Language Control Diffusion: Efficiently Scaling through Space, Time, and Tasks

Edwin Zhang
Department of Computer Science
University of California, Santa Barbara
ete@ucsb.edu

Yujie Lu
Department of Computer Science
University of California, Santa Barbara
something@ucsb.edu

Shinda Huang
Department of Computer Science
University of California, Santa Barbara
shinda@ucsb.edu

William Wang
Department of Computer Science
University of California, Santa Barbara
something@ucsb.edu

Amy Zhang
Department of Electrical and Computer Engineering
The University of Texas at Austin
Meta AI
something@uta.edu

Abstract

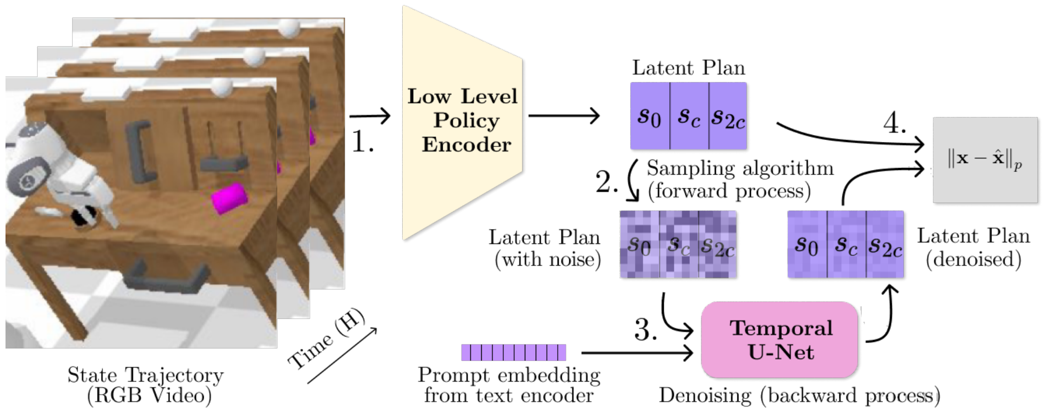
Training generalist agents is difficult across several axes, requiring us to deal with high-dimensional inputs (space), long horizons (time), and multiple novel tasks. Recent advances with architectures have allowed for improved scaling along one or two of these dimensions, but are still computationally prohibitive to use. In this paper, we propose to address all three axes by leveraging **L**anguage to **C**ontrol **D**iffusion models as a hierarchical planner conditioned on language (LCD). We effectively and efficiently scale diffusion models for planning in extended temporal, state, and task dimensions to tackle long horizon control problems conditioned on natural language instructions. We compare LCD with other state-of-the-art models on the CALVIN language robotics benchmark and find that LCD outperforms other SOTA methods in multi-task success rates with single task success rates of 6.1% higher than the previous state-of-the-art, whilst improving inference speed over other comparable diffusion models by 3.3x~15x. We show that LCD can successfully leverage the unique strength of diffusion models to produce coherent long range plans while addressing their weakness in generating low-level details and control.

1 Introduction

Generalist agents are characterized by the ability to plan over long horizons, understand and respond to human feedback, and generalize to new tasks based on that feedback. Language conditioning is an intuitive way to specify tasks, with a built-in structure enabling generalization to new tasks. There have been many recent developments to leverage language for robotics and downstream decision-making and control [1–6]. However, while language is useful for task specification and generalization, it will not necessarily help with planning over long horizons.

Code and visualizations available at `https://language-control-diffusion.github.io/website/`. Our task is best exemplified by videos available at this url.

Preprint. Under review.



1.
Low Level Policy Encoder
Latent Plan
s_0 s_c s_{2c}
2. Sampling algorithm (forward process)
Latent Plan (with noise)
Latent Plan (denoised)
3.
Temporal U-Net
4.
$\|\mathbf{x}-\hat{\mathbf{x}}\|_p$
State Trajectory (RGB Video)
Time (H)
Prompt embedding from text encoder
Denoising (backward process)

Figure 1: An overview of our high-level policy training pipeline. The frozen low-level policy encoder is used to encode a subsampled sequence of RGB observations into a lower dimensional latent space (1), which will be used later on as goals for the goal-conditioned low-level policy. We then noise this latent plan according to a uniformly sampled timestep from the diffusion process' variance schedule (2), and train a Temporal U-Net conditioned on natural language embeddings from a frozen upstream large language model to reverse the noising process (3), effectively learning how to conditionally denoise the latent plan. To train the U-Net, one can simply use the p-norm between the predicted latent plan and the ground truth latent plan as the loss (4). We use $p = 1$ in practice following [7].

Current methods have a few pitfalls. Many existing language-conditioned control methods assume access to a high-level discrete action space (e.g. switch on the stove, walk to the kitchen) provided by a lower level skill oracle [1, 2, 8, 9]. The generative large language model (LLM) will typically decompose some high-level language instruction into a set of predefined skills, which are then executed by a control policy or oracle. However, a fixed set of predefined skills may preclude the ability to generalize to novel environments and tasks. In addition, one of the most important design considerations for these models is deciding what level of abstraction the communication protocol between the LLM and the policy should take. A tradeoff arises from this question, where one must decide between the amount of reasoning to distribute over the LM versus the underlying policy. We can increase the burden on the LLM by having it output text at a low-level of abstraction that is extremely simple to follow, or we can increase the burden on the policy by having it execute text at a higher level of abstraction. Much prior work has been done on increasing the reasoning load of the language encoder by leveraging the capabilities of LLMs to generate planning code [10], calculate affordances of separate skills [1], or perform chain of thought reasoning during rollout [11], but scaling the low-level policy for interpreting higher level instruction has remained relatively underexplored, an essential property necessary when the LLM fails to reason or gives overly broad instructions.

One promising candidate that has emerged for long horizon planning is denoising diffusion models. Text-to-image diffusion models [12–15] have recently been able to successfully take advantage of LLMs to generate incredibly detailed scenes. In addition, diffusion models have recently been proposed and successfully applied for low-dimensional, long-horizon planning and offline reinforcement learning (RL) [7, 16]. In particular, Diffuser [7] has emerged as an especially promising planner with several properties suited for language conditioned control: flexibility in task specification, temporal compositionality, and ability to scale to long-horizon settings.

However, Diffuser [7] does not work directly out of the box when scaling to pixels. This is perhaps unsurprising, as a high-dimensional Diffuser is effectively a text-to-video diffusion model, and even internet-scale video diffusion models have demonstrated only mediocre understanding of physics and temporal coherence over fine details [17, 18]. This is because the training objective for generative models has no notion of the underlying task, meaning they must model the entire scene with equal weighting. This includes potentially task-irrelevant details that may hinder or even prevent solving the actual control problem [19, 20], which leads to catastrophic failure in control where fine details and precision are essential. Additionally, training a video diffusion model is generally computationally expensive and may take several days or weeks to train, which leaves such an approach out of reach to most researchers [21]. This problem is exacerbated when considering that at inference time multiple forward passes (often >20) are required for generating even a single state and action, meaning full text to video diffusion models are inefficient and likely impractical for the real-time sampling demands of robotics and control. In summary, diffusion models are computationally prohibitive to run on high-dimensional input spaces and also tend to be inaccurate at low-level control.

We address both of these issues by proposing the usage of a hierarchical diffusion policy. By utilizing the representation of a goal-conditioned policy as a low-level policy (LLP), we can effectively solve both the representation and efficiency issue by outputting states in a low-dimensional goal space directly into the LLP. This also allows us to arbitrarily scale the difficulty of the low-level policy learning problem by controlling the horizon length from which the goal state is set from the current state. This direction is promising, as it avoids defining the communication layer altogether and enables generating actions directly from high-level text without a human-defined communication protocol in between. In consequence, this hierarchical approach allows us to scale the diffusion model along three orthogonal axes: the **Spatial dimension** through a low-dimensional representation that has been purposely optimized for control, the **Time dimension** through a temporal abstraction enabled by utilizing a goal conditioned low-level policy (LLP), as the LLP can use goal states several timesteps away from the current state, and the **Task dimension** through language, as the diffusion model acts as a powerful interpreter for plugging any large language model into control. In addition, the entire pipeline is extremely fast and simple to train, as we utilize DDIM [22], a temporal abstraction on the horizon, as well as a low-dimensional representation for generation. We are able to achieve an average of 88.7% success rate across all tasks on the challenging CALVIN benchmark. Additionally, we elucidate where diffusion models for text-to-control work well and highlight their limitations. Finally, we explore the grounding between language and state-actions via evaluation of the task generalization capabilities of our hierarchical diffusion policy.

In summary, our core contributions are: **1)** We propose an effective method for improving diffusion policies' scaling to high-dimensional state spaces, longer time horizons, and more tasks by incorporating language and by scaling to pixel-based control. **2)** We significantly improve both the training and inference time of diffusion policies through DDIM, temporal abstraction, and careful analysis and choice of image encoder. **3)** A successful instantiation of our language control diffusion model that substantially outperforms the state of the art on the challenging CALVIN benchmark.

2 Background

Reinforcement Learning. We formulate the environment as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, p)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R}$, discount factor γ, and transition dynamics p. At each time step t, agents observe a state $s \in \mathcal{S} \subseteq \mathbb{R}^n$, take an action $a \in \mathcal{A} \subseteq \mathbb{R}^m$, and transition to a new state s' with reward r following $s', r \sim p(\cdot, \cdot|s, a)$. The goal of RL is then to learn either a deterministic policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ or a stochastic policy $a \sim \pi(\cdot|s)$ that maximises the policy objective, $J(\pi) = \mathbb{E}_{a\sim\pi, s'\sim p} \sum_{t=0}^{\infty} \gamma^t r_t$.

Language Conditioned RL. We consider a language-conditioned RL setting where we assume that the true reward function $\mathcal{R}$ is unknown, and must be inferred from a natural language instruction $L \in \mathscr{L}$. Formally, let $\mathcal{F}$ be the function space of $\mathcal{R}$. Then the goal becomes learning an operator from the language instruction to a reward function $\psi : \mathscr{L} \mapsto \mathcal{F}$, and maximizing the policy objective conditioned on the reward function $\psi(L)$: $J(\pi(\cdot \mid s, \mathcal{R})) = \mathbb{E}_{a\sim\pi, s\sim p} \sum_{t=0}^{\infty} \gamma^t r_t$. This formulation can be seen as a contextual MDP [23], where language is seen as a context variable that affects reward but not dynamics. Note that the space of tasks that can be specified by language is much larger than that of reward, due to the Markov restriction of the latter [24]. For example, "pour the milk" and "pour the milk after five o'clock" are both valid instructions, but are indistinguishable from a reward function if the state does not contain temporal information. We assume access to a prior collected dataset $\mathcal{D}$ of N annotated trajectories $\tau_i = \langle (s_0, a_0, ...s_T), L_i \rangle$. The language conditioned policy π_β, or the behavior policy, is defined to be the policy that generates the aforementioned dataset. This general setting can be further restricted to prohibit environment interaction, which recovers offline RL or imitation learning (IL). In this paper, we assume access to a dataset of expert trajectories, such that $\pi_\beta =$ optimal policy $\pi^\star$. Although this may seem like a strong assumption, the setting still poses a challenging learning problem as many unseen states will be encountered during evaluation and must be generalized to. Several prior methods have failed in this setting [25, 26].

Goal Conditioned Imitation Learning and Hierarchical RL. Goal-conditioned reinforcement learning (RL) is a subfield of RL that focuses on learning policies that can achieve specific goals or objectives, rather than simply maximizing the cumulative reward. This approach has been extensively studied in various forms in the literature [2, 27–30], although we focus on the hierarchical approach

[31]. Following [32], we formulate the framework in a two-level manner where a high level policy $\pi_{\text{hi}}(\tilde{a}|s)$ samples a goal state $g = \tilde{a}$ every c time steps. We refer to c as the temporal stride. A low level policy (LLP) $\pi_{\text{lo}}(a|s_t, g_t)$ then attempts to reach this state, which can be trained through hindsight relabelling [33] the offline dataset $\mathcal{D}$.

3 The Language Control Diffusion (LCD) Framework

In this section, we develop the Language Control Diffusion framework to enable scaling to longer horizons, improve the generalization capabilities of current language conditioned policies by avoiding the usage of a predefined low level skill oracle, and sidestep the computational prohibitiveness of training diffusion models for control from high-dimensional state spaces by proposing the usage of a vastly more efficient hierarchical RL framework. We start by deriving the high-level diffusion policy training objective, and go on to give a theoretical analysis on the suboptimality bounds of our approach by making the mild assumption of Lipschitz transition dynamics. We then solidify this theoretical framework into a practical algorithm by describing the implementation details of both the high-level and low-level policy. Here we also detail the key components of our method that most heavily affect training and inference efficiency, and the specific model architectures used in our implementation for the CALVIN benchmark [34].

3.1 Diffusion Policies in Hierarchical RL

High-level Diffusion Policy Objective. We first describe our problem formulation and framework in detail. Since we assume that our dataset is optimal, the policy objective reduces to imitation learning:

$$\min_{\pi} \mathbb{E}_{s,\mathcal{R}\sim\mathcal{D}} \left[D_{\text{KL}} \left(\pi_\beta(\cdot \mid s, \mathcal{R}), \pi(\cdot \mid s, \mathcal{R}) \right) \right]. \tag{1}$$

As we tackle the problem from a planning perspective, we define a state trajectory generator as $\mathcal{P}$ and switch the atomic object from actions to state trajectories $\boldsymbol{\tau} = (s_0, s_1, ..., s_T)$. Thus we aim to minimize the following KL:

$$\min_{\mathcal{P}} D_{\text{KL}} \left(\mathcal{P}_\beta(\boldsymbol{\tau} \mid \mathcal{R}), \mathcal{P}(\boldsymbol{\tau} \mid \mathcal{R}) \right) = \min_{\mathcal{P}} \mathbb{E}_{\boldsymbol{\tau},\mathcal{R}\sim\mathcal{D}} \left[\log \mathcal{P}_\beta(\boldsymbol{\tau} \mid \mathcal{R}) - \log \mathcal{P}(\boldsymbol{\tau} \mid \mathcal{R}) \right]. \tag{2}$$

This can be reformulated into the following diffusion training objective:

$$\min_{\theta} \mathbb{E}_{\boldsymbol{\tau}_0,\boldsymbol{\epsilon}} [\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{\tau}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2]. \tag{3}$$

Here t refers to a uniformly sampled diffusion process timestep, $\boldsymbol{\epsilon}_t$ refers to noise sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $\boldsymbol{\tau}_0$ refers to the original denoised trajectory $\boldsymbol{\tau}$, α_t denotes a diffusion variance schedule and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. We refer to Appendix C for a more detailed derivation of our objective.

Near Optimality Guarantees. In order to theoretically justify our approach and show that we can safely do diffusion in the LLP encoder latent space without loss of optimality up to some error ϵ, we prove that a near optimal policy is always recoverable with the LLP policy encoder under some mild conditions: that our low-level policy has closely imitated our expert dataset and that the transition dynamics are Lipschitz smooth. Lipschitz transition dynamics is a fairly mild assumption to make [35, 36]. Several continuous control environments will exhibit a Lipschitz transition function, as well as many robotics domains. If an environment is not Lipschitz, it can be argued to be in some sense unsafe [37]. Moreover, one could then impose additional action constraints on such an environment to make it Lipschitz again.

In this paper, we consider high dimensional control from pixels, and thus factorize our low level policy formulation into $\pi_{\text{lo}}(a|s_t, g_t) := \phi(\mathcal{E}(s_t), g_t)$ where $z := \mathcal{E}(s_t)$ defines an encoder function that maps s into a latent representation z and ϕ translates the representation z into a distribution over actions a given a goal g. Note that ϕ is an arbitrary function, and can be made as simple as a matrix or as complex as another hierarchical model. Let $\pi_{\text{lo}}(s) := \phi \circ \mathcal{E}(s)$ be a deterministic low-level policy. Similar to Nachum et al. [38], we can then define the *sub-optimality* of the action and state space induced by $\tilde{\mathcal{A}} = \tilde{\mathcal{S}} = \mathcal{E}(s)$ to be the difference between the best possible high-level policy $\pi^*_{\text{hi}} = \text{argmax}_{\pi\in\Pi} J(\pi_{\text{hi}})$ within this new abstract latent space and the best policy in general:

$$\text{SubOpt}(\mathcal{E}, \phi) = \sup_{s\in S} V^{\pi^*}(s) - V^{\pi^*_{\text{hi}}}(s). \tag{4}$$

Algorithm 1 Hierarchical Diffusion Policy Training

Input: baseline goal-conditioned policy $\pi_{\text{lo}} := \phi(\mathcal{E}(s_t), g_t)$, diffusion variance schedule α_t, temporal stride c, language model ρ

Output: trained hierarchical policy $\pi(a_t|s_t) := \pi_{\text{lo}}(a_t|s_t, \pi_{\text{hi}}(g_t|s_t))$, where g_t is sampled every c time steps from π_{hi} as the first state in $\boldsymbol{\tau}^c$.

```
1: Collect dataset $\mathcal{D}_{\text{onpolicy}}$ by rolling out trajectories $\boldsymbol{\tau} \sim \pi_{\text{lo}}, \rho$
2: Instantiate $\pi_{\text{hi}}$ as diffusion model $\epsilon_\theta(\boldsymbol{\tau}_{\text{noisy}}, t, \rho(L))$
3: repeat
4:     Sample mini-batch $(\boldsymbol{\tau}, L) = B$ from $\mathcal{D}_{\text{onpolicy}}$.
5:     Subsample $\boldsymbol{\tau}^c = (\mathcal{E}(s_0), \mathcal{E}(s_c), \mathcal{E}(s_{2c}), ..., \mathcal{E}(s_T))$.
6:     Sample diffusion step $t \sim$ Uniform$(\{1, ..., T\})$, noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
7:     Update high-level policy $\pi_{\text{hi}}$ with gradient $-\nabla_\theta \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{\tau}^c + \sqrt{1-\bar{\alpha}_t}\epsilon, t, \rho(L))\|^2$
8: until converged
```

Intuitively, this captures how much potential performance we stand to lose by abstracting our state and action space.

Proposition 3.1. *If the transition function $p(s'|s,a)$ is Lipschitz continuous with constant K_f and $\sup_{s\in S, a\in A} |\pi_{\text{lo}}(s) - a^*| \leq \epsilon$, then*

$$\text{SubOpt}(\mathcal{E}, \phi) \leq \frac{2\gamma}{(1-\gamma)^2} R_{max} K_f \text{dom}(P(s'|s,a))\epsilon. \tag{5}$$

Crucially, this shows that the suboptimality of our policy is bounded by ϵ, and that our framework is able to recover a near-optimal policy with fairly mild assumptions on the environment. We give a detailed proof in Appendix D. The beauty of this result lies in the fact that due to the nature of the LLP's training objective of minimizing reconstruction error, we are directly optimizing for Proposition 3.1's second assumption. Moreover if this training objective converges, it achieves an appromixate π^*-irrelvance abstraction [39]. Therefore, utilizing the LLP encoder space for diffusion is guaranteed to be near-optimal as long as the LLP validation loss is successfully minimized.

3.2 Practical Instantiation of Language Control Diffusion

We now describe our practical implementation and algorithm instantiation in this section. We first give an overview of our high-level policy instantiation before specifying the implementation of our low-level policy. Finally, we describe our model architecture in detail.

High-Level Policy. In Algorithm 1 we clarify the technical details of our high-level diffusion policy training. Before this point, we assume that a low-level policy (LLP) has been trained. Whilst our framework is flexible enough to use any LLP trained arbitrarily as long as it belongs to the factorized encoder family described in Section 3.1, in our practical implementation we first train our LLP through hindsight relabelling. After the policy has been trained, we then take the trained encoder representation and use it to compress the entire offline dataset of states. This is useful from a practical perspective as it caches the computation of encoding the state, often allowing for 100-200x less memory usage at train time. This enables significantly more gradient updates per second and batch sizes, and is largely why our method is drastically more efficient than prior work utilizing diffusion for control. After caching the dataset with the pretrained encoder, we then induce a temporal abstraction by subsampling trajectories from the dataset, taking every c^{th} state in line 5 of Algorithm 1, where c refers to the temporal stride. This further increases efficiency and enables flexible restructuring of computational load between the high-level and low-level policy, simply by changing c. These states can then be trained with the typical diffusion training algorithm, detailed in lines 6-7.

Low-Level Policy. We adopt the HULC architecture [3] for our low-level policy. In HULC, the authors propose an improved version of the hierarchical Multi-Context Imitation Learning (MCIL). Note that this means that our method is a successful instantiation of a three-level hierarchical model, as we generate the goal states with diffusion *into their high-level policy*. HULC utilizes hierarchy by generating global discrete latent plans and learning local policies that are conditioned on this global plan. Their method focuses on several small components for increasing the effectiveness of text-to-control policies, including the architectures used to encode sequences in relabeled imitation learning, the alignment of language and visual representations, and data augmentation and optimization techniques.

Model Architecture. We adopt T5-XXL [40] as our textual encoder, which contains 11B parameters and outputs 4096 dimensional embeddings. T5-XXL has similarly found success in the text to image diffusion model Imagen [17]. We utilize a temporal U-Net [7], which performs 1D convolutions across the time dimension of the latent plan rather than the 2D convolution typical in text-to-image generation. This is motivated by our desire to preserve equivariance along the time dimension but not the state-action dimension. In addition, we modify the architecture in Janner et al. [7] by adding conditioning via cross attention in a fashion that resembles the latent diffusion model [21]. Finally, we use DDIM [22] during inference for increased computational efficiency and faster planning. DDIM uses strided sampling and is able to capture nearly the same level of fidelity as typical DDPM sampling [14] with an order of magnitude speedup. For rolling out the latent plans generated by the denoiser, we resample a new sequence of goals g with temporal stride or frequency c, until either the task is completed successfully or the maximum horizon length is reached. Our low-level policy takes over control between samples, with goals generated by the high-level policy as input.

4 Experiments

In our experiments we aim to answer the following questions: 1) Does a diffusion-based approach perform well for language-conditioned RL? 2) How much efficiency is gained by planning in a latent space? Finally, 3) is a hierarchical instantiation of the diffusion model really necessary, and what tradeoffs are there to consider?

Table 1: **Our main result**. We compare success rates between our diffusion model and prior benchmarks on multitask long-horizon control (MT-LHC) for 34 disparate tasks. We report the mean and standard deviation across 3 seeds for our method with each seed evaluating for 1000 episodes. We bold the highest performing model in each benchmark category.

Horizon	GCBC	MCIL	HULC (LLP only)	Diffuser-1D	Diffuser-2D	Ours
One	64.7 ± 4.0	76.4 ± 1.5	82.6 ± 2.6	47.3 ± 2.5	37.4 ± 3.2	$\mathbf{88.7 \pm 1.5}$
Two	28.4 ± 6.2	48.8 ± 4.1	64.6 ± 2.7	18.8 ± 1.8	9.3 ± 1.3	$\mathbf{69.9 \pm 2.8}$
Three	12.2 ± 4.1	30.1 ± 4.5	47.9 ± 3.2	5.9 ± 0.4	1.3 ± 0.2	$\mathbf{54.5 \pm 5.0}$
Four	4.9 ± 2.0	18.1 ± 3.0	36.4 ± 2.4	2.0 ± 0.5	0.2 ± 0.0	$\mathbf{42.7 \pm 5.2}$
Five	1.3 ± 0.9	9.3 ± 3.5	26.5 ± 1.9	0.5 ± 0.0	0.07 ± 0.09	$\mathbf{32.2 \pm 5.2}$
Avg horizon len	1.11 ± 0.3	1.82 ± 0.2	2.57 ± 0.12	0.74 ± 0.03	0.48 ± 0.09	$\mathbf{2.88 \pm 0.19}$

4.1 Experimental Setup

Dataset and Metric. We evaluate on the CALVIN benchmark [34], a challenging multi-task, long-horizon robotics benchmark. After pretraining our low-level policy on all data, we freeze the policy encoder and train the diffusion temporal U-Net using the frozen encoder. We roll out all of our evaluated policies for 1000 trajectories on all 34 tasks, and all comparisons are evaluated in their official repository[1].

Baselines. As introduced in Section 3.2, we compare against the prior state of the art, HULC and MCIL [3, 41]. HULC provides a strong baseline as it is also a hierarchical method. MCIL (Multicontext Imitation Learning) uses a sequential CVAE to predict next actions from image or language-based goals, by modelling reusable latent sequences of states. GCBC (Goal-conditioned Behavior Cloning) simply performs behavior cloning without explictly modeling any latent variables like MCIL, and represents the performance of using the simplest low-level policy. Finally, Diffuser generates a trajectory through reversing the diffusion process. Diffuser is most comparable to our method, however, it utilizes no hierarchy. To ensure a fair comparison, we rigorously follow their evaluation procedure and build directly from their codebase. MCIL and GCBC results are taken from Mees et al. [3], whilst HULC results are reproduced from the original repository. Diffuser was retrained by following the original author's implementation.

4.2 Performance of LCD

We outperform prior methods on the challenging multi-task long-horizon control (MT-LHC) benchmark, and improve

Table 2: Task generalization of LCD on a collection of five held out tasks. We test with 3 seeds and report the mean and std, evaluating on 20 rollouts per task for a total of 100 evaluations.

Task	Diffuser-1D	Ours
Lift Pink Block Table	31.67 ± 10.27	$\mathbf{55.00 \pm 16.33}$
Lift Red Block Slider	13.35 ± 10.25	$\mathbf{88.33 \pm 8.50}$
Push Red Block Left	1.64 ± 2.35	$\mathbf{35.00 \pm 7.07}$
Push Into Drawer	3.34 ± 4.71	$\mathbf{90.00 \pm 10.80}$
Rotate Blue Block Right	5.00 ± 4.10	$\mathbf{36.67 \pm 14.34}$
Avg SR	12.67 ± 3.56	$\mathbf{61.00 \pm 7.79}$

[1]`https://github.com/lukashermann/hulc/tree/fb14d5461ae54f919d52c0c30131b38f806ef8db`

on the strongest prior model's average performance on horizon length one tasks by $\mathbf{6.1\%}$ as shown in Table 1. In order to further elucidate why our method works better than HULC, we do a deeper analysis on several failure cases that we observed and show how LCD corrects them. In addition, in Appendix K we give a breakdown of individual task success rates, where we find that LCD significantly outperforms HULC in 15 of the 34 tasks, outperforms HULC in 23 of the 34 tasks, and improves on the average success rate of single tasks by $\mathbf{3.33\%}$. Note that the average success rate differs from Table 1, as the distribution of tasks for MT-LHC is not uniform because the CALVIN benchmark filters out infeasible trajectories. A visualization of the first task MT-LHC distribution is provided for future work in Appendix G.

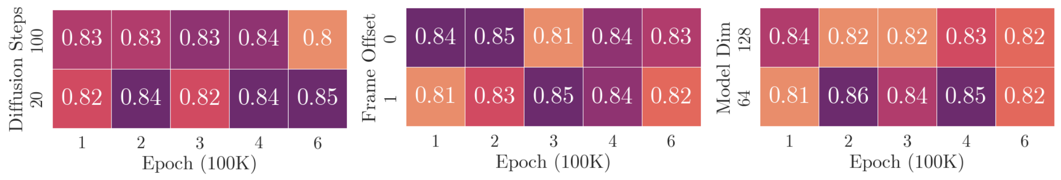
4.3 Task Generalization and Efficiency

We test the task generalization of LCD on a collection of five held out tasks in Table 2. This benchmark is extremely difficult as it requires zero-shot generalization to a new task, which requires generalization across language, actions, and states. We find that LCD is able to successfully generalize, and is able to compose several disparate concepts from the training dataset together such as the color of the blocks, verbs such as lift and push, and object positions and state.

Table 3: Wall clock times for training. Latent dims denotes the size of the latent space that we perform the diffusion generation in. We compare against two variants of Diffuser. Diffuser-1D is the same model as presented in Table 1 which utilizes a VAE trained from scratch on the dataset, whilst Diffuser-2D utilizes a large pretrained VAE from Stable Diffusion [21]. Inference time (sec) refers to the average amount of time taken in seconds to produce an action. LCD is 3.3x-15x faster during inference and 1.5x-3.7x faster during training compared to Diffuser-1D and Diffuser-2D.

	HULC	Diffuser-1D	Diffuser-2D	Ours (HLP only)	Ours (full)
Training (hrs)	82	20.8	49.2	**13.3**	95.3
Inference time (sec)	**0.005**	1.11	5.02	0.333	0.336
Model size	47.1M	74.7M	125.5M	**20.1M**	67.8M
Latent dims	N/A	256	1024	**32**	**32**
Avg ∇ updates/sec	.5	4	2.1	**6.25**	**6.25**

Through the usage of DDIM, temporal abstraction, and low-dimensional generation, we find in Table 3 that LCD is significantly faster during rollout and training than Diffuser. In addition, our method is significantly faster to train than HULC, albeit slower during rollout. This is to be expected, as HULC is not a diffusion model and only requires a single forward pass for generation. However, when comparing to other diffusion models we find that our method is 3.3x-15x faster during inference and 1.5x-3.7x faster during training. All numbers are taken from our own experiments and server for reproducibility and fairness, including baselines.



(a) Diffusion Steps (b) Frame Offset (c) Model Dim

Figure 2: Single task success rates from MT-LHC across different epochs and hyperparameters. Our diffusion model is robust to various hyperparameters such as diffusion steps, frame offset, hidden dimensions, and number of parameters.

4.4 Robustness to Hyperparameters

In Figure 2 we show that our diffusion method is robust to several hyperparameters including the number of diffusion steps/number of function evaluations (NFE), a frame offset o, and the hidden dimensions of the model. NFE is typically a key parameter determining the quality of the representations. However, we found that the performance of our method is relatively insensitive to NFE. Frame offset o controls the spacing between goal state in the input data, which affects the temporal resolution of the representations. We consider augmenting our dataset by adding o during the sampling of our goal state s_{c+o} for potentially improving generalization. However, we find that this effect is more or less negligible. Finally, our method is also robust with respect to the number of parameters, and instantiating a larger model by increasing the model dimensions does not induce overfitting.

4.5 Is Diffusion Really Necessary?

Table 4: Ablation of our method by comparing against a simple four layer MLP with 1024 hidden dimensions (4.8M total parameters) as high level policy. We use the same methodology as in Table 1, and report the mean and standard deviation across 3 seeds for our method with each seed evaluated for 1000 episodes.

Task	MLP	Ours
One	82.1 ± 4.0	$\mathbf{88.7 \pm 1.5}$
Two	61.0 ± 6.25	$\mathbf{69.9 \pm 2.8}$
Three	49.2 ± 4.15	$\mathbf{54.5 \pm 5.0}$
Four	32.1 ± 2.0	$\mathbf{42.7 \pm 5.2}$
Five	25.6 ± 0.9	$\mathbf{32.2 \pm 5.2}$
Avg SR	2.59 ± 0.01	$\mathbf{2.88 \pm 0.19}$

In order to analyze whether diffusion actually improves HULC or if the gain comes just from the usage of a high-level policy, we perform an ablation study in Table 4 by using a simple MLP as a high-level policy, which receives the ground truth validation language embeddings as well as the current state, and attempts to predict the next goal state. An example of the ground truth language is "go push the red block right", which is never explicitly encountered during training. Instead, similar prompts of the same task are given, such as "slide right the red block". We find that even given the ground truth validation language embeddings (the MLP does not need to generalize across language), this ablation significantly underperforms LCD, and slightly underperforms HULC. This suggests that our gains in performance over HULC truly do come from a better modeling of the goal space by diffusion.

4.6 Summary of Critical Findings

Diffuser fails to plan in original state space. After investigating the performance of diffuser, our findings in Figure 3 indicate that Diffuser fails to successfully replan in high dimensional state spaces when using a Variational Autoencoder (VAE). Based on our observations, we hypothesize that the failure of Diffuser to replan successfully is due to the diffusion objective enabling interpolation between low density regions in the VAE's latent space by the nature of the training objective smoothing the original probability distribution of trajectories. In practice, this interpolation between low density regions corresponds to physically infeasible trajectories. This suggests that interpolation between low density regions in the VAE's latent space is a significant factor in the failure of diffuser to plan and replan successfully. Our results highlight the need for better representation, and for further research of scaling diffusion models to high dimensional control.

Representation is essential for effective diffusion. We find that having a good representation is essential for effective diffusion in general control settings, as evidenced in Figure 3. A good representation greatly eases the learning difficulty on the diffusion model by reducing the number of dimensions needed to model, which in turn increases the information density and makes it easier for the model to learn the underlying dynamics of the system. The good representation likely massages the regions of low density latent space between feasible trajectories into areas that still correspond to physically feasible regions when decoded by the low level policy. This enables the model to generalize well to unseen situations, improving its robustness and flexibility. Additionally, by having a good representation, the diffusion model is able to successfully generalize to new scenarios, which is crucial for the model's effectiveness in a real-world application. Our results highlight the importance of latent representation in the diffusion process.

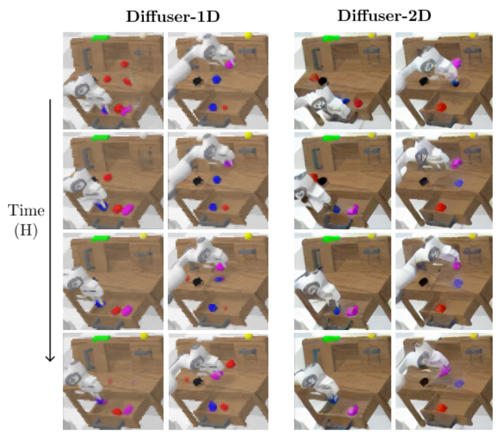


Figure 3: Denoised Latent Representations. Directly using latent diffusion models fails. Hallucination occurs on a β-TC VAE trained from scratch on the CALVIN dataset (Diffuser-1D), and loss of fine details occurs with SD v1.4's [21] internet-scale pretrained autoencoder (Diffuser-2D). For more and enlarged samples please refer to Appendix H.

The encoder representation of deep goal-conditioned RL models can be effectively used for hierarchical RL. Our results imply that the encoder representation of deep reinforcement learning models have the potential to be effectively utilized in hierarchical reinforcement learning. In modern deep policy learning from pixels, we propose an effective task-aware representation can be extracted by using the latent space of an intermediate layer as an encoder. Although we have demonstrated the effectiveness of this approach by instantiating it successfully in a hierarchical diffusion model, this approach is general and can be applied to any generative model in hierarchical RL. Our results suggest that using a shared policy encoder between the high and low level policies can improve the effectiveness and efficiency of generative modelling in hierarchical RL.

5 Related Work

Text-to-Control Models. Text-to-control models or language-conditioned policies [4–6] have been explored in the RL community for improving generalization to novel tasks and environments [6, 42–45]. Although they are not the only way to incorporate external knowledge in the form of text to decision making tasks [46, 47], they remain one of the most popular [48–50]. However, language grounding in RL remains notoriously difficult, as language vastly underspecifies all possible configurations of a corresponding state [51]. Modern text to control models often still struggle with long-horizon language commands and misinterpret the language instruction. Hu et al. [52] 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. [29] consider training a language encoder-decoder for policy shaping, Hanjie et al. [53] utilize an attention module conditioned on textual entities for strong generalization and better language grounding. Zhong et al. [54] propose a model-based objective to deal with sparse reward settings with language descriptions and Mu et al. [55] also tackle the sparse reward settings through the usage of intrinsic motivation with bonuses added on novel language descriptions. However, only Hu et al. [52] consider the long-horizon setting, and they do not consider a high-dimensional state space. Jang et al. [56] carefully examine the importance of diverse and massive data collection in enabling task generalization through language and propose a FiLM conditioned CNN-MLP model [57]. Much work has attempted the usage of more data and compute for control [58–60]. However, none of these methods consider the usage of diffusion models as the medium between language and RL.

Diffusion Models. Diffusion models such as DALL-E 2 [13] and GLIDE [12] have recently shown promise as generative models, with state-of-the-art text-to-image generation results demonstrating a surprisingly deep understanding of semantic relationships between objects and the high fidelity generation of novel scenes. Stable diffusion, an instantiation of latent diffusion [21], has also achieved great success and is somewhat related to our method as they also consider performing the denoising generation in a smaller latent space, albeit with a variational autoencoder [61] rather than a low level policy encoder. Given the success of denoising diffusion probabilistic models [14] in text-to-image synthesis [62], the diffusion model has been further explored in both discrete and continuous data domains, including image and video synthesis [15, 63], text generation [64], and time series [65]. Video generation models are especially relevant to this work, as they are a direct analogue of diffusion planning model Diffuser [7] in pixel space without actions. Diffuser first proposed to transform decision making into inpainting and utilize diffusion models to solve this problem, which much work has followed up on [66–68]. Specifically, 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, [16] 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.

6 Conclusion and Limitations

Learning atomic sub-skills through language is critical to scaling to more complex and open environments. We explore solving this problem through learned state and temporal abstractions, and show that the strengths of diffusion models can be leveraged for long horizon plans, and their weaknesses at low-level detail generation can be managed through learning low-level, goal-conditioned policies with imitation learning. Experiments and qualitative analysis demonstrate the simplicity and effectiveness of our model, showing that LCD can achieve state-of-the-art performance on a competitive language-conditioned control benchmark from rich observations, over long horizons, and generalize to new scenarios. For future work, one could extend LCD to incorporate additional levels of hierarchy and scale to even longer horizons. Additionally, a deeper analysis on the interplay of different representations and the diffusion model could be performed. Finally, one could further probe the task generalization, and potentially improve generalization on the language side by leveraging better pre-trained models and incorporating that distilled knowledge to improve reasoning and planning for control.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL `https://arxiv.org/abs/2204.01691`.

[2] Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

[3] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data, 2022. URL `https://arxiv.org/abs/2204.06252`.

[4] Corey Lynch and Pierre Sermanet. Grounding language in play. *CoRR*, abs/2005.07648, 2020. URL `https://arxiv.org/abs/2005.07648`.

[5] Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. *Language as an Abstraction for Hierarchical Deep Reinforcement Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[6] Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Language-conditioned goal generation: a new approach to language grounding for rl. *LAREL Workshop 2020*, abs/2006.07043, 2020.

[7] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis, 2022. URL `https://arxiv.org/abs/2205.09991`.

[8] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *ICML*, 2022.

[9] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[10] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[11] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[12] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16784–16804. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/nichol22a.html`.

[13] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL `https://arxiv.org/abs/2204.06125`.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

[15] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. URL `https://arxiv.org/abs/2204.03458`.

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

[17] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[18] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[19] 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.

[20] 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*, 2022.

[21] 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.

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

[23] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[24] David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the Expressivity of Markov Reward. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7799–7812. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/4079016d940210b4ae9ae7d41c4a2065-Paper.pdf`.

[25] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[27] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress, 2018. URL `https://arxiv.org/abs/1806.06877`.

[28] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations, 2021. URL `https://arxiv.org/abs/2102.06177`.

[29] Brent Harrison, Upol Ehsan, and Mark O Riedl. Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616*, 2017.

[30] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/schaul15.html`.

[31] 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.

[32] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[33] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[34] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022.

[35] Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 264–273. PMLR, 2018.

[36] Hassan K. Khalil. Nonlinear systems third edition. 2008.

[37] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

[38] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=H1emus0qF7`.

[39] Lihong Li, Thomas Walsh, and Michael Littman. Towards a unified theory of state abstraction for mdps. 01 2006.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[41] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[42] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3D world. *arXiv preprint arXiv:1706.06551*, 2017.

[43] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

[44] Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Emergent systematic generalization in a situated agent, 2019.

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

[46] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

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

[48] 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, 2020.

[49] 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.

[50] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

[51] Willard Van Orman Quine. Word and object mit press. *Cambridge MA*, 1960.

[52] Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. Hierarchical decision making by generating and following natural language instructions. *Advances in neural information processing systems*, 32, 2019.

[53] Austin W Hanjie, Victor Y Zhong, and Karthik Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 4051–4062. PMLR, 2021.

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

[55] Jesse Mu, Victor Zhong, Roberta Raileanu, Minqi Jiang, Noah Goodman, Tim Rocktäschel, and Edward Grefenstette. Improving intrinsic exploration with language abstractions. *arXiv preprint arXiv:2202.08938*, 2022.

[56] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[57] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[58] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[59] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[60] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

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

[62] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/sohl-dickstein15.html`.

[63] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. URL `http://jmlr.org/papers/v23/21-0635.html`.

[64] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation, 2022. URL `https://arxiv.org/abs/2205.14217`.

[65] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. 2021. doi: 10.48550/ARXIV.2101.12072. URL `https://arxiv.org/abs/2101.12072`.

[66] Yilun Dai, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*, 2023.

[67] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[68] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

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

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

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

[72] 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*, 2020.

[73] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Appendix

Outline of the Appendix

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 exmaine 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 Hyper-parameter settings and training details

For all methods we proposed in Table 1, Table 2, Table 6, 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.

B.1 HP and training details for methods in Table 1 and Table 6.

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 Table 6.

C Training Objective Derivation

To model this, we turn to diffusion models [69], whom we borrow much of the following derivation from. Inspired by non-equilibrium thermodynamics, the common forms of diffusion models [14, 22, 70] 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 [70]. Since the backwards density also follows a Gaussian, it suffices to predict μ_θ and Σ_θ which parameterize the backwards distribution:

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

In practice, Σ_θ is often fixed to constants, but can also be learned through reparameterization. Following [14] we consider learning only μ_θ, which can be computed just as a function of τ_t and $\epsilon_\theta(\tau_t, t)$. One can derive that $\boldsymbol{\tau}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\tau}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ for $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, 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 ϵ given by the tractable backwards density. Assuming we have ϵ_θ, 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 [71]:

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

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 [14], we arrive at the objective used in the practical algorithm **??**:

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

Furthermore, thanks to the connection between noise conditioned score networks and diffusion models [14, 72], 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 ϵ_θ 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.

D 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 [38]. 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.

[38] 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 D.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 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 D.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 $C = \frac{2\gamma}{(1-\gamma)^2} R_{max} K_f \text{dom}(P(s'|s,a))$. □

E Inference Pipeline

We give an overview of LCD's inference pipeline below. For our ablations, we are also able to use this pipeline to decode the latent states into pixel space to analyze and interpret the plans generated by the denoising autoencoder. We give examples of these decoded plans in Appendix H.

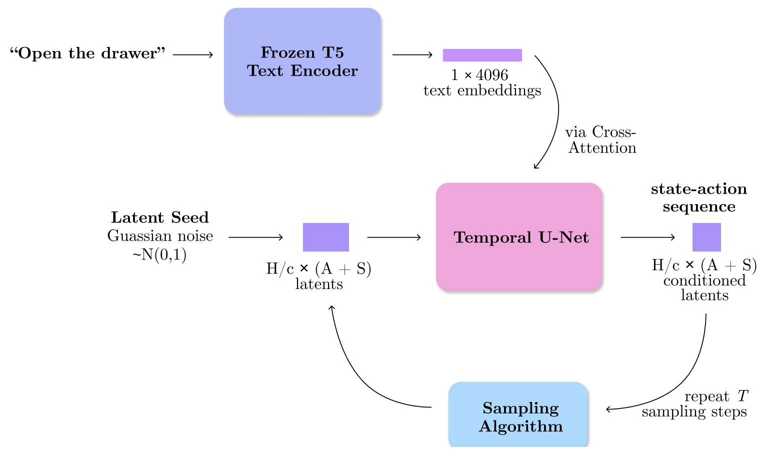


Figure 4: An overview of our inference pipeline. We pass sampled noise to our denoising autoencoder along with an encoded language prompt. The autoencoder is instantiated as a temporal U-Net. By repeating this process iteratively, we are able to generate high-fidelity latent plans conditioned on language.

F 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 H.2.

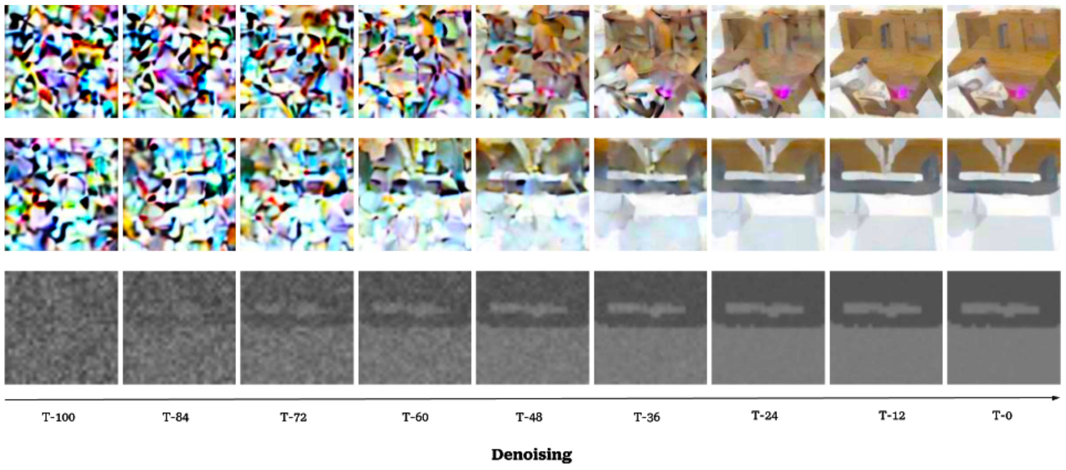


Figure 5: An overview of our Denoising process. In Figure 5 and Figure 6, we give an example of the denoising process of one of our ablations, the Diffuser-2D model. This model utilizes the 2D autoencoder of [21] with [7].

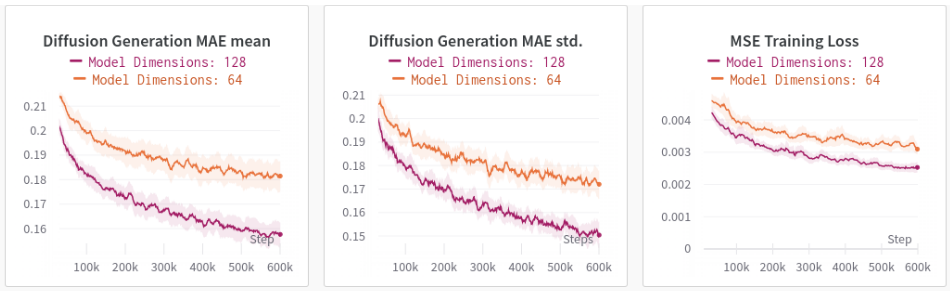


Figure 6: Diffusion Loss Comparison. Here we give study how varying the Diffusion model's size changes the performance of the model. As can be seen, scaling the model from 64 hidden dimensions to 128 strictly increases generation quality, and would likely follow scaling laws observed in [73].

G Task Distribution

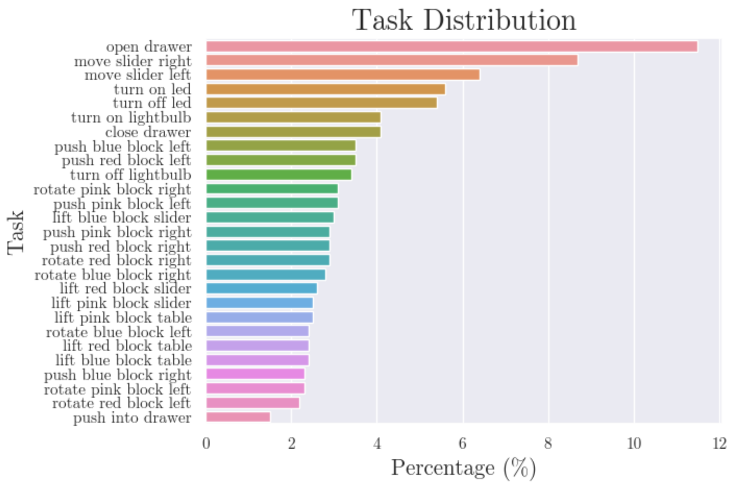
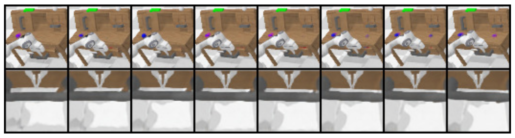


Figure 7: The Evaluation Task Distribution. We visualize the distribution of all the tasks considered in our experiments in Figure 7. Note the long-tailedness of this distribution, and how it skews evaluation scores upwards if one can solve the relatively easier tasks that occur most frequently, such 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.

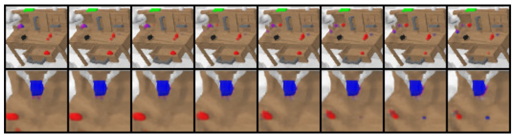
H Representation Failures

H.1 Diffuser-1D (β-TC VAE Latent Representation) Failures

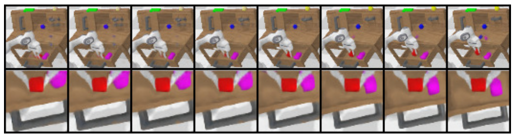
We give a few failure cases of decoded latent plans, where the latent space is given by a trained from scratch β-TC VAE on the CALVIN D-D dataset. The top row of each plan comes from the static camera view, whilst the bottom one comes from the gripper camera view (a camera located at the tool center 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.



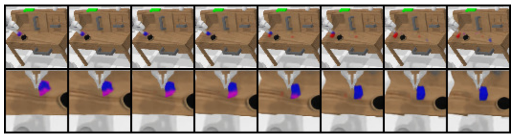
(a) An example of the Close Drawer Task. Notice the flickering block on the top right of the table. Also not the entangled red and blue blocks at the top left of the table.



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



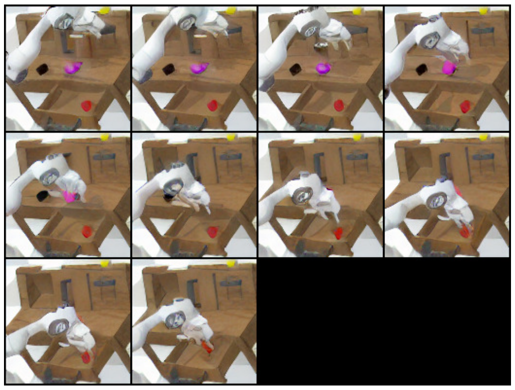
(c) An example of the Lift Red Block Drawer task. Two blocks begin to appear on the table at the end of generation. The red block is also not clearly generated in the first frame.

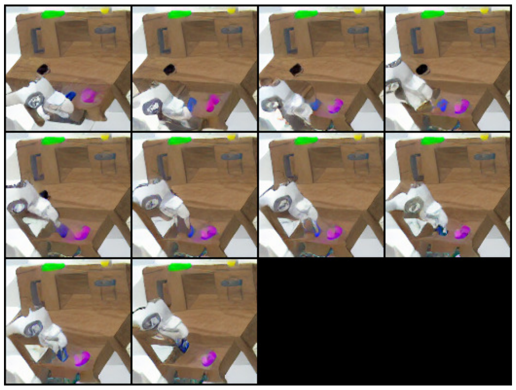


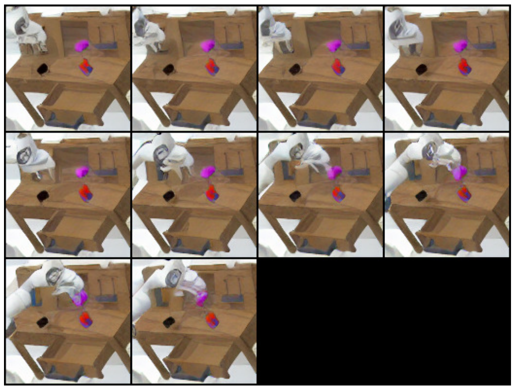
(d) An example of the Push Blue Block Right task. The blue block on the table becomes red by the end of the static view, whereas the opposite happens in the gripper view.

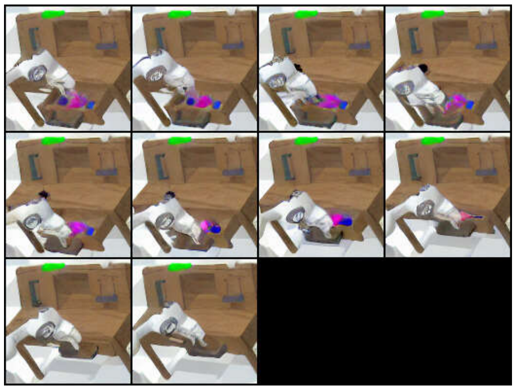
H.2 Diffuser-2D (Stable Diffusion Latent Representation) Failures

We additionally give some failure cases for Diffuser-2D. For more information on the training of this model, please refer to Appendix F. 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.

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

In order to better understand whether the representation failures found in Appendix H 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 10. 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.

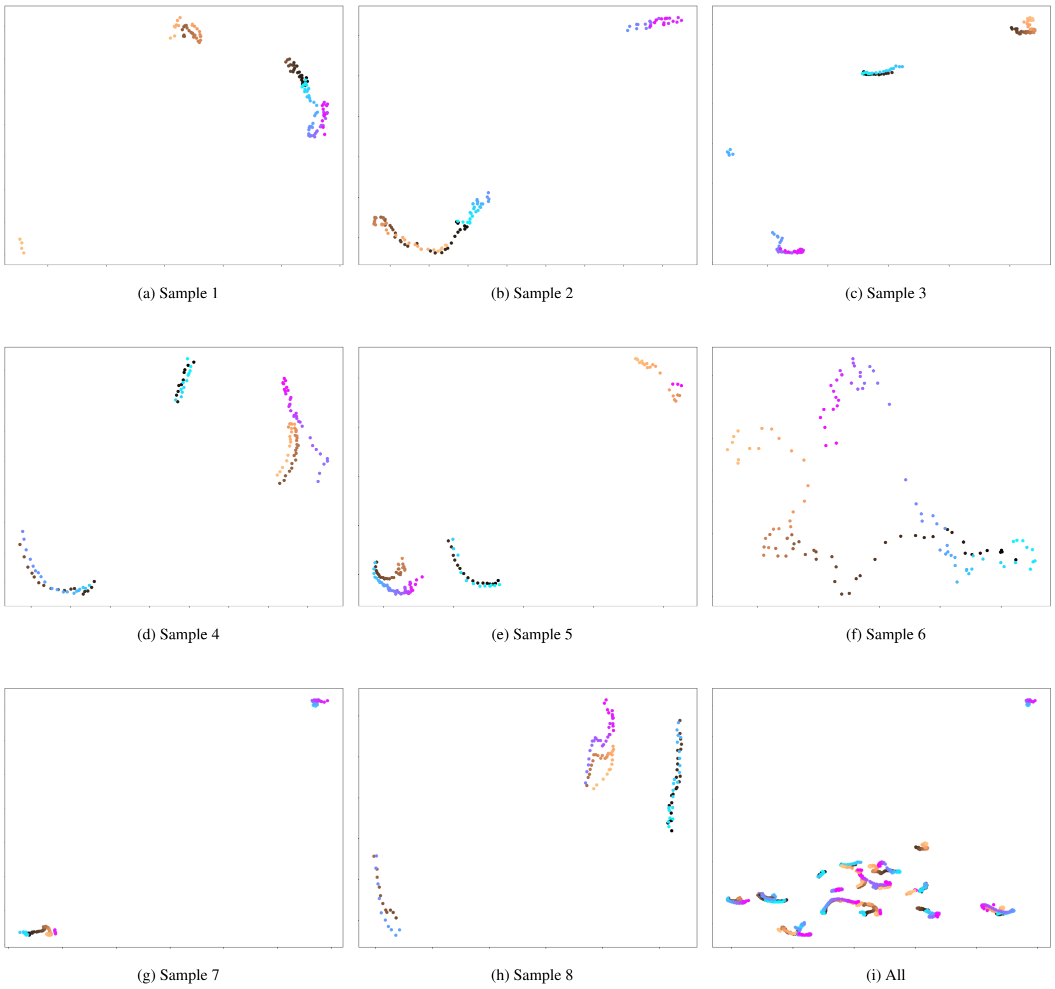


Figure 10: TSNE visualization of GT-VAE trajectory vs. Diffuser-1D trajectory, where the purple and light blue color range is the ground truth VAE, and the copper color range is Diffuser-1D. All states are normalized, and all trajectories are taken from the task "lift pink block table".

J HULC Latent Plan TSNE

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

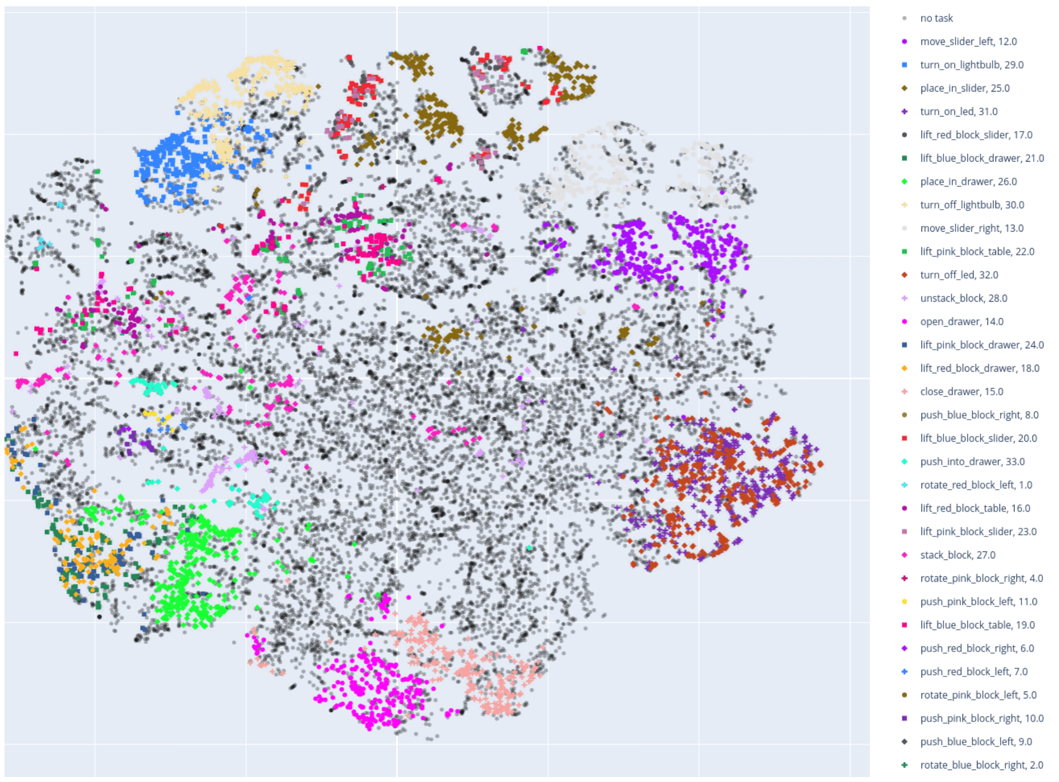


Figure 11: TSNE of Latent Plan. We give a TSNE embedding of the latent plan space of HULC in Figure 11. The latent plan space is the communication layer between the high level policy and low level policy of the HULC model, which corresponds to the intermediate layer between the lower level and lowest 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.

K Comparison of success rates (SR) across single tasks, evaluated with HULC and LCD

Table 6: Comparison of success rates (SR) across single tasks, evaluated with HULC and LCD. We again report the performance for the mean and standard deviation across 3 seeds for all methods, with runs taken from the MT-LHC benchmark in Table 1.

Task	HULC	Ours
Close Drawer	99.50 ± 0.36	95.97 ± 3.46
Lift Blue Block Drawer	86.97 ± 13.01	90.83 ± 9.47
Lift Blue Block Slider	58.80 ± 6.99	$\mathbf{78.77 \pm 5.20}$
Lift Blue Block Table	72.73 ± 7.89	76.07 ± 2.03
Lift Pink Block Drawer	67.93 ± 9.39	76.27 ± 22.80
Lift Pink Block Slider	71.50 ± 4.54	$\mathbf{79.30 \pm 2.01}$
Lift Pink Block Table	68.00 ± 5.60	70.47 ± 3.07
Lift Red Block Drawer	$\mathbf{91.67 \pm 11.79}$	77.00 ± 4.71
Lift Red Block Slider	71.97 ± 2.57	79.70 ± 7.32
Lift Red Block Table	60.87 ± 5.84	$\mathbf{69.43 \pm 2.77}$
Move Slider Left	96.27 ± 1.14	96.27 ± 1.96
Move Slider Right	99.27 ± 0.24	97.67 ± 2.19
Open Drawer	97.83 ± 1.43	98.07 ± 0.50
Place In Drawer	95.93 ± 1.37	95.90 ± 1.79
Place In Slider	70.90 ± 5.44	$\mathbf{81.10 \pm 0.14}$
Push Blue Block Left	60.10 ± 9.69	71.93 ± 6.82
Push Blue Block Right	27.60 ± 9.10	33.10 ± 2.86
Push Into Drawer	72.13 ± 5.74	75.47 ± 5.29
Push Pink Block Left	66.73 ± 17.27	64.80 ± 11.98
Push Pink Block Right	57.67 ± 4.46	51.17 ± 6.04
Push Red Block Left	64.17 ± 4.71	55.30 ± 16.32
Push Red Block Right	28.73 ± 16.07	37.83 ± 3.27
Rotate Blue Block Left	59.63 ± 7.38	$\mathbf{70.03 \pm 2.56}$
Rotate Blue Block Right	62.57 ± 4.17	$\mathbf{72.77 \pm 6.76}$
Rotate Pink Block Left	71.53 ± 6.99	75.07 ± 9.15
Rotate Pink Block Right	55.93 ± 4.29	$\mathbf{64.00 \pm 4.84}$
Rotate Red Block Left	67.43 ± 5.15	$\mathbf{81.23 \pm 8.86}$
Rotate Red Block Right	69.53 ± 5.44	74.57 ± 2.11
Stack Block	$\mathbf{36.83 \pm 2.84}$	32.00 ± 3.56
Turn Off Led	98.30 ± 1.87	96.87 ± 1.80
Turn Off Lightbulb	98.33 ± 1.25	95.67 ± 3.78
Turn On Led	98.73 ± 0.93	97.63 ± 1.73
Turn On Lightbulb	$\mathbf{98.97 \pm 0.40}$	95.60 ± 1.58
Unstack Block	94.53 ± 4.53	90.80 ± 3.72
Unstack Block	94.53 ± 4.53	90.80 ± 3.72
Avg Success Rate	70.68 ± 2.65	74.01 ± 2.64

L Model Card for Language Control Diffusion

A hierarchical diffusion model for long horizon language conditioned planning.

L.1 Model Details

L.1.1 Model Description

A hierarchical diffusion model for long horizon language conditioned planning.

L.2 Uses

L.2.1 Direct Use

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

L.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

L.2.3 Out-of-Scope Use

Discrimination in real-world decision making, military usage

L.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.

L.4 Training Details

L.4.1 Training Data

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

L.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

L.6 Technical Specifications

L.6.1 Model Architecture and Objective

Temporal U-Net, Diffusion objective

L.6.2 Compute Infrastructure

Hardware Nvidia Titan RTX , Nvidia A10

Software Pytorch