Emergent Quantized Communication

Boaz Carmeli, Ron Meir, Yonatan Belinkov*

Technion – Israel Institute of Technology
boaz.carmeli@campus.technion.ac.il, rmeir@ee.technion.ac.il, belinkov@technion.ac.il

Abstract

The field of emergent communication aims to understand the characteristics of communication as it emerges from artificial agents solving tasks that require information exchange. Communication with discrete messages is considered a desired characteristic, for both scientific and applied reasons. However, training a multi-agent system with discrete communication is not straightforward, requiring either reinforcement learning algorithms or relaxing the discreteness requirement via a continuous approximation such as the Gumbel-softmax. Both these solutions result in poor performance compared to fully continuous communication. In this work, we propose an alternative approach to achieve discrete communication – quantization of communicated messages. Using message quantization allows us to train the model end-to-end, achieving superior performance in multiple setups. Moreover, quantization is a natural framework that runs the gamut from continuous to discrete communication. Thus, it sets the ground for a broader view of multi-agent communication in the deep learning era.

1 Introduction

A key aspect in emergent communication systems is the channel by which agents communicate when trying to accomplish a common task. Prior work has recognized the importance of communicating over a discrete channel (Havrylov and Titov 2017; Lazaridou and Baroni 2020; Vanneste et al. 2022). From a scientific point of view, investigating the characteristics of communication that emerges among artificial agents may contribute to our understanding of human language evolution. And from a practical point of view, discrete communication is required for natural human–machine interfaces. Thus, a large body of work has been concerned with enabling discrete communication in artificial multi-agent systems (Foerster et al. 2016; Havrylov and Titov 2017, *inter alia)*. However, the discretization requirement poses a significant challenge to *neural* multi-agent systems, which are typically trained with gradient-based optimization. Two main approaches have been proposed in the literature for overcoming this challenge, namely using reinforcement learning (RL) algorithms (Williams
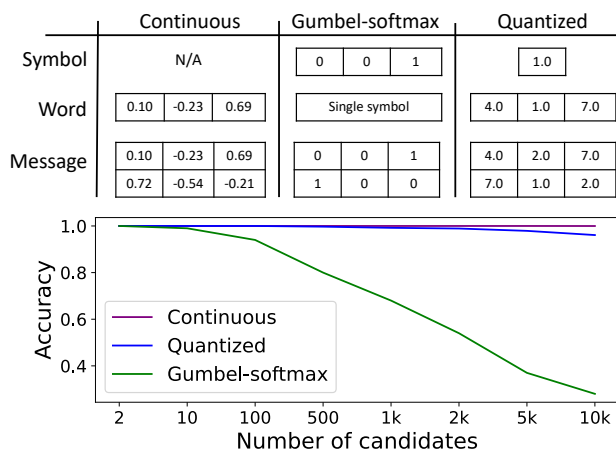
Figure 1: **Top**: Symbol, word, and message elements for continuous, Gumbel-softmax, and quantized communication modes. **Bottom**: Accuracy (Y-axis) achieved by the three communication modes vs. number of candidates (X-axis), in the Object game. Continuous communication leads to good performance on the end task but does not use symbols. Gumbel-softmax sends one word per symbol, but requires a recurrent channel and does not work well in practice. Quantized communication enables discrete and successful communication. Detailed channel parameters are provided in section 5.

1992; Lazaridou, Peysakhovich, and Baroni 2016) or relaxing the discrete communication with continuous approximations such as the Gumbel-softmax (Jang, Gu, and Poole 2016; Havrylov and Titov 2017). The RL approach maintains discreteness, but systems optimized with the Gumbel-softmax typically perform better in this setting. However, Gumbel-softmax training is effectively done with continuous communication. Both discretization approaches perform far worse than a system with fully continuous communication. In short, The more discrete the channel, the worse the system's performance.

In this work, we propose a new framework for discrete communication in multi-agent systems, based on quantization (Figure 1, top). Drawing inspiration from work on ef-

ficient neural network quantization during training and inference (Banner et al. 2018; Wang et al. 2018; Choi et al. 2018)), we quantize the message delivered between the agents. We investigate two learning setups: First, training is done with continuous communication, while inference is discretized by quantization, similar to the common scenario when using continuous approximations like Gumbel-softmax. Second, we investigate the effects of quantizing the messages during both training and inference.

We experimentally validate our approach in multiple scenarios. We consider three different games that fall into the well-known design of referential games, where a sender transmits information about a target object, which a receiver needs to identify (Lewis 2008; Lazaridou, Peysakhovich, and Baroni 2016; Choi, Lazaridou, and De Freitas 2018; Guo et al. 2019). Our objects include synthetic discrete objects, images, and texts. We also experiment with a variant, which we call the classification game, where the receiver needs to identify the class to which the object belongs. In all cases, we find our quantized communication to outperform the standard approach using Gumbel-softmax by a large margin, often even approaching the performance with fully continuous communication (Figure 1, bottom).

Finally, we investigate the quantized communication by varying the granularity of quantization. This allows us to cover a much wider range of discreteness levels than has previously been possible. We analyze which aspects of the communication channel are most important for accomplishing the agents' task and how they affect the resulting language. We find that quantization, even an extreme one, works surprisingly well given a long enough message. Evidently, quantized communication with a binary alphabet performs almost as well as continuous communication.

In summary, this work develops a new framework for discrete communication in multi-agent systems, setting the ground for a broader investigation of emergent artificial communication and facilitating future work on interfacing with these systems.

2   Background

We begin with a formal definition of the multi-agent communication setup, often called "emergent multi-agent communication" (Lazaridou and Baroni 2020). In this setup, a sender and a receiver communicate in order to accomplish a given task. In the **referential game**, the sender needs to transmit information about a target object, which the receiver uses to identify the object from a set of candidates. In the **classification game**, the sender again transmits information about an object, but the receiver needs to identify the *class* the object belongs to, rather than its identity. Notably, the two games require significantly different communication. While in the referential game the sender needs to accurately describe the target, in the classification game the sender needs to describe the target's class (see Appendix A.2 for details).

Formally, we assume a world $\mathbb{O}$ with $|\mathbb{O}|$ objects.[1] At each

---

[1]We defer details on the type of objects to Section 4. For now, one can think of objects as images, texts, etc.
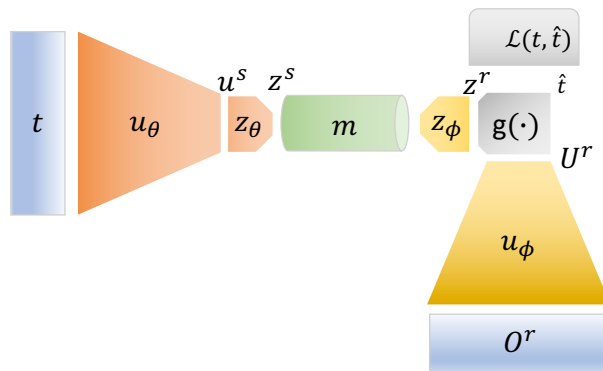

Figure 2: The emergent communication setup. Sender network is at the left, Receiver network is at the bottom right, and m is the communication channel.

turn, n candidate objects $\mathbb{C} = \{c\}^n \subseteq \mathbb{O}$, are drawn uniformly at random from $\mathbb{O}$. One of them is randomly chosen to be the target t, while the rest, $\mathbb{D} = \mathbb{C} \setminus t$, serve as distractors.

Figure 2 illustrates the basic setup. At each turn, the sender S encodes the target object t via its encoder network u_θ, such that $u^s = u_\theta(t) \in \mathbb{R}^d$ is the encoded representation of t. It then uses its channel network z_θ to generate a message m, $m = z_\theta(u^s) = z_\theta(u_\theta(t))$. The channel and message have certain characteristics that influence both the emergent communication and the agents' performance in the game, and are described in Section 2.1.

At each turn, the receiver R encodes each candidate object c_i ($i = 1, 2, \ldots, n$) via its encoder network u_ϕ, to obtain $u_\phi(c_i)$. We write $U^r \in \mathbb{R}^{\mathbb{C} \times d}$ to refer to the set of encoded candidate representations, each of dimension d. The receiver then decodes the message via its decoder channel network z_ϕ, obtaining $z^r = z_\phi(m) \in \mathbb{R}^d$. Next, the receiver computes a score matching each of the encoded candidates to the decoded message. The receiver then calculates prediction scores $\tilde{t} = \text{softmax}(z^r \cdot U^r)$. At test time, the receiver's predicted target object is the one with the highest score, namely $\hat{t} = \text{argmax}_i \tilde{t}$. During training, the entire system is optimized end-to-end with the cross-entropy loss between the correct target t and the predicted target $\hat{t}$. The trainable parameters are all the parameters of both sender and receiver networks, θ and ϕ.

2.1   Communication Elements

A key aspect of the emergent communication setup is the message (m in Figure 2). In this work we compare three communication modes that generate this message: continuous (CN) uses a continuous message, while Gumbel-softmax (GS) and quantized (QT) use a discrete message. We start by describing the communication elements common to all modes, and then provide more details on the unique aspects of each communication mode.

Formally, we define three communication elements, namely, symbol, word and message. Figure 1 provides an example for each element.

- **Symbol** is the atomic element of the communication. An alphabet is a collection of symbols. The alphabet size is a parameter of the quantized and Gumbel-softmax communication modes, while continuous communication uses real-numbers in m, corresponding to an uncountable alphabet.
- **Word** is the basic message element. A word is represented with a one-dimensional vector. In continuous communication, this vector is composed of floating point numbers, for quantized communication it is composed of integers, and for Gumbel-softmax communication it is a one-hot vector.
- **Message** is a sequence of one or more words, which the sender sends to the receiver. An instantaneous (Instant) channel is capable of sending (and receiving) only single-word messages, while a Recurrent channel sends (receives) multi-word messages with a recurrent neural network (RNN).

2.2 Communication Modes

In this section we describe two known communication modes: continuous and Gumbel-softmax. These communication modes serve as baselines. In the following section we describe our quantized communication.

Continuous Communication In continuous communication, words are represented with floating point vectors (see Figure 1). Though continuous, one may think of each vector element as if it represents a symbol, and the vector itself represents a word. Continuous communication is expected to lead to good performance, provided that the channel has sufficient capacity. With continuous communication, the system can easily be trained end-to-end with back-propagation.

Gumbel-softmax The Gumbel-softmax is a continuous approximation for a categorical distribution. In the communication context, a discrete message is approximated via a sampling procedure. Details are given elsewhere (Havrylov and Titov 2017; Jang, Gu, and Poole 2016) and implementation specifics are provided in Appendix A.4. The end result is a continuous message, where each word has the size of the alphabet and holds *one* (approximate) symbol. This allows for end-to-end optimization with gradient methods, and for discrete communication at inference time. However, the channel capacity is limited, and a large alphabet size is both inefficient (due to the need to sample from a large number of categories) and does not perform well in practice.

3 Quantized Communication

Quantization techniques aim to reduce model size and computation cost while maintaining a similar level of performance to the original model (Banner et al. 2018). The key quantization idea is to replace floating-point representations of model weights and/or activations with integers. We emphasize that, while quantization has a specific purpose in mind (efficiency), it renders the neural network discrete by definition. Allowing gradients to flow though the network during back-propagation enables end-to-end gradient-based optimization of the network with off-the-shelf optimizers.

Algorithm 1: Quantizing continuous communication.

```
1: msg: continuous message          ▷ Floating point vector
2: S: scaling factor                ▷ Set the alphabet range
3: procedure NORMALIZE(msg)
4:     min_elem ← min(msg.elements)
5:     max_elem ← max(msg.elements)
6:     msg ← (msg.elems − min_elem)/max_elem
7:     return msg
8: end procedure
9: procedure QUANTIZE(msg)
10:     msg ← Normalize(msg)
11:     s_msg ← msg/S                ▷ Scale to range
12:     qt_msg ← quantize(s_msg)     ▷ Integer vector
13:     deqt_msg ← dequant(qt_msg)   ▷ Rounded float
14:     discrete_msg ← deqt_msg          ▷ For logging
15:     deqt_msg ← deqt_msg ∗ S          ▷ Scale back
16:     return (deqt_msg, discrete_msg)
17: end procedure
18: msg ← Quantize(msg)
```

3.1 Quantized Communication Method

We follow the quantization definition and notation provided by Gholami et al. (2021). The quantization operator is defined by $Q(r) = \text{Int}(r/S) - Z$ where r is a real-valued floating-point tensor, S is a real-valued scaling scalar, and Z is an integer zero point, which we set to zero. The Int() function maps a real value to an integer value through a rounding operation (e.g., round to nearest integer and truncation). This operator, also known as *uniform quantization* (Gholami et al. 2021), results in quantized values that are uniformly spaced.[2]

One can recover floating-point values r from the quantized values $Q(r)$ through dequantization, $\hat{r} = S(Q(r)+Z)$. Obviously, the recovered real values $\hat{r}$ will not exactly match r due to the rounding operation. This rounding mismatch is a core difference between continuous and quantized communication.

The quantized operator's scaling factor S essentially divides a given range of real values r into a number of partitions. Specifically, we define the scaling factor S to be $S = \frac{[\beta - \alpha]}{|v|}$, and we set v to be the alphabet size. In this work we normalize message values to the range $[0, 1]$, thus $\beta = 1, \alpha = 0$. Emprically, message normalization improves results for both quantized and continuous communication. Notably, the rounding error of $\hat{r}$ is linearly correlated with the alphabet size.

This procedure results in a quantization algorithm, presented in Algorithm 1, which maps each message to a set of symbols from the alphabet. The quantization algorithm allows **fine-grained control over channel capacity**. Capacity can be controlled by both the alphabet size and the word length. The total number of unique words allowed by the channel is given by channel-capacity = alphabet-size$^{\text{word-length}}$.

---

[2]Future work may explore communication with non-uniform quantization schemes (Gholami et al. 2021).

3.2 Training with Quantization

Notably, one may choose to apply the quantization algorithm during both training and inference, or only during inference. Quantization only during inference is similar to the basic Gumbel-softmax setup described above, where training is done with a continuous approximation and inference is discrete. Quantization during training makes the system non-differentiable due to the rounding operation. In this case, we use the straight-through estimator (STE; Bengio, Léonard, and Courville 2013), which approximates the non-differentiable rounding operation with an identity function during the backward pass. This is similar to what is known in the Gumbel-softmax literature (Jang, Gu, and Poole 2016) as the straight-through option, where the softmax is replaced with argmax during the forward path.

4 Experimental Setup

4.1 Games and Datasets

We run our experiments on four games from three datasets.

Synthetic objects. This dataset is based on Egg's object game (Kharitonov et al. 2019). Each object has 4 attributes. Each attribute has 10 possible values, and different attributes share the same set of values. Thus, the dataset contains 10^4 objects which are uniquely identified by four discrete values.

Images. We use the Egg implementation of the image game from Lazaridou, Peysakhovich, and Baroni (2016).[3] The Dataset contains images from ImageNet (Deng et al. 2009). The training set contains $46,184$ images, distributed evenly over 463 classes, out of which we randomly choose 8032. The validation and test sets have $67,521$ images each, split over the same classes. We randomly choose distractors from classes other than the targets'.

Texts. We use a short text dataset, named Banking77 (Casanueva et al. 2020), which we refer to as Sentences. It contains $10,000$ sentences, classified into 77 classes, each represented with a meaningful class name. The sentences are user queries to an online customer support banking system, while the classes are user intents. We use the Sentences dataset for two different games: **Sent-Ref** is a referential game, that is, the receiver needs to identify the sentence. **Sent-Cls** is a classification game, where the receiver receives a set of candidate classes and needs to identify the class of the target sent by the sender.

Data Splits. In all experiments we split the data 80/10/10 into training, validation, and test sets, respectively. For Image and Sentence-Ref games, both targets (sender-side objects) and candidates (receiver's side) are mutually exclusive across splits. For Object and Sentence-Cls game, targets are mutually exclusive while candidates are shared across splits. Table 1 provides summary statistics of the datatsets.

---

[3]https://dl.fbaipublicfiles.com/signaling_game_data

Dataset	#Objects	#Train	#Valid	#Test	Max
Object	10K	8000	1000	1000	10K
Image	181K	8032	1024	1024	100
Sent-Ref	10K	7997	1001	1001	77
Sent-Cls	10K/77	7953	1004	1042	77

Table 1: Sizes of datasets and splits for each game. 77 is the number of classes in the Banking dataset. Max is the maximum number of candidates used for evaluating the game.

4.2 Agents' Architecture

Encoding Agents. We refer by sender and receiver encoding agents to the u_θ and u_ϕ networks, respectively, as described in Section 2. For Object and Image games, we follow the architecture provided by the Egg implementation (Kharitonov et al. 2019). The agents in the Object game uses a single fully-connected (FC) layer to encode the objects. The agents in the Image game use a FC network followed by two convolutional layers and a second FC layer. In the Image game, the sender uses all candidates for encoding the target (referred to as 'informed-sender' by Lazaridou, Peysakhovich, and Baroni (2016)). In all other games the sender encodes only the target object. For the Sentence games (both referential and classification), we use a distilbert-base-uncased model (Sanh et al. 2019) from Huggingface (Wolf et al. 2020) as the sentence encoder without any modification.[4] Appendix A.4 provides more details.

Communication Channels. We refer by sender and receiver channels to the z_θ and z_ϕ networks, respectively, as described in Section 2. We experiment with two architectures for the communication channel: Instant and Recurrent. **Instant** simply passes the sender's encoded representation of the target through a FC network to scale it to the word length and send it to the receiver. The receiver's Instant channel decodes the message with a FC feed-forward network and compares it with the candidates' encoded representations as described in Section 2. The **Recurrent** channel enables sending and receiving multi-word messages. We adapt the Recurrent channel implemented in Egg to work with continuous and quantized communication. More details on channel configuration are provided in Appendix A.4

4.3 Number of Candidates and Distractors

Most earlier emergent communication setups use a limited number of distractors (Mu and Goodman 2021; Li and Bowling 2019). Recent work (Chaabouni et al. 2021; Guo et al. 2021) reports the effect that an increased number of distractors has on accuracy results. The number of distractors affects results in two complementary ways. On the one hand, adding more distractors renders the receiver's task

---

[4]Importantly, in this work we aim to evaluate communication performance across various settings, and not necessarily find the best-performing encoding network. Nevertheless, we find our setup to achieve close to state-of-the-art results in the Sentence-Cls game, as shown in Section 5.

harder during inference. On the other hand, during training, distractors serve as negative samples which are known to improve learning Mitrovic, McWilliams, and Rey (2020). Based on these observations, our experimental environment lets us decouple the number of negative examples during training, from the number of distractors used for evaluation. In all our experiments we train the system with a large number of negative samples (serves as distractors) and report results on an increasing number of candidates, always including the target as one of them.

4.4 Evaluation Metrics

In this work we report prediction accuracy as the main metric. Similar to Guo et al. (2021), we observe a correlation between the number of unique messages (NoUM) and accuracy, and report this measurement as well. Recent work (Chaabouni et al. 2021; Yao et al. 2022) reports that the popular topographic similarity metric (Brighton and Kirby 2006; Lazaridou et al. 2018) does not correlate well with accuracy, especially when measured on a large number of distractors. In our work we observed the same effect, so we refrain from reporting this metric.

4.5 Training Details

We performed hyper-parameter tuning on the validation set and report results on the test set. As systems have many hyper-parameters to tune we tried to reduce changing most parameters between different setups to a minimum. However, we ran an extensive hyper-parameter search for alphabet size and message length for the Gumbel-softmax communication to insure that we report the best possible results for this communication mode. Quantized communication required only minimal tuning and still outperformed Gumbel-softmax across all setups. We report more details on configurations and hyper-parameters in Appendix A.4.

Each experiment took under 24 hours on a single v100 GPU and a CPU with 128GB RAM. We run each experiment three times with different random seeds and report average performance. Variance is generally small (Appendix A.1).

5 Results

We first experiment with quantization only during inference and compare its performance to continuous and Gumbel-softmax communication. Our main results are presented in Figure 3.[5] The graphs show test accuracy results (Y-axis) against the number of candidates (X-axis) for the three communication modes over two channel architectures for the four games. As expected, performance generally degrades when increasing the number of candidates in most setups. This is especially evident in Gumbel-softamx (GS)
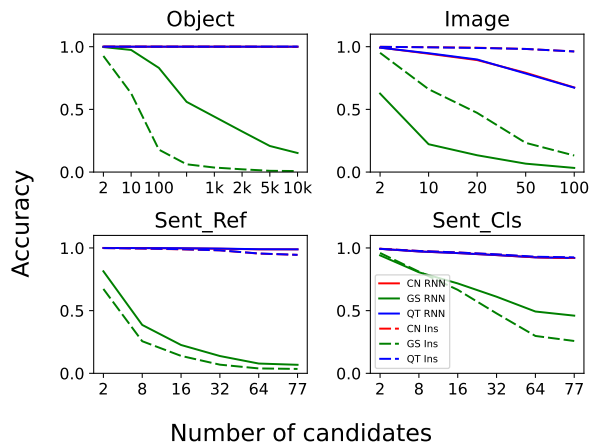
Figure 3: Communication results for four games for Instant (Ins) and Recurrent (RNN) channels, using quantization only during inference. The CN and QT results are essentially the same, thus overlapping in the plots. Channel parameters for the experiments are provided in footnote 5.

communication (green lines), while continuous (CN, red) and quantized (QT, blue) communication modes scale much more gracefully with the number of candidates. Notably, the **quantized communication is on-par with the fully continuous communication in all cases**.

Considering the different games, in the Object game, continuous and quantized communication perform perfectly, while Gumbel-softamx suffers from a large number of candidates. In the other games, there is a slight performance degradation with continuous and quantized communication.

Next we compare the performance with communication using Instant vs. Recurrent channels. Recall that the Instant channel has a much more limited capacity (each message is one-word long) compared to the Recurrent channel (each message is made of multiple words). The Gumbel-softmax communication suffers most clearly from the Instant channel: In all but the Image game, it performs worse than Gumbel-softmax with the Recurrent channel. The gap is especially striking in the Object game (compare green dashed and solid lines). The poor performance of Gumbel-softmax with the Instant channel can be explained by the limited channel capacity: The sender in this case can generate up to 100 unique messages (each containing just a single symbol), which are just 1.0% of the unique objects in the Object game. Thus it has little hope of correctly identifying the target object. One might hope that a Recurrent channel would help Gumbel-Softamx perform better, as it has the required capacity to uniquely represent all objects (10^6 unique messages). However, even with this capacity, performance on a high number of candidates is low. We attribute the poor performance to the difficulty to optimize RNNs with discrete communication. This might also explain why Gumbel-softmax with an instant channel works better than that with the Recurrent channel in the Image game.

In contrast to Gumbel-softmax, quantized communication

	Instant				Recurrent			
	Train+Inf		Only Inf		Train+Inf		Only Inf	
Game	2	Max	2	Max	2	Max	2	Max
Object	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Image	0.51	0.01	0.99	0.96	0.51	0.01	0.99	0.67
Sent-Ref	0.95	0.29*	0.99	0.95	0.99	0.74*	1.0	0.99
Sent-Cls	0.97	0.58*	0.99	0.92	0.98	0.63	0.99	0.93

Table 2: Comparing quantization during both training and inference (Train+Inf) to quantization only during inference (Only Inf), for Instant and Recurrent channels, with 2 candidates and the maximum number of candidates in each game (Max). Reported results are for an average of three runs. Standard deviations for all reported results are under 0.02, apart for those marked by ∗, which are under 0.07.

does not suffer from the limited capacity problem, nor from the optimization difficulty. In both Instant and Recurrent channels, quantized communication leads to excellent performance, even in the face of a large number of candidates.

We note in passing, that for the sentence classification game (Sent_Cls), we get results that are on par with state-of-the-art classification results for this dataset (Qi et al. 2020; Zhang et al. 2021), even though we used a very different setup, that of communication between agents.

5.1 Quantization During Training

So far we reported results when quantization was applied only at inference time. Here we compare with applying quantization also during training. Table 2 reports accuracy results for two settings, using either 2 or a maximum number of candidates (varying by game).

In the Object game, we report perfect performance in all communication settings, whether using quantization in training and inference or only at inference. In the other games, quantization only during inference leads to better performance than quantization during both training and inference. This may be explained by the added constraint of learning to perform a task with genuine discrete communication. However, it is striking that even with quantization during training, results are quite good in many cases, especially with the Recurrent channel. We conclude that quantization is a beneficial approach for learning to communicate with a discrete channel. Note that since we used a very naive quantization scheme during training, it is likely that more elaborate discretization schemes, e.g., non-uniform (Choi et al. 2018), may greatly improve performance.

5.2 Communication Analysis

Figure 4 analyzes the quantized communication results for the Object game over an Instant channel. Results are obtained from a test set with 1000 unique targets. Appendix A.3 provides a similar analysis for the other games, showing largely consistent results.

The top heatmaps show performance in various settings, organized according to word length (X-axis) and alphabet size (Y-axis). The bottom heatmaps show the number of unique messages (NoUM) sent by the sender in each
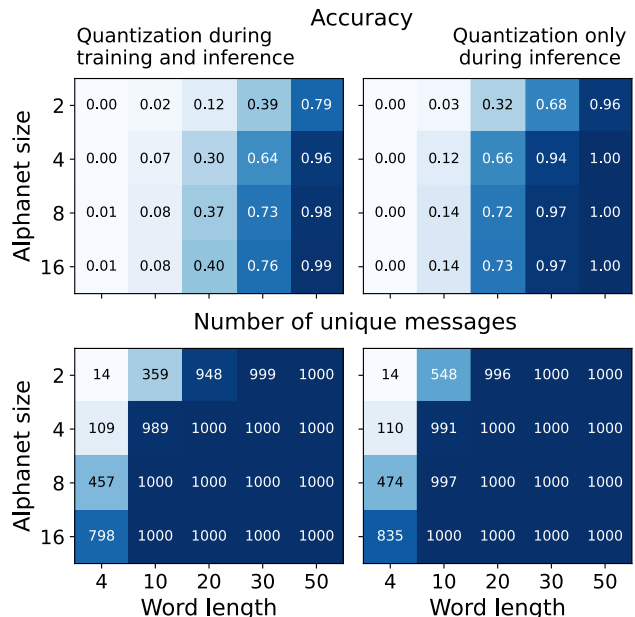
Figure 4: Showing accuracy (top) and number of unique messages (NoUM) (bottom) as a function of alphabet size and word length for the Object game with an Instant channel. Comparing quantization during both training and inference (left) or only during inference (right).

configuration (1000 max). We compare quantization during both training and inference (left heatmaps) with quantization only during inference (right heatmaps). As before, quantization during both training and inference performs worse than quantization only during inference.

Clearly, increasing the channel capacity (moving to the bottom-right corner in the heatmaps) improves accuracy results and increases the NoUM sent by the sender up to a maximum of 1000. Increasing word length (moving to the right) improves results substantially for all alphabet sizes, and reaches a maximal performance for a length of 50, given alphabet sizes larger than 4 for quantization during inference only (top right). For quantization during both training and inference, the same results are achieved with an alphabet size larger than 16 (top left). Interestingly, increasing the alphabet size (moving to the bottom in the heatmaps) has a smaller effect. With long enough words, the system performs almost optimally, even with a very small alphabet (e.g., 2 and 4 symbols), resembling findings by Freed et al. (2020). As seen by comparing the top and bottom heatmaps, the NoUM correlates with performance. Interestingly, having a NoUM equal to the number of unique targets is a necessary but not sufficient condition for perfect performance.

Finally, we compare the number of unique messages in Gumbel-softmax and quantized communication modes across the four games (Table 3). The number of unique messages generated by quantized communication equals (in 3 out of 4 games) or almost equals (in the Image game) the number of unique targets. In contrast, Gumbel-softmax does not generate enough unique messages. The Gumbel-softmax

| | | Instant | | Recurrent | |
Game	#Targets	GS	QT	GS	QT
Object	1000	58	1000	948	1000
Image	1024	16	1016	391	1016
Sent_Ref	1001	8	1001	116	1001
Sent_Cls	1042	20	1042	125	1042

Table 3: Number of unique messages for GS and QT communication modes, with Instant and Recurrent channels. Channel parameters are provided in footnote 5. QT results are for quantization only during inference.

with a Recurrent channel produces many more messages than the Instant channel. However, only for the Object game does it generate nearly enough unique messages.

6 Related work

The field of emergent communication gained renewed interest in recent years with the advances of deep neural networks and natural language processing (Lazaridou and Baroni 2020). Despite significant advances, approaches for generating discrete messages from neural networks remain scarce. Multi-agent reinforcement learning (RL) and Gumbel-Softmax (GS) are the two alternative approaches used by the community.

6.1 Multi-Agent Reinforcement Learning

The work by Foerster et al. (2016) is probably the first to suggest methods for learning multi-agent communication. Many studies in the emergent communication field (Lazaridou, Potapenko, and Tieleman 2020) use RL and variants of the REINFORCE algorithm (Williams 1992) for solving the referential game (Foerster et al. 2016; Lazaridou, Peysakhovich, and Baroni 2016; Chaabouni et al. 2021). Vanneste et al. (2022) provide a comprehensive review of the various ways to overcome the discretization issue within multi-agent environments. Notably, they found that none of the surveyed methods is best in all environments, and that the optimal discretization method greatly depends on the environment. Somewhat close to our approach, Freed et al. (2020) propose an elaborate stochastic quantization procedure, which relies on adding stochastic noise as part of an encoding/decoding procedure, and evaluate it in path-finding and search problems. In contrast, our approach is simple and deterministic, and works exceptionally well in the referential and classification games.

6.2 Gumbel-softmax Communication

Gumbel-softmax (Jang, Gu, and Poole 2016) enables discrete communication by sampling from a categorical Gumbel distribution. It allows gradients to flow through this non-differentiable distribution by replacing it with a differentiable sample from a Gumbel-softmax distribution. Havrylov and Titov (2017) compare communication with RL and Gumbel-softmax and observe that the latter converges much faster and results in more effective protocols.

Since then, many studies have used Gumbel-softmax in the emergent communication setup (Resnick et al. 2019; Guo et al. 2021; Mu and Goodman 2021; Dessì, Kharitonov, and Marco 2021). Though widely used as the default method for overcoming the discretization difficulty, Gumbel-Softmax still suffers from at least two severe limitations. First, it uses a one-hot vector to encode symbols, limiting capacity. Second, it requires sampling from a distribution, making optimization more expensive and less accurate.

6.3 Quantization

In the context of neural networks, quantization is a method for reducing model size and computation cost while maintaining performance. More generally, quantization, as a method to map input values in a large (often continuous) set to output values in a small (often finite) set, has a long history (Gray and Neuhoff 1998). The fundamental role of quantization in modulation and analog-to-digital conversion was first recognized during the early development of pulse-code modulation systems, especially in the work of Oliver, Pierce, and Shannon (1948), and with the seminal coding theory work by Shannon (1948). Recently, intensive research on quantization shows great and consistent success in both training and inference of neural networks using 8-bit number representations, and even less (Banner et al. 2018; Wang et al. 2018; Choi et al. 2018). In particular, breakthroughs of half-precision and mixed-precision training (Courbariaux, Bengio, and David 2014; Gupta et al. 2015) significantly contributed to vast performance improvements. Notably, moving from floating-point to integer computation renders many operations non-differentiable. To overcome this subtlety, a straight-through estimator (STE) (Bengio, Léonard, and Courville 2013) is often used.

7 Conclusions

Research on emergent communication between artificial agents strives for discrete communication. However, common methods such as continuous relaxations via Gumbel-softmax lag far behind continuous communication in terms of performance on the agents' task. In this work we propose an alternative approach that achieves discrete communication via message quantization, while enabling simple end-to-end training. We show that our quantized communication allows us to run the gamut from continuous to discrete communication by controlling the quantization level, namely, the size of the used alphabet and the word length. When applying quantization during inference, we observe extremely good results, even for a very small alphabet size. We also investigated the option to apply quantization during training, demonstrating superior performance compared to the common Gumbel-softmax approach, albeit inferior to inference-only quantization.

Future work may explore more elaborate quantization schemes for message discretization, during either training or inference. We believe that the quantization approach offers a good test bed for investigating emergent communication in multi-agent systems.

Acknowledgements

References

Banner, R.; Hubara, I.; Hoffer, E.; and Soudry, D. 2018. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Brighton, H.; and Kirby, S. 2006. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial life*, 12(2): 229–242.

Casanueva, I.; Temčinas, T.; Gerz, D.; Henderson, M.; and Vulić, I. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Chaabouni, R.; Strub, F.; Altché, F.; Tarassov, E.; Tallec, C.; Davoodi, E.; Mathewson, K. W.; Tieleman, O.; Lazaridou, A.; and Piot, B. 2021. Emergent communication at scale. In *International Conference on Learning Representations*.

Choi, E.; Lazaridou, A.; and De Freitas, N. 2018. Compositional obverter communication learning from raw visual input. *arXiv preprint arXiv:1804.02341*.

Choi, J.; Chuang, P. I.-J.; Wang, Z.; Venkataramani, S.; Srinivasan, V.; and Gopalakrishnan, K. 2018. Bridging the accuracy gap for 2-bit quantized neural networks (qnn). *arXiv preprint arXiv:1807.06964*.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2014. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dessì, R.; Kharitonov, E.; and Marco, B. 2021. Interpretable agent communication from scratch (with a generic visual processor emerging on the side). *Advances in Neural Information Processing Systems*, 34: 26937–26949.

Foerster, J.; Assael, I. A.; De Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29.

Freed, B.; Sartoretti, G.; Hu, J.; and Choset, H. 2020. Communication learning via backpropagation in discrete channels with unknown noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 7160–7168.

Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M. W.; and Keutzer, K. 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

Gray, R. M.; and Neuhoff, D. L. 1998. Quantization. *IEEE transactions on information theory*, 44(6): 2325–2383.

Guo, S.; Ren, Y.; Havrylov, S.; Frank, S.; Titov, I.; and Smith, K. 2019. The emergence of compositional languages for numeric concepts through iterated learning in neural agents. *arXiv preprint arXiv:1910.05291*.

Guo, S.; Ren, Y.; Mathewson, K.; Kirby, S.; Albrecht, S. V.; and Smith, K. 2021. Expressivity of Emergent Language is a Trade-off between Contextual Complexity and Unpredictability. *arXiv preprint arXiv:2106.03982*.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *International conference on machine learning*, 1737–1746. PMLR.

Havrylov, S.; and Titov, I. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *Advances in neural information processing systems*, 30.

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

Kharitonov, E.; Chaabouni, R.; Bouchacourt, D.; and Baroni, M. 2019. EGG: a toolkit for research on Emergence of lanGuage in Games. *arXiv preprint arXiv:1907.00852*.

Lazaridou, A.; and Baroni, M. 2020. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*.

Lazaridou, A.; Hermann, K. M.; Tuyls, K.; and Clark, S. 2018. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*.

Lazaridou, A.; Peysakhovich, A.; and Baroni, M. 2016. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

Lazaridou, A.; Potapenko, A.; and Tieleman, O. 2020. Multi-agent communication meets natural language: Synergies between functional and structural language learning. *arXiv preprint arXiv:2005.07064*.

Lewis, D. 2008. *Convention: A philosophical study*. John Wiley & Sons.

Li, F.; and Bowling, M. 2019. Ease-of-teaching and language structure from emergent communication. *Advances in neural information processing systems*, 32.

Mitrovic, J.; McWilliams, B.; and Rey, M. 2020. Less can be more in contrastive learning.

Mu, J.; and Goodman, N. 2021. Emergent Communication of Generalizations. *Advances in Neural Information Processing Systems*, 34: 17994–18007.

Oliver, B.; Pierce, J.; and Shannon, C. E. 1948. The philosophy of PCM. *Proceedings of the IRE*, 36(11): 1324–1331.

Qi, H.; Pan, L.; Sood, A.; Shah, A.; Kunc, L.; Yu, M.; and Potdar, S. 2020. Benchmarking commercial intent detection services with practice-driven evaluations. *arXiv preprint arXiv:2012.03929*.

Resnick, C.; Gupta, A.; Foerster, J.; Dai, A. M.; and Cho, K. 2019. Capacity, bandwidth, and compositionality in emergent language learning. *arXiv preprint arXiv:1910.11424*.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Vanneste, A.; Vanneste, S.; Mets, K.; De Schepper, T.; Mercelis, S.; Latré, S.; and Hellinckx, P. 2022. An Analysis of Discretization Methods for Communication Learning with Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2204.05669*.

Wang, N.; Choi, J.; Brand, D.; Chen, C.-Y.; and Gopalakrishnan, K. 2018. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Yao, S.; Yu, M.; Zhang, Y.; Narasimhan, K. R.; Tenenbaum, J. B.; and Gan, C. 2022. Linking Emergent and Natural Languages via Corpus Transfer. *arXiv preprint arXiv:2203.13344*.

Zhang, J.; Bui, T.; Yoon, S.; Chen, X.; Liu, Z.; Xia, C.; Tran, Q. H.; Chang, W.; and Yu, P. 2021. Few-shot intent detection via contrastive pre-training and fine-tuning. *arXiv preprint arXiv:2109.06349*.

A Appendix

A.1 Detailed results

Tables 5, 6, 7, and 8 provide results for the Object, Image, Sentence-referential, and Sentence-classification games, respectively, which complement the main results from Section 5. Each table contains results for six configurations: three communication modes (continuous, Gumbel-softmax, and quantized) and two channel types (Instant and Recurrent). The tables also specify the channel settings (alphabet size, word length, and message length) in each configuration.

The tables provide average results and standard deviations over three runs with different random seeds. Variance is generally small, with the exception of Gumbel-softmax with a Recurrent channel, which exhibits larger variance, attesting to the optimization difficulties in this setup.

A.2 Referential and Classification Games

While in the referential game the receiver and sender share objects form the same world, in the classification game they do not. In the classification game, each object $o = (o_s, o_r)$ is composed of two elements: a sample $o_s \in \mathbb{O}_s$ and a label $o_r \in \mathbb{O}_r$. The sender's targets are drawn from the set of possible samples, $\mathbb{O}_s$, while the receiver's candidates are drawn from the set of possible labels, $\mathbb{O}_r$. For example, in our experiments with the Sent-Cls dataset, the samples are user queries and the labels are their intents. Importantly, while the sender's objects are unique, the receiver's objects are not; for example, two user queries may have the same intent. The game's goal is to match the sender's target o_s with the receiver's target o_r.

Two complementary differences exist between Referential and Classification games. On the one hand, it is easy to notice that with continuous communication a trivial solution exists for referential games, given that: (1) The same encoder is used by the sender and the receiver to encode the target and the candidates, respectively. (2) The channel has enough capacity to transmit the encoded target within a single message. Assuming these two conditions hold, the sender can simply send the encoded target to the receiver. The receiver then just needs to compare the target with its candidates to find the correct one. Thus, one can expect perfect performance in these cases.

The classification game, on the other hand, requires mapping $|\mathbb{O}_s|$ objects to $|\mathbb{O}_r|$ classes. Thus, the task can be solved by sending just $|\mathbb{O}_r|$ unique messages, one for each class. Still, messages are very different from the objects they describe, thus continuous communication does not have a foreseen advantage over other communication types in the Classification game, beyond those related to optimization differences.

A.3 Language Analysis for the Three Games

Figures 5, 6, and 7 provide heatmaps for the Image, Sent_Ref and Sent_Cls games, complementing the analysis from Section 5.2. Each figure compares accuracy results (top) and number of unique messages (bottom) for quantization during both training and inference (left) or only during inference (right). In all games, increasing the number of symbols

		Object		Image		Sent-Ref/Cls	
		Inst	Rnn	Inst	Rnn	Inst	Rnn
Sender	Channel						
	embedding	–	1024	–	50	–	768
	hidden	W_len	1024	W_len	20	W_len	768
	Agent						
	embedding	–	–	50	50	–	768
	hidden	W_len	1024	W_len	20	W_len	768
	kernel	–	–	50	50	–	–
Receiver	Channel						
	embedding	–	1024	–	50	–	768
	hidden	W_len	1024	W_len	20	W_len	768
	Agent						
	embedding	–	–	50	50	–	768
	hidden	W_len	1024	W_len	20	W_len	768

Table 4: Network sizes used in the different games. Sender's channel, sender's encoding agent, receiver channel and receiver encoding agent configuration pertained to z_θ, u_θ, z_ϕ, u_ϕ networks as described in Figure 2. For Instant channel, dimension of sender's channel output and receiver's channel input networks is set by the desired word length (w_len).

improves the results. In addition, in all games quantization during training degrades the performance, in line with the analysis in the main paper. In the Image game (Figure 5), quantization during training prevents system convergence, resulting in failure to learn.

Interestingly, in the Sentence-Cls game (Figure 7), the number of unique messages is equal to the number of targets, similar to other games, while clearly for this game, it would have been sufficient to have one unique message per class, that is, 77 messages.

A.4 Model structure and hyper-parameters

Table 4 shows the sizes of neural networks used in the different games. The Object and Image games are based on the Egg (Kharitonov et al. 2019) implementation. The Sent-Ref and Sent-Cls are based on the Egg channel architecture as well but use different agent encoders. The Object game uses one-layer fully-connected feed-forward neural networks for the sender and receiver. The Image game uses one fully-connected layer, two convolutional layers, and another fully-connected layer for the sender and receiver, following the implementation in Egg. The Sentence games use a Distilled BERT backbone (Sanh et al. 2019), which is frozen in the referential sentence game and fine-tuned in the classification sentence game, in order to achieve the best possible classification results.

In the experiments, we report results when varying the communication elements (alphabet size, word length, and message length). If not mentioned specifically, we use the best tuned values. All hyper-parameters were tuned on the validation set of each dataset. We found it most important

to tune the Gumbel-softmax communication. The best performing values were alphabet size of 100 (for both Instant and Recurrent channels) and message length of 6 (for Recurrent channel). Other configurations of Gumbel-softmax which we experimented with are the temperature (best value was $t = 1.0$) and the option to train with straight-through, which led to performance degradation and is thus disabled in all experiments.

We generally found the quantized communication to be less sensitive to the the channel characteristics. The main paper reports a detailed analysis in Section 5.2. If not reported otherwise, we keep the message length in both continuous and quantized communication to be the same as in Gumbel-softmax, that is, 6 words.

An important option is the number of negative samples (distractors) used during training, which is independent from the number of candidates used for evaluation. For the Object game we used all $10K$ objects, excluding the target, as negatives (distractors). For Sent_Cls we used the maximum number of classes (76, excluding the target class) as negatives. We kept this number the same for the Sent_Ref for consistency considerations. For the Image game we use 99 distractors, due to memory limitations.

The main training hyper-parameters we tuned were the number of epochs, the learning rate, and the batch size. We trained networks for 50 epochs, which was sufficient for convergence. We allowed a patience of 50 epochs to find the best epoch, evaluated on the validation set. The best learning rate was 1e-5 for Object and Sentence games, and 1e-3 for the Image game. The best batch size was 32.

Figure 6 (Sent_Ref game, Instant channel)

Accuracy — Quantization during training and Inference (left)

Alphabet size \ Word length	4	10	20	30	50	100
2	0.01	0.01	0.04	0.05	0.12	0.29
4	0.01	0.03	0.04	0.06	0.10	0.20
8	0.02	0.02	0.04	0.05	0.14	0.26
16	0.01	0.02	0.05	0.05	0.14	0.30

Accuracy — Quantization only during Inference (right)

Alphabet size \ Word length	4	10	20	30	50	100
2	0.03	0.21	0.47	0.54	0.71	0.85
4	0.05	0.28	0.60	0.67	0.83	0.94
8	0.05	0.29	0.62	0.68	0.85	0.94
16	0.05	0.29	0.61	0.68	0.85	0.95

NoUM — Quantization during training and Inference (left)

Alphabet size \ Word length	4	10	20	30	50	100
2	1	4	49	388	586	978
4	7	117	459	937	957	1000
8	24	532	975	1001	1001	1001
16	68	982	1001	1001	1001	1001

NoUM — Quantization only during Inference (right)

Alphabet size \ Word length	4	10	20	30	50	100
2	4	262	936	987	1000	1001
4	18	932	999	1001	1001	1001
8	69	1001	1001	1001	1001	1001
16	214	1001	1001	1001	1001	1001

Figure 6: Accuracy (top) and number of unique messages (NoUM) (bottom) as a function of alphabet size and word length for the **Sent_Ref** game with an Instant channel, comparing quantization during both training and inference (left) or only during inference (right).

Figure 5 (Image game, Instant channel)

Accuracy — Quantization during training and Inference (left)

Alphabet size \ Word length	4	10	20	30	50
2	0.01	0.01	0.01	0.02	0.02
4	0.01	0.01	0.01	0.02	0.02
8	0.01	0.01	0.01	0.02	0.02
16	0.01	0.01	0.01	0.02	0.02

Accuracy — Quantization only during Inference (right)

Alphabet size \ Word length	4	10	20	30	50
2	0.12	0.60	0.88	0.89	0.92
4	0.23	0.88	0.96	0.95	0.96
8	0.24	0.91	0.98	0.97	0.97
16	0.24	0.91	0.98	0.97	0.97

NoUM — Quantization during training and Inference (left)

Alphabet size \ Word length	4	10	20	30	50
2	1	1	1	1	1
4	1	1	1	1	1
8	1	1	1	1	1
16	1	1	1	1	1

NoUM — Quantization only during Inference (right)

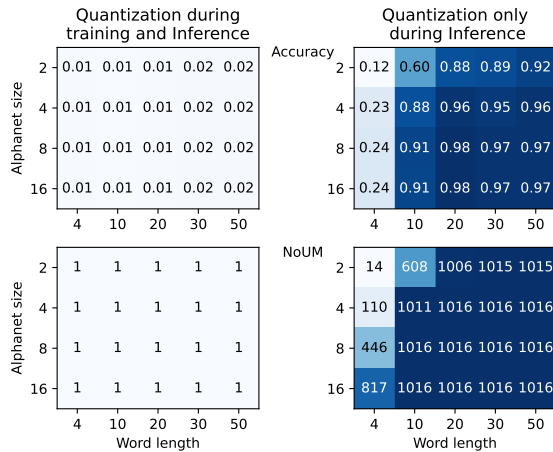| Alphabet size \ Word length	4	10	20	30	50
2	14	608	1006	1015	1015
4	110	1011	1016	1016	1016
8	446	1016	1016	1016	1016
16	817	1016	1016	1016	1016

Figure 5: Accuracy (top) and number of unique messages (NoUM) (bottom) as a function of alphabet size and word length for the **Image** game with an Instant channel, comparing quantization during both training and inference (left) or only during inference (right).

Figure 7 (Sent_Cls game, Instant channel)

Accuracy — Quantization during training and Inference (left)

Alphabet size \ Word length	4	10	20	30	50	100
2	0.01	0.03	0.06	0.11	0.12	0.20
4	0.04	0.09	0.11	0.17	0.19	0.31
8	0.05	0.12	0.17	0.26	0.34	0.48
16	0.04	0.14	0.24	0.36	0.48	0.61

Accuracy — Quantization only during Inference (right)

Alphabet size \ Word length	4	10	20	30	50	100
2	0.15	0.78	0.89	0.90	0.91	0.92
4	0.79	0.91	0.91	0.92	0.92	0.92
8	0.89	0.92	0.92	0.91	0.92	0.92
16	0.89	0.92	0.91	0.92	0.92	0.92

NoUM — Quantization during training and Inference (left)

Alphabet size \ Word length	4	10	20	30	50	100
2	1	5	53	385	592	1024
4	7	118	463	961	985	1042
8	26	561	1010	1039	1041	1042
16	73	1019	1042	1042	1042	1042

NoUM — Quantization only during Inference (right)

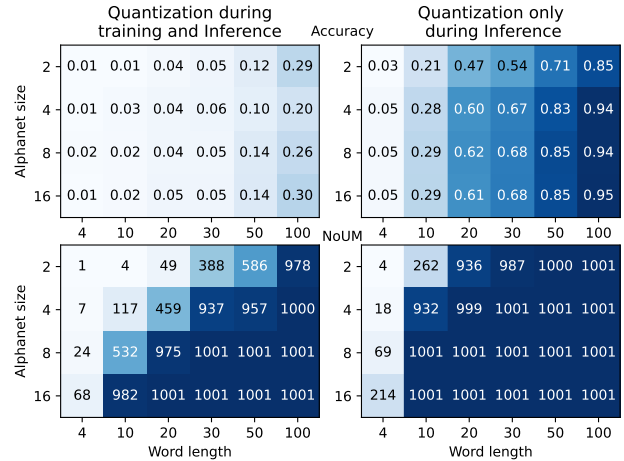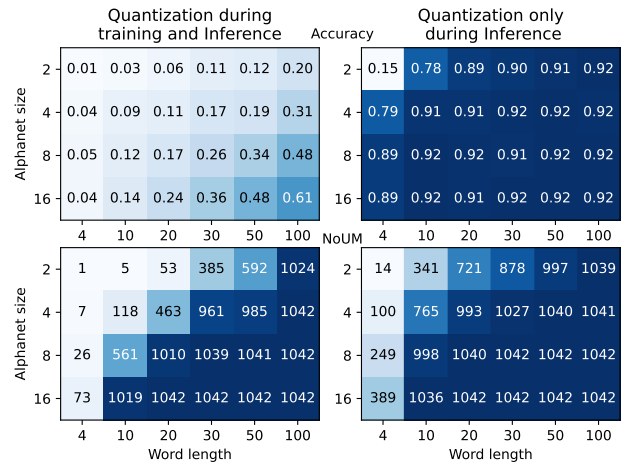| Alphabet size \ Word length	4	10	20	30	50	100
2	14	341	721	878	997	1039
4	100	765	993	1027	1040	1041
8	249	998	1040	1042	1042	1042
16	389	1036	1042	1042	1042	1042

Figure 7: Accuracy (top) and number of unique messages (NoUM) (bottom) as a function of alphabet size and word length for the **Sent_Cls** game with an Instant channel, comparing quantization during both training and inference (left) or only during inference (right).

	Configuration				Number of candidates							
Comm Type	A-size	W-len	M-len		2	10	100	500	1000	2000	5000	10000
CN-RNN	N/A	100	6	avg	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
				std	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GS-RNN	10	10	6	avg	0.997	0.974	0.831	0.560	0.442	0.324	0.208	0.152
				std	0.003	0.021	0.108	0.223	0.217	0.189	0.140	0.110
QT-RNN	10	100	6	avg	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
				std	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CN-Inst	N/A	100	1	avg	1.000	1.000	1.000	1.000	1.000	1.000	0.998	0.998
				std	0.000	0.000	0.000	0.000	0.001	0.001	0.003	0.003
GS-Inst	100	100	1	avg	0.926	0.628	0.179	0.062	0.036	0.022	0.010	0.008
				std	0.004	0.024	0.003	0.009	0.002	0.003	0.005	0.003
QT-Inst	10	100	1	avg	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
				std	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 5: **Object** game test results for the 6 different communication types on an increasing number of candidates. Standard deviation is over three runs.

	Configuration				Number of candidates				
Comm Type	A-size	W-len	M-len		2	10	20	50	100
CN-RNN	N/A	100	6	avg	0.993	0.944	0.893	0.792	0.675
				std	0.004	0.012	0.019	0.035	0.039
GS-RNN	50	50	6	avg	0.626	0.222	0.134	0.067	0.033
				std	0.209	0.213	0.150	0.085	0.043
QT-RNN	10	100	6	avg	0.992	0.948	0.898	0.786	0.673
				std	0.005	0.008	0.018	0.034	0.041
CN-Inst	N/A	100	1	avg	0.999	0.996	0.992	0.982	0.962
				std	0.001	0.003	0.004	0.008	0.006
GS-Inst	50	50	1	avg	0.951	0.661	0.472	0.233	0.132
				std	0.005	0.024	0.027	0.019	0.007
QT-Inst	10	100	1	avg	0.999	0.995	0.990	0.982	0.962
				std	0.000	0.001	0.002	0.006	0.007

Table 6: **Image** game test results for the 6 different configuration on an increasing number of candidates. Standard deviation is over three runs.

	Configuration				Number of candidates					
Comm Type	A-size	W-len	M-len		2	8	16	32	64	77
CN-RNN	N/A	100	6	avg	1.000	0.999	0.997	0.994	0.989	0.989
				std	0.000	0.000	0.003	0.002	0.001	0.001
GS-RNN	100	100	6	avg	0.814	0.386	0.226	0.138	0.078	0.068
				std	0.120	0.165	0.084	0.058	0.041	0.037
QT-RNN	10	100	6	avg	1.000	0.999	0.998	0.995	0.990	0.989
				std	0.000	0.000	0.002	0.001	0.002	0.001
CN-Inst	N/A	100	1	avg	0.999	0.995	0.989	0.981	0.955	0.945
				std	0.001	0.002	0.002	0.003	0.009	0.007
GS-Inst	100	100	1	avg	0.673	0.256	0.139	0.069	0.039	0.035
				std	0.036	0.045	0.012	0.015	0.006	0.008
QT-Inst	10	100	1	avg	0.999	0.996	0.990	0.981	0.956	0.945
				std	0.001	0.002	0.003	0.003	0.009	0.006

Table 7: **Sentence referential (Sent_Ref)** game test results for the 6 different communication types on an increasing number of candidates. Standard deviation is over three runs.

	Configuration				Number of candidates					
Comm Type	A-size	W-len	M-len		2	8	16	32	64	77
CN-RNN	N/A	100	6	avg	0.993	0.972	0.959	0.943	0.925	0.920
				std	0.001	0.003	0.004	0.006	0.008	0.009
GS-RNN	10	10	6	avg	0.941	0.806	0.717	0.611	0.493	0.460
				std	0.023	0.045	0.045	0.040	0.032	0.031
QT-RNN	10	10	6	avg	0.993	0.972	0.959	0.943	0.925	0.920
				std	0.001	0.002	0.003	0.004	0.007	0.009
CN-Inst	N/A	100	1	avg	0.994	0.976	0.964	0.948	0.927	0.920
				std	0.002	0.005	0.007	0.009	0.013	0.014
GS-Inst	100	100	1	avg	0.960	0.811	0.666	0.478	0.298	0.258
				std	0.004	0.014	0.022	0.033	0.038	0.038
QT-Inst	10	100	1	avg	0.993	0.976	0.964	0.949	0.930	0.925
				std	0.002	0.002	0.004	0.006	0.008	0.009

Table 8: **Sentence classification (Sent_Cls)** game test results for the 6 different communication types on an increasing number of candidates. Standard deviation is over three runs.