Adaptive Discrete Communication Bottlenecks with Dynamic Vector Quantization for Heterogeneous Representational Coarseness

Dianbo Liu[1], Alex Lamb[1], Xu Ji[1], Pascal Junior Tikeng Notsawo[1], Michael Mozer[2], Yoshua Bengio[1, 4], Kenji Kawaguchi [3]
[1] Mila-Quebec AI Institute
[2] Google Research, Brain Team
[3] National Univeristy of Singapore
[4] CIFAR AI Chair
liudianbo@gmail.com, alex6200@gmail.com

Abstract

Vector Quantization (VQ) is a method for discretizing latent representations and has become a major part of the deep learning toolkit. It has been theoretically and empirically shown that discretization of representations leads to improved generalization, including in reinforcement learning where discretization can be used to bottleneck multi-agent communication to promote agent specialization and robustness. The discretization tightness of most VQ-based methods is defined by the number of discrete codes in the representation vector and the codebook size, which are fixed as hyperparameters. In this work, we propose learning to dynamically select discretization tightness conditioned on inputs, based on the hypothesis that data naturally contains variations in complexity that call for different levels of representational coarseness which is observed in many heterogeneous data sets. We show that dynamically varying tightness in communication bottlenecks can improve model performance on visual reasoning and reinforcement learning tasks with heterogeneity in representations.

Introduction

Discretization of latent representations via vector quantization is a method for improving the robustness and generalization of learned models (Oord, Vinyals, and Kavukcuoglu 2017; Liu et al. 2021). Replacing a continuous representation with a discrete representation, and limiting the capacity of the discrete representation, both improve generalization guarantees (Liu et al. 2021). Discretization imposes a bottleneck (Tishby, Pereira, and Bialek 2000) as the representation can take fewer values, and reducing capacity of the discrete representation tightens this bottleneck.

Discretization can be used to bottleneck communication in modular inference models, for example in multi-agent reinforcement learning. Modular inference combines outputs across modules into new module inputs while restricting the number of modules that communicate (Goyal et al. 2019). To make up for the loss in expressivity from being restricted to few communicative modules in each timestep, the model is forced to learn to perform inference in steps of specialized skill as opposed to applying all skills at once, which specializes the modules by definition since few are active in each step (Darwen and Yao 1996; Goyal et al. 2019; Bengio 2017; Lamb et al. 2021). The communication bottleneck is tightened by switching from continuous to discrete representations (Liu et al. 2021). Beyond improvements in generalization error from decreasing capacity, it has been hypothesized that the inductive bias of bottlenecked communication between modules improves generalization by 1) reflecting the true causal structure of data generated by sparse interactions between few variables, 2) increasing robustness when modules are recombined in novel ways, compared to unspecialized modules with all-to-all communication, 3) improving sample efficiency since specialized modules require fewer training points to learn (Goyal et al. 2019; Bengio 2017).

However, discrete bottleneck methods typically lack adaptability. This work improves on the vector quantization method of Liu et al. (2021) by making tightness of the bottleneck dynamic. Instead of a single discretization function that maps all inputs to a discrete space with fixed size, we use a pool of discretization functions with varying levels of output capacity, and choose the function applied for a given input using key-query attention between representations of the input and discretization functions.

The hypothesis is that this improves generalization because the optimal level of discretization suggested by the bounds, which is the tightest bottleneck such that training error can still be minimized (Liu et al. 2021), is unlikely to be the same for all regions of a data distribution. The shortest description that captures adequate information in inputs for performing well on a task generally varies with the input, for example images contain different numbers of objects, and gameplay involves goals of varying complexity. In terms of generalization error, using a single discretization capacity is potentially wasteful for simpler inputs, because the generalization gap can be reduced by selectively imposing a tighter representation bottleneck on the former without increasing training error. To minimize the model selecting looser bottlenecks than necessary, we use an objective function that penalizes the choice of bottleneck proportional to its capacity.

In summary, the contributions of our work are as follows:

- We propose a dynamic vector quantization method (DVQ) that adaptively chooses the number of discrete codes and the codebook size that control tightness of the bottleneck.
- Our theoretical analysis shows that dynamic adjustment of

the bottleneck improves generalization error under the sufficient condition of tighter average bottleneck and equal training loss.

- We empirically show improvement in performance by using DVQ to discretize inter-component communication within a deep learning model and inter-agent communication between agents, compared to using VQ with fixed bottleneck capacity.

Method

Communication Discretization

The process of converting data with continuous attributes into data with discrete attributes is called discretization (Chmielewski and Grzymala-Busse 1996). In this study, we use discrete latent variables to quantize information communicated among different modules in a similar manner to codebooks (Liu et al. 2021), which is a general version of vector quantization (VQ-VAE, Oord, Vinyals, and Kavukcuoglu (2017)) where the hidden representation is a list of discretized codes instead of a single discretized code. The discretization process for each vector $h \in \mathcal{H} \subset \mathbb{R}^m$ is described as follows. First, vector h is divided into G segments $s_1, s_2, \ldots, s_G$ with $h = \text{CONCATENATE}(s_1, s_2, \ldots, s_G)$, where each segment $s_i \in \mathbb{R}^{m/G}$ with $\frac{m}{G} \in \mathbb{N}^+$. Second, each continuous segment s_i is discretized separately by being mapped to a discrete latent space vector $e \in \mathbb{R}^{L\times(m/G)}$ where L is the size of the discrete latent space (i.e., an L-way categorical variable):

$$e_{o_i} = \text{DISCRETIZE}(s_i), \quad \text{where } o_i = \underset{j\in\{1,\ldots,L\}}{\text{argmin}} ||s_i - e_j||.$$

These discretized codes, which we call the factors of continuous representation h, are concatenated to obtain the final discretized vector z as

$$z = \text{CONCATENATE}(\text{DISCRETIZE}(s_1), \text{DISCRETIZE}(s_2), ..., \text{DISCRETIZE}(s_G)). \tag{1}$$

The multiple steps described above can be summarized by $z = q(h, L, G)$, where $q(\cdot)$ is the whole discretization process using the codebook, L is the codebook size, and G is number of segments or factors per vector.

The loss for model training is $\mathcal{L} = \mathcal{L}_{\text{task}} + \mathcal{L}_{discretization}$ where

$$\mathcal{L}_{discretization} = \frac{1}{G}\Bigg(\sum_i^G ||\,\text{sg}(s_i) - e_{o_i}||_2^2 + \beta \sum_i^G ||s_i - \text{sg}(e_{o_i})||_2^2\Bigg). \tag{2}$$

$\mathcal{L}_{\text{task}}$ is the loss for the specific task, *e.g.*, cross entropy loss for classification or mean square error loss for regression, sg refers to a stop-gradient operation that blocks gradients from flowing into its argument, and β is a hyperparameter which controls the reluctance to change the code. The term $\sum_i^G ||\,\text{sg}(s_i) - e_{o_i}||_2^2$ is the codebook loss, which only applies to the discrete latent vector and brings the selected e_{o_i} close to the output segment s_i. The term $\sum_i^G ||s_i - \text{sg}(e_{o_i})||_2^2$ is the commitment loss, which only applies to the target segment s_i and trains the module that outputs s_i to make s_i stay close to the chosen discrete latent vector e_{o_i}. Following the original codebooks and VQ-VAE papers, we found 0.25 to be a good value for β, and e was initialized using k-means clustering on vectors h with $k = L$.

Dynamic Bottlenecks

Instead of a single codebook and discretization bottleneck, we use multiple bottlenecks where the tightness of each bottleneck is defined by the number of factors and codebook size. A pool of N discretization functions $Q = \{q_t\}_{t\in[N]}$ is made available to all representations being discretized, where the number of factors and codebook size for each discretization function are given by G_t and L_t respectively. The discretization functions in the pool do not share parameters nor codebooks with each other. Each of the discretization functions is associated with a signature key vector $k_t \in \mathbb{R}^l$ which is randomly initialized and learned in the training process. Key-value attention is conducted between k_t and query $f(h) \in \mathbb{R}^l$ where h is the continuous representation being discretized and f is a single layer neural network projector. Gumbel-softmax (Jang, Gu, and Poole 2016) is applied on the attention scores to make a one-hot selection of which q_t to use, with categorical distribution π^h over discretization functions for h given by

$$\pi^h(t) = \frac{\exp(k_t^\intercal f(h))}{\sum_{j\in[N]} \exp(k_j^\intercal f(h))}. \tag{3}$$

Bottleneck Tightness

In order to learn a communication bottleneck with as few bits as necessary for minimizing training error, we introduce pressure to choose q_t with low G_t and L_t. This pressure is implemented with capacity penalty $\mathcal{C}_{bottlenecking}$:

$$\mathcal{C}_{bottlenecking} = G_t \ln(L_t) \tag{4}$$

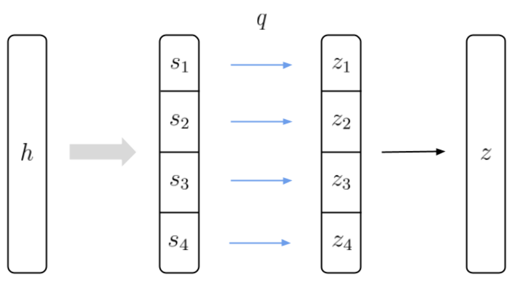
$$\mathcal{L} = \mathcal{L}_{task} + \alpha\mathcal{L}_{discretization} + \beta\mathcal{C}_{bottlenecking} \tag{5}$$

where $\mathcal{L}$ is the overall loss for the representation being discretized, $\mathcal{L}_{task}$ is the task loss, $\mathcal{L}_{discretization}$ is the sum of commitment and codebook losses from the discretization process and $\mathcal{C}_{bottlenecking}$ is the bottlenecking cost. α and β are hyperparameters that are chosen using a validation set.

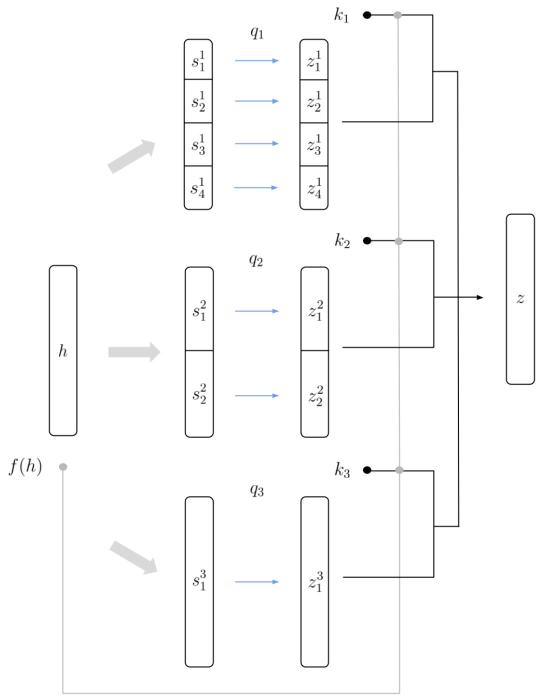
In our experiments, we found that including a continuous-valued function in the pool, corresponding to expressivity in the limit of codebook size $L_t \to \infty$, improved performance in some tasks. In this case L_t was set to be a large number (10^9) in the penalty term $\mathcal{C}_{bottlenecking}$.

Architectural Choices

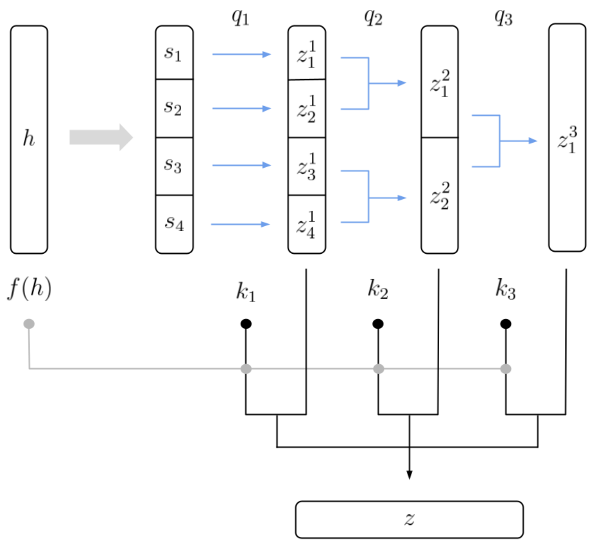
Given continuous representation h, the bottleneck can be enforced in either a flat or hierarchical manner (fig. 1). In the flat case, the bottlenecked representation is the output of a single $q_t \in Q$ selected with key-value attention. In the hierarchical case, all functions $\{q_t\}_{t\in[N]}$ are utilized by first ordering them in order of descending G_t, and setting the input segments of each function to be concatenated factors produced by the previous function, with h as input into the first function. Then output of a single $q_t \in Q$ is selected with key-value attention.



(a) Fixed bottleneck with a single discretization function (Liu et al. 2021).



(b) Adaptive bottleneck with flat attention over discretization functions (ours).



(c) Adaptive bottleneck with hierarchical attention over discretization functions (our method).

Figure 1: Dynamic discrete bottlenecks can be implemented as a flat function of continuous input h (center), or as an iterative function that produces progressively coarser outputs (right). Bottleneck functions q_t for $t \in [N]$, have different capacity and separate parameters. One bottleneck is selected for each input using key-value attention between representations of the input and discretization functions, $f(h)$ and k_t respectively.

Theoretical Analysis

In this section, we show that the adaptive discretization process has a potential advantage in improving the performance of the final model by better trading off the balance between the generalization and expressivity adaptively for each input. Moreover, our analysis predicts that the additional regularization, $C_{bottlenecking}$, in our algorithm plays an important role for this tradeoff, and that the number of adaptive bottlenecks N cannot be arbitrarily large. To achieve this goal, we follow the abstract framework of Liu et al. (2021). That is, let ϕ be an arbitrary function, which can refer to the composition of an evaluation criterion and the rest of the network following (adaptive) discretization bottlenecks. Given any function $\phi : \mathbb{R}^m \to \mathbb{R}$ and any family of sets $S = \{S_1, \ldots, S_K\}$ with $S_1, \ldots, S_K \subseteq \mathcal{H}$, let us define the corresponding function ϕ_k^S by $\phi_k^S(h) = 1\{h \in S_k\}\phi(h)$ for all $k \in [K]$, where $[K] = \{1, 2, \ldots, K\}$. Let $e^{(t)} \in \mathbb{R}^{L_t \times (m/G_t)}$ be fixed and we denote by $(Q_{k_t}^{(t)})_{k_t \in [L_t^{G_t}]}$ all the possible values after the discretization process for t-th adaptive bottleneck: i.e., $q(h, L_t, G_t) \in \cup_{k_t \in [L_t^{G_t}]}\{Q_{k_t}^{(t)}\}$ for all $h \in \mathcal{H}$. We define $k_{\max} = \max_{t \in [N]} L_t^{G_t}$ and $Q_{k_t}^{(t)} = \emptyset$ for all $k_t > L_t^{G_t}$. Under this abstract setting with $t = N = 1$, the previous paper (Liu et al. 2021) proved their main theoretical result, showing that the models with non-adaptive discretization process have advantages over the continuous models without it (we present a slightly tighter version of the previous paper's results in Appendix). Thus, in this section, we focus on the comparison of adaptive and non-adaptive discretization processes.

To analyze the adaptive discretization process, we introduce the additional notation: let $q_t(h)$ be the discretization process with a particular bottleneck $t \in [N]$, and $q(h)$ be the whole discretization process as $q(h) = q_{\hat{t}(h)}(h)$ where $\hat{t}(h)$ is the result of the key-value attention. Define $I_t = \{i \in [n] : \hat{t}(h_i) = t\}$, which is the set of the indices of training samples that end up using the t-th adaptive bottleneck.

The following theorem extends the main theorem from Liu et al. (2021) to the setting of adaptive bottlenecking and shows the advantage of the adaptive version over the non-adaptive version:

Theorem 1. *Let $N \in \mathbb{N}_+$ and $S_k = \{Q_k^{(t)}\}_{t=1}^N$ for all $k \in [k_{\max}]$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an iid draw of n examples $(h_i)_{i=1}^n$, the following holds for any $\phi : \mathbb{R}^m \to \mathbb{R}_+$ and $k \in [k_{\max}]$:*

$$\mathbb{E}_h[\phi_k^S(q(h))] \le \frac{1}{n}\sum_{i=1}^n \phi_k^S(q(h_i)) + \alpha(\mathcal{J}_1 + \mathcal{J}_2), \quad (6)$$

where $\alpha = \sup_{h \in \mathcal{H}} \phi_k^S(h)$,

$$\mathcal{J}_1 = \sum_{t=1}^N \frac{|I_t|}{n}\sqrt{\frac{G_t \ln(L_t) + \ln(N/\delta)}{2n}}, \text{ and}$$

$$\mathcal{J}_2 = 1\{N \ge 2\}\sqrt{\frac{2N\ln 2 + 2\ln(1/\delta)}{n}}.$$

Proof. The proof is presented in Appendix □

Theorem 1 recovers the previous result of (Liu et al. 2021) when we set $N = 1$ as desired. That is, by setting $N = 1$, the inequality (6) in Theorem 1 becomes

$$\mathbb{E}_h[\phi_k^S(q(h))] \le \frac{1}{n}\sum_{i=1}^n \phi_k^S(q_1(h_i)) \quad (7)$$
$$+ \alpha\sqrt{\frac{G_1 \ln(L_1) + \ln(1/\delta)}{2n}},$$

which is the previous bound in Theorem 1 of Liu et al. (2021). Thus, we successfully generalized the previous analysis framework to cover both the adaptive and non-adaptive versions in an unified manner.

Theorem 1 shows that there are two ways that the adaptive version can improve the non-adaptive version; i.e., the potential improvement happens when $\sum_{t=1}^N \frac{|I_t|}{n}\sqrt{G_t \ln(L_t)} < \sqrt{G_1 \ln(L_1)}$ or $\frac{1}{n}\sum_{i=1}^n \phi_k^S(q(h_i)) < \frac{1}{n}\sum_{i=1}^n \phi_k^S(q_1(h_i))$. The first criterion is met when the weighted average of the adaptive bottleneck sizes $G_t \ln(L_t)$ is smaller than the pre-fixed bottleneck size $G \ln(L) = G_1 \ln(L_1)$. This is indeed encouraged by the additional regularization, $C_{bottlenecking}$, in our algorithm. The second criterion is satisfied when the training loss with adaptive bottlenecks is less than that with a fixed bottleneck.

Theorem 1 provides the insight that the benefit of the adaptive bottlenecks lies in the tradeoff between the expressivity (to minimize the training loss $\frac{1}{n}\sum_{i=1}^n \phi_k^S(q(h_i))$) and generalization (to minimize the term $\sum_{t=1}^N \frac{|I_t|}{n}\sqrt{G_t \ln(L_t)}$). For a fixed bottleneck, the training loss tends to decrease as G_1 increases because increasing G_1 improves the expressivity and trainability. However, increasing G_1 results in a worse bound on the generalization gap as the gap scales as $\sqrt{G_1/n}$ in Theorem 1. Thus, we have a tradeoff between the expressivity and generalization. For the adaptive bottlenecks, different values of G_t are used for different samples. As a result, the adaptive bottlenecks can have a better tradeoff between expressivity and generalization by only using the necessary expressivity or bottleneck G_t (and L_t) for each sample to reduce $\sum_{t=1}^N \frac{|I_t|}{n}\sqrt{G_t \ln(L_t)}$ while minimizing the training loss $\frac{1}{n}\sum_{i=1}^n \phi_k^S(q(h_i))$.

For example, consider a scenario with a subset of training samples that require G and L to be extremely large to minimize the training loss for the subset. If we use a pre-fixed bottleneck, we need to make $G_1 \ln(L_1)$ to be extremely large to minimize all training samples. This results in a bad generalization term $\sqrt{G_1 \ln(L_1)/n}$ in Theorem 1. On the other hand, if we use the adaptive bottleneck, we can minimize the training loss for all samples by using a large $G_t \ln(L_t)$ only for the subset while using small values of $G_t \ln(L_t)$ for other samples.

In terms of these two criteria for the tradeoff, the adaptive version seems to be always better as we increase N. However, this better tradeoff comes with a cost. In Theorem 1, the cost is captured by the additional term $\sqrt{\frac{2N\ln 2 + 2\ln(1/\delta)}{n}}$, which increases as N increases. Thus, while the adaptive version is better in the sense of the tradeoff of the two criteria, it comes with the additional cost of $\sqrt{N/n}$ term. This predicts that N cannot be arbitrarily large.

Related Works

Vector Quantization. VQ is motivated by the fundamental result of Shannon's rate-distortion theory (Gersho and Gray 1991; Cover and Thomas 2006): better performance can always be achieved by coding vectors instead of scalars, even if the sequence of source symbols are independent random variables. K-means (MacQueen et al. 1967) is the prime method for VQ. The K-means algorithm clusters data by trying to separate the samples into n groups of equal variance, this by minimizing the intra-class inertia. Despite the performance of this algorithm, K-means based VQ has an exponential complexity in encoding (computation and memory) and decoding (memory) (Tan et al. 2018). Various improvements to K-means have been proposed to address complexity issues (e.g., product quantization) (Ge et al. 2014), but their performance is suboptimal compared to some deep neural network (DNN) based approaches (Tan et al. 2018), where the idea is to map input data from the original high dimensional space to the DNN latent space with lower dimensionality, and apply K-means to the latent codes. Because K-means in the latent space is sub-optimal for VQ, Tan et al. (2018) proposed DeepVQ, a fully-DNN architecture for vector quantization, in the context of data compression. DeepVQ is an autoencoder that overcomes the complexity issue by directly mapping to the binary index of the codeword. Recent works (Rolfe 2017; Maddison, Mnih, and Teh 2017) proposed novel reparameterization methods to handle the non-gradient issue for discrete random variables in VAE. VQ-VAE (Oord, Vinyals, and Kavukcuoglu 2017) avoids such problem by using the identity function, namely copying gradients from the decoder input to encoder output (Bengio, Léonard, and Courville 2013).

Bottlenecking inter-module communication within a model. Many methods have been used in recent years to enable efficient communication between specialized components of machine learning models, from attention mechanisms for selectively communicating information between specialized components in machine learning models (Goyal et al. 2019, 2021b,a) and transformers (Vaswani et al. 2017; Lamb et al. 2021); collective memory and shared parameters for multi-agent communication (Pesce and Montana 2020), node attributes in graph-based models (Koller and Friedman 2009; Battaglia et al. 2018) for relational reasoning, dynamical systems simulation, multi-agent systems, and in many other areas. While most of inter-specialist communication mechanisms operates in a pairwise symmetric manner, Goyal et al. (2021b) introduced a bandwidth limited communication channel to allow information from a limited number of modules to be broadcast globally to all modules, inspired by Global workspace theory (Baars 2019). Recently, Liu et al. (2021) showed that replacing a continuous representation with a discrete representation, and limiting the discrete representation to a short list of codes from a small codebook, both improve generalization guarantees. Following VQ-VAE (Oord, Vinyals, and Kavukcuoglu 2017), Liu et al. (2021) proposed discrete-valued neural communication (DVNC) to improve systematic generalization in a variety of architectures, including transformers (Vaswani et al. 2017; Lamb et al. 2021), RIMs (Goyal et al. 2020) , and graph neural networks (Kipf, van der Pol, and Welling 2020).

Bottlenecking inter-agent communication in multi-agent RL. A wide range of multi-agent applications have benefitted from inter-agent message passing including distributed smart grid control (Pipattanasomporn, Feroze, and Rahman 2009), consensus in networks (You and Xie 2011), multi-robot control (Ren and Sorensen 2008), autonomous vehicle driving (Petrillo et al. 2018), elevators control (Crites and Barto 1998) and for language learning in two-agent systems (Lazaridou, Peysakhovich, and Baroni 2017). An important challenge in MARL is how to facilitate communication among interacting agents, especially in tasks requiring synchronization (Scardovi and Sepulchre 2009; Wen et al. 2012). For example, in the multi-agent deep deterministic policy gradient (Lowe et al. 2020), which extends the actor-critic algorithm (Degris, White, and Sutton 2013); the input size of each critic increases linearly with the number of agents, which hinders its scalability (Jiang and Lu 2018). To overcome this, Pesce and Montana (2020) provides the agents with a shared communication device that can be used to learn from their collective private observations and share relevant messages with others in the centralised learning and decentralised execution paradigm (Foerster et al. 2016; Kraemer and Banerjee 2016; Oliehoek, Spaan, and Vlassis 2008). However, their approach is limited to small-scale systems. Iqbal and Sha (2019) proposed Multi-Actor-Attention-Critic to learn decentralised policies with centralised critics, which selects relevant information for each agent at every time-step through an attention mechanism. Many other communication mechanisms have been proposed, such as CommNet (Sukhbaatar, Szlam, and Fergus 2016), IC3NEt (Singh, Jain, and Sukhbaatar 2018), BiCNet (Peng et al. 2017), attention (Jiang and Lu 2018) and soft-attention (Das et al. 2020) based, master-slave architecture (Kong et al. 2017), Feudal Multiagent Hierarchies (Ahilan and Dayan 2019), Bayesian Action Decoder (Foerster et al. 2019).

Experiments

In our experiments we test the hypothesis that imposing a dynamic bottleneck on communication facilitates modularization and improved out-of-distribution generalization to data with heterogeneous representational coarseness compared to a fixed bottleneck. We test our method in three settings, 1) inter-component communication among different components in the same model and test on in-distribution homogeneous data , 2) same setting as 1) but test on out-of-distribution data with representational coarseness heterogeneity among different data points and 3) inter-agent communication in cooperative multi-agent reinforcment learning (MARL) which natural have heterogeneity among observations of different agents.

Task/Model	**Continuous**	**DVNC**	**Adaptive Quantization(ours)**	**Adaptive Hierarchical(ours)**
Alien	0.130 ± 0.023	0.152 ± 0.026	0.170 ± 0.075	**0.177 ± 0.057**
BankHeist	0.397 ± 0.043	0.371 ± 0.057	0.406 ± 0.037	**0.414 ± 0.084**
Berzerk	0.436 ± 0.250	0.584 ± 0.011	**0.630 ± 0.016**	0.580 ± 0.021
Boxing	0.873 ± 0.021	0.908 ± 0.068	0.929 ± 0.031	**0.957 ± 0.041**
MsPacman	**0.152 ± 0.037**	0.135 ± 0.030	0.054 ± 0.002	0.057 ± 0.005
Pong	0.169 ± 0.047	0.201 ± 0.035	0.205 ± 0.068	**0.225 ± 0.031**
shapes	0.674 ± 0.055	0.672 ± 0.053	0.664 ± 0.034	**0.692 ± 0.065**
SpaceInvaders	0.138 ± 0.037	0.199 ± 0.085	**0.258 ± 0.103**	0.232 ± 0.076

Table 1: Performance of different methods in bottlenecking inter-module communication in a visual reasoning model on in-distribution test data

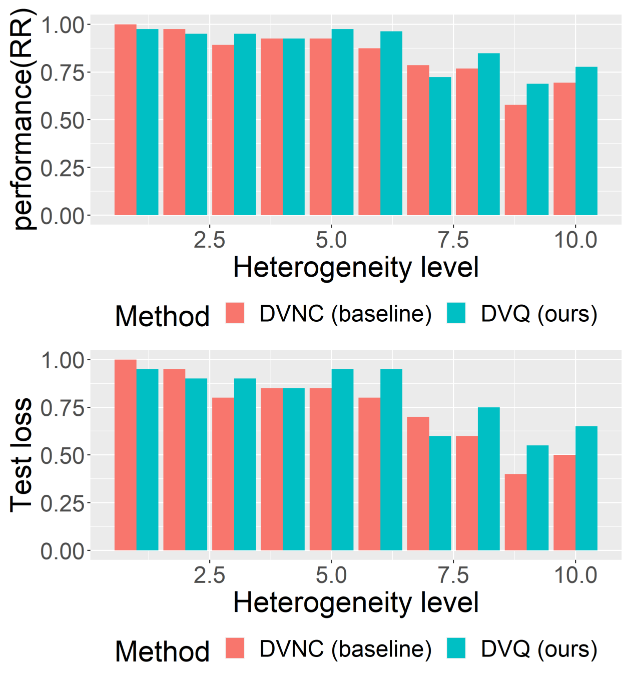


Figure 2: Adaptive hierarchical bottlenecking show performance advantages in out-of-distribution (OOD) test data with high levels of heterogeneity. Graph neural networks with adaptive hierachical bottlenecking (our method) in communication show high performance (left) and lower test loss (right) than bottlenecking with fixed tightness (Liu et al. 2021) on OOD test data with heterogeneous samples with different image cropping. Heterogeneity levels indicate number of different augmentations applied on images in the OOD test data set.

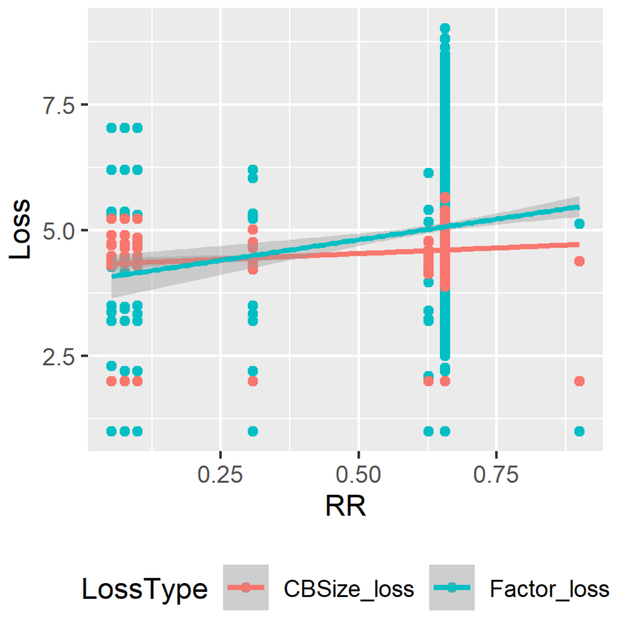


Figure 3: Tighter bottlenecks are used in difficult tasks. A positive correlation of both factor size (blue) and codebook size (red) with model performance measured by reciprocal rank (RR). The higher the RR, the better the model performs in the task.

Experimental Setup

Discretize inter-module communication within a model. We explore the effects of using DVQ to bottleneck inter-module communication for visual reasoning tasks. The tasks are to predict object movement in a grid world (referred to as **"Shapes"**) and 7 different Atari game environments. In all of the environments, changes in each image frame depend on the previous image frame and the actions applied to different objects. Objects are captured by a convolutional neural network and are passed to a Graph neural network (GNN). Positions of objects are captured by nodes in the GNN and the relative positions among different objects are communicated through the network a discrete manner (Kipf, van der Pol, and Welling 2020). In this work we apply either DVQ or DVNC to discretize the vector sum of edges each node is connected to. In the heterogeneous out-of-distribution test data set, a different set of environmental images were collected compared with the training set and different types of image cropping were applied to the images in the OOD test sets. Each image only receive one type of croppining and the total number of different types of cropping applied to the whole test set is defined as the heterogeneity level of the data set. None of the image cropping types in OOD test set were seen in training set.

Task/Model	**Continuous**	**DVNC**	**Adaptive Quantization(ours)**	**Adaptive Hierarchical(ours)**
Alien	0.090 ± 0.013	0.102 ± 0.016	0.117 ± 0.045	**0.117 ± 0.037**
BankHeist	0.198 ± 0.022	0.251 ± 0.052	0.270 ± 0.049	**0.271 ± 0.033**
Berzerk	0.335 ± 0.149	0.483 ± 0.009	0.501 ± 0.015	**0.512 ± 0.015**
Boxing	0.678 ± 0.022	0.701 ± 0.018	0.710 ± 0.033	**0.761 ± 0.021**
MsPacman	**0.099 ± 0.012**	0.110 ± 0.030	0.050 ± 0.002	0.051 ± 0.003
Pong	0.109 ± 0.037	0.120 ± 0.014	0.135 ± 0.023	**0.145 ± 0.032**
shapes	0.410 ± 0.045	0.559 ± 0.066	0.600 ± 0.024	**0.601 ± 0.035**
SpaceInvaders	0.102 ± 0.017	0.109 ± 0.035	0.168 ± 0.63	**0.182 ± 0.086**

Table 2: Adaptive bottlenecking in inter-module communication in a visual reasoning model on show performance advantages on out-of-distribution test data with high level of heterogeneity(heterogeneity=10)

Bottleneck communication among reinforcement learning agents. To investigate the potential of using DVQ to bottleneck communication among different reinforcement learning agents, we conducted experiments in two MARL environments with cooperative tasks. Multi-Agent Particle Environment is a 2D cooperative MARL environment originally proposed in the MADDPG paper (Lowe et al. 2020). There are 3 agents and 3 landmarks in a 2D space. In this study, we use the cooperative navigation task (called **"SimpleSpread"**) where agents are required to cover all landmarks while minimizing inter-agent collisions. The action space is discrete and agents can move either up, down, left or right in addition to a no-move "action". Each agent only has a partial view of the environment. In this work, we allow agents to encode their partial view and send it as messages to other agents. We apply discretization bottlenecks on the messages each agent receives. The second MARL environment we use is GhostRun. The environment consists of multiple agents, each with a partial view of the ground below them. There are multiple ghosts (red dots) moving randomly in the environment. All the agents work as a team to escape from the ghosts. Once an agent has ghosts in its view, the team receives negative rewards which are multiplied by the total number of ghosts in the joint view of all agents. Similar to the Particle environment, we allow agents to communicate with each other by sending their encoded partial view as messages, and we discretize the messages received by each agent.

Inter-module Communication

The bottlenecks were applied on edges of graphs in the GNN linking nodes that represent objects in the image (Liu et al. 2021). The visual reasoning tasks require the model to forecast future scenes in the environments based on current state and actions if available. Out of the 8 visual reasoning tasks, DVQ slightly outperforms or matches fixed-bottleneck DVNC in 7 tasks on the in-distribution test set (table 1), with the adaptive hierarchical architecture best on average. On the out-of-distribution test set with heterogeneity level 10, DVQ significantly outperforms fixed-bottleneck DVNC in 7 tasks (table 2), with the adaptive hierarchical architecture best on average. In addition, we show that on average, DVQ have similar performance as DVNC when heterogeneity level is low but have better performance when heterogeneity level is high (figure 2), which agrees with our hypothesis that adaptive bottlenecking helps in data with heterogeneous representational coarseness. In addition, our ablation analyses show that the observation we made above are robust to different number of factors, codebook sizes, and other hyperparameters (appendix figure).

Inter-agent Communication In Cooperative MARL

Next we investigate the potential benefits of using DVQ to bottleneck information exchange among multiple agents in two cooperative MARL tasks. Agents communicate with each other in a broadcasting manner where each agent j sends identical messages $m_{j,t}$, the encoded partial observation of agent j, to all other team members simultaneously. Each agent in the cooperative game receive $M_t = \{m_{j,t} | 0 \leq j < N_{agents}\}$ from all agents in the team at the same time and reads M_t using an attention mechanism $m'_{i,t} = \text{softmax}\left(\frac{q^m_{i,t} \kappa_t}{M} * t\sqrt{d_m}\right)$ where $m'_{i,t}$ is the final message agent i received from others at time step t and $q^m_{i,t} = f_q(m_{j,t})$ and $\kappa_t = f_m(M * t)$. Both f_m and f_q are MLP encoders. At each time step, for each agent, $m'_{i,t}$ is discretized. In DVQ, $L \in [16, 64, 256]$ and $G \in [1, 2, 4]$. In the VQ baseline, $L = 16$ and $G = 1$. In the Particle environment, both dynamic hierarchical quantization and dynamic quantization outperform VQ with fixed coarseness. In the GhostRun environment, dynamic hierarchical quantization outperforms the baseline and dynamic quantization ties with the baseline (Figures in Appendix)

Task Difficulty And Bottleneck Tightness

In the sections above, our experiments showed that DVQ outperforms VQ with fixed capacity in various tasks. Next, we seek to understand behaviors of DVQ. In the visual reasoning tasks, difficulty varies significantly among different games. For example, object movement in some Atari games are more unpredictable than others. In addition, even within the same game, difficulty many vary among episodes. This motivates us to ask the question of whether the tightness of bottleneck introduced by DVQ is influenced by how hard a task is. To answer this, we first quantify difficulty of the task based on the test performance of the visual reasoning model. Reciprocal Ranks (RR) is used as the performance metric where higher RR means better performance. Next, we break down the capacity penalty into the factor loss, which corresponds to penalty as a result of the number of factors, and codebook size loss, which is the term that penalizes high capacity caused by large codebook sizes. Our analysis shows that both factor loss and codebook size loss have positive correlation with RR, across the 8 tasks (Figure 3), with stronger correlation in the case of number of factors. Depending on the direction of causality in this observation, it suggests that tighter bottlenecks are picked for hard tasks either because of difficulty of optimization (when the optimal level of bottleneck tightness is not found, the model tends to undershoot capacity) or because harder tasks call for stronger regularization from tighter bottlenecks.

Discretization Difficulty and Bottleneck Tightness

In addition to difficulty of the task itself, another dimension to consider is the difficulty of discretization, which is largely determined by distribution of the learned representation and dynamics of the model. We estimate difficulty of discretization using the discretization loss obtained during evaluation. Recall that the discretization loss is averaged over factors. We observed that DVQ tends to increase its capacity by increasing both the number of factors and codebook size when the discretization loss is high (appendix figures), which is reasonable as high expressivity in inputs puts positive pressure on decreasing the tightness of the bottleneck.

Conclusion

Effective communication between different specialized components of a model or between different agents in a MARL system requires compatible representations and synchronized messages.

We have shown theoretically and empirically the benefits of using a set of discretization functions with varying levels of output capacity and choosing the function applied for a given input using key-query attention, instead of using a single discretization function that maps all inputs to a discrete space of fixed size. Adaptively bottlenecking capacity makes tightness dependent on inputs, which allows the generalization gap to be decreased. The experiments show that performance is improved compared to using a fixed capacity bottleneck across a range of tasks in visual reasoning and multi-agent reinforcement learning.

References

Ahilan, S.; and Dayan, P. 2019. Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning. arXiv:1901.08492.

Baars, B. J. 2019. *On Consciousness: Science & Subjectivity*. Nautilus Press.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C.; Song, F.; Ballard, A.; Gilmer, J.; Dahl, G.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261.

Bengio, Y. 2017. The consciousness prior. *arXiv preprint arXiv:1709.08568*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432.

Chmielewski, M. R.; and Grzymala-Busse, J. W. 1996. Global discretization of continuous attributes as preprocessing for machine learning. *International journal of approximate reasoning*, 15(4): 319–331.

Cover, T. M.; and Thomas, J. A., eds. 2006. *Elements of Information Theory*. Wiley.

Crites, R. H.; and Barto, A. G. 1998. Elevator Group Control Using Multiple Reinforcement Learning Agents. *Mach. Learn.*, 33(2–3): 235–262.

Darwen, P.; and Yao, X. 1996. Automatic modularization by speciation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, 88–93. IEEE.

Das, A.; Gervet, T.; Romoff, J.; Batra, D.; Parikh, D.; Rabbat, M.; and Pineau, J. 2020. TarMAC: Targeted Multi-Agent Communication. arXiv:1810.11187.

Degris, T.; White, M.; and Sutton, R. S. 2013. Off-Policy Actor-Critic. arXiv:1205.4839.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. arXiv:1605.06676.

Foerster, J. N.; Song, F.; Hughes, E.; Burch, N.; Dunning, I.; Whiteson, S.; Botvinick, M.; and Bowling, M. 2019. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. arXiv:1811.01458.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2014. Optimized Product Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4): 744–755.

Gersho, A.; and Gray, R. M., eds. 1991. *Vector Quantization and Signal Compression*. Norwell, MA, USA: Kluwer Academic Publishers.

Goyal, A.; Didolkar, A.; Ke, N. R.; Blundell, C.; Beaudoin, P.; Heess, N.; Mozer, M.; and Bengio, Y. 2021a. Neural Production Systems. arXiv:2103.01937.

Goyal, A.; Didolkar, A.; Lamb, A.; Badola, K.; Ke, N. R.; Rahaman, N.; Binas, J.; Blundell, C.; Mozer, M.; and Bengio, Y. 2021b. Coordination Among Neural Modules Through a Shared Global Workspace. arXiv:2103.01197.

Goyal, A.; Lamb, A.; Hoffmann, J.; Sodhani, S.; Levine, S.; Bengio, Y.; and Schölkopf, B. 2019. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*.

Goyal, A.; Lamb, A.; Hoffmann, J.; Sodhani, S.; Levine, S.; Bengio, Y.; and Schölkopf, B. 2020. Recurrent Independent Mechanisms. arXiv:1909.10893.

Iqbal, S.; and Sha, F. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. arXiv:1810.02912.

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

Jiang, J.; and Lu, Z. 2018. Learning Attentional Communication for Multi-Agent Cooperation. arXiv:1805.07733.

Kipf, T.; van der Pol, E.; and Welling, M. 2020. Contrastive Learning of Structured World Models. arXiv:1911.12247.

Koller, D.; and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press. ISBN 0262013193.

Kong, X.; Xin, B.; Liu, F.; and Wang, Y. 2017. Revisiting the Master-Slave Architecture in Multi-Agent Deep Reinforcement Learning. arXiv:1712.07305.

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

Lamb, A.; He, D.; Goyal, A.; Ke, G.; Liao, C.-F.; Ravanelli, M.; and Bengio, Y. 2021. Transformers with Competitive Ensembles of Independent Mechanisms. *arXiv preprint arXiv:2103.00336*.

Lazaridou, A.; Peysakhovich, A.; and Baroni, M. 2017. Multi-Agent Cooperation and the Emergence of (Natural) Language. arXiv:1612.07182.

Liu, D.; Lamb, A. M.; Kawaguchi, K.; ALIAS PARTH GOYAL, A. G.; Sun, C.; Mozer, M. C.; and Bengio, Y. 2021. Discrete-valued neural communication. *Advances in Neural Information Processing Systems*, 34.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2020. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv:1706.02275.

MacQueen, J.; et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, 281–297. Oakland, CA, USA.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. arXiv:1611.00712.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. 2008. Optimal and Approximate Q-Value Functions for Decentralized POMDPs. *J. Artif. Int. Res.*, 32(1): 289–353.

Oord, A. v. d.; Vinyals, O.; and Kavukcuoglu, K. 2017. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*.

Peng, P.; Wen, Y.; Yang, Y.; Yuan, Q.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. arXiv:1703.10069.

Pesce, E.; and Montana, G. 2020. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning*, 109(9-10): 1727–1747.

Petrillo, A.; Salvi, A.; Santini, S.; and Valente, A. 2018. Adaptive multi-agents synchronization for collaborative driving of autonomous vehicles with multiple communication delays. *Transportation Research Part C: Emerging Technologies*, 86: 372–392.

Pipattanasomporn, M.; Feroze, H.; and Rahman, S. 2009. Multi-agent systems in a distributed smart grid: Design and implementation. In *2009 IEEE/PES Power Systems Conference and Exposition*, 1–8.

Ren, W.; and Sorensen, N. 2008. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56: 324–333.

Rolfe, J. T. 2017. Discrete Variational Autoencoders. arXiv:1609.02200.

Scardovi, L.; and Sepulchre, R. 2009. Synchronization in networks of identical linear systems. *Automatica*, 45(11): 2557–2562.

Singh, A.; Jain, T.; and Sukhbaatar, S. 2018. Learning when to Communicate at Scale in Multiagent Cooperative and Competitive Tasks. arXiv:1812.09755.

Sukhbaatar, S.; Szlam, A.; and Fergus, R. 2016. Learning Multiagent Communication with Backpropagation. arXiv:1605.07736.

Tan, D.-K. L.; Le, H.; Hoang, T.; Do, T.-T.; and Cheung, N.-M. 2018. DeepVQ: A Deep Network Architecture for Vector Quantization. In *CVPR Workshops*.

Tishby, N.; Pereira, F. C.; and Bialek, W. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. arXiv:1706.03762.

Wen, G.; Duan, Z.; Yu, W.; and Chen, G. 2012. Consensus in Multi-agent Systems with Communication Constraints. *International Journal of Robust and Nonlinear Control*, 22: 170 – 182.

You, K.; and Xie, L. 2011. Network Topology and Communication Data Rate for Consensusability of Discrete-Time Multi-Agent Systems. *IEEE Transactions on Automatic Control*, 56(10): 2262–2275.