Vector Quantization in the Brain: Grid-like Codes in World Models

Xiangyuan Peng[*] 1900012762peng@pku.edu.cn
Xingsi Dong[*†] dxs19980605@pku.edu.cn
Si Wu[†] siwu@pku.edu.cn

PKU-Tsinghua Center for Life Sciences, Academy for Advanced Interdisciplinary Studies.
IDG/McGovern Institute for Brain Research, Center of Quantitative Biology, Peking University.
School of Psychological and Cognitive Sciences,
Key Laboratory of Machine Perception (Ministry of Education).
[*]: Equal contribution.
[†]: Corresponding authors.

Abstract

We propose Grid-like Code Quantization (GCQ), a brain-inspired method for compressing observation–action sequences into discrete representations using grid-like patterns in attractor dynamics. Unlike conventional vector quantization approaches that operate on static inputs, GCQ performs spatiotemporal compression through an action-conditioned codebook, where codewords are derived from continuous attractor neural networks and dynamically selected based on actions. This enables GCQ to jointly compress space and time, serving as a unified world model. The resulting representation supports long-horizon prediction, goal-directed planning, and inverse modeling. Experiments across diverse tasks demonstrate GCQ's effectiveness in compact encoding and downstream performance. Our work offers both a computational tool for efficient sequence modeling and a theoretical perspective on the formation of grid-like codes in neural systems.

1 Introduction

VQ-VAE [1] introduces discrete latent variables into the autoencoding framework through vector quantization (VQ) [2], allowing the model to compress high-dimensional continuous inputs into discrete, tokenized representations. This capability has led to its widespread application across various domains, including images [3, 4], video [5], speech [6], actions [7], and multimodal data [8], demonstrating its versatility in handling complex, diverse inputs. The success of VQ-VAE underscores the utility of compressing inputs into reusable codes as a general computational strategy for preprocessing and organizing data across a wide range of tasks.

Biological systems face the similar challenge: how to process and represent high-dimensional, continuous inputs arising from multiple sensory and motor modalities. In parallel, the brain exhibits grid-like codes (GCs), which serve as general-purpose neural patterns for encoding information. GCs are extensively observed across various brain regions. Initially identified in the medial entorhinal cortex for spatial navigation [9], GCs have since been observed in the neocortex [10, 11, 12] and associated with representing abstract concepts beyond space, such as time and relational knowledge [10, 11, 13, 14]. This widespread neural activity is characterized by bump-like patterns, periodicity, and typically disentangled representations.

Building on this insight, we propose a brain-inspired VQ method, Grid-like Code Quantization (GCQ), which uses the principles of GCs to structure the codebook. Specifically, we use continuous attractor neural networks (CANNs) [15, 16, 17] to generate grid-like activity patterns, where each stable

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

state—bump—acts as a codeword. Due to the finite number of neurons, these bumps naturally form a discretized representation [18]. Unlike traditional VQ methods that use a static codebook, GCQ introduces an action-conditioned codebook: a dynamic set of codewords formed by CANN-generated bumps whose transitions are modulated by actions. This enables GCQ to perform quantization not on isolated observations, but on observation–action sequences, allowing the representations to capture temporal dependencies and behavioral context. Moreover, assigning distinct CANNs to different action types naturally yields disentangled representations, facilitating generalization and compositionality.

The overall GCQ pipeline follows an encoder–quantizer–decoder architecture, adapted for action-conditioned sequence compression (Fig. 2). Specifically, the model processes an observation–action sequence, where the action sequence is used to construct an action-conditioned codebook, and the observation sequence is passed through the encoder to produce a corresponding latent sequence. This latent sequence is then quantized via template matching with the action-conditioned codebook. The matched codewords are passed to the decoder, which reconstructs the original observation sequence. Since the codebook is fixed, training requires only a commitment loss and a reconstruction loss. To enable gradient flow through the discrete quantization step, we use a straight-through estimator (STE).

GCQ is a dynamic compression approach that operates on observation–action sequences, and therefore serves as a form of world model [19, 20]. Unlike prior world models that rely on a two-stage design to separately compress space and time—typically using models like VQ-VAE for static spatial observations and autoregressive models [21] for temporal dynamics—GCQ performs spatial and temporal compression jointly.

In summary, our contributions are as follows:

- To the best of our knowledge, GCQ is the first model to unify spatial and temporal compression through an action-conditioned quantization process. This enables direct compression of observation–action sequences, offering an integrated alternative to conventional two-stage world models. (Sec. 4)
- GCQ's spatiotemporal compression yields a cognitive map, which supports long-horizon prediction, goal-directed planning, and the derivation of an inverse model. In particular, goal-directed planning becomes computationally simple, as it reduces to finding a sequence of valid bump transitions on the map. (Sec. 5).
- GCQ offers insights into the formation of GCs in the brain, enhancing our understanding of neural representations (Sec. 6).

2 Related Work

VQ methods Vanilla VAEs [22] often suffer from posterior collapse in their latent spaces when compressing high-dimensional data [23], impairing downstream tasks. VQ-VAEs [1] address this by enforcing a structured latent space through discretization. Due to their superior compression efficiency and tokenization paradigm, VQ has become a standardized module in single-modal preprocessing pipelines in machine learning [3, 4]. In multimodal settings, these compressed tokens further act as a universal interface across modalities [5]. Meanwhile, numerous studies have proposed diverse codebook designs to enhance compression rates [24, 25, 26]. Unlike most learnable codebooks, FSQ uses a predefined codebook. Similarly, our GCQ utilizes a fixed codebook derived from continuous attractor dynamics. Critically, our method diverges from conventional VQ approaches by performing sequence-to-sequence template matching rather than single-frame matching.

World models [19, 20] provide a framework for predicting future observations conditioned on actions. Most world models based on encoder–decoder architectures first compress observations using a VAE, and then model temporal dynamics in the latent space using temporal predictors such as RNNs [27, 28], Transformers [29], S4 models [30], or continuous Hopfield networks [31]. These approaches typically follow a two-stage design, with spatial and temporal compression handled separately. In contrast, GCQ is also an encoder–decoder world model, but it performs spatial and temporal compression jointly. There also exist decoder-only world models [32] that skip explicit compression and directly predict future observations. However, these models often struggle with planning due to the high computational cost of operating in the raw observation space. GCQ, by

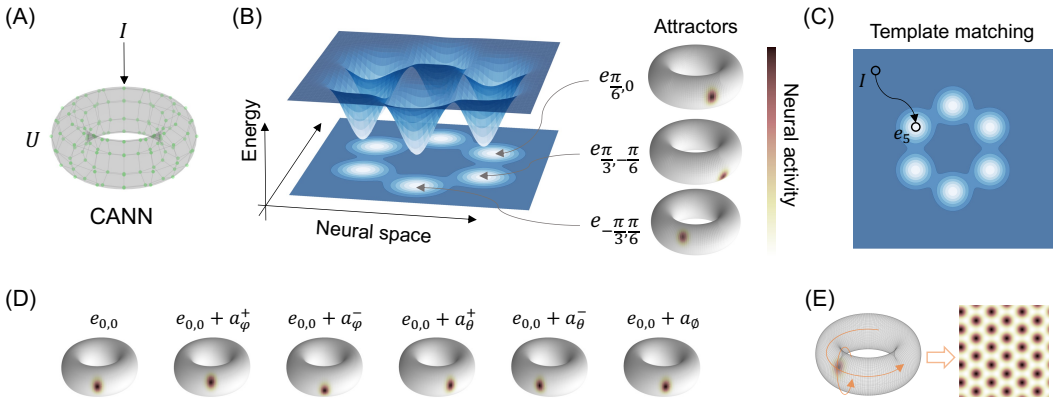

Figure 1: (A) Schematic of a CANN: Each green dot represents a neuron uniformly distributed on a torus. The neurons receive external input I. (B) Energy landscape of CANN dynamics: Each local minimum in the energy landscape corresponds to an attractor state, which manifests as a 2D Gaussian bump on the torus. (C) Template matching via CANN dynamics: The CANN inherently performs template matching between the external input I and its attractor states. The input I is matched to the attractor that maximizes their inner product. (D) Attractor transition: Under four distinct actions, the attractor initially at position $(0, 0)$ stabilizes to four new attractor states. (E) Due to the periodic boundary conditions of the CANN, bump movements along the two axes naturally form grid-like patterns.

compressing both space and time into a compact latent representation, enables more efficient planning and inference.

Cognitive map with CANNs Unlike classical attractor networks [33]—which store discrete, unstructured patterns—CANNs encode structured patterns organized by metric relationships. This geometric regularity facilitates flexible state transitions through predefined operators [34], enabling operations like metric-based navigation and relational inference. Recent advances [35] have harnessed predefined CANNs as structured latent states for representation learning, empirically validating their ability to model neural population dynamics. Further work [36] proposes that structured latent spaces can map biologically to the entorhinal-hippocampal loop, a core circuit for spatial and episodic memory. However, existing implementations rely on biologically constrained online learning, which limits scalability. Our GCQ framework uses offline learning, enhancing parallelism and enabling application to large-scale datasets.

3 CANNs and Template Matching

In this section, we will briefly introduce CANNs, explain how they can form bumps as attractor states. In parallel, for VQ, the latent state obtained by the encoder must undergo template matching with codewords. We will demonstrate that CANNs inherently implement template matching between representations and bump states through their intrinsic dynamics. Finally, we will show how transitions between distinct attractor states can be mediated by actions.

GCs can naturally be modeled by bumps in CANNs (Fig. 1E). The formation of CANNs does not require complex optimization but relies on translation-invariant connectivity and periodic boundary conditions. CANNs have been widely used as canonical models to elucidate the encoding of features in neural systems, including, for example, the encoding of orientation [37], head direction [38] and spatial location [17, 39]. CANNs can be expressed through various mathematical formulations. Here, we adopt a relatively concise form [16] to demonstrate their principles. We consider N^2 neurons distributed on a toroidal ($S^1 \times S^1$) surface. These neurons are indexed by their positions on the torus $\theta \in \{\theta_i\}_{i=1}^N$ and $\varphi \in \{\varphi_j\}_{j=1}^N$, where θ_i and φ_j are uniformly distributed over $(-\pi, \pi]$ (Fig. 1A). Let $U_{\theta,\varphi}(t)$ and $r_{\theta,\varphi}(t)$ denote the synaptic input and firing rate, respectively, of the neuron located

at (θ, φ) at time t. The dynamics of the CANN are governed by:

$$\tau \frac{\partial U_{\theta,\varphi}(t)}{\partial t} = -U_{\theta,\varphi}(t) + \rho \sum_{\theta',\varphi'} W_{\theta,\varphi}(\theta', \varphi') r_{\theta',\varphi'}(t) + I_{\theta,\varphi}(t), \tag{1}$$

where τ is the synaptic time constant and ρ is the neuronal density. $W_{\theta,\varphi}(\theta', \varphi')$ is the recurrent neuronal connections weights between neuron (θ, φ) and neuron (θ', φ'),

$$W_{\theta,\varphi}(\theta', \varphi') = \frac{J}{2\pi a^2} \exp\left[-\frac{||\theta - \theta'||_S^2 + ||\varphi - \varphi'||_S^2}{2a^2}\right]. \tag{2}$$

The norm $\|\cdot\|_S$ denotes the shortest path between two points on the circle, ensuring periodic boundary and translation-invariant conditions. The parameters J and a control the strength and width of the Gaussian connectivity, respectively. The nonlinear relationship between the firing rate $r_{\theta,\varphi}(t)$ and the synaptic input $U_{\theta,\varphi}(t)$ is implemented by divisive normalization, which is written as,

$$r_{\theta,\varphi}(t) = \frac{U_{\theta,\varphi}^2(t)}{1 + k\rho \sum_{\theta',\varphi'} W_{\theta,\varphi}(\theta', \varphi') U_{\theta',\varphi'}^2(t)}, \tag{3}$$

where k controls the normalization strength. In reality, divisive normalization could be implemented by shunting inhibition [40].

Previous studies [16, 41] have established that the CANN dynamics governed by Eq. (1) possess N^2 stationary states (attractors) when the external input $I_{\theta,\varphi}(t) = 0$ (Fig. 1B). Each state corresponds to a 2D Gaussian bump on the torus, centered at coordinates (θ, φ), with the firing rate of the neuron at position (θ', φ') given by:

$$e_{\theta,\varphi}(\theta', \varphi') = A \exp\left[-\frac{\|\theta - \theta'\|_S^2 + \|\varphi - \varphi'\|_S^2}{2a^2}\right], \tag{4}$$

where $A = \left[1 + (1 - 32\pi a^2 k / J^2 \rho)^{1/2}\right] / (4\pi a^2 k \rho)$ is the amplitude. When $I_{\theta,\varphi}(t)$ is a constant input, prior work [42] demonstrated that after its removal, the network converges to an attractor determined by:

$$\theta^*, \varphi^* = \max_{\theta,\varphi} \sum_{\theta',\varphi'} e_{\theta,\varphi}(\theta', \varphi') I_{\theta',\varphi'}, \tag{5}$$

which demonstrates that CANN dynamics effectively perform template matching between the input I and the N^2 attractors according to their inner product. (Fig. 1C).

The bump in CANNs exhibit high mobility, enabling controlled movement through mechanisms such as: anti-symmetric connections [43], negative feedback [44, 43], velocity neurons [45]. Such bump displacements correspond to transitions between attractor states. For the toroidal CANN described above, each attractor can undergo local two-dimensional displacements in the θ, φ plane. We define two orthogonal action bases aligned with the θ and φ axes (Fig. 1D),

$$a_\theta^{\pm} = e_{\theta \pm \Delta\theta, \varphi} - e_{\theta,\varphi}, \quad a_\varphi^{\pm} = e_{\theta, \varphi \pm \Delta\varphi} - e_{\theta,\varphi}. \tag{6}$$

where $\Delta\varphi$ and $\Delta\theta$ denote a small displacement step.

4 Grid-like Code Quantization

In this section, we first introduce the action-conditioned codebook in GCQ and the template matching process for sequences. We then describe how GCQ enables bidirectional mapping between real-world actions and latent transitions, and propose a greedy operator for measuring distances on the cognitive map to support inverse modeling and planning.

4.1 Action-conditioned codebook and sequence matching

We first introduce the key difference between GCQ and VQ from a high-level perspective. In the VQ method, the encoder first compresses the observation o into s, which is then matched to the closest codes in the codebook through template matching, producing $\hat{s}$. The decoder then reconstructs $\hat{o}$ from $\hat{s}$. In GCQ, the input consists of an action-observation sequence $\{o_1, a_1, o_2, a_2, ..., o_n\}$. The encoder

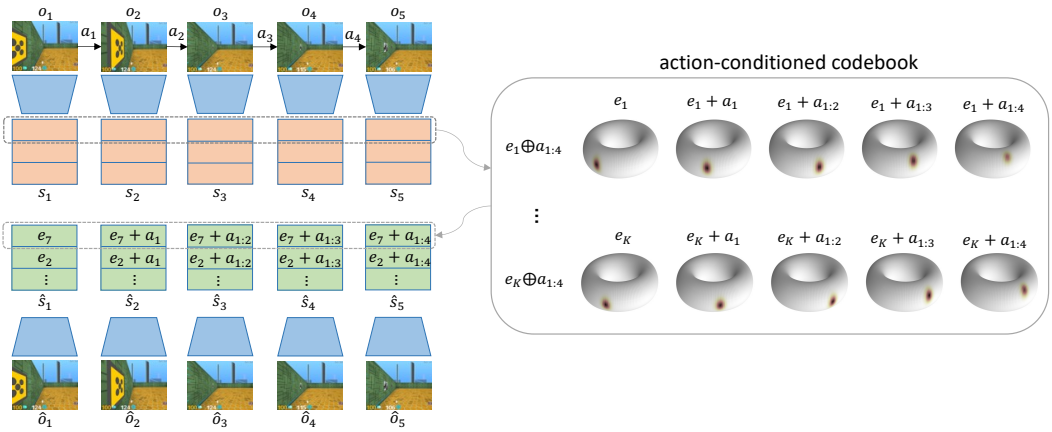

Figure 2: Schematic of GCQ: The action-observation sequence is encoded by the encoder into a latent state composed of $m = 3$ codes. Through sequence template matching with the action-conditioned codebook, the decoder reconstructs the predicted observations. The gray arrows indicate the template matching process, and the gray dashed boxes represent the matching targets.

compresses the observation sequence $o_{1:n} = \{o_1, o_2, ..., o_n\}$ into $s_{1:n}$, which is then matched to the closest codes in the action-conditioned codebook via template matching, yielding $\hat{s}_{1:n}$. The decoder then reconstructs $\hat{o}_{1:n}$ from $\hat{s}_{1:n}$.

In GCQ, each code corresponds to an attractor in the CANN (Fig. 1B). In the previous section, we used θ and φ to index different attractors; for simplicity, we will now use natural numbers as attractor indices. Each code consists of d neurons, and the codebook contains K attractors. A simple implementation sets $K = d$, where each attractor's center coincides with a single neuron. Alternatively, we can set $K > d$, causing some attractor centers to fall between two neurons. In practice, different combinations of K and d can be selected. The state representation $s_i \in \mathbb{R}^{m \times d}$, meaning that s_i is composed of m codes.

Additionally, we manually define a mapping between the action sequence $a_i \in \mathcal{A}$ from the dataset and the action combinations applied to the CANNs. For notational simplicity, we hereafter use a_i to refer to an action in either the original space or the CANN space. In the latter context, $a_i \in \mathbb{R}^{m \times d}$ represents the composite action over m bumps, with its component $a_i^j \in \mathbb{R}^d$ denoting the action applied to the j-th bump in Eq.(6). Each CANN supports five distinct actions, resulting in up to 5^m possible action combinations across m CANNs. Since this mapping is injective, the discrete action space must satisfy $|\mathcal{A}| \leq 5^m$. For continuous actions, a CANN can define transitions in two directions, imposing the constraint $\dim(\mathcal{A}) \leq 2m$.

After establishing the mapping, we quantize the latent representation $s_{1:n} = \{s_{1:n}^j\}_{j=1}^m$. This representation consists of a set of m parallel sequences, where each $s_{1:n}^j$ corresponds to a sequence from one of the m CANNs (as depicted by the dashed lines in Fig. 2). The quantization process is performed independently for each of these m sequences. For each latent sequence $s_{1:n}^j$, we perform a template matching procedure. This involves comparing $s_{1:n}^j$ against a set of K candidate trajectories. Each candidate trajectory is generated by applying the known action sequence $a_{1:n-1}^j$ to a base bump state e_i. We denote this operation as:

$$e_i \oplus a_{1:n-1}^j = \{e_i, e_i + a_1^j, \ldots, e_i + a_{1:n-1}^j\}, \tag{7}$$

where $e_i + a_{1:n-1}^j = e_i + \sum_{t=1}^{n-1} a_t^j$. The index k of the best-matching codeword for the j-th latent sequence is found by minimizing a distance metric (e.g., the L2 norm) between the latent sequence and each of the K candidate trajectories:

$$k_j = \arg \min_{i \in \{1,..,K\}} ||s_{1:n}^j - (e_i \oplus a_{1:n-1}^j)|| \tag{8}$$

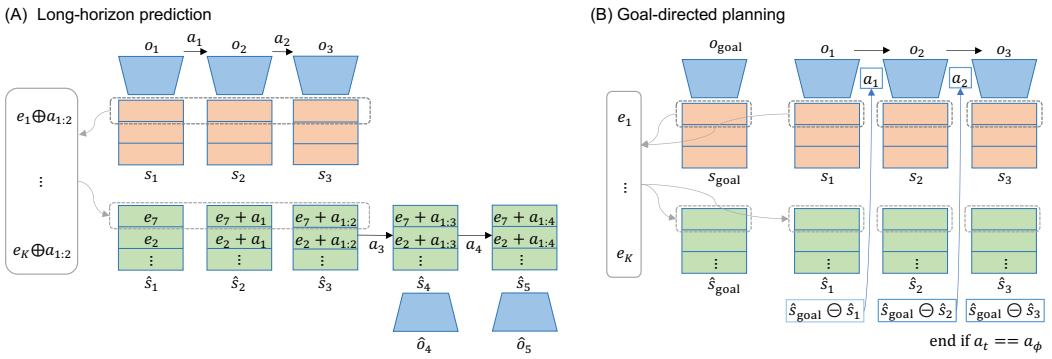

Figure 3: (A) Schematic of long-horizon prediction. The figure illustrates the process of initializing with a sequence of length 3 and predicting two future observations. (B) Schematic of goal-directed planning. Some gray arrows are omitted for clarity. The blue arrows represent the mapping from actions in the latent space to real agent actions. Through iterative action generation, environment interaction, and observation, the agent continues until it outputs a no-op action $a_\emptyset$, indicating that the goal has been reached.

Finally, the quantized sequence $\hat{s}^j_{1:n}$ is constructed using this optimal codeword e_{k_j}. The complete quantized representation $\hat{s}_{1:n}$ is the collection of these individually quantized sequences:

$$\hat{s}^j_{1:n} = e_{k_j} \oplus a^j_{1:n-1}, \quad \text{and} \quad \hat{s}_{1:n} = \{\hat{s}^j_{1:n}\}_{j=1}^m \tag{9}$$

When computing the loss in GCQ using backpropagation (BP), we adopt the same straight-through estimator (STE) as in the VQ method, copying gradients from the decoder input to the encoder output to enable gradient flow to the encoder. GCQ uses two loss terms: a reconstruction loss and a commitment loss:

$$L = ||o_{1:n} - \hat{o}_{1:n}||^2 + \beta \left\| s_{1:n} - \text{sg}\left[\hat{s}_{1:n}\right] \right\|^2 \tag{10}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operation and β adjusts the strength of the commitment loss.

In GCQ, the encoder and decoder are not designed in the same way as in conventional VQ models. Traditional VQ architectures often use ResNet-based building blocks, which provide each code with only a limited receptive field. As a result, modifying a single code typically leads to only local changes in the reconstructed observation. In contrast, GCQ assigns each code to an action, and altering the action can result in global changes to the observation. This necessitates that each code has access to global information during encoding and decoding. To address this, we explore three architectural variants for the encoder and decoder: (1) ResNet followed by a fully connected layer, (2) ViT [46], and (3) a hybrid of ResNet and ViT. Among these, ViT achieves the best trade-off in terms of parameter efficiency, training stability, and overall performance (Table.1).

4.2 Operations on cognitive map

GCQ uses a structured latent space, allowing an agent's actions in the real environment to correspond to simple movements of bumps within the latent space. In effect, GCQ constructs a space defined by bump dynamics, which can be interpreted as a cognitive map. By establishing a mapping between observations and this map, actions in the real space can be projected onto the map to determine position changes, and conversely, movements within the map can be mapped back to real-space actions. This bidirectional mapping enables GCQ to support both inverse modeling and goal-directed planning. Specifically, to compute the distance between two states s_i and s_j, we define an operation on the cognitive map. Since bump movements are action-driven and only valid actions produce feasible transitions, we introduce the following operation:

$$s_i \ominus s_j = \arg\min_{a \in \mathcal{A}} |s_j + a - s_i|. \tag{11}$$

This operation represents a greedy step: it selects the best valid action a that moves s_j one step closer to s_i.

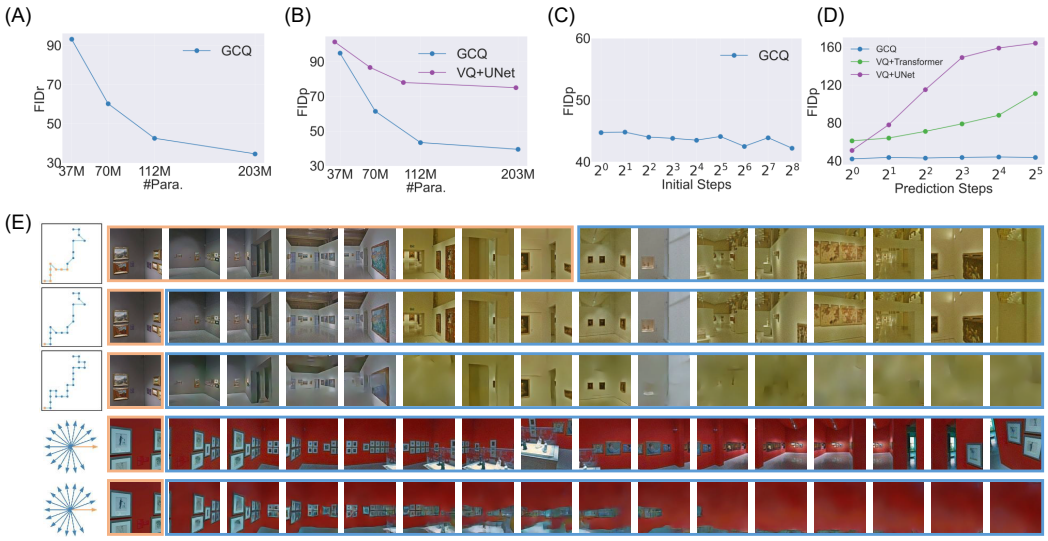

Figure 4: (A)(B) Reconstruction FID and prediction FID for GCQ and VQ+UNet across different model sizes. (C) Prediction FID of GCQ varies with changes in the initialization length. (D) Prediction FID of GCQ (#Para:112M), VQ+UNet (96M) and VQ+Transformer (121M) changes as the prediction length increases. (E) Predictions on GSV dataset. The first patch in each row represents the trajectory drawn by the action. The first three rows correspond to movement actions, while the last two rows correspond to rotation actions. In practice, different actions are encoded within a single GCQ, enabling their use. For visualization convenience, they are plotted separately here. Rows 1, 2, and 4 show GCQ predictions with different initialization lengths (orange frames), predicting subsequent observations (blue frames). Rows 3 and 5 show VQ+UNet predictions under the same conditions. As the prediction length increases, the images become blurry.

5 Experiment

As a spatiotemporal compression model, GCQ is first evaluated in ablation studies to demonstrate its ability to compress and reconstruct observations. We then show that GCQ, when used as a world model, supports long-horizon prediction, goal-directed planning, and inverse modeling. Compared to traditional two-stage models, GCQ exhibits superior performance in long-range prediction tasks.

Datasets. We evaluate GCQ on four datasets, all of which contain image-based observations. The 2DMaze [47] dataset is a virtual environment where actions correspond to the agent's movements. Each observation contains a full view of the maze, providing complete information. The Google Street View (GSV) dataset represents real-world environments with partial observations; the actions include both translational movements and rotational head turns in two directions. In the MPI3D [48] and 3DShapes [49] datasets, actions are defined as abstract feature-level changes.

Baselines. We compare GCQ with traditional two-stage world models. VQ-VAE is used in the first stage for spatial compression. The codebook size in GCQ and VQ-VAE is kept the same for a fair comparison. For modeling temporal relationships, we use a UNet that predicts the next latent state s_{t+1} based on the current latent state s_t and action a_t. We refer to this baseline as 'VQ+UNet.' For action embedding in the UNet, we follow the approach from LAPO [7]. To further model temporal dependencies, we also adopt a Transformer-based architecture following TransDreamer [29]. We refer to this baseline as 'VQ+Transformer.'

Evaluation Metrics. To evaluate the quality of the model-generated observations, we report peak signal-to-noise ratio (PSNR) for pixel-level reconstruction fidelity, and use the Fréchet Inception Distance (FID) [50] to assess the quality of generated images.

Ablations We first conducted ablation experiments on the GSV dataset. Table 1 presents the performance of three different encoder-decoder network building blocks. It can be observed that the ViT and Hybrid models achieve better performance with fewer parameters. However, during the

experiments, we found that the Hybrid model was less stable in training and converged more slowly than ViT. Therefore, unless otherwise specified, all subsequent experiments utilized the ViT-structured network. The GCQ exhibits scalability with model size similar to VQ+UNet, both in reconstruction and prediction. (Fig. 4A,B).

Model type	Image size	#Para.	FIDr↓	FIDp↓	PSNRr↑	PSNRp↑
Resnet	$3 \times 80 \times 40$	330M	48.05	48.57	25.70	25.59
Hybrid	$3 \times 80 \times 40$	64M	21.29	22.31	29.07	28.54
	$3 \times 128 \times 128$	140M	41.55	41.91	26.31	25.59
ViT	$3 \times 80 \times 40$	90M	13.27	13.92	31.32	31.34
	$3 \times 128 \times 128$	112M	42.56	43.41	27.82	27.77

Table 1: Model comparisons on the Street View dataset. FIDr, FIDp, PSNRr, and PSNRp represent the FID and PSNR scores for reconstruction and prediction, respectively. ↓ and ↑ indicate that lower or higher values are better. The ResNet model was not trained on higher resolution images because its architecture includes a fully connected layer after the convolutional backbone, resulting in an excessively large number of parameters.

We also make the bump-like codes in the codebook learnable by using the following loss function:

$$L = ||o_{1:n} - \hat{o}_{1:n}||^2 + \beta \left\| s_{1:n} - \text{sg}\left[\hat{s}_{1:n}\right] \right\|^2 + \gamma \left\| \text{sg}\left[s_{1:n}\right] - \hat{s}_{1:n} \right\|^2 \tag{12}$$

However, our experiments show that making the codes learnable actually degrades performance. We attribute this to the fact that, unlike the relatively simple codes in VQ, our codes exhibit more complex dynamic relationships. Allowing the codes themselves to be trained may therefore reduce training stability.

Model	FIDp↓	PSNRp↑
GCQ (fixed)	43.41	27.77
GCQ (learnable)	47.76	24.48

Table 2: Effect of learnable vs. fixed codes in GCQ.

Long-horizon prediction. After being initialized with an observation-action sequence, GCQ can perform actions directly in the latent space to predict future observations (Fig. 3A). Notably, its prediction performance remains stable regardless of the length of the initialization sequence (Fig. 4C; Fig. 4E, rows 1–2). As shown in Fig. 4D, the performance of the VQ method degrades as the prediction horizon increases, whereas GCQ maintains robust predictive quality due to its stable latent structure (Fig. 4E, rows 2–3). This is a key advantage of GCQ: by constructing a consistent cognitive map, it effectively addresses the instability issues commonly seen in current world models [51]—such as inaccurate predictions after completing a full rotation in the environment (Fig. 4E, rows 4–5). GCQ also demonstrates strong zero-shot prediction capabilities on relatively simple datasets. As shown in Fig. 5, rows 1–2, the model produces reasonable predictions in environments it has never encountered during training. Furthermore, by treating abstract feature transitions as a form of action, GCQ can also be used to predict observation changes driven by abstract-level variations. These predictions likewise exhibit long-range stability (Fig. 5, rows 3–4).

Goal-directed planning. Given a goal and an initial position, the GCQ can utilize the distance in the cognitive map to generate the most desirable action for the current step. After executing the action, a new observation is obtained, and this process is iterated, continuously reducing the distance to the goal in the cognitive map until the goal is reached (Fig. 3B, Fig. 6 rows, 1–2). The computation of the action at each step is of constant complexity.

Inverse model. Given a sequence of observations, the GCQ can first map them onto the cognitive map and then use goal-directed planning to determine the action or sequence of actions between adjacent observations, thus implementing the inverse model. The corresponding action sequence can be applied to the latent representation of another observation, and using the prediction capability, the generated sequence under this set of actions can be obtained (Fig. 6 rows, 3–4).

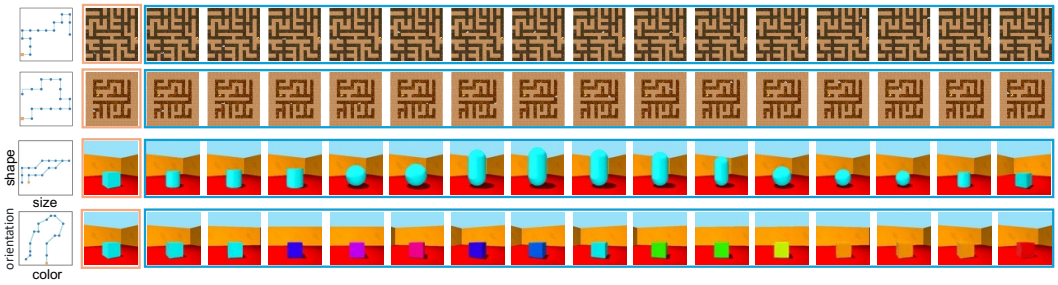

Figure 5: With the same setup as Fig. 4E, Rows 1–2: Prediction on the 2DMaze dataset. Rows 3–4: Prediction on the 3DShapes dataset.

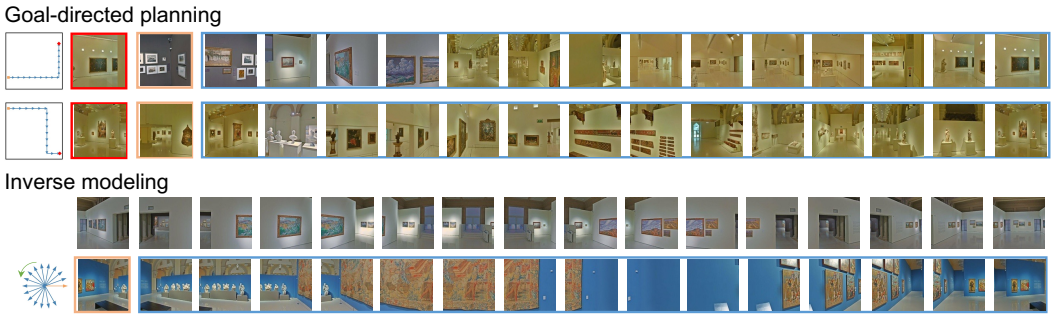

Figure 6: Rows 1–2: Goal-directed planning. The first patch shows the trajectory after planning, with orange indicating the starting point, red indicating the endpoint, and blue representing the planned route. The subsequent red-framed patches represent the endpoint, orange-framed patches represent the starting point, and blue-framed patches represent intermediate observations encountered during the process. Rows 3–4: Inverse modeling. The top row displays the given observation sequence. The bottom row starts with the first patch showing the action trajectory inferred from the observation sequence, with orange indicating the given initial observation, followed by the sequence generated based on the action trajectory.

6 Discussion

In this work, we introduced GCQ, a brain-inspired framework for compressing observation–action sequences into discrete, structured representations. GCQ uses continuous attractor dynamics to generate grid-like codewords, and selects them in an action-conditioned manner to capture both spatial and temporal dependencies. This spatiotemporal quantization process produces compact latent representations that serve as cognitive maps, enabling long-horizon prediction, goal-directed planning, and inverse modeling. Our experiments demonstrate that GCQ supports generalization across tasks while offering interpretability through its structured latent space.

Insights for Neuroscience. Beyond its practical performance, GCQ also offers a new computational hypothesis for the emergence of GCs in the brain. Traditionally, the formation of neurons with structured tuning properties was approached through handcrafted models [52], which provided only limited explanatory power. In contrast, the machine learning paradigm offers a data-driven framework: artificial neural networks are optimized to perform cognitive tasks, and their internal representations are analyzed to reveal emergent coding principles. Following this approach, prior studies have shown that GCs can arise when networks are trained to perform path integration under biologically constraints [53, 54, 55]. Subsequent work has emphasized the importance of predictive rather than reconstructive objectives [56] and extended the analysis to more general frameworks such as world models and predictive learning [57]. Efforts to induce disentangled representations through architectural or loss function constraints have further refined these insights [58, 59]. However, the robustness of grid-like pattern emergence remains debated [60], with some studies [61] suggesting that specific architectural features (e.g., one-hot inputs) are necessary.

Recent developmental findings add a new dimension to this discussion. Experiments show that toroidal activity patterns emerge in the medial entorhinal cortex even before sensory experience [62]. Intriguingly, such toroidal structures can be naturally modeled by bump attractors in CANNs. This suggests that the brain may possess preconfigured low-dimensional structures capable of bump activity, even prior to learning. Consistent with this, recent work has argued that GCs likely emerge from internal CANN mechanisms rather than from purely feedforward architectures [63].

This leads us to a novel hypothesis inspired by GCQ: GCs may arise not from optimizing networks, but from learning to map sensory experience onto a set of preexisting bump-based activity patterns. In GCQ, the codebook is defined by CANN-generated bumps before learning begins. The learning process then consists of associating observation–action sequences with combinations of these fixed codewords. Similarly, we speculate that the brain may use a fixed set of toroidal patterns—produced by CANNs—as a biological codebook. Through experience, the brain learns to map external sensory inputs onto these internal structures, endowing them with meaning and interpretability, allowing for the decoding of grid-like patterns. This perspective suggests a unified model of how the brain may simultaneously achieve compression and semantic organization of sensory information.

Static Setting. We also evaluated GCQ in a static setting, where the sequence length is 1. In this case, we compared GCQ with VQ-VAE on ImageNet [64], finding that GCQ suffered minimal performance degradation. This suggests that the use of a fixed codebook does not significantly harm performance on static tasks. For further details, refer to Appendix A.

Scalability of Action. Our current work utilizes 2D attractors, where each has 5 potential transitions (four shifts and one stationary). With such m CANNs, the model can represent 5^m distinct actions. If we use P-dimensional attractors, the number of states per CANN will become $2P+1$, yielding a total action space of $(2P+1)^5$. Therefore, GCQ can be scaled to higher-dimensional action spaces by adjusting both m and P. For the experiments in this paper, which involve relatively low-dimensional actions, 2D attractors are sufficient.

Future Work. A promising direction for advancing GCQ lies in enabling the encoder and decoder to process entire sequences holistically, rather than treating each sequence element independently. Incorporating ViTs with spatial-temporal attention could serve as an effective approach toward this goal. Moreover, scaling GCQ to larger and more diverse datasets would facilitate a deeper investigation into its generalization capabilities and robustness across a broader range of tasks and domains.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. T2421004 to S.W.), the Science and Technology Innovation 2030-Brain Science and Brain-inspired Intelligence Project (no. 2021ZD0200204, S.W.).

References

[1] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[2] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[4] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22563–22575, 2023.

[6] Tuan Vu Ho, Quoc Huy Nguyen, Masato Akagi, and Masashi Unoki. Vector-quantized variational autoencoder for phase-aware speech enhancement. 2022.

[7] Dominik Schmidt and Minqi Jiang. Learning to act without actions. *arXiv preprint arXiv:2312.10812*, 2023.

[8] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024.

[9] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[10] Christian F Doeller, Caswell Barry, and Neil Burgess. Evidence for grid cells in a human memory network. *Nature*, 463(7281):657–661, 2010.

[11] Alexandra O Constantinescu, Jill X O'Reilly, and Timothy EJ Behrens. Organizing conceptual knowledge in humans with a gridlike code. *Science*, 352(6292):1464–1468, 2016.

[12] Joshua Jacobs, Christoph T Weidemann, Jonathan F Miller, Alec Solway, John F Burke, Xue-Xin Wei, Nanthia Suthana, Michael R Sperling, Ashwini D Sharan, Itzhak Fried, et al. Direct recordings of grid-like neuronal activity in human spatial navigation. *Nature neuroscience*, 16(9):1188–1190, 2013.

[13] KiJung Yoon, Sam Lewallen, Amina A Kinkhabwala, David W Tank, and Ila R Fiete. Grid cell responses in 1d environments assessed as slices through a 2d lattice. *Neuron*, 89(5):1086–1099, 2016.

[14] Dmitriy Aronov, Rhino Nevers, and David W Tank. Mapping of a non-spatial dimension by the hippocampal–entorhinal circuit. *Nature*, 543(7647):719–722, 2017.

[15] Shun-ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2):77–87, 1977.

[16] Si Wu, Kosuke Hamaguchi, and Shun-ichi Amari. Dynamics and computation of continuous attractors. *Neural Computation*, 20(4):994–1025, 2008.

[17] Yoram Burak and Ila R Fiete. Accurate path integration in continuous attractor network models of grid cells. *PLoS computational biology*, 5(2):e1000291, 2009.

[18] Marcella Noorman, Brad K Hulse, Vivek Jayaraman, Sandro Romani, and Ann M Hermundstad. Maintaining and updating accurate internal representations of continuous variables with a handful of neurons. *Nature Neuroscience*, 27(11):2207–2217, 2024.

[19] Jiirgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent n eu al networks for dynamic reinforcement learning and planning in non-stationary environments, 1990.

[20] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1), 2022.

[21] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] Ananya Harsh Jha, Saket Anand, Maneesh Singh, and VS Rao Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–820, 2018.

[24] Sander Dieleman, Charlie Nash, Jesse Engel, and Karen Simonyan. Variable-rate discrete representation learning. *arXiv preprint arXiv:2103.06089*, 2021.

[25] Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*, 2018.

[26] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

[27] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

[28] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[29] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[30] Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. *arXiv preprint arXiv:2403.04253*, 2024.

[31] James Whittington, Timothy Muller, Shirely Mark, Caswell Barry, and Tim Behrens. Generalisation of structural knowledge in the hippocampal-entorhinal system. *Advances in neural information processing systems*, 31, 2018.

[32] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. *arXiv preprint arXiv:2412.03572*, 2024.

[33] S-I Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, 100(11):1197–1206, 1972.

[34] Junfeng Zuo, Ying Nian Wu, Si Wu, and Wenhao Zhang. The motion planning neural circuit in goal-directed navigation as lie group operator search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[35] Xingsi Dong, Xiangyuan Peng, and Si Wu. Predictive learning in energy-based models with attractor structures. *arXiv preprint arXiv:2501.13997*, 2025.

[36] Sarthak Chandra, Sugandha Sharma, Rishidev Chaudhuri, and Ila Fiete. Episodic and associative memory from spatial scaffolds in the hippocampus. *Nature*, pages 1–13, 2025.

[37] R Ben-Yishai, R Lev Bar-Or, and H Sompolinsky. Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences*, 92(9):3844–3848, 1995.

[38] Kechen Zhang. Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *The Journal of Neuroscience*, 16(6):2112–2126, 1996.

[39] Bruce L McNaughton, Francesco P Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678, 2006.

[40] Simon J Mitchell and R Angus Silver. Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron*, 38(3):433–445, 2003.

[41] C. C Alan Fung, K. Y. Michael Wong, and Si Wu. A moving bump in a continuous manifold: A comprehensive study of the tracking dynamics of continuous attractor neural networks. *Neural Computation*, 22(3):752–792, 2010.

[42] Sophie Deneve, Peter E Latham, and Alexandre Pouget. Reading population codes: a neural implementation of ideal observers. *Nature Neuroscience*, 2(8):740–745, 1999.

[43] Yuanyuan Mi, CC Fung, KY Wong, and Si Wu. Spike frequency adaptation implements anticipative tracking in continuous attractor neural networks. *Advances in neural information processing systems*, 27, 2014.

[44] K Wong, He Wang, Si Wu, and Chi Fung. Attractor dynamics with synaptic depression. *Advances in Neural Information Processing Systems*, 23, 2010.

[45] Wenhao Zhang, Ying Nian Wu, and Si Wu. Translation-equivariant representation in recurrent networks with a continuous manifold of attractors. *Advances in Neural Information Processing Systems*, 35:15770–15783, 2022.

[46] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[47] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

[48] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[49] Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

[50] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[51] Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: Rnns, transformers, and s4. *Advances in Neural Information Processing Systems*, 36:72904–72930, 2023.

[52] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[53] Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.

[54] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018.

[55] Ben Sorscher, Gabriel Mel, Surya Ganguli, and Samuel Ocko. A unified theory for the origin of grid cells through the lens of pattern formation. *Advances in neural information processing systems*, 32, 2019.

[56] Stefano Recanatesi, Matthew Farrell, Guillaume Lajoie, Sophie Deneve, Mattia Rigotti, and Eric Shea-Brown. Predictive learning as a network mechanism for extracting low-dimensional latent space representations. *Nature communications*, 12(1):1417, 2021.

[57] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[58] James CR Whittington, Will Dorrell, Surya Ganguli, and Timothy EJ Behrens. Disentanglement with biological constraints: A theory of functional cell types. *arXiv preprint arXiv:2210.01768*, 2022.

[59] Yusi Chen, Huanqiu Zhang, Mia Cameron, and Terrence Sejnowski. Predictive sequence learning in the hippocampal formation. *Neuron*, 112(15):2645–2658, 2024.

[60] Ben Sorscher, Gabriel C Mel, Aran Nayebi, Lisa Giocomo, Daniel Yamins, and Surya Ganguli. When and why grid cells appear or not in trained path integrators. *bioRxiv*, pages 2022–11, 2022.

[61] Rylan Schaeffer, Mikail Khona, and Ila Fiete. No free lunch from deep learning in neuroscience: A case study through models of the entorhinal-hippocampal circuit. *Advances in neural information processing systems*, 35:16052–16067, 2022.

[62] M. GUARDAMAGNA, E. HERMANSE, J. CARPENTER, C. LYKKEN, B. DUNN, E. I. MOSER, and M.-B. MOSER. Experience-independent emergence of toroidal and ring manifolds in the entorhinal cortex. *Neuroscience Meeting Planner. Chicago, IL. : Society for Neuroscience, 2024. Online*, (PSTR190.07. 2024), 2024.

[63] Richard J Gardner, Erik Hermansen, Marius Pachitariu, Yoram Burak, Nils A Baas, Benjamin A Dunn, May-Britt Moser, and Edvard I Moser. Toroidal topology of population activity in grid cells. *Nature*, 602(7895):123–128, 2022.

[64] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Section3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 5 and the code in supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have included the code for reproducing the main results in supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 5.

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We used fid and PSNR to evaluate our work.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section B

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the related paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: See the documentation in supplementary material.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLM is used only for writing.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Static Setting

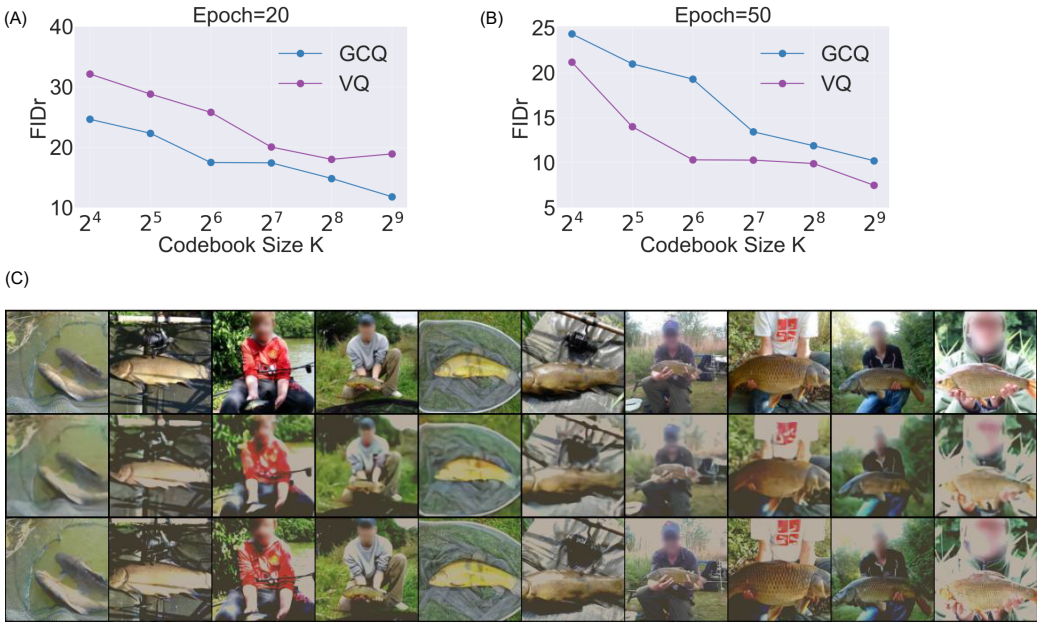

Figure 7: To evaluate the performance of GCQ in a static setting, we conducted experiments on the CIFAR-10 and ImageNet datasets. Figures (A) and (B) show that, for both GCQ and VQ, the FIDr decreases as the codebook size K increases. At 20 training epochs, GCQ outperforms VQ, whereas at 50 epochs VQ achieves better results. Figure (C) compares the reconstructions produced by GCQ and VQ on ImageNet: the first row shows the original images, the second row the reconstructions obtained with GCQ, and the third row those obtained with VQ.

B Experiment Details

Here are the hyperparameters used in the experiment. All programs run on an NVIDIA A100-SXM4-80GB. The experiments reported in this paper, including the ViT, ResNet, and Hybrid networks, required 8-12 hours of training each. For ViT and hybrid architectures, we trained for 40 epochs with a learning rate of 1e-4; for the ResNet network, we trained for 100 epochs with a learning rate of 3e-4. All training runs used the Adam optimizer.

C Python Implementation

Our Python implementation of GCQ is fully vectorized, relying exclusively on matrix operations without any for loops. This design makes it highly amenable to parallelization.

```
def forward(self, latents: Tensor, label: Tensor) -> Tuple[Tensor, Tensor]:
    """
    Vector quantization for sequence data with parallel processing support for batch dimensions

    Args:
        latents: Tensor of shape [B x S x D x H x W]
        label: Tensor of shape [B x S x 4] representing the change from current frame to next frame

    Returns:
        quantized: Quantized tensor of shape [B x S x D x H x W]
        vq_loss: Vector quantization loss
```

```
    """
    B, S, D, H, W = latents.shape
    N = H * W // 4
    device = latents.device

    # Reshape to processable form
    latents = latents.view(B, S, D, 4, N).permute(0, 3, 4, 1, 2).contiguous()  # [B, 4, N, S, D]
    factor_latents = latents.view(B, 4, N, S * D)

    # Construct expanded embeddings without loops
    # label: [B, S, 4] -> rearrange to [B, 4, S]
    shift_amounts = (label.permute(0, 2, 1).long() * self.step)  # [B, 4, S]
    # Construct indices for rolling: for each shift_amount against self.embedding_weight
    # First generate indices [K], then calculate new indices via broadcasting: new_idx = (arange(K) - shift) % K
    k_idx = torch.arange(self.K, device=device).view(1, 1, 1, self.K)  # [1,1,1,K]
    # After expanding shift_amounts: [B,4,S,1]
    rolled_indices = (k_idx - shift_amounts.unsqueeze(-1)) % self.K  # [B, 4, S, K]
    # Using rolled_indices to extract new embeddings from embedding_weight, resulting shape [B,4,S,K,D]
    expanded_embed = self.embedding_weight[rolled_indices]  # [B,4,S,K,D]
    # Adjust dimensions: exchange [K] and [S] positions before merging S and D
    expanded_embed = expanded_embed.permute(0, 1, 3, 2, 4).contiguous()  # [B,4,K,S,D]
    # Finally reshape to [B, 4, K, S*D]
    expanded_embedding = expanded_embed.view(B, 4, self.K, S * D)

    # Calculate nearest neighbor indices
    # factor_latents: [B,4,N,S*D]; expanded_embedding: [B,4,K,S*D]
    A = factor_latents  # [B,4,N,S*D]
    B_expand = expanded_embedding  # [B,4,K,S*D]
    A_sq = (A ** 2).sum(dim=-1, keepdim=True)  # [B,4,N,1]
    B_sq = (B_expand ** 2).sum(dim=-1).unsqueeze(-2)  # [B,4,1,K]
    cross = 2 * torch.matmul(A, B_expand.transpose(-1, -2))  # [B,4,N,K]
    dist = A_sq + B_sq - cross  # [B,4,N,K]
    encoding_inds = dist.argmin(dim=-1)  # [B,4,N]

    # Sample vectors from expanded_embedding according to indices using torch.gather
    # expanded_embedding shape [B,4,K,S*D], sampling on dim=2
    encoding_inds_exp = encoding_inds.unsqueeze(-1).expand(-1, -1, -1, S * D)  # [B,4,N,S*D]
    embedding_results = torch.gather(expanded_embedding, 2, encoding_inds_exp)  # [B,4,N,S*D]

    commitment_loss = F.mse_loss(embedding_results.detach(), factor_latents)
    embedding_results = factor_latents + (embedding_results - factor_latents).detach()  # [B,4,N,S*D]
    embedding_results = embedding_results.view(B, 4, N, S, D)
    embedding_results = embedding_results.permute(0, 3, 4, 1, 2).contiguous().view(B, S, D, H, W)
    return embedding_results, commitment_loss
```