z}}(e|z) = \int_{-\infty}^{+\infty} P_{\mathbf{e}|\mathbf{z},\mathbf{m},\epsilon}(e|z,m_a,\epsilon) P_{\mathbf{m}|\mathbf{z},\epsilon}(m_a|z,\epsilon) P_{\boldsymbol{\epsilon}}(\epsilon) d\epsilon \tag{10}$$

$$+ \int_{-\infty}^{+\infty} P_{\mathbf{e}|\mathbf{z},\mathbf{m},\epsilon}(e|z,m_b,\epsilon) P_{\mathbf{m}|\mathbf{z},\epsilon}(m_b|z,\epsilon) P_{\boldsymbol{\epsilon}}(\epsilon) d\epsilon \tag{11}$$

Given $z$ and $\epsilon$, the quantization procedure is deterministic, and can be represented as $P_{\mathbf{m}|\mathbf{z},\epsilon}(m|z,\epsilon) = \mathbb{1}_{L(m) \leq z+\epsilon < U(m)}$. Additionally, the reconstruction error is deterministic given $z$, $m$, and $\epsilon$, and is given by $e = \hat{z} - z = C(m) - \epsilon - z$. The distribution over reconstruction error given $z$, $m$, and $\epsilon$ can therefore be represented as $P_{\mathbf{e}|\mathbf{z},\mathbf{m},\epsilon}(e|z,m_a,\epsilon) = \boldsymbol{\delta}(C(m) - \epsilon - z)$. $P_{\boldsymbol{\epsilon}}$ is a uniform ranging from $-\frac{\delta}{2}$ to $+\frac{\delta}{2}$, and can therefore be represented as $P_{\boldsymbol{\epsilon}}(\epsilon) = \frac{1}{\delta}\mathbb{1}_{-\frac{\delta}{2} \leq \epsilon < +\frac{\delta}{2}}$. Substituting these expressions into 10, we get:

$$P_{\mathbf{e}|\mathbf{z}}(e|z) = \frac{1}{\delta}\Bigg( \int_{-\infty}^{+\infty} \boldsymbol{\delta}(C(m_a) - \epsilon - z)) \mathbb{1}_{L(m_a) \leq z+\epsilon < U(m_a)} \mathbb{1}_{-\frac{\delta}{2} \leq \epsilon < +\frac{\delta}{2}} d\epsilon \tag{12}$$

$$+ \int_{-\infty}^{+\infty} \boldsymbol{\delta}(C(m_b) - \epsilon - z) \mathbb{1}_{L(m_b) \leq z+\epsilon < U(m_b)} \mathbb{1}_{-\frac{\delta}{2} \leq \epsilon < +\frac{\delta}{2}} d\epsilon \Bigg). \tag{13}$$

Adjusting the bounds of the above integrals to omit regions where the integrand is zero:

$$P_{\mathbf{e}|\mathbf{z}}(e|z) = \frac{1}{\delta}\Bigg( \int_{-\frac{\delta}{2}}^{U(m_a)-z} \boldsymbol{\delta}(C(m_a) - \epsilon - z)) \mathbb{1}_{L(m_a) \leq z+\epsilon < U(m_a)} \mathbb{1}_{-\frac{\delta}{2} \leq \epsilon < +\frac{\delta}{2}} d\epsilon \tag{14}$$

$$+ \int_{L(m_b)-z}^{+\frac{\delta}{2}} \boldsymbol{\delta}(C(m_b) - \epsilon - z) \mathbb{1}_{L(m_b) \leq z+\epsilon < U(m_b)} \mathbb{1}_{-\frac{\delta}{2} \leq \epsilon < +\frac{\delta}{2}} d\epsilon \Bigg). \tag{15}$$

Evaluating the integral yields:

$$P_{\mathbf{e}|\mathbf{z}}(e|z) = \frac{1}{\delta}\left( \mathbb{1}_{C(m_a)+\frac{\delta}{2}-z > e \geq C(m_a)-U(m_a)} + \mathbb{1}_{C(m_b)-L(m_b) > e \geq C(m_b)-\frac{\delta}{2}-z} \right) \tag{16}$$

$$= \frac{1}{\delta}\left( \mathbb{1}_{U(m_a)-z > e \geq -\frac{\delta}{2}} + \mathbb{1}_{+\frac{\delta}{2} > e \geq U(m_a)-z} \right) \tag{17}$$

$$= \frac{1}{\delta}\mathbb{1}_{+\frac{\delta}{2} > e \geq -\frac{\delta}{2}} \tag{18}$$

$$= U\left(\frac{-\delta}{2}, +\frac{\delta}{2}\right) \tag{19}$$