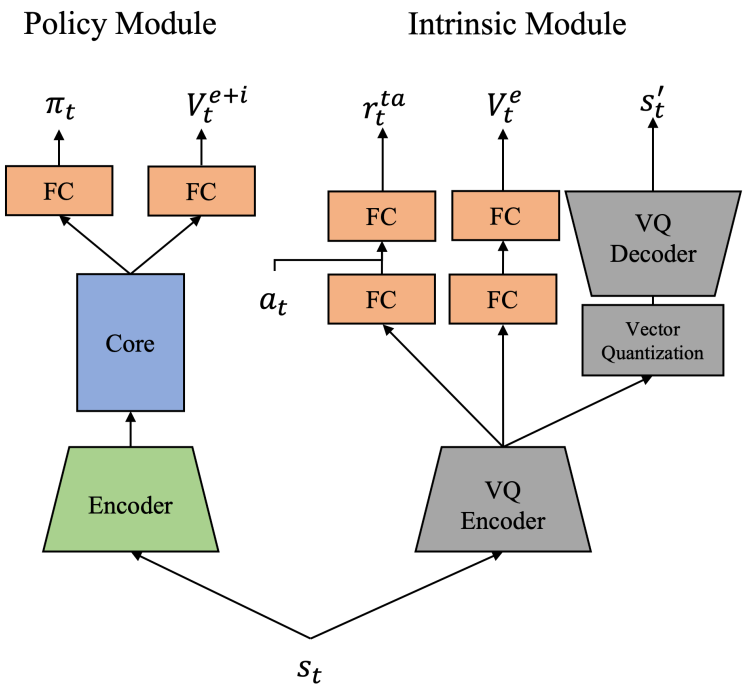


Figure 7: Diagram of network architecture for the policy module and the intrinsic module.

A Architecture Details

We employ two modules, the policy module and the intrinsic module. The overall network architectures for these modules are described in Figure 7 and their details are presented in Table 1 and Table 2. The base architectures of the policy network for MiniGrid and DMLab are based on that of AGAC and IMPALA, respectively. One difference is that the value head of the policy module, which originally outputs the value estimation of extrinsic rewards V^e, estimates the value function for the sum of extrinsic and intrinsic rewards, denoted V^{e+i}. The value estimation for extrinsic rewards is now done in the intrinsic module, as shown in the right hand side of Figure 7.

B Hyperparameters and Training Details

All baseline models as well as LECO are trained using the V-trace actor-critic framework, IMPALA [12], where the policy and the value networks are learned from trajectories collected from multiple actors running asynchronously in a distributed system. The experimentation code for all methods are implemented on top of Sample Factory [27], a code base designed for high throughtput rate during IMPALA-based training. The total number of actors used in MiniGrid tasks and DMLab tasks are 20 and 240, respectively, spread across multiple nodes. Table 3 presents our common hyperparameter values for all models on MiniGrid and DMLab tasks. It is noted that the ObstructedMaze tasks in MiniGrid used $\gamma = 0.8$ while all other tasks, including the DMLab tasks, used $\gamma = 0.99$. We observe that ObstructedMaze tasks, especially the ObstructedMaze-Full is solved with a lower γ value.

For RND, NovelD, and RIDE on DMLab tasks, we search over learning-rate $\in \{0.0001, 0.0002, 0.0004\}$, and the intrinsic reward coefficient $\in \{0.1, 0.01, 0.005\}$ and select the learning rate as 0.0001 and the intrinsic reward coefficient as 0.01.

Table 1: Policy Module

Module	**Layers**	
	MiniGrid	DMLab
Preprocessing	Normalization $\in (-1, 1)$	
ConvBlock(k, c, s)	[Conv2d($k \times k, c$, stride=s)]	
Residual Block(k, c)	[ConvBlock($k \times k, c$, stride=1), ReLU, ConvBlock($k \times k, c$, stride=1), ReLU]	
Encoder	ConvBlock(3,32,2) ELU ConvBlock(3,32,2) ELU ConvBlock(3,32,2) ELU FC $\times$ 1	ConvBlock(3,16,1), MaxPool(3 $\times$ 3,stride=2) ReLU Residual Block(3,16) $\times$ 2 ConvBlock(3,32,1), MaxPool(3 $\times$ 3,stride=2) ReLU Residual Block(3,32) $\times$ 2 ConvBlock(3,32,1), MaxPool(3 $\times$ 3,stride=2) ReLU Residual Block(3,32) $\times$ 2 FC $\times$ 1
Core	LSTM(256) $\times$ 1	LSTM(256+$\lvert\mathcal{A}\rvert$+1) $\times$ 2
Head	Policy: FC(256,$\lvert\mathcal{A}\rvert$) Value: FC(256, 1)	Policy: FC(256+$\lvert\mathcal{A}\rvert$+1,$\lvert\mathcal{A}\rvert$) Value: FC(256+$\lvert\mathcal{A}\rvert$+1, 1)

Table 2: Intrinsic Module

Module	**Layers**	
	MiniGrid	DMLab
Preprocessing	Upsample ($7 \times 7 \rightarrow 12 \times 12$) Downsample ($96 \times 72 \rightarrow 96 \times 64$) Normalization $\in (-1, 1)$	
ConvBlock(k,s)	[Conv2d($k \times k, 64$, stride=s), BatchNorm2d]	
Residual Block	[ReLU, ConvBlock(3, 1), ReLU, ConvBlock(1,1)]	
VQ-Encoder	ConvBlock(3, 1) [ConvBlock(4, 2), ReLU, AvgPool, Conv2d($1 \times 1, 64$, stride=1)] $\times$ 3 ConvBlock(4, 2) Residual Block $\times$ 2	
VQ-Decoder	Symmetric architecture of Encoder	
Modulator	FC(576, 512)	FC(384, 512) ReLU FC(512 + $\lvert\mathcal{A}\rvert$, $\lvert\mathcal{A}\rvert$) tanh
Value Head	FC(576, 512)	FC(384, 512) ReLU FC(512, 1)

C Ablation Study

C.1 Hash size.

In our VQ-based hashing, used in LECO and the baselines, the hash size is defined as the spatial size $w \times h$ of the output of the encoder of VQ-VAE. Accordingly, the capacity of the hash is defined as $K^{w \times h}$ where K is the codebook size. This capacity can be critical in measuring the state novelty based on the episodic count, as it determines the level of the state compression and the corresponding

Table 3: Hyperparameters

Parameter	Value	
	MiniGrid	DMLab
Learning rate (η)	0.0003	0.0001
Learning rate (η_{ta})	0.0003	0.3η
Batch Size	16	48
Entropy Coefficient	0.0005	0.003
Unroll Length	96	
γ	0.99^*	
α	0.01	
λ	0.5	
V-trace ρ	1.0	
V-trace c	1.0	
Optimizer	Adam	
Adam ϵ	1e-6	
Adam β_1	0.9	
Adam β_2	0.999	

Table 4: Hashing parameters

Method	Parameter	Value	
		MiniGrid	DMLab
VQ	spatial size ($w \times h$)	3×3	3×2
	codebook size (K)	8	24
AE-LSH [33]	SimHash dimension	25	62
	$b(s)$ size (bits)	256	512
DSC [10]	spatial size ($w \times h$)	3×3	3×2
	intensity (K)	11	4

state clustering. If the capacity is too large then most states would be mapped to different hashes and if it is too small then most states would be mapped to the same hash.

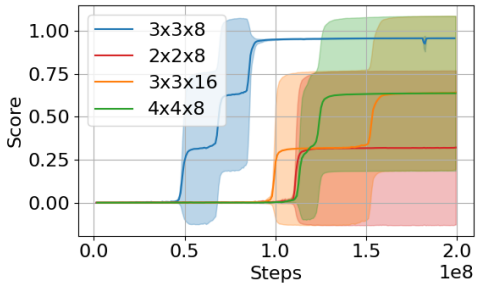
In MiniGrid, we search over the spatial size $\in \{2 \times 2, 3 \times 3, 4 \times 4\}$ and the codebook size $\in \{8, 16\}$. In DMLab, we search over the spatial size $\in \{3 \times 2, 6 \times 4\}$ and the codebook size $\in \{12, 24, 36\}$. The key results of this grid search for LECO are shown in Figure 8. As shows in the figure, $3 \times 3 \times 8$ performs the best in MiniGrid tasks and $3 \times 2 \times 24$ performs the best in the DMLab tasks and thus the two parameter values are selected as the default values for our hashing as in Table 4.

For AE-LSH, we search over the hash size $\in \{9, 15, 25, 35\}$ in MiniGrid, and $\in \{24, 42, 62, 82\}$ in DMLab and select 25 and 62 as the hash size of each task. For a given state, DSC hash is defined as follows: 1) downsample it to $w \times h$ by nearest neighbor algorithm, 2) rescale the intensity so that they are integers between 0 and K for each channel of the state, and 3) serialize it. For DSC in MiniGrid, we search over the downsampled size $\in \{2 \times 2, 3 \times 3, 4 \times 4\}$, and in DMLab we search over the downsampled size $\in \{3 \times 2, 6 \times 4\}$ and the intensity $\in \{4, 8\}$, and select $3 \times 3 \times 11$ and $3 \times 2 \times 4$ as the hash size for MiniGrid and DMLab, respectively.

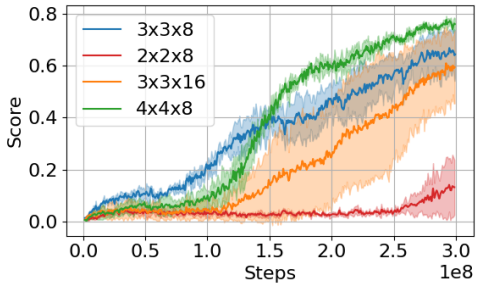
C.2 Study of λ.

Equation 1 indicates the possibility of adjusting the weight to the task-specific modulation. In order to show the effect of this weight, we run an ablation on Equation 1 where the equation is restated here for convenience:

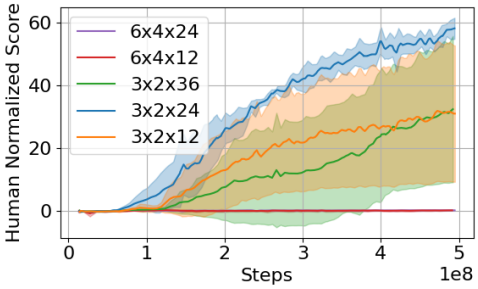
$$r_t^i(a_{t-1}, s_t, a_t, s_{t+1}) = (1 - \lambda) r_t^{\text{ep}}(s_{t+1}) + \lambda r_t^{\text{ta}}(a_{t-1}, s_t, a_t).$$



(a) MiniGrid-KeyCorridor-S6R3



(b) MiniGrid-ObstructedMaze-Full



(c) DMLab-lasertag_three_opponents_large

Figure 8: Performance comparison across different VQ hashing parameters ($w \times h \times K$).

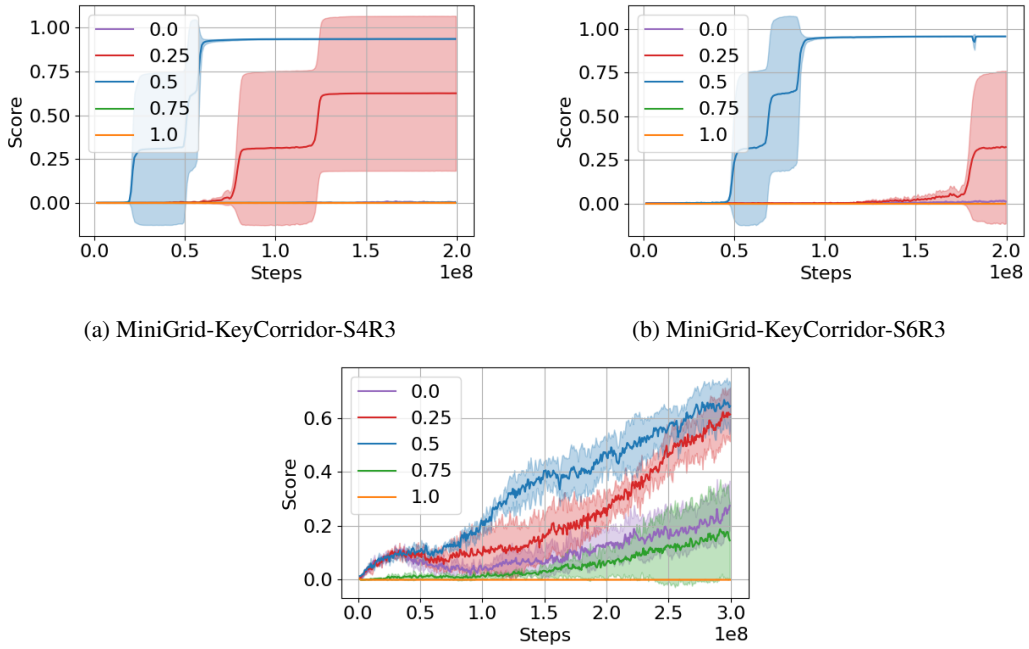
We fix λ as a constant hyperparameter and its value is determined by grid search on three MiniGrid tasks as shown in Figure 9 of Appendix. On the other hand, lambda can be adaptively calculated relative to the magnitude of the novelty from episodic counts, however, we leave it as the future work.

When $\lambda = 0$, the algorithm is equivalent to using episodic count only, which is denoted as VQ-only in the main text. When $\lambda = 1.0$, the algorithm is similar to using LIRPG [40] only, however, note that the architecture of the task-specific modulator are fairly different from the intrinsic module in LIRPG, and the final optimization loss of the intrinsic module in LECO is also different in that it includes a VQ-VAE loss as a part through the shared encoder. When $\lambda = 0.5$, it is equivalent to LECO. As shown in Figure 9, having only either the episodic state novelty or the task-specific modulation results in failure in solving the task. On the other hand, putting more weight on the episodic state novelty ($\lambda = 0.25$) relatively performs better than small weight ($\lambda = 0.75$), however the performance is best when the two terms are balanced ($\lambda = 0.5$).

D Discussions

D.1 Task-irrelevant actions in count-only methods.

In order to solve the KeyCorridor tasks, an agent is required to obtain a key, open a locked door, and obtain the target object, which is a colored ball. Without task-specific modulation the agent can easily get stuck on repeating meaningless actions such as dropping the key or opening and closing the door, to obtain task-irrelevant state novelties. Figure 10 visually depicts such phenomenon. In this figure we provide action heat maps on three distinct episodes. Each map displays the accumulated counts on three specific actions ($pickup, drop, toggle$) from different episodes which are sampled by the final checkpoint of each model. Red colored grids represent the agent's execution of specific action for more than once in that position. For example, in *episode-1* of the top-row, the agent performs the specific action twice below of the gray key. In top-row of the figure we can observe the agent of VQ-only performing the $toggle$ action multiple times in front of the doors and the $pickup$ & $drop$ action repeatedly in empty spaces. Moreover, the agent opens the door to the room with the target object but does not proceed to picking it up (see *Episode-3*). On the other hand, the bottom row of



(a) MiniGrid-KeyCorridor-S4R3

(b) MiniGrid-KeyCorridor-S6R3

(c) MiniGrid-ObstructedMaze-Full

Figure 9: Performances according to different λ's. The labels indicate the normalized λ such that the sum of the weights of the episodic state novelty and task-specific modulation is equal to 1.

the same figure shows that LECO focuses on achieving the task objective by faithfully conducting task-related actions.

D.2 Task-specific modulation in r^i.

The task-specific modulation of LECO is designed to dynamically control exploration and exploitation during training. Here, we conceptually illustrate the trends of LECO's intrinsic reward r^i as the learning progresses. In the beginning of training, before collecting enough extrinsic rewards, the agent would not have any information about task-relevant and -irrelevant behaviors. In hard exploration problems, the agent cannot obtain extrinsic rewards without utilizing the state novelty (i.e. exploration) in the environment. Assuming that exploration by episodic state novelty is enough to guide the agent to achieve some extrinsic rewards, those extrinsic rewards would cause r^{ta} to be positive to encourage the agent to explore more in that direction. As the agent learns explorative behaviors and collects some extrinsic rewards, task-specific modulation will start to distinguish task-relevant from the irrelevant amongst those learned behaviors. Then, the agent transitions to focusing on the task-relevant behaviors and eventually ignore the task-irrelevant behaviors by decaying the intrinsic reward as a whole. The results in Figure 13 depicts the described decaying trend. We can also observe that the dynamics of LECO's intrinsic reward r^i is different for each task; r^i decays more moderately as the relative difficulty and extrinsic reward sparsity increases. The difficulty and sparsity roughly increases from (a)→(c), (d)→(f) and from (g)→(h).

D.3 Comparisons of hashing results.

First, We qualitatively show some weaknesses of AE-LSH compared to VQ. In general, both hash methods map similar states into the same hash, however, some differences can be observed in the obtained hashing results. As shown in Figure 14 and Figure 15, in Minigrid, AE-LSH maps important states (i.e. states in which the agent realizes a target object or states just before obtaining a target object) and relatively less important states into the same hash. This obstructs correctly distinguishing task-specific state novelty for efficient exploration. On the other hand, VQ more clearly clusters states to the hash according to the relative importance.

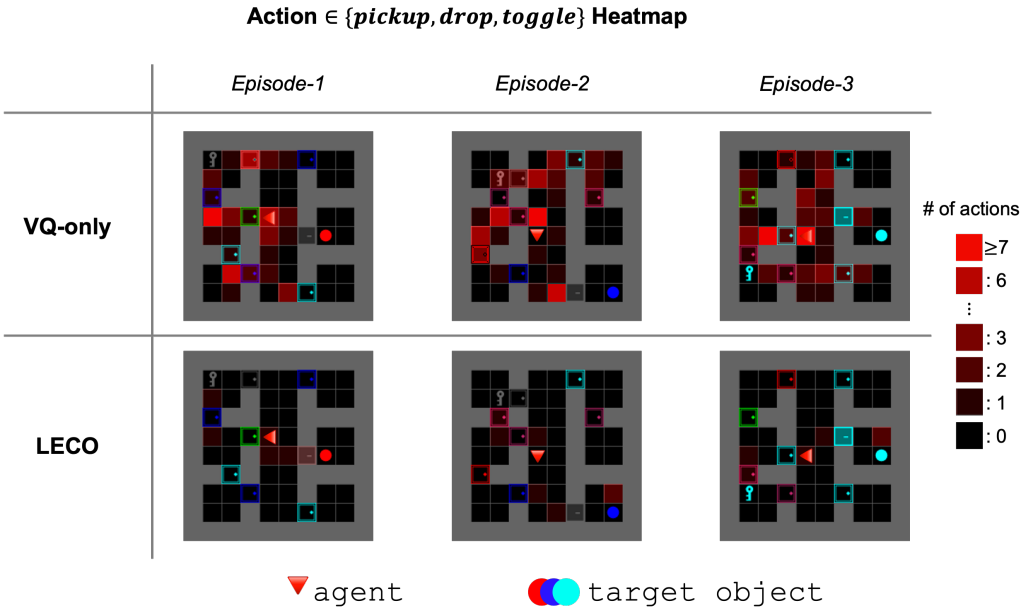


Figure 10: Action heat maps of three different episodes on the task of `KeyCorridor-S4R3`, Count only based, VQ-only, (**Top-Row**) and LECO (**Bottom-Row**). The red triangle is added to indicate the initial position of the agent. The ball, whose color is randomly chosen at the start of each episode, is the target object that the agent has to pickup. Unlike KeyCorridor, for the tasks of ObstructedMaze family, the goal object is the blue ball, whose color stays consistent.

In DMLab-lasertags, the difference in performance between AE-LSH and VQ is also reflected in the hashing results. As shown in Figure 16, Figure 17, and Figure 18, AE-LSH maps states to hashes according to visual similarities whereas VQ learns to map according to the importance represented via the presence of the opponents in sight.

Table 5: Mean *new-hash rates* on five episodes sampled on DMLab-lasertag.

`three_opponents_small`		`three_opponents_large`	
LECO(AE-LSH)	LECO(VQ)	LECO(AE-LSH)	LECO(VQ)
86.8%	71.6%	68.2%	58.6%

We quantitatively show the effectiveness of VQ compared to AE-LSH. As shown in Table 5, on the DMLab-Lasertags, the new hash rates, which measure how often the states are mapped into a new hash within an episode, obtained by VQ are smaller than those by AE-LSH even though the entire hash space of VQ is larger than AE-LSH (i.e. $6^{24} > 2^{62}$). This means that VQ makes similar states to be more grouped into the same hash.

D.4 Noisy-TV on MiniGrid.

Provided by [28] is a variation of MultiRoom task that brings the concept of Noisy-TV problem into the MiniGrid environment setting, named `MiniGrid-MultiRoomNoisyTV-N7-S4`. In this version of the task, the color of the ball changes to a random color when the agent performs a specific action, creating a visual novelty irrelevant to the objective of the task, which is to reach the goal [28]. In Figure 11, we present the learning curves of LECO and baselines on this Noisy-TV environment. The result shows that all methods that use the episodic intrinsic reward, including LECO, are able to successfully solve the noisy environments. In this experiment, We conjecture that the task-specific modulator in LECO requires a little more experiences than NovelD since it is updated upon the extrinsic rewards that are sparse in the early RL phase. In a similar argument, DSC, which requires

no training, is most efficient in the noisy TV problem. Moreover, in fact NovelD converges stably in about 3e7 frames (vs. 5e7 frames for LECO).

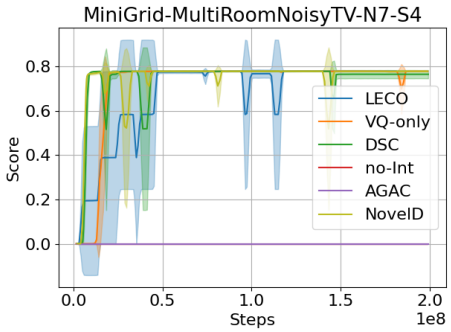


Figure 11: Performance comparison on Noisy-TV environment. Along with its variants, LECO is able to bypass the Noisy-TV problem. The shaded area represents a range of a standard deviation over 3 runs of different random seeds.

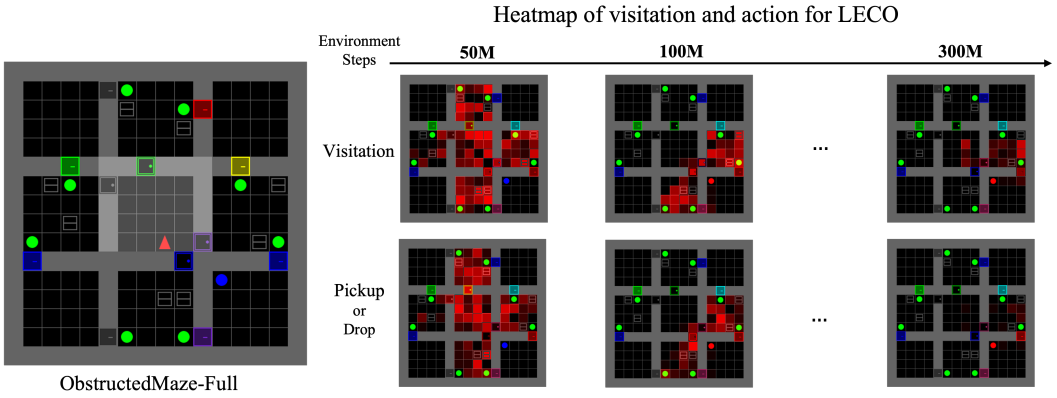
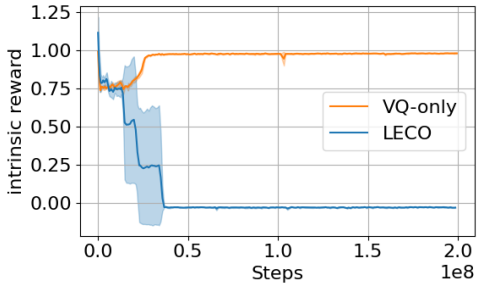
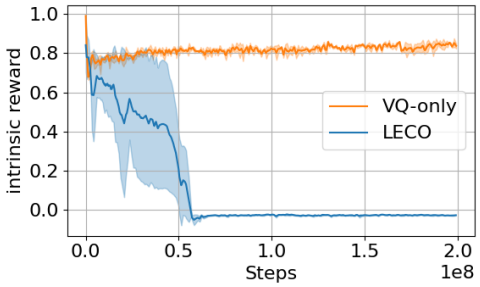


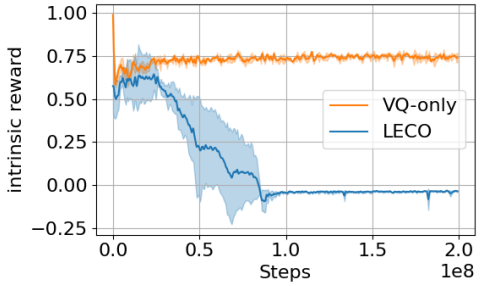
Figure 12: `ObstructedMaze-Full` Environment (**Left**), Visitation heatmaps (**Right Top**) and Action heatmaps (**Right Bottom**): LECO successfully solves this task from 100M steps, and does not repeat meaningless pickup & drop actions to obtain the state novelty at 300M steps.



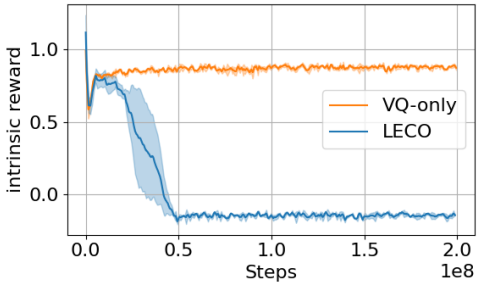
(a) MiniGrid-MultiRoom-N6



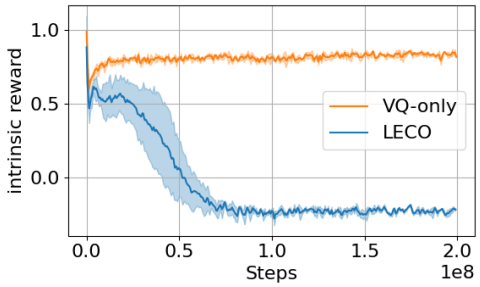
(b) MiniGrid-KeyCorridor-S4R3



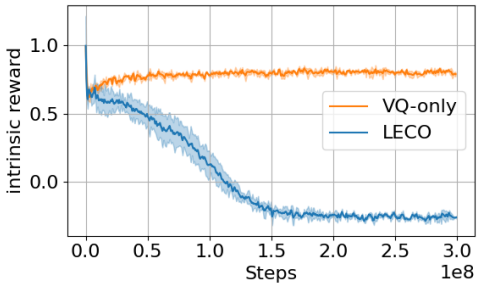
(c) MiniGrid-KeyCorridor-S6R3



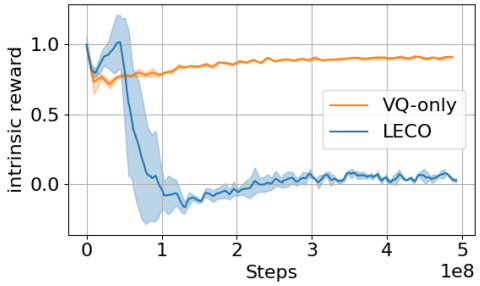
(d) MiniGrid-ObstructedMaze-1Q



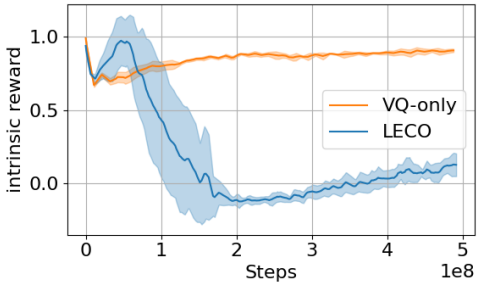
(e) MiniGrid-ObstructedMaze-2Q



(f) MiniGrid-ObstructedMaze-Full



(g) DMLab-lasertag_three_opponents_small



(h) DMLab-lasertag_three_opponents_large

Figure 13: Change in r^i for different tasks during training.

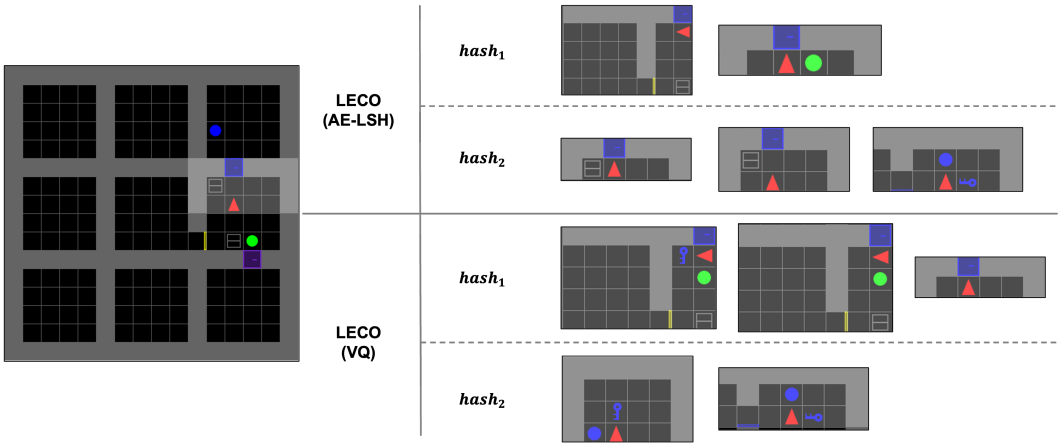


Figure 14: (**Left:**) An episode of `ObstructedMaze-1Q` Environment. The bright area is the partial observation. In this episode, the agent must find a blue key, open the locked blue door, and obtain the blue ball. (**Right:**) Hash samples from the episodic hash memory of LECOs. Each row represents a hash index and the partially observed states that are mapped into this hash.

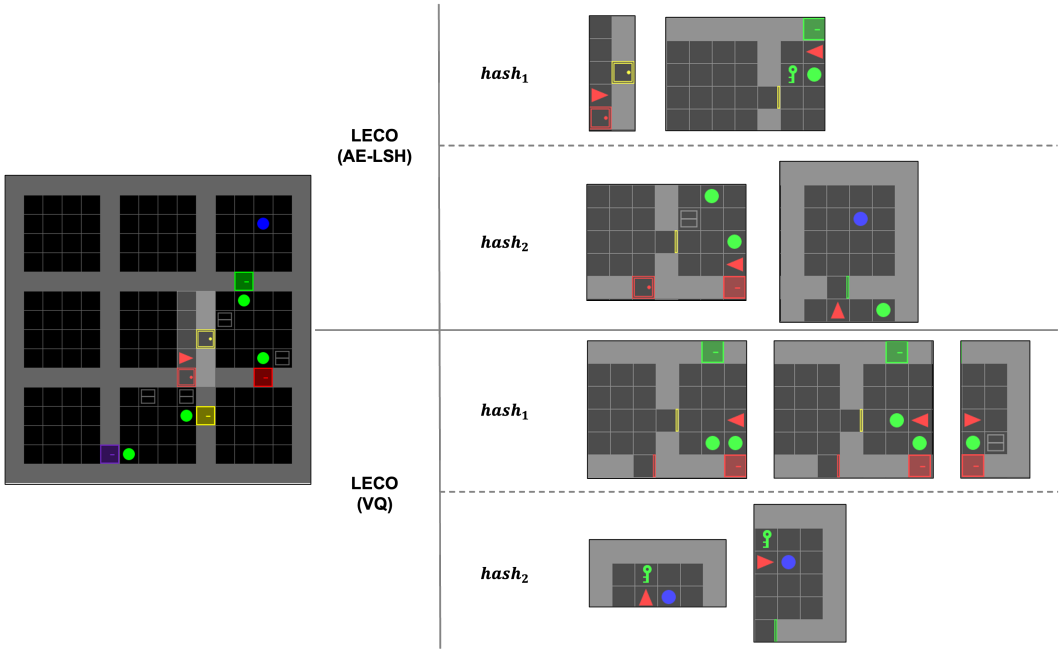


Figure 15: (**Left:**) An episode of `ObstructedMaze-2Q` Environment. The bright area is the partial observation. In this episode, the agent must find a green key, open the locked green door, and obtain the blue ball. (**Right:**) Hash samples from the episodic hash memory of LECOs. Each row represents a hash index and the partially observed states that are mapped into this hash.

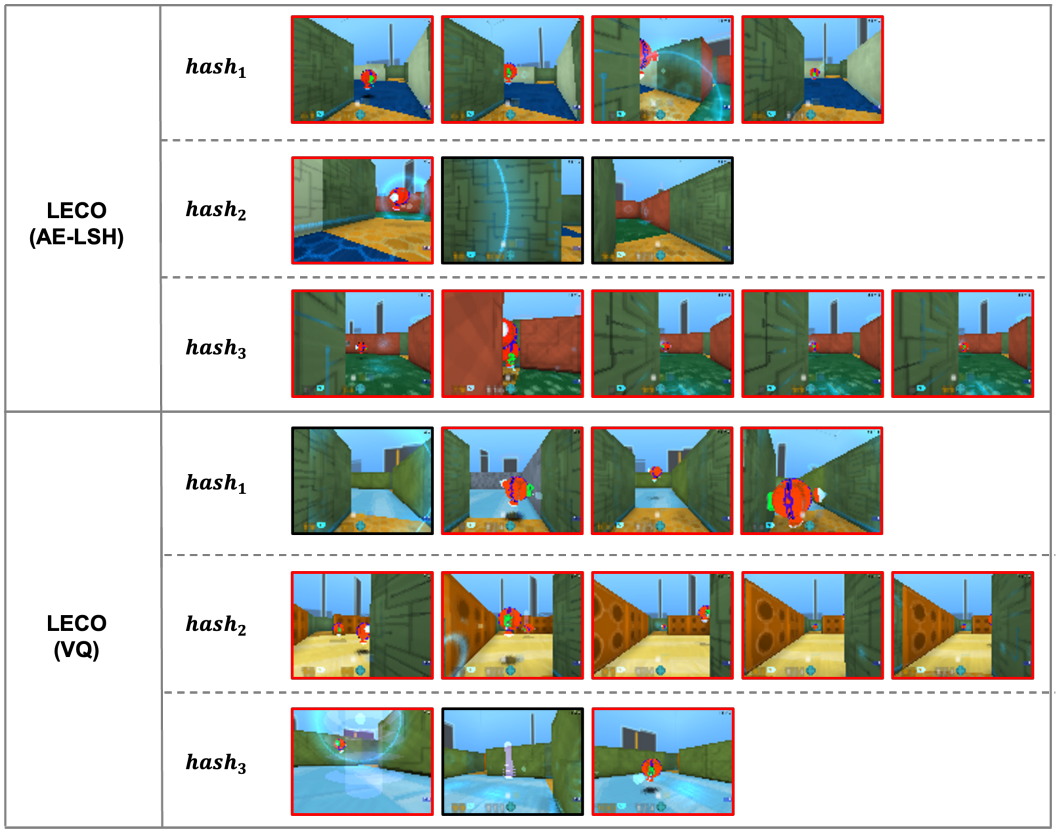


Figure 16: Hash samples of the episodic hash memory of LECOs on DMLab-lasertag_three_opponents_small. Each row represents a hash index and the partially observed states that are mapped into this hash. The state that contained the opponents is highlighted in red.

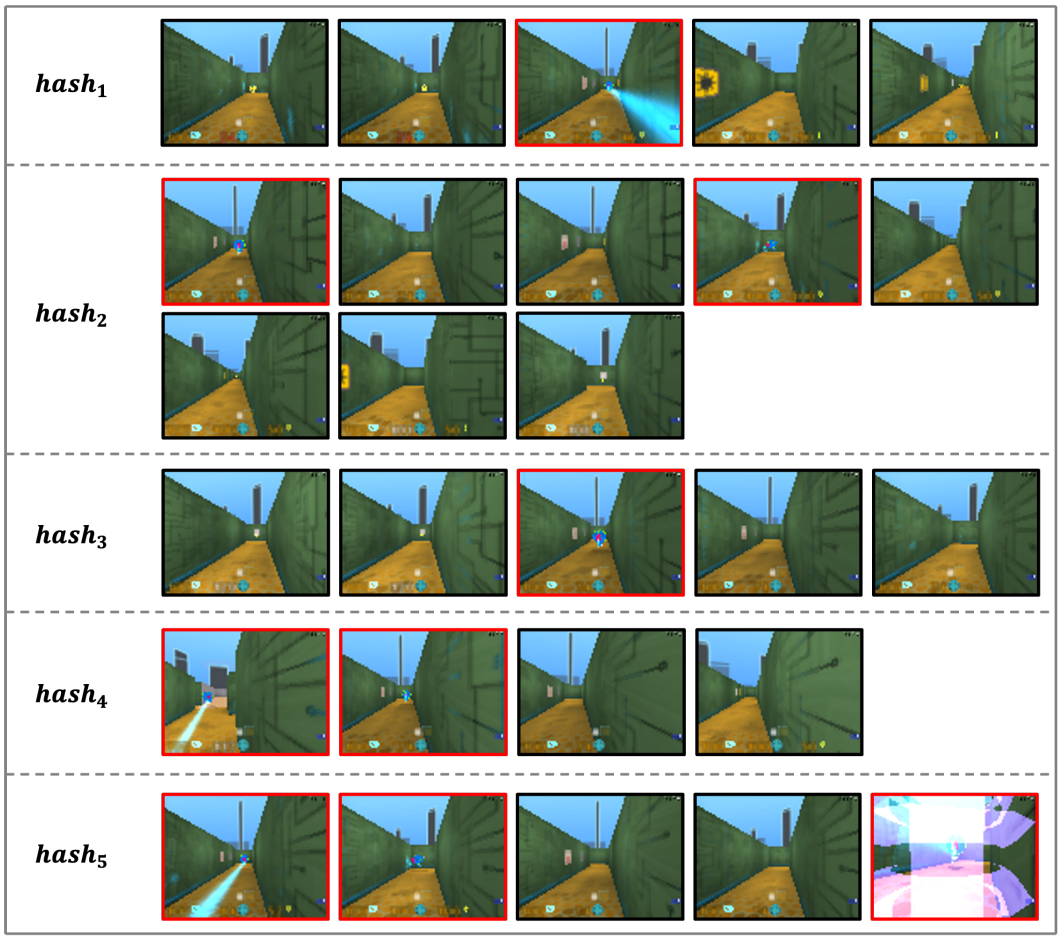


Figure 17: Hash samples of the episodic hash memory of LECO(AE-LSH) on DMLab-lasertag_three_opponents_large. Each row represents a hash index and the partially observed states that are mapped into this hash. The state that contained the opponents is highlighted in red.

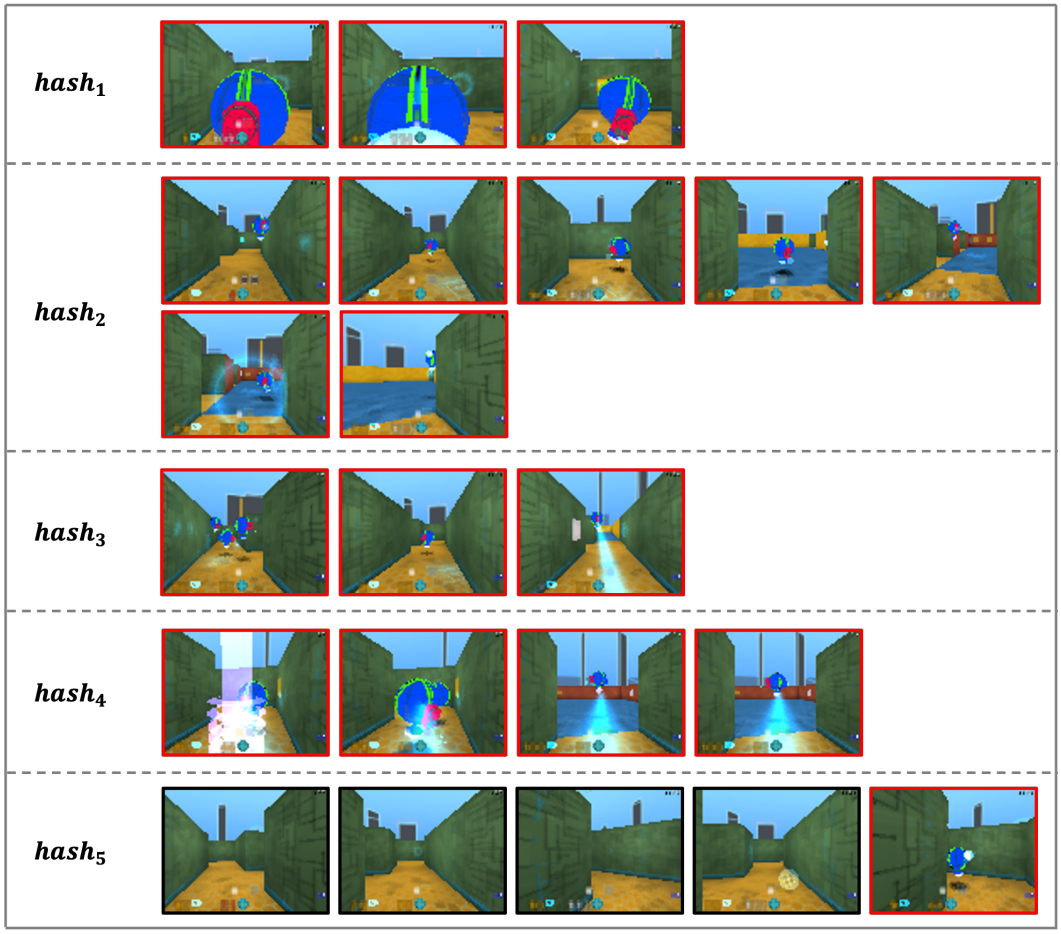


Figure 18: Hash samples of the episodic hash memory of LECO(VQ) on DMLab-lasertag_three_opponents_large. Each row represents a hash index and the partially observed states that are mapped into this hash. The state that contained the opponents is highlighted in red.