Appendix

Primary Keywords: *Learning, Imitation Learning, Information Theory*

Algorithm Optimization

Practical Optimization

In Equation 3 of the main paper, we adopt $H(z)$ instead of $H(z|s)$ to optimize the latent skill representation. Skills are mostly irrelevant to the state of the instruction following tasks. Formally, we build the mutual-information between latent skill z and s:

$$I(z;s) = H(z|s) - H(z),$$

where $H(z|s)$ is the conditional entropy of the skill z given the state s, and $H(z)$ is the entropy of the skill z. We assume that the mutual information between skill and state is low, in which case $H(z|s)$ can be approximated by $H(z)$. The entropy of skills measures the degree of randomness or disorder. As we introduce code reinitialization to make discrete skill selection rather than training only a few codes (as shown in Figure 6(a)), we implicitly increase the randomness of skills.

Mutual Information entropy loss

Here, we explain how to transform the objective from maximizing entropy to minimizing MSE error and incorporate it into VQVAE training; we provide the following proof: First, we have our optimal goal from Equation 3.

$\mathcal{F}(\theta,\phi) = H(z|s) + E[\log p(z|s,l)] + E[\log q(l|\mathbf{z})]$

In our implimentation we have skill encoder $p_\theta(z|s,l)$ and language decoder $q_\phi(\mathbf{z})$. The first term is optimized by code reinitialization, which is mentioned in the appendix. Here, to prove how we transfer maximizing conditional entropy into minimizing MSE loss. The conversion mode of the third term of the formula is illustrated below.

$$\mathbf{E}[\log p(z \mid l)] = \sum_{i=1}^{m} \log p\left(l_i \mid z_i;\theta\right)$$

$$= \sum_{i=1}^{m}\left(\log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{\left(l_i - \mathrm{q}_\phi\left(z_i\right)\right)^2}{2\sigma^2}\right)\right)$$

$$= \frac{1}{2\sigma^2}\sum\left(l_i - \mathrm{q}_\phi(z_i)\right)^2 - \mathrm{m}\ln\sigma\sqrt{2\pi}$$

By removing the constant term, we can equivalently maximize entropy, equivalent to minimizing the mean squared error (MSE) loss. In Equation 5, we optimize this term in each episode by concatenating skill into trajectory instruction and compute MSE loss with language embedding from the CLIP encoder. The second term can also be optimized with $|p_\theta(s,l) - sg(z_q)|_2^2$.

Implementation Details

We provide details on the implementation of our LCSD algorithm in this section.

Language Encoder: We use a pre-trained CLIP ResNet50 text encoder network as our language encoder. This part is used only to decouple the complex language and will not be involved in the subsequent updates.

Observation Encoder: We use linear MLP layers as our state encoder. In the LORel environment, image observations are encoded by a convolution network similar to the original paper.

Skill Encoder: We use linear MLP as our skill encoder with language embeddings and states as input. The output skills correspond to the states of each step.

Skill Decoder: We use a three-linear-layer MLP as our language decoder with unique skill sets as our input. The output vector is consistent with the dimensionality of the language embedding.

Diffusion Model: We define a conditional U-net model for fusing different modal information. This denoising model also extends to the language condition, which only requires changing the initialization dimension. The denoising network structure is shown in Figure 1.

Our implementation uses the PyTorch framework on an NVIDIA A100 GPU[1]. Parameters of that LCSD method used on different environments are listed in Table 4. For tasks of different difficulty, we use different sizes of codebook to measure. We use 20 skill codes for Babyai and Calvin as these environments do not contain many skills. For the LORel environment, we use 30 skill codes to represent more skills. Our method remains robust in varying hyperparameters, which is shown in Section .

[1]Our PyTorch code and datasets would be public after publishing

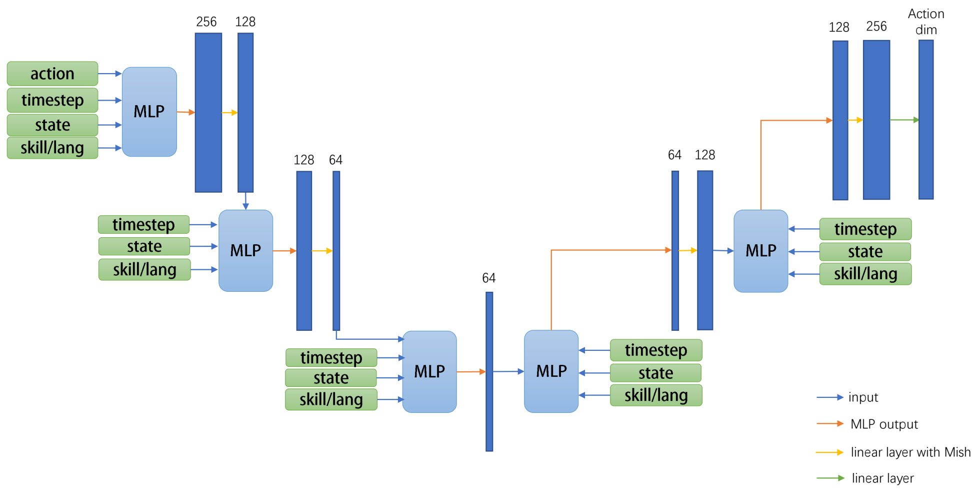


Figure 1: **Denoising Network Architecture**. We use a network similar to U-net. Different linear layers reduce and expand the spatial dimensions while fusing multimodal features.

Table 1: **Parameters of LCSD**

Hyperparameter	BabyAI	LORel	Calvin
Behavior loss Weight	5.0	5.0	5.0
Skill loss Weight	2.0	2.0	5.0
Reconstruct loss Weight	0.01	0.01	0.1
commitment Weight	1.0	1.0	1.0
Batch size	256	256	128
Codebook Dim	16	16	16
Skill Number	20	30	20
Reconstruct Option	3	4	2
Policy lr	2e-4	2e-4	2e-4
Skill lr	1e-4	1e-4	1e-4
Timesteps	50	100	50

Experimental Details

We describe detailed information about the evaluation environments and the specific demonstration trajectories.

Babyai: Babyai is a two-dimensional grid game with multiple task settings, where the agent performs a series of subtasks according to the language instructions, such as ”open the yellow door” and ”go to the key behind you”. To evaluate our skill-based model, we select three relatively challenging tasks: GoToSeq, SynthSeq, and BossLevel. We use 10,000 trajectories for each task in our experiments. An example of BabyAI BossLevel sequence is shown in Figure 2.

LORel: LORel is a robot manipulation environment based on Metaworld in a tabletop scenario, where a simulated Sawyer robot manipulates a faucet, a drawer, and two cups in different colors. Each trajectory is labeled with a language and contains one or more tasks, such as ”Move black mug left and turn faucet left”. Two trajectories of different tasks are shown in Figure 3. We collect an offline dataset of 50,000 trajectories in our experiments from the official code.

Calvin: Calvin is a manipulation environment that uses a Franka Panda arm in four different settings. We adopt the tasks proposed by the original paper and modify the evaluation on six tasks: Open Drawer, Close Drawer, Turn on Lightbulb, Turn off Lightbulb, Move Slider Left, and Move Slider Right. We directly select 1,216 trajectories from the Calvin-D dataset that are relevant to the above six tasks as our offline dataset. Since RGB image inputs depend on factors such as image encoders and multiple perspectives, in order to eliminate interference and focus solely on evaluating the underlying policy and their performance on DT, we directly selected the 21-dimensional perspective state of the Calvin environment as the input for LCSD. As this type of state space does not include the spatial aspect of the ball, we excluded tasks related to the ball in our task selection, which is mentioned above. The dataset can be downloaded from the official link.

Table 2 lists the language labels used in our training dataset for the Calvin environment. We observe that the language labels for each task in Calvin are not standardized but vary as much as possible, similar to the different test settings in LORel. However, unlike LORel, the training dataset in Calvin consists of diverse language instructions, making it more challenging to train our LCSD algorithm.

As a result, we find that the language condition model outperformed the skill condition in the Calvin environment, where we use a diverse offline dataset with varying instruction labels. This outcome is reasonable, given the nature of the training dataset in Calvin. It demonstrates the importance of considering the language condition in our LCSD algorithm when dealing with diverse and varied language instructions. We are pleased to see that our LCSD maintains a high level of performance in all tasks.

Additional Experiments

Effect of Skill and Reconstruction Option Numbers

Codebook size and the number of reconstruction options are important hyperparameters that may affect skill learning

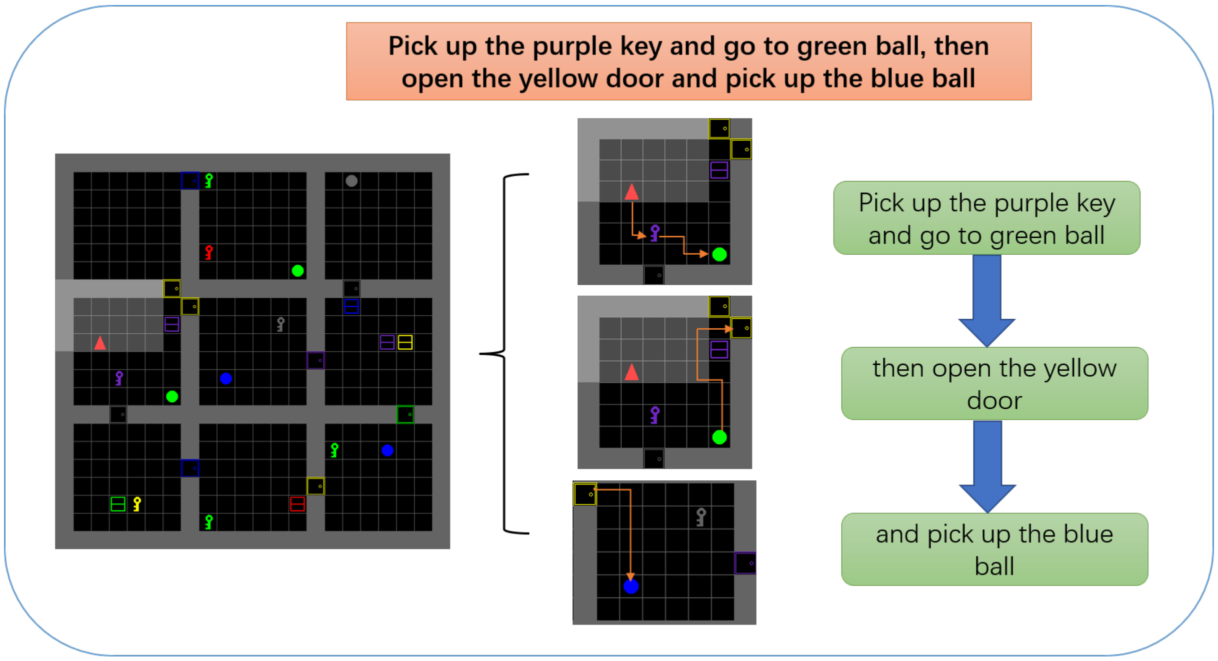

Pick up the purple key and go to green ball, then open the yellow door and pick up the blue ball
Pick up the purple key and go to green ball
then open the yellow door
and pick up the blue ball

Figure 2: Babyai Bosslevel on task *Pick up the purple key and go to green ball, then open the yellow door and pick up the blue ball .*

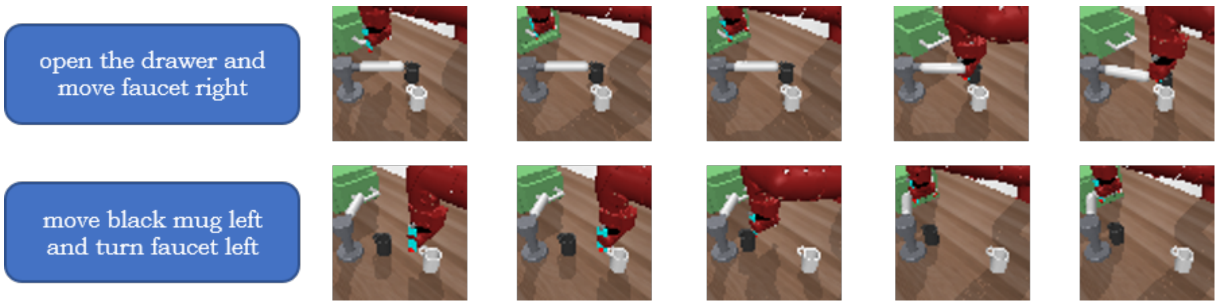

open the drawer and move faucet right
move black mug left and turn faucet left

Figure 3: LORel trajectory on two different tasks

Table 2: Calvin dataset language settings

Catagory	Detail tasks	Training Language	Evaluation language	Num
lightbulb	turn on/off lightbulb	toggle the light switch to turn on/off the light bulb turn on/off the light bulb move the light switch to turn on/off the light bulb	use the switch to turn on/off the light bulb	95
drawer	open/close drawer	open/close the cabinet drawer grasp the drawer handle and open/close it pull/push the handle of the drawer	pull/push the handle to open/close the drawer	303
slider	move slider left/right	grasp the door handle, then slide the door to the left/right move the door all the way to the left/right slide/push the door to the left/right	push the sliding door to the left/right side	382
light	push down the button to turn on/off the led	press the button to turn on/off the led light turn on/off the led light toggle the button to turn on/off the led light	push the sliding door to the left/right side	382
Unknown tasks		Turn on/off green/yellow lamp move/toggle the light switch to turn off the yellow/green light		227

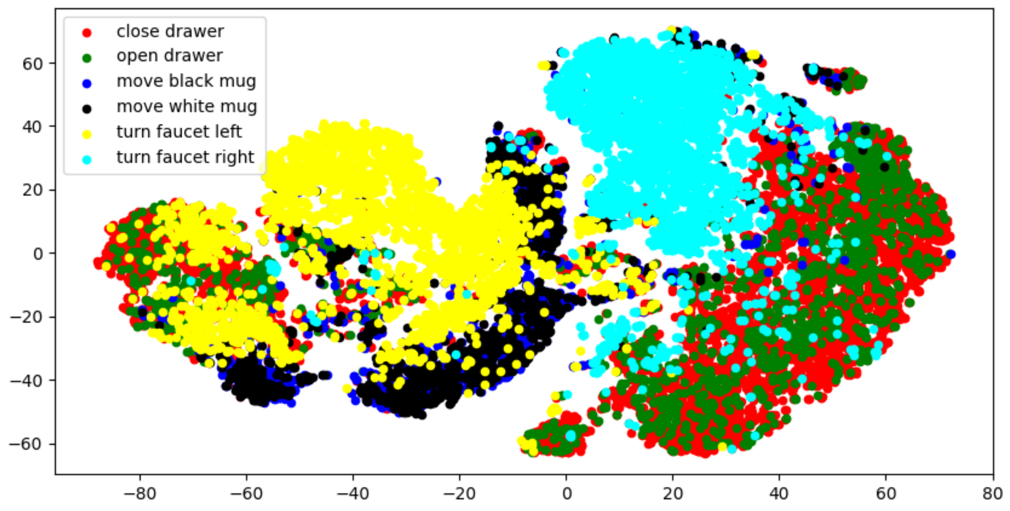


Figure 4: Data visualize in LORel.

Table 4: Diffusion language policy performance on Babyai

Timestep	SynthSeq	GoToSeq	BossLevel
100	54.1%	65.2%	55.2%
50	52.2%	65.4%	48.5%

performance. In our approach, the upper limit of the skill set used by the decoder needs to be adjusted depending on the task, similar to how codebook size is adjusted in VQ-VAE. In Figure 10, we have shown the stability of skills generated by LCSD in varying codebook sizes. We further show our success rate in LORel environments in Table 3 to see that the overall result is stable in the process of parameter change. (Note that this success rate is an overall result of all tasks, which is different from the main paper.)

Table 3: LCSD performance on LORel with different sizes of codebook and number of reconstruction options

CodebookOption	3	4	5
20	35.10%	35.34%	38.92%
30	37.40%	38.70%	36.36%
40	37.66%	38.96%	37.92%

In Table 3, We find that the best result is achieved when LCSD with the setting of 40 skills and four reconstruction options. Meanwhile, we observe that the performance of each parameter fluctuated within a normal range. These results confirm the robustness of our LCSD algorithm on these two hyperparameters. It demonstrates that our algorithm can handle varying codebook sizes and the number of reconstruction options while still achieving stable performance.

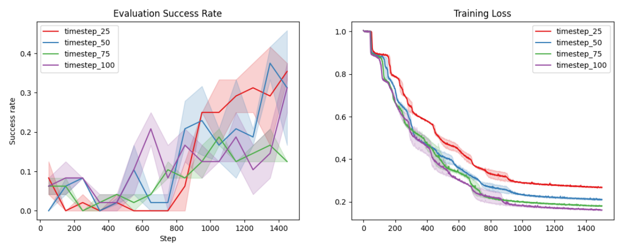


Figure 5: Diffusion performance and training loss on Calvin with different timesteps.

How do Diffusion timestep affect results

Additionally, we evaluate the effect of timestep on the results of behavioral cloning in the DDPM process. A larger timestep can provide a more accurate distribution for the input recovery, but it will also take more time during evaluation. We evaluate our Diffusion language-conditioned policy on the BabyAI environment with two timestep settings. In the BabyAI environment, the performance of LCSD is not significantly influenced by timestep, as shown in Table 4.

We conduct more ablation experiments on the Calvin environment with 25, 50, 75, and 100 timesteps for six tasks, as shown in Figure 5. In the complex Calvin dataset, we observe that a larger timestep does not correspond to better results. As seen in the table, timesteps of 25 and 50 are the most appropriate in both performance and efficiency, while timesteps of 75 perform the worst, even though the training loss decreased more quickly. We believe this is because a good fitting is not always reflected in the performance in non-optimal diverse datasets. However, with our diffusion model, we can handle both large and precise as well as small and diverse datasets with appropriate timesteps.

Skill Analysis

We analyze the effectiveness of our skill learning method in the context of Decision Transformer (DT) by comparing it with the original DT-based skill learning model, LISA. In Figure 6, we show the skill correlation map for the original LISA model, which only trained on two skills during VQ quantization. However, we can select more skills with our code reinitialization method, as shown in Figure 6(b). This demonstrates the generalizability of our skill-learning method to other imitation models.

To more specifically describe the meaning of the skill map, Calvin's skill map is analyzed in detail in Figure 7. There are six high-frequency words selected in code 2 that combine to form a single skill language: Push the sliding door to the left. In code 18, we can see a combination of multiple expressions of a task: pull/open the drawer/handle. This phenomenon shows that the skill learning of LCSD can fully understand the potential association of human semantics and then better generalize to more tasks.

In Figure 8, we compare the performance of the DT model with and without code reinitialization. We find that the original DT model failed to achieve practical training in almost all task settings because skill learning is limited to index collapse. On the other hand, the advantage of skill dispersion with code reinitialization is reflected in the test accuracy.

In summary, our skill-learning method is effective not only for our LCSD algorithm but also for other imitation models, such as DT-based models. The results suggest that our code reinitialization method can help to generate discrete meaningful skills, leading to better test accuracy and improved skill learning.

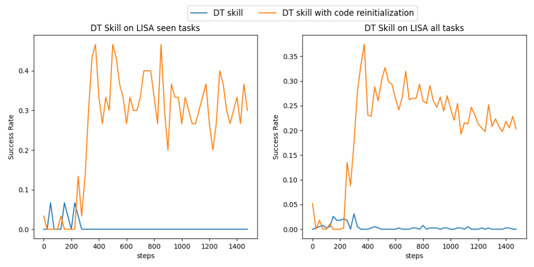


Figure 8: DT model with code reinitialization success rate on LORel.

Different skill map in Varying codebook Size

To test the robustness of LCSD, we use the skill-language correspondence graph to identify whether different codebook sizes influence skill learning. For comparison, we also conduct experiments using a basic skill-encoder-only diffusion model, namely the Diffusion Encoder skill model(Abbreviated as Diffusion encoder in Figures). Firstly, we plot our basic diffusion encoder skill model(Diffusion with LISA) in Figure 9 on the codebook with sizes of 30 and 40. From the figures, we observe that skill learning stacked in index collapse, which was more serious when using a larger codebook.

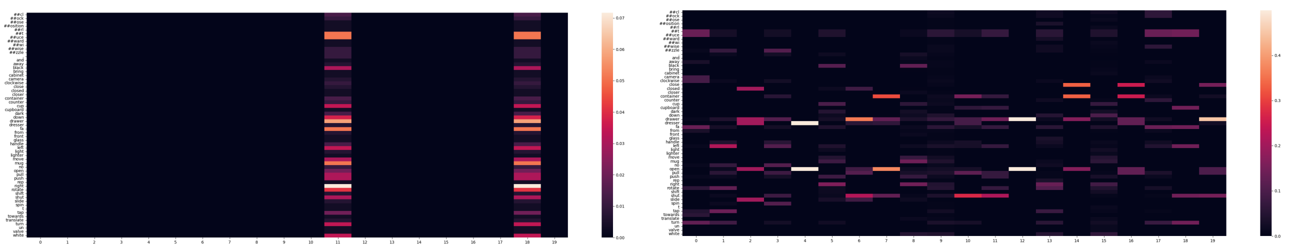
Next, we test our LCSD algorithm with the same number of skills in LORel, as shown in Figure 10. We observe that LCSD can learn diverse and precise skills regardless of the codebook size, demonstrating the robustness of our algorithm.

Skill Maps in Different Environment

This section presents our skill correlation maps on the BabyAI and Calvin environments. In the first two tasks of BabyAI, GoToSeq, and SynthSeq, the skills are diverse enough as the tasks are relatively simple to handle. We plot the skill maps for the BabyAI BossLevel tasks to demonstrate the performance of different skill-learning methods in complex language-conditioned tasks, as shown in Figure 11. In contrast, the skills learned by the diffusion encoder are still limited to a small portion of the codebook, and index collapse will be alleviated after adding a language decoder, as shown in Figure 11(b) and Figure 11(c). It is evident that LCSD generates better discrete and interpretable skills among all the methods in Figure 11(a).

In the Calvin environment, we find that at a codebook size of 20, the skills learned by the diffusion encoder are close in dispersion to the skills generated by LCSD(Comparing two panels of Figure 12). However, as the number of skills increased, the codebook fell into a small subset of skill vector during skill learning, as shown in Figure 13(a). In contrast, LCSD is able to maintain skill diversity even with a larger codebook, which corresponds to our stable performance on different tasks (Figure 13(b)).

In summary, our skill correlation maps in different environments demonstrate the effectiveness of our LCSD algorithm in learning diverse and precise skills, even in complex language-conditioned tasks. The ability of our algorithm to maintain skill diversity regardless of the codebook size and number of reconstruction options is a crucial advantage, enabling us to handle large and complex datasets with varying hyperparameters.



(a) skill-word correlation map (b) skill-word correlation map with code reinitialization

Figure 6: Skill Map on DT model with LISA or LCSD skill learning method.

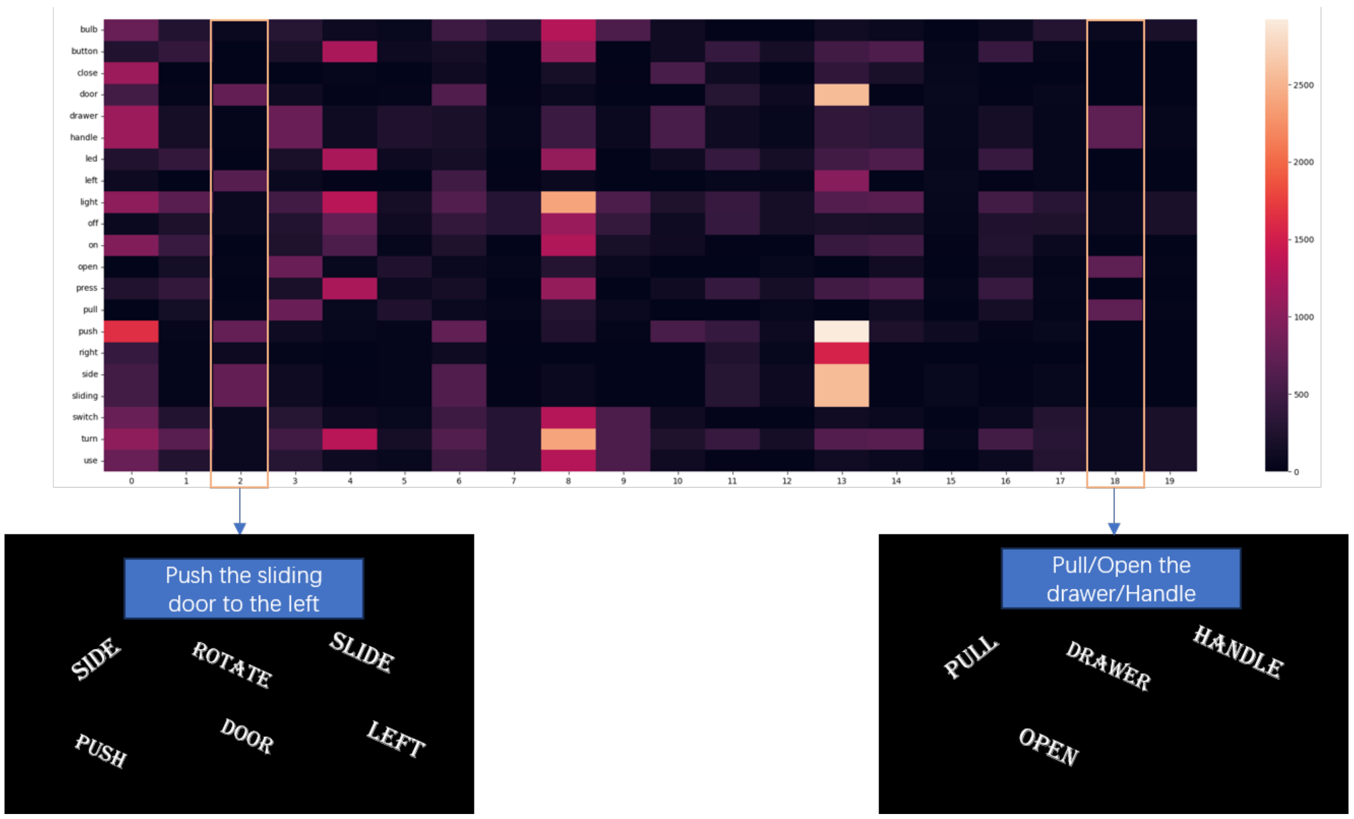
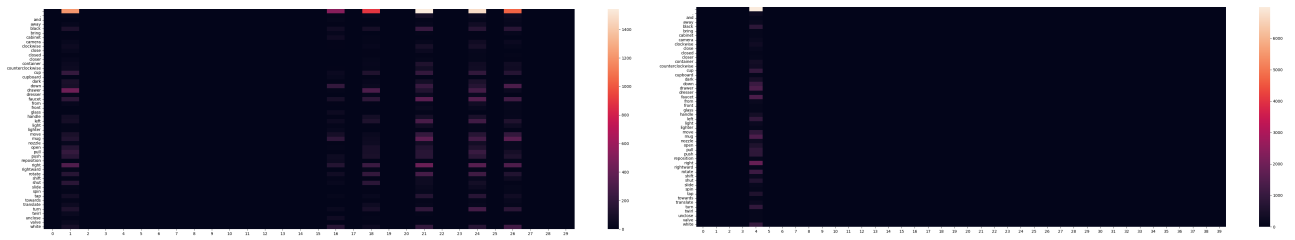
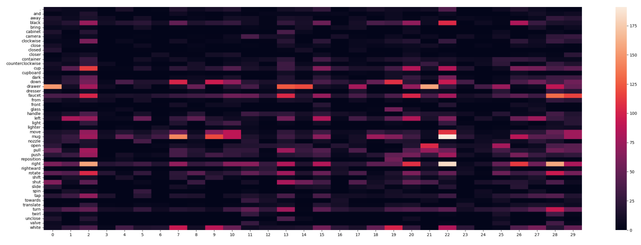


Figure 7: **Explanation of certain skill codes in CALVIN**. Skill maps show how our skills are correlated with words in evaluation. As can be seen from the second column, six words are chosen and can be formed into a sentence: Push the Sliding door to the left. Also, in code 18, this skill corresponds to the task ”open drawer” and can recognize different representations: open/pull, drawer/handle.



(a) Diffusion LISA skill map on 30 codes (b) Diffusion LISA skill map on 40 codes

Figure 9: Diffusion LISA Skill map in LORel state environment.


(a) LCSD skill map with 30 codes

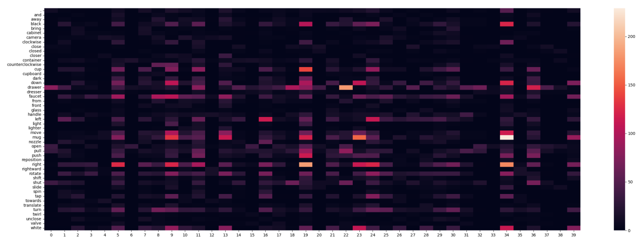
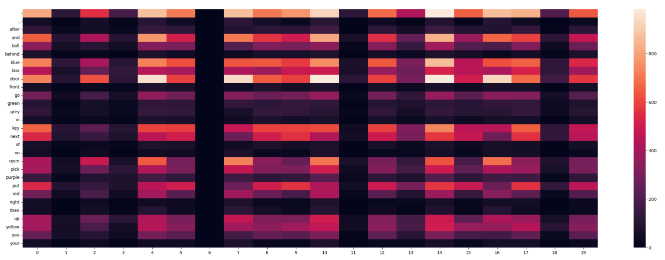
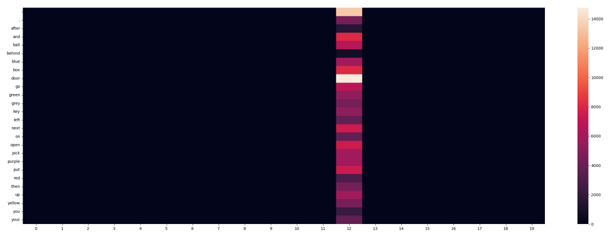

(b) LCSD skill map with 40 codes

Figure 10: LCSD Skill map in LORel state environment.


(a) LCSD skill map


(b) Diffusion LISA skill map

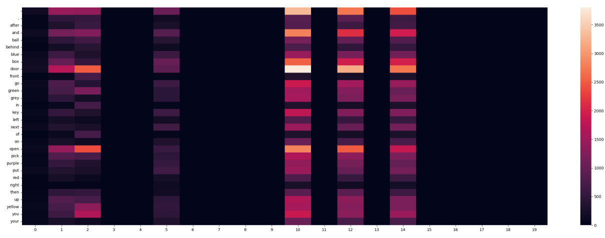
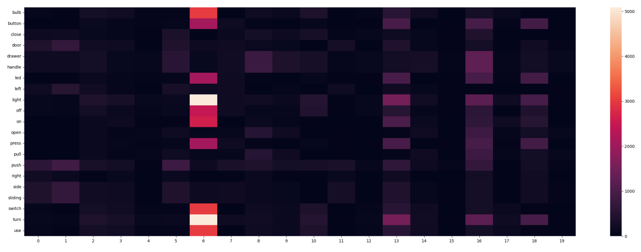

(c) Diffusion LISA skill map with reinit

Figure 11: Skill map in BabyAI Bosslevel


(a) Diffusion Encoder skill map with 20 skills

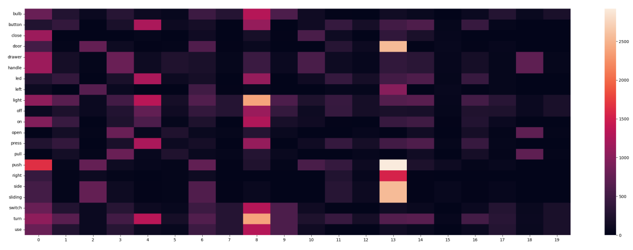
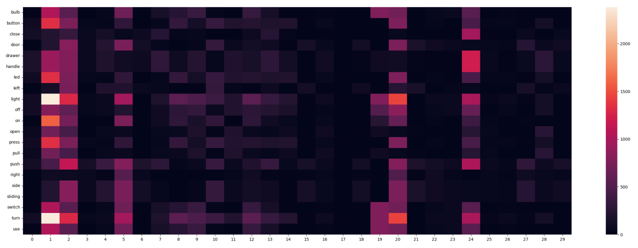
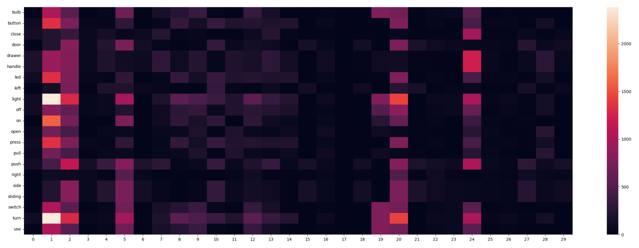

(b) LCSD skill map with 20 skills

Figure 12: Skill map in Calvin on codebook length of 20



(a) Diffusion Encoder skill map with 30 skills



(b) LCSD skill map with 30 skills

Figure 13: Skill map in Calvin on codebook length of 30