Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015b.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Alan M Turing and J Haugeland. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, pp. 29–56, 1950.

Hu Wang, Qi Wu, and Chunhua Shen. Soft expert reward learning for vision-and-language navigation. In *European Conference on Computer Vision*, pp. 126–141. Springer, 2020a.

Tongzhou Wang, Simon S Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised mdps: Learning world models better than the world itself. *arXiv preprint arXiv:2206.15477*, 2022a.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Vision-language navigation policy learning and adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4205–4216, 2020b.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning, 2022b. URL https://arxiv.org/abs/2208.06193.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics, 2011.

Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Kaizhi Zheng, Xiaotong Chen, Odest Chadwicke Jenkins, and Xin Eric Wang. Vlmbench: A compositional benchmark for vision-and-language manipulation. *arXiv preprint arXiv:2206.08522*, 2022.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. Rtfm: Generalising to novel environment dynamics via reading. *arXiv preprint arXiv:1910.08210*, 2019.

Victor Zhong, Jesse Mu, Luke Zettlemoyer, Edward Grefenstette, and Tim Rocktäschel. Improving policy learning via language dynamics distillation. *arXiv preprint arXiv:2210.00066*, 2022.

Hongkuan Zhou, Zhenshan Bing, Xiangtong Yao, Xiaojie Su, Chenguang Yang, Kai Huang, and Alois Knoll. Language-conditioned imitation learning with base skill priors under unstructured data. *arXiv preprint arXiv:2305.19075*, 2023.

Appendix

TABLE OF CONTENTS

Please refer to our website `https://language-control-diffusion.github.io/website/` for more qualitative results in video format. We release our code and models at `https://github.com/language-control-diffusion/code`.

A BROADER IMPACT

In this paper, we present research on diffusion-based models for reinforcement learning. While our work has the potential to advance the field, we recognize the importance of being transparent about the potential societal impact and harmful consequences of our research.

Safety: Our research focuses on the development of diffusion models for reinforcement learning, and we do not anticipate any direct application of our technology that could harm, injure, or kill people.

Discrimination: We recognize the potential for LCD to be used in ways that could discriminate against or negatively impact people. As such, we encourage future work to take steps to mitigate potential negative impact, especially in the areas of healthcare, education, or credit, and legal cases.

Surveillance: LCD did not involve the collection or analysis of bulk surveillance data.

Deception and Harassment: We recognize the potential for our technology to be used for deceptive interactions that could cause harm, such as theft, fraud, or harassment. We encourage researchers to communicate potential risks and take steps to prevent such use of LCD.

Environment: Our research required the usage of GPUs, which may be potentially energy intensive and environmentally costly. However, we seek to minimize the usage of GPUs through LCD's efficiency and the impact of our work could be beneficial to the environment.

Human Rights: We prohibit the usage of LCD that facilitates illegal activity, and we strongly discourage LCD to be used to deny people their rights to privacy, speech, health, liberty, security, legal personhood, or freedom of conscience or religion.

Bias and Fairness: We recognize the potential for biases in the performance of LCD. We encourage researchers building off this work to examine for any biases and take steps to address them, including considering the impact of gender, race, sexuality, or other protected characteristics.

In summary, we recognize the potential societal impact and harmful consequences of our research and encourage researchers to consider these factors when developing and applying new technologies on top of LCD. By doing so, we can help ensure that our work has a positive impact on society while minimizing any potential negative consequences.

B ADDITIONAL RELATED WORK

Here we gives a more detailed treatment to additional related work.

B.1 TEXT-TO-CONTROL MODELS

Modern text to control models often still struggle with long-horizon language commands and misinterpret the language instruction. Hu et al. (2019) attempt to solve a long-horizon Real-Time Strategy (RTS) game with a hierarchical method utilizing language as the communication interface between the high level and low level policy, whilst Harrison et al. (2017) consider training a language encoder-decoder for policy shaping, Hanjie et al. (2021) utilize an attention module conditioned on textual entities for strong generalization and better language grounding. Zhong et al. (2022) propose a model-based objective to deal with sparse reward settings with language descriptions and Mu et al. (2022) also tackle the sparse reward settings through the usage of intrinsic motivation with bonuses added on novel language descriptions. However, only Hu et al. (2019) consider the long-horizon setting, and they do not consider a high-dimensional state space. Jang et al. (2022) carefully examine the importance of diverse and massive data collection in enabling task generalization through language and propose a FiLM conditioned CNN-MLP model (Perez et al., 2018). Much work has attempted the usage of more data and compute for control (Brohan et al., 2022; Jaegle et al., 2021; Reed et al., 2022). However, none of these methods consider the usage of diffusion models as the medium between language and RL.

B.2 DIFFUSION MODELS

One driver of this recent success in generative modeling is the usage of *classifier-free guidance*, which is amenable to the RL framework through the usage of language as a reward function. The language command is a condition for optimal action generation, which draws direct connection to control as inference (Levine, 2018). Given the success of denoising diffusion probabilistic models (Ho et al., 2020) in text-to-image synthesis (Sohl-Dickstein et al., 2015a), the diffusion model has been further explored in both discrete and continuous data domains, including image and video synthesis (Ho et al., 2022b;c), text generation (Li et al., 2022b), and time series (Rasul et al., 2021). To expound further on Janner et al. (2022)'s Diffuser, they diffuse the state and actions jointly for imitation learning and goal-conditioned reinforcement learning through constraints specified through classifier guidance, and utilize Diffuser for solving long-horizon and task compositional problems in planning. Instead of predicting the whole trajectory for each state, (Wang et al., 2022b) apply the diffusion model to sample a single action at a time conditioned by state. However, neither of these works considers control from pixels, or utilizing language for generalization.

C Reasoning for usage of HULC as Low Level Policy (LLP)

We justify the usage of using a hierarchical model as our LLP here. We would like to clarify that we use a hierarchical approach for our low-level policy for two main reasons. First, we aim to leverage the strongest existing baseline to maximize our chances of creating a SOTA model. Second, we encountered issues replicating the results for the GCBC flat policy from its original paper (Mees et al., 2022b), making it less favorable for our approach. It is also important to note that there is, in principle, no reason why LCD cannot work with a flat policy, and this would be a straightforward addition for future work. One successful instantatiation of a direct text to video generation model is (Dai et al., 2023). However, they again avoid the issue of directly generating low level actions by using a low-level controller or an inverse dynamics model which we are able to directly solve for.

D Hyper-parameter settings and training details

For all methods we proposed in Table 1, Table 2, **??**, and Table 5, we obtain the mean and standard deviation of each method across 3 seeds. Each seed contains the individual training process and evaluates the policy for 1000 episodes.

Baselines are run with either 8 Titan RTX or 8 A10 GPUs following the original author guidelines, whilst our experiments are run with a single RTX or A10. In total this project used around 9000 hours of compute.

D.1 HP and training details for methods in Table 1 and **??**.

Model	Module	Hyperparameter	Value
HULC	Trainer	Max Epochs	30
		β for KL Loss	0.01
		λ for Contrastive Loss	3
		Optimizer	Adam
		Learning Rate	2e-4
	Model	Transformer Hidden Size	2048
		Language Embedding Size	384
LCD	Gaussian Diffusion	Action Dimension	32
		Action Weight	10
		Loss Type	L2
		Observation Dimension	32
		Diffusion Steps	20
		Model Dimension	64
	Trainer	EMA Decay	0.995
		Label Frequency	200000
		Sample Frequency	1000
		Batch Size	512
		Learning Rate	2e-4
		Train Steps	250k
		Number of Steps Per Epoch	10000
		Normalizer	Gaussian Normalizer
		Frame Offset	0

Table 5: Hyperparameters for our methods in Table 1 and **??**.

E HYPER-PARAMETERS FOR TABLE 4

E.1 HYPERPARAMETERS

We control all other evaluation parameters not listed in Table E.1 to be the same as in our prior experiments for the MLP and Diffusion high-level policy. These hyperparameters can be found in Appendix D.

	Hyperparameter	Value
Transformer	Number of gradient steps	100 K
	Mini-batch size	512
	Transformer hidden dim	4096
	Transformer hidden layers	3
	Final Hidden Activation	ReLU
	Final Hidden Dim	4096
	Number of heads	8
	Dropout	0.1
	Learning Rate	$2\mathrm{e}-4$
	Layer Norm	32

Table 6: Hyperparameters for our methods in Table 4.

E.2 GRIDSEARCH

We gridsearch over the following parameters and pick the best performing combination from the following:

- Num of transformer layers: [4, 8, 24]
- Width of transformer layers: [2048, 4096, 8192]
- Learning rate: [2e-5, 2e-4, 4e-4]

F TRAINING OBJECTIVE DERIVATION

To model this, we turn to diffusion models (Weng, 2021), whom we borrow much of the following derivation from. Inspired by non-equilibrium thermodynamics, the common forms of diffusion models (Sohl-Dickstein et al., 2015b; Ho et al., 2020; Song et al., 2020a) propose modeling the data distribution $p(\boldsymbol{\tau})$ as a random process that steadily adds increasingly varied amounts of Gaussian noise to samples from $p(\boldsymbol{\tau})$ until the distribution converges to the standard normal. We denote the forward process as $f(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t-1})$, with a sequence of variances $(\beta_0, \beta_1 ... \beta_T)$. We define $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.

$$f(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0) = \prod_{t=1}^{T} f(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t-1}), \quad \text{where } f(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t-1}) = \mathcal{N}(\boldsymbol{\tau}_t; \sqrt{1-\beta_t}\boldsymbol{\tau}_{t-1}, \beta_t \mathbf{I}). \tag{6}$$

One can tractably reverse this process when conditioned on $\tau_{\mathbf{0}}$, which allows for the construction of a sum of the typical variational lower bounds for learning the backward process' density function (Sohl-Dickstein et al., 2015b). Since the backwards density also follows a Gaussian, it suffices to predict μ_θ and Σ_θ which parameterize the backwards distribution:

$$p_\theta\left(\boldsymbol{\tau}_{t-1} \mid \boldsymbol{\tau}_t\right) = \mathcal{N}\left(\boldsymbol{\tau}_{t-1}; \boldsymbol{\mu}_\theta\left(\boldsymbol{\tau}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\boldsymbol{\tau}_t, t\right)\right). \tag{7}$$

In practice, Σ_θ is often fixed to constants, but can also be learned through reparameterization. Following (Ho et al., 2020) we consider learning only μ_θ, which can be computed just as a function of $\boldsymbol{\tau}_t$ and $\epsilon_\theta(\boldsymbol{\tau}_t, t)$. One can derive that $\boldsymbol{\tau}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\tau}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, through a successive reparameterization of (6) until arriving at $f(\boldsymbol{\tau}_t|\boldsymbol{\tau}_0)$. Therefore to sample from $p(\boldsymbol{\tau})$, we need only to learn ϵ_θ, which is done by regressing to the ground truth $\boldsymbol{\epsilon}$ given by the tractable backwards density. Assuming we have $\boldsymbol{\epsilon_\theta}$, we can then follow a Markov chain of updates that eventually converges to the original data distribution, in a procedure reminiscent of Stochastic Gradient Langevin Dynamics (Welling & Teh, 2011):

$$\boldsymbol{\tau}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{\tau}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\boldsymbol{\tau}_t, t\right)\right) + \sigma_t \mathbf{z}, \quad \text{where } \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{8}$$

To learn ϵ_θ, we can minimize the following variational lower bound on the negative log-likelihood:

$$\begin{aligned}
L_{\text{CE}} &= -\mathbb{E}_{q(\boldsymbol{\tau}_0)} \log p_\theta(\boldsymbol{\tau}_0) \\
&= -\mathbb{E}_{q(\boldsymbol{\tau}_0)} \log \Big(\int p_\theta(\boldsymbol{\tau}_{0:T}) d\boldsymbol{\tau}_{1:T} \Big) \\
&= -\mathbb{E}_{q(\boldsymbol{\tau}_0)} \log \Big(\int q(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0) \frac{p_\theta(\boldsymbol{\tau}_{0:T})}{q(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0)} d\boldsymbol{\tau}_{1:T} \Big) \\
&= -\mathbb{E}_{q(\boldsymbol{\tau}_0)} \log \Big(\mathbb{E}_{q(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0)} \frac{p_\theta(\boldsymbol{\tau}_{0:T})}{q(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0)} \Big) \\
&\leq -\mathbb{E}_{q(\boldsymbol{\tau}_{0:T})} \log \frac{p_\theta(\boldsymbol{\tau}_{0:T})}{q(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0)} \\
&= \mathbb{E}_{q(\boldsymbol{\tau}_{0:T})} \Big[\log \frac{q(\boldsymbol{\tau}_{1:T}|\boldsymbol{\tau}_0)}{p_\theta(\boldsymbol{\tau}_{0:T})} \Big] = L_{\text{VLB}}. \\
L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{\text{KL}}(q(\boldsymbol{\tau}_T|\boldsymbol{\tau}_0) \parallel p_\theta(\boldsymbol{\tau}_T)) \\
L_t &= D_{\text{KL}}(q(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t+1}, \boldsymbol{\tau}_0) \parallel p_\theta(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t+1})) \\
&\qquad \text{for } 1 \leq t \leq T-1 \text{ and} \\
L_0 &= -\log p_\theta(\boldsymbol{\tau}_0|\boldsymbol{\tau}_1).
\end{aligned} \tag{9}$$

Which enables us to find a tractable parameterization for training, as the KL between two Gaussians is analytically computable.

$$
\begin{aligned}
L_t &= \mathbb{E}_{\boldsymbol{\tau}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}_t,t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\boldsymbol{\tau}_t,\boldsymbol{\tau}_0)-\boldsymbol{\mu}_\theta(\boldsymbol{\tau}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\boldsymbol{\tau}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{\tau}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big)-\frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{\tau}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t,t)\Big)\|^2\Big] \\
&= \mathbb{E}_{\boldsymbol{\tau}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\boldsymbol{\tau}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{\tau}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t,t)\|^2\Big].
\end{aligned}
\tag{10}
$$

After removing the coefficient at the beginning of this objective following (Ho et al., 2020), we arrive at the objective used in the practical algorithm Algorithm 1:

$$
\mathbb{E}_{\boldsymbol{\tau}_0,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{\tau}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t,t)\|^2]. \tag{11}
$$

Furthermore, thanks to the connection between noise conditioned score networks and diffusion models (Song et al., 2020b; Ho et al., 2020), we are able to state that $\epsilon_\theta \propto -\nabla \log p(\boldsymbol{\tau})$:

$$
\begin{aligned}
\mathbf{s}_\theta(\boldsymbol{\tau}_t,t) &\approx \nabla_{\boldsymbol{\tau}_t}\log p(\boldsymbol{\tau}_t) \\
&= \mathbb{E}_{q(\boldsymbol{\tau}_0)}[\nabla_{\boldsymbol{\tau}_t}p(\boldsymbol{\tau}_t|\boldsymbol{\tau}_0)] \\
&= \mathbb{E}_{q(\boldsymbol{\tau}_0)}\Big[-\frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t,t)}{\sqrt{1-\bar{\alpha}_t}}\Big] \\
&= -\frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t,t)}{\sqrt{1-\bar{\alpha}_t}}.
\end{aligned}
\tag{12}
$$

Therefore, by using a variant of $\boldsymbol{\epsilon_\theta}$ conditioned on language to denoise our latent plans, we can effectively model $-\nabla_\tau \mathcal{P}_\beta(\boldsymbol{\tau} \mid \mathcal{R})$ with our diffusion model, iteratively guiding our generated trajectory towards the optimal trajectories conditioned on language.

G PROOF OF 3.1

Proof. The proof is fairly straightforward, and can be shown by translating our definition of suboptimality into the framework utilized by (Nachum et al., 2019). We are then able to leverage their first theorem bounding suboptimality by the Total Variation (TV) between transition distributions to show our result, as TV is bounded by the Lipschitz constant multiplied by the domain of the function.

(Nachum et al., 2019) first define a low level policy generator Ψ which maps from $S \times \tilde{A}$ to Π. Using the high level policy to sample a goal $g_t \sim \pi_{\text{hi}}(g|s_t)$, they use Ψ to translate this to a policy $\pi_t = \Psi(s_t, g_t)$, which samples actions $a_{t+k} \sim \pi_t(a|s_{t+k}, k)$ from $k \in [0, c-1]$. The process is repeated from s_{t+c}. Furthermore, they define an inverse goal generator $\varphi(s, a)$, which infers the goal g that would cause Ψ to yield an action $\tilde{a} = \Psi(s, g)$. The following can then be shown:

Theorem G.1. *If there exists $\varphi : S \times A \to \tilde{A}$ such that,*

$$\sup_{s \in S, a \in A} D_{TV}(P(s'|s,a)||P(s'|s, \Psi(s, \varphi(s,a)))) \leq \epsilon, \tag{13}$$

then $\text{SubOpt}'(\Psi) \leq C\epsilon$*, where* $C = \frac{2\gamma}{(1-\gamma)^2} R_{max}$.

Note that their SubOpt' is different from ours; whilst we defined in terms of the encoder $\mathcal{E}$ and action generator ϕ, they define it in terms of Ψ. Note, however, that the two are equivalent when the temporal stride $c = 1$, as Ψ becomes $\pi_{\text{lo}} = \phi \circ \mathcal{E}$. It is essential to note that when using a goal conditioned imitation learning objective, as we do in this paper, π_{lo} becomes equivalent to an inverse dynamics model $\text{IDM}(s, \mathcal{E}(s)) = a$ and that $\varphi(s, a)$ becomes equivalent to $\mathcal{E}(s')$. This is the key to our proof, as the second term in the total variation of G.1 reduces to

$$\begin{aligned} &P(s'|s, \Psi(s, \varphi(s,a)))) \\ &= P(s'|s, \Psi(s, \mathcal{E}(s'))) \\ &= P(s'|s, a+\epsilon)). \end{aligned} \tag{14}$$

Since we have that the transition dynamics are Lipschitz:

$$\begin{aligned} &\int_{\mathcal{A},\mathcal{S}} |P(s'|s,a) - P(s'|s,a+\epsilon))| \; d\nu \\ &\leq \int_{\mathcal{A},\mathcal{S}} K_f |a - (a+\epsilon)| \, d\nu \\ &= K_f \epsilon \int_{\mathcal{A},\mathcal{S}} d\nu \\ &= K_f \epsilon \, \text{dom}(P(s'|s,a)) \end{aligned} \tag{15}$$

Which we can then plug into 13 to obtain the desired $C = \frac{2\gamma}{(1-\gamma)^2} R_{max} K_f \text{dom}(P(s'|s,a))$. □

H DIFFUSER-2D

Here we give details for our strongest Diffusion-based ablation, which uses Stable Diffusion's VAE for generating latent plans, which outputs a latent 2D feature map, with height and width 1/8 of the original image. Plans are sampled with a temporal stride of 7, such that each trajectory covers a total of 63 timesteps with $t = 0, 7, 14...63$. Overall, generation quality tends to be higher and more temporally coherent than that of the 1D model, but low level details still not precise enough for control from pixels. For examples of model outputs, please refer to subsection J.2.

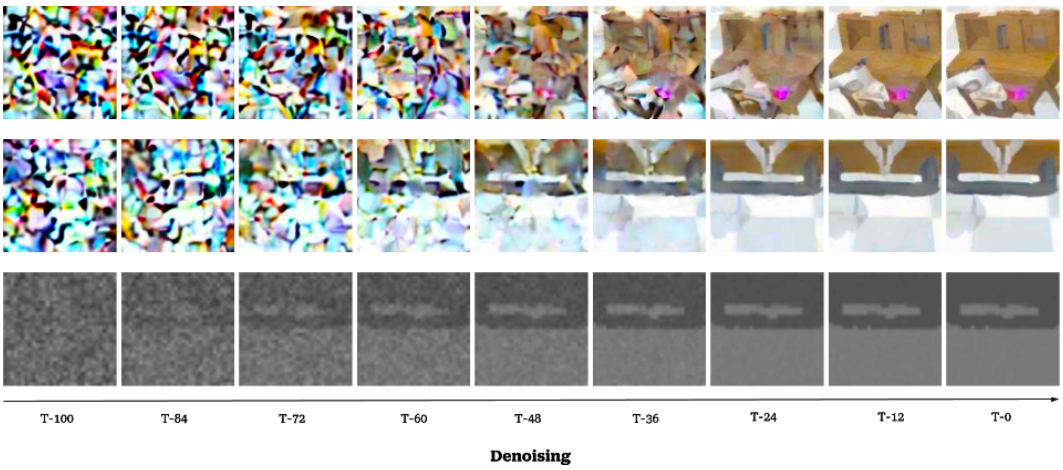


Figure 4: An overview of our Denoising process. In Figure 4 and Figure 5, we give an example of the denoising process of one of our ablations, the Diffuser-2D model. This model utilizes the 2D autoencoder of (Rombach et al., 2022) with (Janner et al., 2022).

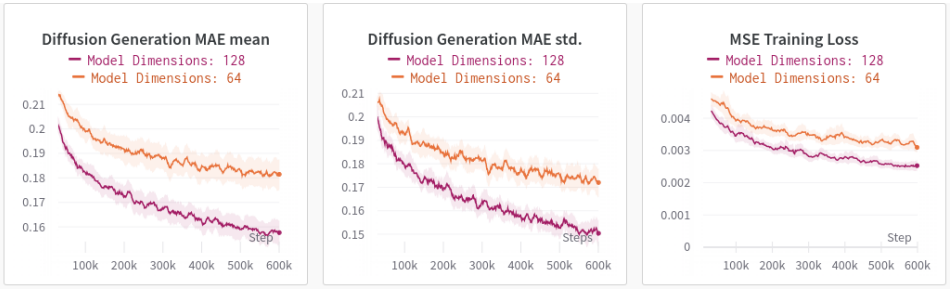


Figure 5: Diffusion Loss Comparison. Here we give study how varying the Diffusion model's size changes the performance of the model. As can be seen, scaling the model from 64 hidden dimensions to 128 strictly increases generation quality, and would likely follow scaling laws observed in (Kaplan et al., 2020).

I TASK DISTRIBUTION

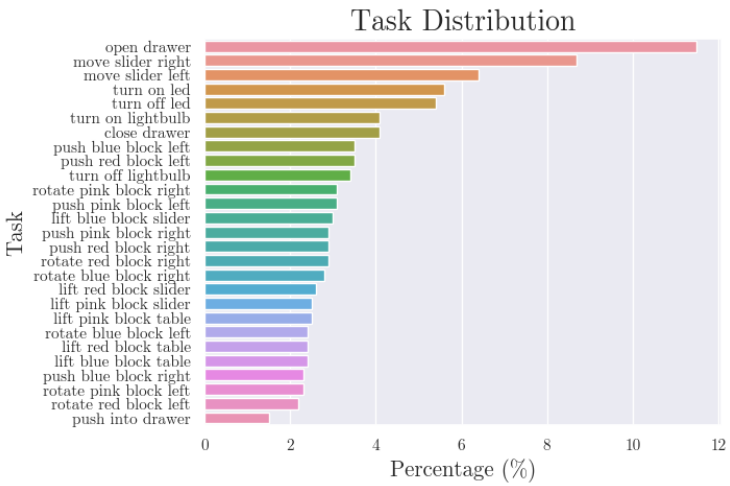
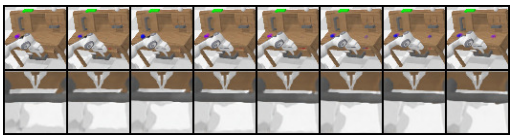


Figure 6: The Evaluation Task Distribution. We visualize the distribution of all the tasks considered in our experiments in Figure 6. Note the long-tailedness of this distribution, and how it skews evaluation scores upwards if one can solve the relatively easier tasks that occur most frequently, such as Open Drawer, Move Slider Right, and Move Slider Left. These tasks only deal with static objects, meaning there is very little generalization that is needed in order to solve these tasks when compared to other block tasks involving randomized block positions.

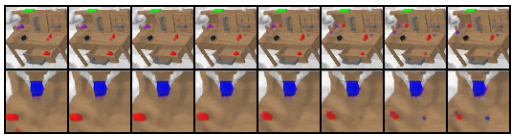
J REPRESENTATION FAILURES

J.1 DIFFUSER-1D (BETA-TC VAE LATENT REPRESENTATION) FAILURES

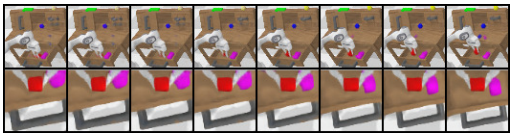
We give a few failure cases of decoded latent plans, where the latent space is given by a trained from scratch β-TC VAE on the CALVIN D-D dataset. The top row of each plan comes from the static camera view, whilst the bottom one comes from the gripper camera view (a camera located at the tool center point of the robot arm). The VAE is trained by concatenating the images in the channel dimension, and compressing to 128 latent dimensions. Plans are sampled with a temporal stride of 9, such that each trajectory covers a total of 63 timesteps with $t = 0, 9, 18...63$. Interestingly, we found that replanning during rollout did not work, precluding the possibility of success on CALVIN with our implementation of this method.



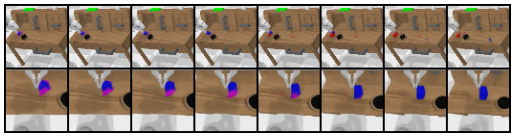
(a) An example of the Close Drawer Task. Notice the flickering block on the top right of the table. Also not the entangled red and blue blocks at the top left of the table.



(b) An example of the Lift Blue Block Slider Task. The gripper view is temporally incoherent, red and blue blocks in slider are entangled.



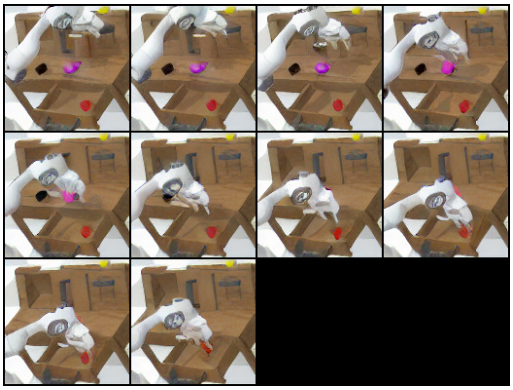
(c) An example of the Lift Red Block Drawer task. Two blocks begin to appear on the table at the end of generation. The red block is also not clearly generated in the first frame.



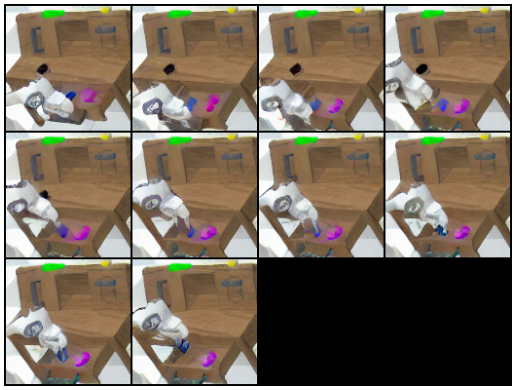
(d) An example of the Push Blue Block Right task. The blue block on the table becomes red by the end of the static view, whereas the opposite happens in the gripper view.

J.2 DIFFUSER-2D (STABLE DIFFUSION LATENT REPRESENTATION) FAILURES

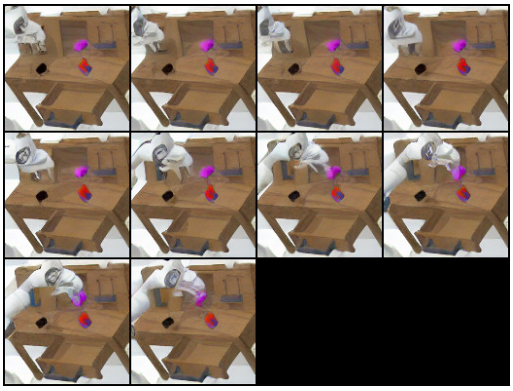
We additionally give some failure cases for Diffuser-2D. For more information on the training of this model, please refer to Appendix H. We also found that replanning during rollout did not work with this model.



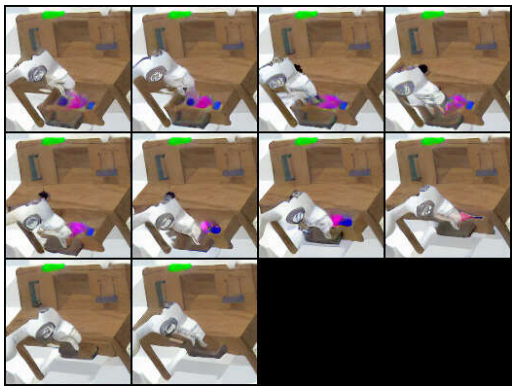
(a) An example of the Lift Red Block Drawer Task. Note the pink block that disappears.



(b) An example of the Lift Blue Block Drawer Task. The gripper arm is entangled with the block.



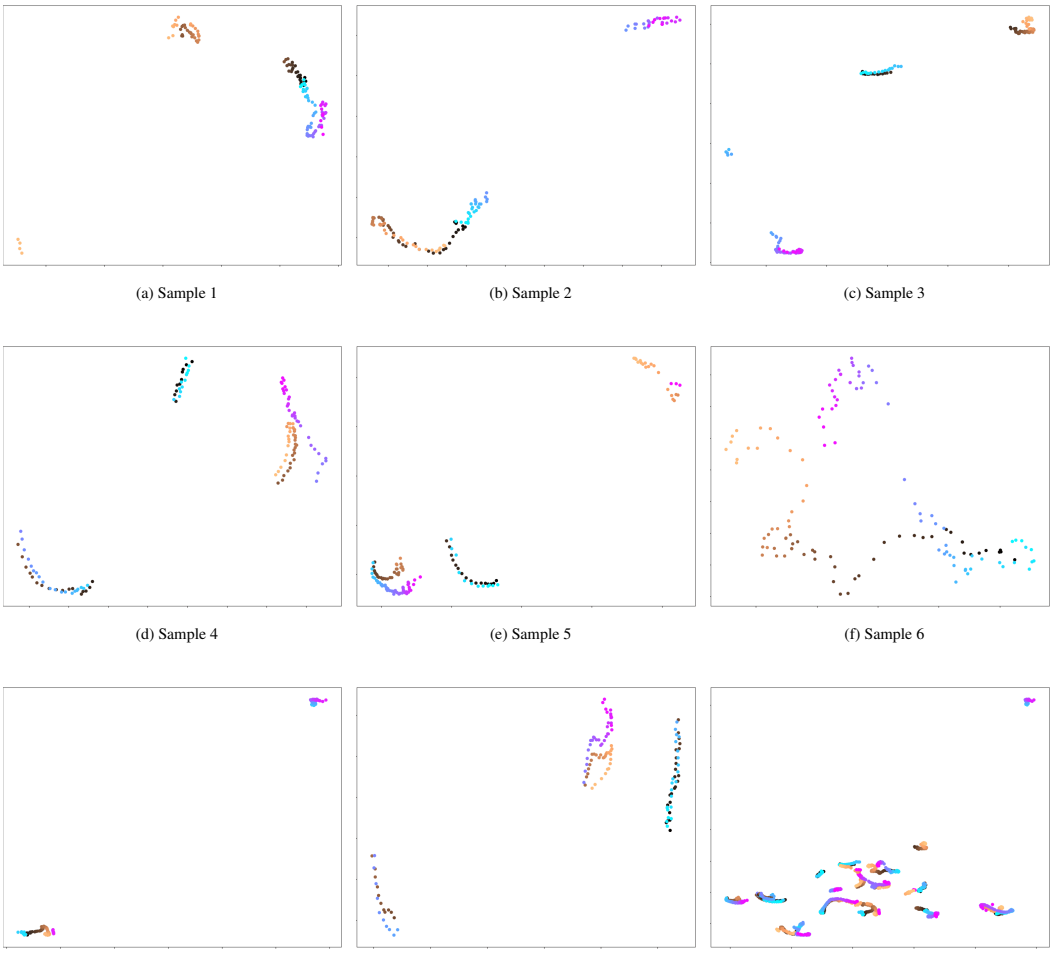
(c) An example of the Lift Pink Block Slider Task. Note the entangled red/blue blocks.



(d) An example of the Close Drawer Task. Note the entangled pink/blue blocks.

K TSNE Comparison between Groud Truth (GT) trajectory and Diffuser-1D (DM) trajectory

In order to better understand whether the representation failures found in Appendix J are a result of the underlying encoder or the diffusion model, we visualize the TSNE embeddings of an encoded successful trajectory from the dataset, which we refer to as a Ground Truth trajectory, and the TSNE embeddings of generated trajectories from Diffuser-1D (DM) in Figure 9. If we observe that the DM's embedddings are fairly close to the GT-VAE's, then we can reasonably presume that the VAE is the failure mode, whereas if the trajectories are wildly different this would imply that the DM is failing to model the VAE's latent distribution properly. Here, all samples other than 6 appear to fairly close, so we suspect that the failure case lies in the underlying latent distribution and not the DM's modeling capabilities. This is further backed by LCD, as we show that by using the proper underlying latent space with a LLP leads to success.



(a) Sample 1 (b) Sample 2 (c) Sample 3

(d) Sample 4 (e) Sample 5 (f) Sample 6

(g) Sample 7 (h) Sample 8 (i) All

Figure 9: TSNE visualization of GT-VAE trajectory vs. Diffuser-1D trajectory, where the purple and light blue color range is the ground truth VAE, and the copper color range is Diffuser-1D. All states are normalized, and all trajectories are taken from the task "lift pink block table".

L HULC LATENT PLAN TSNE

We give TSNE embeddings of several Latent Plans generated during inference by HULC below.

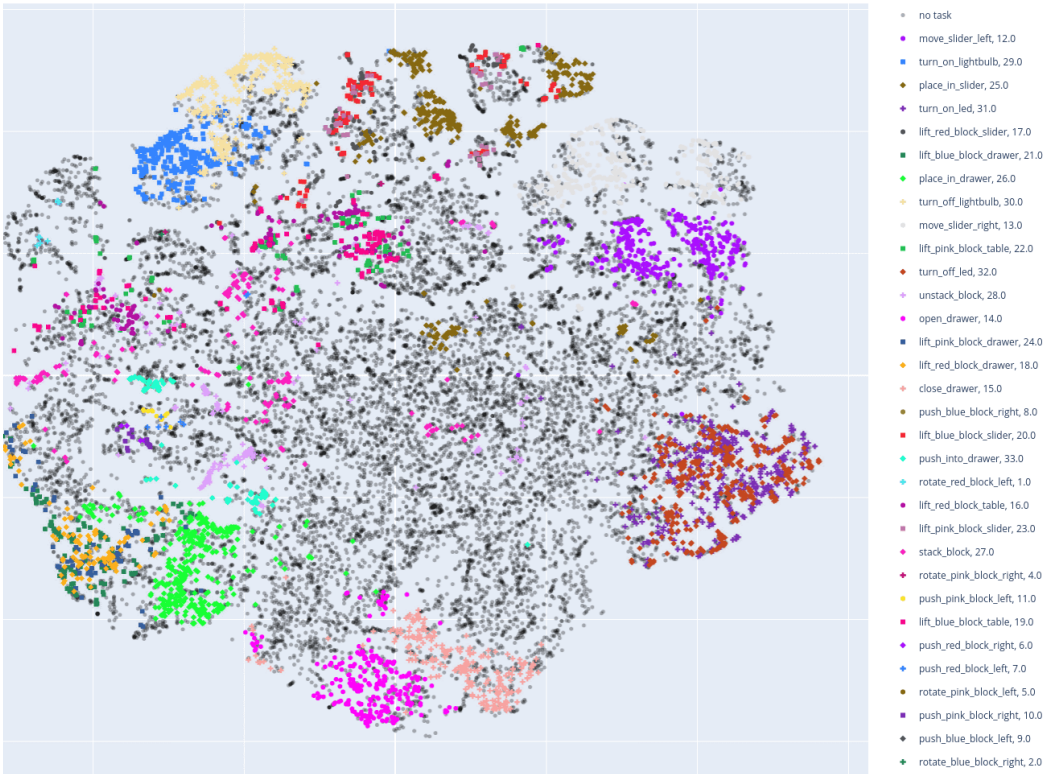


Figure 10: TSNE of Latent Plan. We give a TSNE embedding of the latent plan space of HULC in Figure 10. The latent plan space is the communication layer between the high level policy and low level policy of the HULC model, which corresponds to the intermediate layer between the lower level and lowest level policy in our method. We clarify that this is not the latent goal space that our model does generation in. Our method performs latent generation in the earlier layer from the output of the goal encoder, which corresponds to 32 latent dimensions.

M MODEL CARD FOR LANGUAGE CONTROL DIFFUSION

A hierarchical diffusion model for long horizon language conditioned planning.

M.1 MODEL DETAILS

M.1.1 MODEL DESCRIPTION

A hierarchical diffusion model for long horizon language conditioned planning.

M.2 USES

M.2.1 DIRECT USE

Creating real world robots, controlling agents in video games, solving extended reasoning problems from camera input

M.2.2 DOWNSTREAM USE

Could be deconstructed so as to extract the high level policy for usage, or built upon further by instantiating a multi-level hierarchical policy

M.2.3 OUT-OF-SCOPE USE

Discrimination in real-world decision making, military usage

M.3 BIAS, RISKS, AND LIMITATIONS

Significant research has explored bias and fairness issues with language models (see, e.g., Sheng et al. (2021) and Bender et al. (2021)). Predictions generated by the model may include disturbing and harmful stereotypes across protected classes; identity characteristics; and sensitive, social, and occupational groups.

M.4 TRAINING DETAILS

M.4.1 TRAINING DATA

http://calvin.cs.uni-freiburg.de/

M.5 ENVIRONMENTAL IMPACT

Carbon emissions can be estimated using the Machine Learning Impact calculator presented in Lacoste et al. (2019).

- **Hardware Type:** NVIDIA Titan RTX, NVIDIA A10
- **Hours used:** 9000
- **Cloud Provider:** AWS
- **Compute Region:** us-west-2
- **Carbon Emitted:** 1088.64 kg

M.6 TECHNICAL SPECIFICATIONS

M.6.1 MODEL ARCHITECTURE AND OBJECTIVE

Temporal U-Net, Diffusion objective

M.6.2 COMPUTE INFRASTRUCTURE

Hardware Nvidia Titan RTX , Nvidia A10

Software Pytorch