Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous and Instruction-guided Driving

Brian Yang Huangyuan Su Nikolaos Gkanatsios Tsung-Wei Ke
Ayush Jain Jeff Schneider Katerina Fragkiadaki
Carnegie Mellon University

Abstract

Diffusion models excel at modeling complex and multimodal trajectory distributions for decision-making and control. Reward-gradient guided denoising has been recently proposed to generate trajectories that maximize both a differentiable reward function and the likelihood under the data distribution captured by a diffusion model. Reward-gradient guided denoising requires a differentiable reward function fitted to both clean and noised samples, limiting its applicability as a general trajectory optimizer. In this paper, we propose Diffusion-ES, a method that combines gradient-free optimization with trajectory denoising to optimize black-box non-differentiable objectives while staying in the data manifold. Diffusion-ES samples trajectories during evolutionary search from a diffusion model and scores them using a black-box reward function. It mutates high-scoring trajectories using a truncated diffusion process that applies a small number of noising and denoising steps, allowing for much more efficient exploration of the solution space. We show that Diffusion-ES achieves state-of-the-art performance on nuPlan, an established closed-loop planning benchmark for autonomous driving. Diffusion-ES outperforms existing sampling-based planners, reactive deterministic or diffusion-based policies, and reward-gradient guidance. Additionally, we show that unlike prior guidance methods, our method can optimize non-differentiable language-shaped reward functions generated by few-shot LLM prompting. When guided by a human teacher that issues instructions to follow, our method can generate novel, highly complex behaviors, such as aggressive lane weaving, which are not present in the training data. This allows us to solve the hardest nuPlan scenarios which are beyond the capabilities of existing trajectory optimization methods and driving policies. [1]

[1]Project page: diffusion-es.github.io

1. Introduction

Diffusion models have shown to excel at modeling highly complex and multimodal trajectory distributions for decision-making and control [1, 26]. Reward-gradient guidance [26, 33, 59] has been used to test-time optimize differentiable reward functions by alternating between denoising diffusion steps and backpropagating reward gradients to the noised trajectory. In this way, sampled trajectories are pushed towards the trajectory data manifold while also maximizing the reward function at hand [26]. This decoupling of the reward function from trajectory diffusion permits a single trajectory diffusion model to be used for maximizing a variety of reward functions at test time. Reward-gradient guidance requires the reward function to be differentiable and fitted in both noisy and clean trajectories, which usually requires re-training. This limits its applicability as a general solver for trajectory optimization.

We propose Diffusion-ES, a reward-guided denoising method for optimization of non-differentiable, black-box objectives that samples and mutates trajectories using a diffusion model, guided by a reward function that operates only on the clean, final, denoised samples. Naively combining diffusion with sampling-based optimization does not work: sampling-based optimizers, like CEM [49] or MPPI [63], typically require a large population of samples across multiple iterations of selection and mutation to converge to good solutions, which, when combined with the computational cost of denoising inference, results in a prohibitively slow search process. In Diffusion-ES, *high-scoring trajectories are mutated using a truncated diffusion-denoising process*, by adding a small amount of noise and denoising them back, as shown in Figure 2 *right*. The amount of added noise is progressively decreased across search iterations making Diffusion-ES computationally viable.

The trajectory diffusion model used for test-time optimization in Diffusion-ES can in principle condition on any scene-relevant information to narrow the sampling to a distribution of scene-relevant trajectories. In fact, the amount of conditioning information controls a continuum between

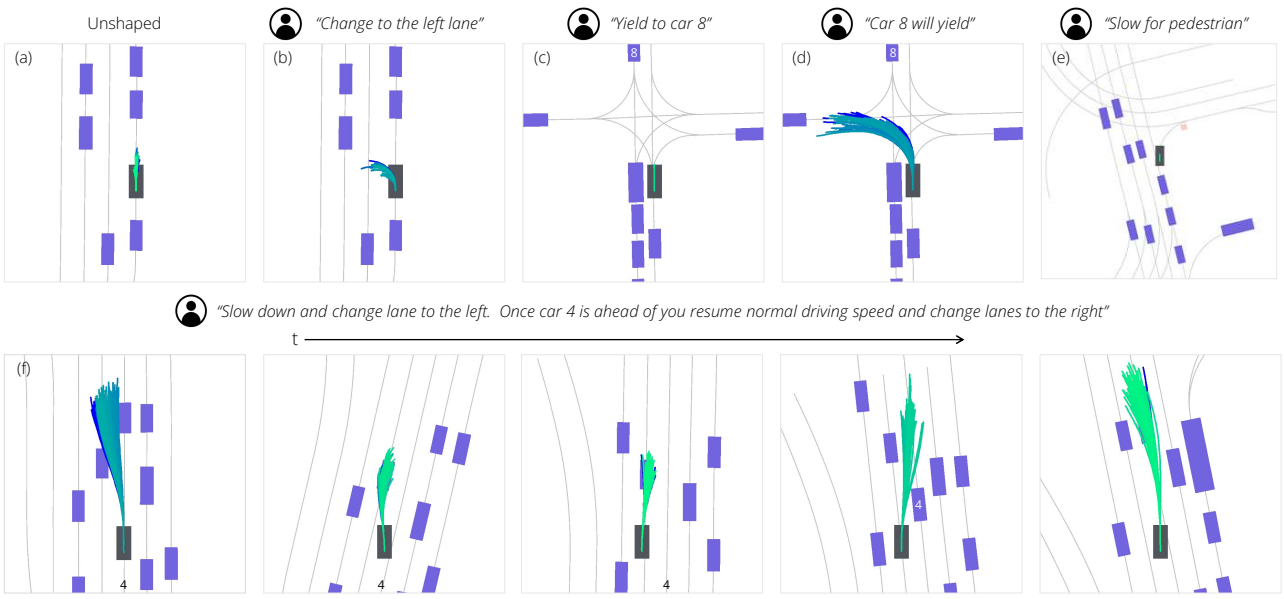

Figure 1. **Diffusion-ES is a test-time trajectory optimization method for arbitrary reward functions that combines generative trajectory models and sampling-based search.** (a) Trajectories generated by Diffusion-ES by optimizing a general driving reward function that encourages following lanes, avoiding collisions, and respecting traffic signs. Diffusion-ES achieves state-of-the-art driving performance in nuPlan [5]. (b-f) Diffusion-ES follows driving instructions in natural language through test-time optimization of language-shaped reward functions, without any additional training. We prompt LLMs to map language instructions to programs that shape the driving reward function, which we then optimize with Diffusion-ES.

train-then-test learning and test-time planning, and a corresponding trade-off between inference speed and out-of-distribution (OOD) generalization: **1.** *The more conditioning information, the narrower the distribution to draw trajectory samples from, the faster the search.* In the extreme, no test-time reward optimization is used and our diffusion model operates as a reactive policy at test time. Indeed, many methods train diffusion policies [9] as *conditional* diffusion trajectory prediction models using rewards as conditioning information to the trajectory diffusion model [1] or finetune a diffusion policy with reinforcement learning [35, 60] or imitation learning [40]. While these methods can handle black-box reward functions or good behaviours to imitate, we show that using them as is or with reward-guidance at test time often under-performs reward-guided denoising of an *unconditional* diffusion model, which completely decouples trajectory and reward modelling. **2.** *The less conditioning information, the wider the distribution to draw trajectory samples from, the slower the search, but the better the generalization to OOD tasks and scenarios* that require novel pairings of trajectories and scene contexts, not present in the training data. Indeed, this is the premise of test-time planning over train-then-test learning: test-time optimization of a composition of energy functions [14, 18], here, the energy of the trajectory data distribution and the energy of arbitrary reward functions, should be able to synthesize novel behaviours not seen at training time.

We show how Diffusion-ES, when combined with an unconditional diffusion model over trajectories, can achieve state-of-the-art planning performance purely through diffusion-guided black-box reward maximization. Our approach is evaluated on nuPlan [5], an established driving benchmark built on real-driving logs and estimated ground-truth perception. We achieve state-of-the-art performance for closed-loop driving, matching the performance of the previous SOTA, PDM-Closed, a sampling-based planner tailored to the nuPlan benchmark [12], as well as reactive driving policies [45, 52], deterministic or diffusion-based. Moreover, we illustrate the flexibility of Diffusion-ES by test-time optimizing language-shaped reward functions generated using few-shot LLM prompting. Using language instructions, we can solve the most challenging nuPlan scenarios, as well as synthesize entirely novel driving behaviors. We then test our model and baselines in their ability to optimize the generated reward functions to elicit the desired behaviours. Qualitative examples of behaviors generated by instruction following using our method can be found in Figure 1. We show Diffusion-ES dramatically outperforms PDM-Closed , other sampling-based planners, as well as ablative versions of Diffusion-ES that either condition the diffusion model on the surrounding scene, or do not use any guidance at all.

In summary, our contributions are as follows:

1. We introduce Diffusion-ES, a trajectory optimization method for optimizing black-box objectives that uses

a trajectory diffusion model for sampling and mutating trajectory proposals during sampling-based search. We show Diffusion-ES matches the SOTA performance of engineered planners in closed-loop driving in nuPlan, and much outperforms them when optimizing more complex reward functions that require flexible driving behaviour, beyond lane following. To the best of our knowledge this is the first work to combine evolutionary search with diffusion models.

2. We show that Diffusion-ES can be used to follow language instructions and steer the closed-loop driving behaviour of an autonomous vehicle by optimizing the LLM-shaped reward functions, *without any training data of language and actions*. We showed that such instruction following can solve the most challenging driving scenarios in nuPlan.
3. We show extensive ablations of our model with varying amount of conditioning information which clearly reveals the trade off between inference speed and out-of-distribution (OOD) generalization in driving.

We believe Diffusion-ES will be useful to the community as a general trajectory optimizer with applicability beyond driving. Our code and models will be publicly available to aid reproducibility in the project webpage: `diffusion-es.github.io`.

2. Related work

Diffusion models for decision-making and trajectory optimization Diffusion models [21, 39, 55, 56] learn to approximate the data distribution through an iterative denoising process and have shown impressive results on image generation [13, 44, 48, 50]. They have been used for imitation learning for manipulation tasks [9, 40, 46, 64], for controllable vehicle motion generation [6, 27, 71] and for video generation of manipulation tasks [15, 67]. Works of [28, 72] use diffusion models to forecast offline vehicle trajectories. To the best of our knowledge, this is the first work to use diffusion models in closed-loop driving.

Learning versus planning for autonomous driving Learning to drive from imitating driving demonstrations is prevalent in the research and development of autonomous vehicles [2, 3, 7, 8, 10, 31, 42, 43, 45, 70]. Many preeminent imitation methods assume the underlying action distribution is unimodal, which is problematic when training from multimodal expert demonstrations. Objectives and architectures that can better handle multimodal trajectory prediction have been proposed [11, 17, 30, 38, 54, 58]. We show that diffusion models are well-suited for driving and can be used to synthesize rich complex behaviors from multimodal demonstrations.

On the other hand, conventional autonomy stacks do not rely on learning at all for decision making, and rather rely on optimizing manually engineered cost functions online [16, 37, 74]. Recently, PDM-Closed [12] achieved state-of-the-art performance on the nuPlan driving benchmark by purely relying on test-time planning and heuristics for selecting trajectory proposals. Other prior works aim to incorporate the benefits of offline learning for test-time planning by performing sampling-based planning over learned cost maps [69] or doing gradient-based optimization over learned dynamics models [47]. We extend this line of work by showing how diffusion-based generative models can be combined with sampling-based planning.

Language-conditioned policies for autonomous driving Recently there has been significant progress made towards language-conditioned policies for driving. GAIA-1 [22] is a generative world model capable of multimodal video generation that leverages video, language and actions to synthesize driving scenarios which can comply with given language instructions. However, GAIA-1 does not execute any actual control inputs.

LLMs trained from Internet-scale text have shown impressive zero-shot reasoning capabilities for a variety of downstream language tasks when prompted appropriately, without any weight fine-tuning [4, 34, 61]. Recent works have shown that LLMs can be prompted to map language instructions to language subgoals [23, 24, 65, 73] action programs [19, 32, 57] or cost maps [25] with appropriate plan-like or program-like prompts. Our work follows few-shot prompting of LLMs to shape driving reward functions. We extend previous methods by using Python generators to produce reward functions which maintain internal state across calls. Works of [36, 62, 66] use LLMs to predict low-level control signals given high-level scene descriptions and language instructions. However, none of these approaches evaluate their performance on closed-loop driving, which is significantly more difficult than open-loop trajectory forecasting [12]. [53] is similar to us, but only considers a highly simplified driving setup and does not report results on a standardized benchmark with strong baselines.

3. Method

3.1. Background

Diffusion models A diffusion model captures the probability distribution $p(x)$ through the inversion of a forward diffusion process, that gradually adds Gaussian noise to the intermediate distribution of a initial sample x. The amounts of added noise depend on a predefined variance schedule $\beta_t \in (0,1)_{t=1}^{T}$, where T denotes the total number of diffusion timesteps. At diffusion timestep t, the forward diffusion process adds noise into x using the formula $x_t = \sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ is a sample from a Gaussian distribution with the same dimensionality as x. Here, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}t = \prod i = 1^t \alpha_i$. For denoising, a neural network $\hat{\epsilon} = \epsilon_\theta(x_t; t)$ takes input as the noisy

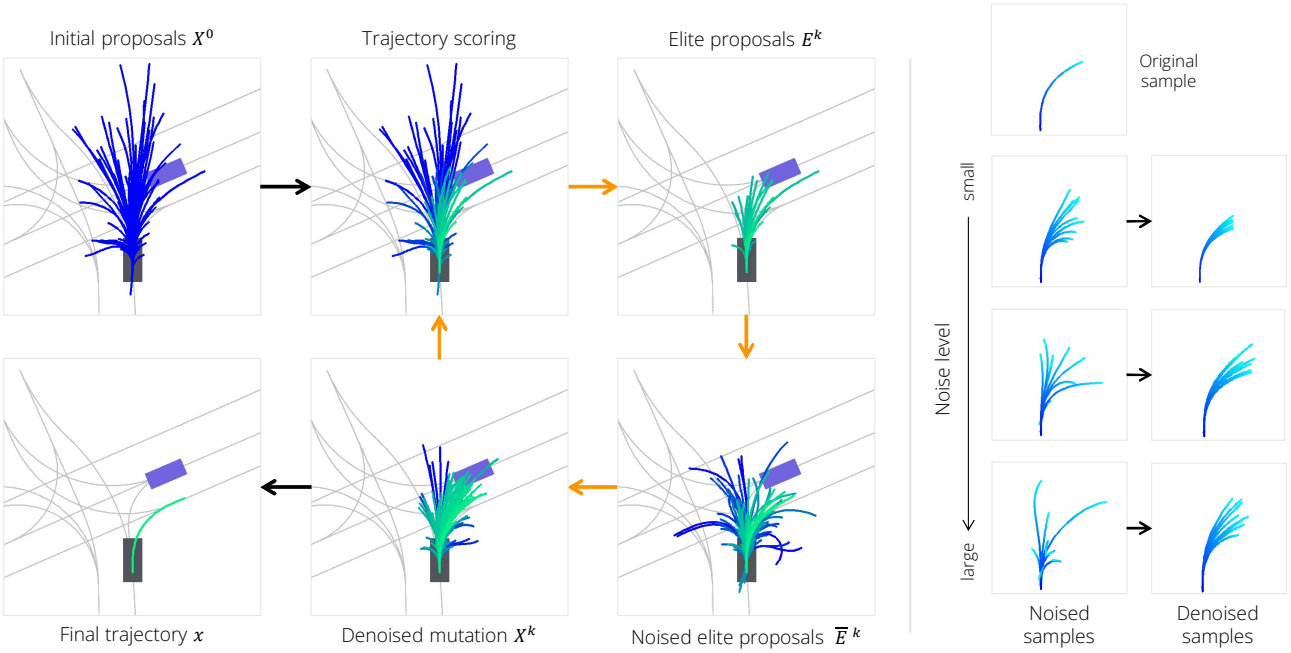

Figure 2. **Overview for Diffusion-ES.** *Left:* Generative evolutionary search with Diffusion-ES. Trajectories are colored by rewards (blue is low, green is high). *Right:* We visualize the mutations for varying noise levels. Color denotes timestep along trajectory. While noise perturbations alone can lead to unrealistic trajectories, denoising helps project samples back onto the trajectory data manifold.

sample x_t and the diffusion timestep t, and learns to predict the added noise ϵ. To generate a sample from the learned distribution $p_\theta(x)$, we start by drawing a sample from the prior distribution $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and iteratively denoise this sample T times with ϵ_θ. The application of ϵ_θ depends on a specified sampling schedule [21, 56], which terminates with x_0 sampled from $p_\theta(x)$. Diffusion models can be easily extended to model $p(x|\mathbf{c})$, where $\mathbf{c}$ is some conditioning signal, such as the expected future rewards , by adding an additional input to the denoising neural network ϵ_θ.

Evolutionary strategies Evolutionary strategies (ES) are a family of population-based *gradient-free* optimization algorithms which can maximize arbitrary black-box reward functions $R(x) : \mathbb{R}^d \to \mathbb{R}$ without any training, where x is the variable we optimize over. ES iteratively update a search distribution $q(x)$ to maximize expected rewards $\mathbb{E}_{x\sim p_\theta(x)}[R(x)]$. While distribution-based ES approaches such as CEM and CMA-ES represent $q(x)$ explicitly (often as a unimodal Gaussian), we can also represent $q(x)$ non-parametrically as a set of high-performing solutions without making strong assumptions about the functional form of q.

3.2. Diffusion-ES

Diffusion-ES is a trajectory optimization method that leverages gradient-free evolutionary search to perform reward-guided sampling from trained diffusion models, for any black-box reward function $R(x)$. Specifically, we use a trained diffusion model ϵ_θ to initialize the sample population and we use a truncated diffuse-denoise process to mutate samples while staying in the data manifold. The control flow of Diffusion-ES is shown in Figure 2 (left) and in Algorithm 1.

Initializing the population with diffusion sampling We begin by sampling an initial population X^0 of M trajectory samples using our diffusion model:

$$X^0 = \{x_i\}_{i=1}^M \sim p_\theta(x), \tag{1}$$

where X^k is the population at iteration k. This involves a complete pass through the reverse diffusion process. Given we use an unconditional diffusion model, these samples are scene agnostic and can always be used without re-sampling them at each timestep. We can also modify the initial population by including samples generated by other approaches or mixing in solutions from the previous timestep to warm-start our optimization.

Sample scoring At each iteration k, we score the samples in our population $\{R(x_i)|x_i \in X^k\}_{i=1}^M$. Note that our population consists of "clean" samples so we do not need a reward function which can handle "noisy" samples, which gives us significant flexibility compared to guidance methods that perform classifier-based guidance.

Selection We use rewards to decide which samples we should select to propagate to the next iteration. Similar

to MPPI [63], we resample X^{k+1} as follows:

$$q(x) = \frac{\exp(\tau R(x))}{\sum_{i=1}^{M} \exp(\tau R(x_i))} \quad (2)$$

$$E^{k+1} = \{x_i \overset{\text{iid}}{\sim} q(x)\}_{i=1}^{M}, \quad (3)$$

where E^{k+1} represents our elite set which is kept from iteration k, and τ is a tunable temperature parameter controlling the sharpness of q.

Mutation using truncated diffusion-denoising We apply randomized mutations to E^{k+1} for exploration. Prior evolutionary search methods resort to naive Gaussian perturbations which do not exploit any prior knowledge about the data manifold. **Our key insight is to leverage a truncated diffusion-denoising process to mutate trajectories in a way the resulting mutations are part of the data manifold.** We can run the first t steps of the forward diffusion process to get noised elite samples $\bar{E}^{k+1}$:

$$\bar{E}^{k+1} = \{\sqrt{\bar{\alpha}_N}x + \sqrt{1-\bar{\alpha}_N}\epsilon | x \in E^{k+1}\}, \quad (4)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. Then we can run the last t steps of the reverse diffusion process to denoise the samples again, giving us clean samples X^{k+1}:

$$X^{k+1} = \{x \sim p_\theta(x|\bar{x}) | \bar{x} \in \bar{E}^{k+1}\}. \quad (5)$$

In practice, the number of timesteps t of the truncated diffusion process is a tunable time-dependent hyperparameter t_k which controls the mutation strength at each iteration k. This is visualized in Figure 2 (right). We find that linearly decaying the number of mutation diffusion steps t_k from 5 to 1 over 20 search steps works best in our experiments.

Algorithm 1 `Diffusion-ES`

Input: Diffusion model p_θ, reward function R, search steps K, population size M, number of noising steps N, variance schedule $\bar{\alpha} = \{\bar{\alpha}_i\}_{i=1}^{N}$.
Initial proposals: $X^0 \leftarrow \{x_i \sim p_\theta(x)\}_{i=1}^{M}$
for search step $k \in (1, \ldots, K)$ **do**
  Score proposals $\{R(x_i)|x_i \in X^{k-1}\}_{i=1}^{M}$
  Compute distribution $q(x) = \frac{\exp(\tau R(x))}{\sum_{i=1}^{M} \exp(\tau R(x_i))}$
  Sample elites from X^{k-1}: $E^k \leftarrow \{x_i \overset{\text{iid}}{\sim} q(x)\}_{i=1}^{M}$
  Renoise elites $\bar{E}^k \leftarrow \{\sqrt{\bar{\alpha}_N}x + \sqrt{1-\bar{\alpha}_N}\epsilon | x \in E^k, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})\}$
  Denoise elites $X^k \leftarrow \{x \sim p_\theta(x|\bar{x}) | \bar{x} \in \bar{E}^k\}$
end for
return output $x = \arg\max_{x \in X^K} R(x)$

3.3. Mapping language instructions to reward functions with LLM prompting

To follow driving instructions given in natural language, we map them to black-box reward functions which we optimize with Diffusion-ES. We adopt a similar approach to [29, 68] which uses LLMs to synthesize reward functions from language instructions. Reward functions are composable and allow us to seamlessly combine language guidance with other constraints. This is crucial in driving where we constantly optimize many different objectives at once (e.g., safety, driver comfort, route adherence).

Similar to prior work, we expose a Python API which can be used to extract information about entities in the road scene. Since many of the basic reward signals in driving do not change from scenario to scenario (e.g., collision avoidance or drivable area compliance), we allow the LLM to write *reward shaping* code which modifies the behavior of the base reward function, as opposed to generating everything from scratch. Reward shaping can add auxiliary reward terms (e.g., a dense lane-reaching reward) or re-weight existing reward terms. We show generated code examples in Figure 3.

Our goal is to handle general and complex language instructions with temporal dependencies, such as *"Change lanes to the left, then pass the car on the right, then take the exit"*. Previous works [25, 68] produce a stationary reward function, i.e., one which is fixed during planning. This can make it challenging to express sequential plans solely through rewards. We find that a much more natural and succinct way of capturing these plans in code is through the use of *generator functions* which retain internal state between calls. All our prompts and code examples can be found in the supplementary file. In Section 4, we show how optimizing these language-shaped reward functions can synthesize rich and complex driving behaviors that comply with the language instructions.

4. Experiments

We first evaluate Diffusion-ES on closed-loop driving in nuPlan [5], an established benchmark that uses estimated perception for vehicles, pedestrians, lanes and traffic signs. Our model and baselines are evaluated on their ability to drive safely and efficiently while having access to close-to-ground-truth perception output provided by the dataset. We also consider a suite of driving instruction following tasks. We map instructions to shaped reward functions with LLM prompting, and evaluate Diffusion-ES and baselines in their ability to optimize the generated reward functions and accurately follow the instructions. Our experiments aim to answer the following questions:

1. How does Diffusion-ES compare to existing sampling-based planners and reward-gradient guidance for trajectory optimization?
2. How does Diffusion-ES compare to SOTA reactive driving policies that directly map environments' state to vehicle trajectories?
3. Can the hardest nuPlan driving scenarios be solved by assuming access to a human teacher giving instructions

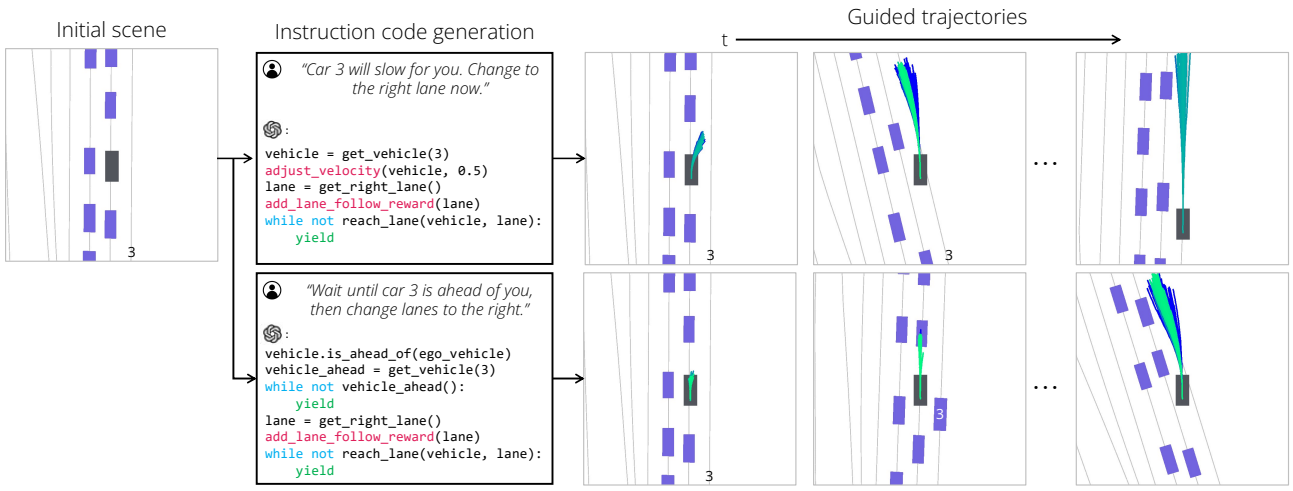

Figure 3. **Shaping reward functions with language instructions using LLM prompting and optimizing them with Diffusion-ES.** For the same initial scene, we show two distinct language instructions, along with their generated reward shaping code and generated trajectories from Diffusion-ES.

in natural language, without any additional training data?
4. Does scene conditioning for the diffusion model benefit Diffusion-ES?

4.1. Closed-loop driving

We evaluate our model on the nuPlan Val14 planning benchmark [12]. We specifically consider the reactive agent track of the nuPlan benchmark since it is the most difficult and realistic of the evaluation settings in nuPlan.

Model setup and hyperparameters For Diffusion-ES, we train a diffusion model over ego-vehicle trajectories consisting of 2D poses (x, y, θ) predicted 8 seconds into the future at 2Hz, leading to an overall action dimension of 48. Unlike prior work [26], we model the distribution over actions only rather than modeling states and actions. We use a population size of $M = 128$ in our experiments. Our diffusion model is trained with $T = 100$ denoising steps.

Reward function We adopt a modified version of the scoring function used in PDM-Closed [12] as our reward function. To compute rewards, we convert our predicted trajectory to low-level control inputs using an LQR tracker. These control inputs are fed into a kinematic bicycle model [41] which propagates the dynamics of the ego-vehicle. We follow [12] and forecast the motion of other agents by assuming constant velocity. These simulated rollouts are then scored following the nuPlan benchmark evaluation metrics. We also add auxiliary reward terms to penalize proximity to the leading agent and enforce speed limits.

Note that this reward function is *not differentiable* due to the tracker and the use of non-differentiable heuristics for assessing traffic violations. Additionally, training a model to regress rewards is challenging since the nuPlan dataset contains no instances of serious traffic infractions.

Evaluation metric We report our results using **driving score**, which aggregates multiple planning metrics related to traffic rule compliance, safety, route progress, and rider comfort. This is the standard evaluation metric used in nuPlan.

Baselines We consider the following baselines:

- *UrbanDriverOL* [51], a deterministic transformer policy trained with behaviour cloning and augmentations. Unlike in [51], the nuPlan implementation does not perform closed-loop training.
- *PlanCNN* [45] a deterministic imitation policy which encodes a rasterized BEV map using a CNN backbone.
- *IDM* [20]: a heuristic rule-based planner which adjusts its speed to maintain a safe distance to the leading vehicle. It is also used to control the behaviour of agents in nuPlan.
- *PDM-Closed* [12]: an MPC-based planner which generates path proposals using lane centerlines, and rolls out trajectories similarly to us. Instead of iteratively optimizing rewards, PDM-Closed simply executes the highest-performing proposal after one round of scoring. It is the current state-of-the-art on the nuPlan Val14 benchmark.
- *Diffusion Policy*: a diffusion model we consider that conditions on scene features to predict a vehicle trajectory directly. We encode scene features using the transformer feature backbone from Urban Driver [52]. We train it with imitation learning and augmentations. This is similar to the unconditional trajectory model in Diffusion-ES with additional conditioning on scene features.

	Method	Driving Score (↑)
Train-then-test	UrbanDriverOL [52]	65
	PlanCNN [45]	72
	Diffusion policy	50
Test-time optimize	IDM [49]	77
	PDM-Closed [12]	**92**
	Diffusion-ES (ours)	**92**

Table 1. **Closed-loop driving results in the val14 split of [12].** Our model matches the performance of previous non-learning based planners and significantly outperforms all other models.

We show quantitative results in Table 1. We draw the following conclusions:

1. Diffusion-ES matches the prior state-of-the-art, PDM-Closed and substantially outperforms all other baselines. PDM-Closed is a sampling-based planner that relies on domain-specific heuristics to generate trajectory proposals whereas Diffusion-ES learns these proposals from data. Both use a similar reward function and dynamics model for the agents in the scene.

2. There is a large gap in performance between reactive neural policies and test-time planners, also pointed out in recent work [12]. We hypothesize that this is because compared to other control benchmarks, nuPlan has a much richer observation space as scenes are densely populated by dynamic actors, many of which are irrelevant to the ego-agent. This can make it challenging for learning-based methods to generalize, which motivates the need of test-time optimization.

3. Diffusion-ES substantially outperforms diffusion policy. Qualitatively, our diffusion policy has a tendency to randomly change lanes, which causes the ego-vehicle to reach out-of-distribution scenarios faster. Diffusion-ES leverages the expressiveness of generative modeling while using test-time optimization to improve generalization.

4.2. Language instruction following

One drawback of the nuPlan driving benchmark is that encourages highly conservative driving behaviors. For example, PDM-Closed [12] holds the state-of-the-art in the nuPlan Val14 benchmark while being unable to change lanes, since its path proposals only consider the lane the ego-vehicle is currently on. However, lane changing is not necessary for good driving performance in the current nuPlan benchmark.

To evaluate Diffusion-ES and baselines in their ability to optimize arbitrary reward functions, we consider eight language instruction following tasks, each taken from an existing driving log in the nuPlan benchmark. In each task, the language instruction requires the ego-vehicle to perform a specific driving maneuver that solves a challenging driving scenario. In most scenarios there will be no examples of the instructed behavior anywhere in nuPlan. For instance, the lane weaving task requires the ego-vehicle to aggressively change multiple lanes in dense urban traffic. Task descriptions, language instructions, and prompts are provided in the appendix. We use the method described in 3.3 to generate executable Python code given a language instruction that adapts the initial reward function of Section 4.1, giving us a language-shaped reward function for each scenario. Our model and baselines will optimize the same language-shaped reward function.

Evaluation metric We evaluate our model and baselines using **task success rate**, which measures how frequently the agent was able to successfully complete the designated task. To increase the difficulty of the tasks, we randomize the behavior of other vehicle agents by adding noise to their IDM parameters at sporadic intervals during each episode. All scores reported are averaged across ten random seeds.

Baselines We compare to the following baselines:

- *PDM-Closed*: this is adapted to this setting by using our language-shaped reward function in place of the original reward function.
- *PDM-Closed-Multilane*: a modified variant of PDM-Closed which considers a wider range of laterally offset paths, allowing for lane changes.
- *Conditional Diffusion-ES*: Diffusion-ES that uses a conditional diffusion model instead of an unconditional one.

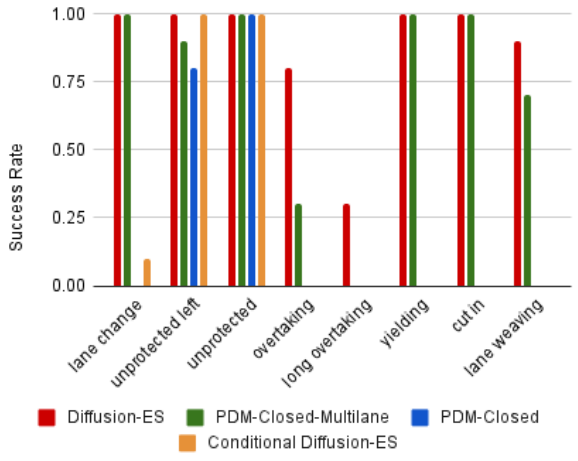

Figure 4. **Following driving instructions.** Diffusion-ES outperforms all baselines in optimizing complex language-shaped reward functions.

Figure 4 shows the success rates on the controllability tasks. We draw the following conclusions:

1. Diffusion-ES outperforms all baselines. Although PDM-Closed-Multilane has substantially improved performance

Method	Lane Error	Speed Error
CEM	2.34	2.05
MPPI	3.29	2.74
Reward-gradient guidance	1.22	0.96
Diffusion-ES (ours)	**0.61**	**0.79**

Table 2. **Lane following in nuPlan.**

compared to PDM-Closed due to more diverse trajectory proposals, it is still weaker than Diffusion-ES on 4 out of 8 tasks. This highlights the weakness of relying on handcrafted rules for proposal generation. **2. Diffusion-ES performs significantly worse with a conditional diffusion model.** The conditional diffusion model is much harder to guide since the scene context causes fewer samples to be in-distribution.

4.3. Lane following

To compare Diffusion-ES against reward-gradient guidance, we consider a simplified lane following task with a differentiable reward function, which consists of two terms: one penalizing lateral deviation from the lane (**lane error**), and one penalizing deviation from a target speed (**speed error**). We sample 14 scenarios, one of each scenario type in nuPlan, and report average planning costs across all scenarios.

Baselines We consider the following baselines:

- *CEM* [49]: a widely used ES method which parameterizes the search distribution q as a Gaussian. CEM iterates between sampling from q and fitting q to the best samples.
- *MPPI* [63]: similar to CEM but rather than keep a fixed number of elites, MPPI samples proportional to rewards.
- *Reward-gradient guidance* [26]: we directly optimize the ground-truth planning objective with gradient descent during the denoising process.

We show quantitative results in Table 2. We draw the following conclusions: **1.Diffusion-ES outperforms the differentiable reward-gradient guidance baseline *even though the objective is differentiable***. We hypothesize that this is because although the ground truth reward function is available, it may not provide suitable guidance for intermediate noisy trajectories. This highlights a key advantage of our method over prior work, which is that we can optimize novel objectives without needing to train a reward regressor on noisy samples. **2. Both diffusion-based methods significantly outperform sampling-based planners that do not leverage diffusion**. This is consistent with our hypothesis that diffusion guidance can optimize trajectories much more efficiently than conventional ES methods. Videos of our method driving in all experimental settings can be found in our project page diffusion-es.github.io.

Method	Wallclock time (s)
Diffusion	1.11 ± 0.02
Diffusion-ES	5.85 ± 0.11
Diffusion-ES (*optimized*)	0.50 ± 0.01

Table 3. **Diffusion-ES runtime comparison.**

4.4. Runtime analysis

Diffusion-ES can be used in real-time with some minor optimizations. By using fewer diffusion steps $T = 10$, smaller population size $M = 32$ and less iterations $K = 2$, our method can be run at the same frequency as the simulator (2 Hz) at a small cost to performance (nuPlan driving score drops from 92 to 91). We report the average wallclock time for inference at every timestep over 100 trials in Table 3.

4.5. Discussion - Limitations - Future work

As seen in Section 4.4, our approach does introduce computational overhead. We believe that these issues can be mitigated by incorporating recent advances in diffusion modeling such as faster samplers. Our reward function assumes other agents will travel at constant velocity, which could clearly be improved. This also assumes that other agents cannot react to the ego-vehicle, which has been shown to be a major limitation for planners in self-driving [47]. However, our instruction following experiments suggest that even if we cannot ever forecast perfectly, we can use language-shaped rewards to solve the hardest driving scenarios. We aim to explore memory-prompted analogical reward shaping for handling long-tail scenarios without a human teacher in our future work.

5. Conclusion

We presented Diffusion-ES, a method for black-box reward guided diffusion sampling. We showed that Diffusion-ES can effectively optimize reward functions in nuPlan for driving and instruction following, and outperforms engineered sampling-based planners, reactive deterministic or diffusion policies, as well as differentiable reward-gradient guidance. We showed how our method can be used to follow language instructions without any language-action trajectory data, simply using LLM prompting to generate shaped reward maps for test-time optimization. Our future work will explore retrieving the right reward shaping to optimize to handle long-tailed driving scenarios in the absence of human teachers. Our experiments show the trade-off between inference speed and OOD generalization during scene conditioning in diffusion policies. Our future work will explore ways to amortize the result of such searches to fast reactive policies, and to balance the two extremes so that a variable amount of compute can be spent depending on the scenario.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, T. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *ArXiv*, abs/2211.15657, 2022. 1, 2

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018. 3

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016. 3

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 3

[5] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles, 2022. 2, 5

[6] João Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. *ArXiv*, abs/2308.01557, 2023. 3

[7] Dian Chen and Philipp Krähenbühl. Learning from all vehicles, 2022. 3

[8] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. *ArXiv*, abs/1912.12294, 2019. 3

[9] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023. 2, 3

[10] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio M. López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410, 2017. 3

[11] Alexander Cui, Abbas Sadat, Sergio Casas, Renjie Liao, and Raquel Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. *CoRR*, abs/2101.06547, 2021. 3

[12] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning, 2023. 2, 3, 6, 7

[13] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233, 2021. 3

[14] Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc, 2023. 2

[15] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *ArXiv*, abs/2302.00111, 2023. 3

[16] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. Baidu apollo em motion planner. *arXiv preprint arXiv:1807.08048*, 2018. 3

[17] Katerina Fragkiadaki, Jonathan Huang, Alex Alemi, Sudheendra Vijayanarasimhan, Susanna Ricco, and Rahul Sukthankar. Motion prediction under multimodality with conditional stochastic networks, 2017. 3

[18] Nikolaos Gkanatsios, Ayush Jain, Zhou Xian, Yunchu Zhang, Christopher Atkeson, and Katerina Fragkiadaki. Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement. In *Robotics: Science and Systems*, 2023. 2

[19] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14953–14962, 2023. 3

[20] Dirk Helbing and Benno Tilch. Generalized force model of traffic dynamics. *Physical review E*, 58(1): 133, 1998. 6

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 3, 4

[22] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023. 3

[23] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022. 3

[24] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan

Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022. 3

[25] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023. 3, 5

[26] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022. 1, 6, 8

[27] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9644–9653, 2023. 3

[28] Chiyu "Max" Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion, 2023. 3

[29] Pushkal Katara, Zhou Xian, and Katerina Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models, 2023. 5

[30] Sanmin Kim, Hyeongseok Jeon, Jun Won Choi, and Dongsuk Kum. Diverse multiple trajectory prediction using a two-stage prediction network trained with lane loss. *IEEE Robotics and Automation Letters*, 8(4): 2038–2045, 2023. 3

[31] Alex Kuefler, Jeremy Morton, Tim Allan Wheeler, and Mykel John Kochenderfer. Imitating driver behavior with generative adversarial networks. *CoRR*, abs/1701.06699, 2017. 3

[32] 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. 3

[33] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In *ICML*, 2023. 1

[34] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021. 3

[35] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. *arXiv preprint arXiv:2304.12824*, 2023. 2

[36] Jiageng Mao, Yuxi Qian, Hang Zhao, and Yue Wang. Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*, 2023. 3

[37] M Montremerlo, J Beeker, S Bhat, and H Dahlkamp. The stanford entry in the urban challenge. *Journal of Field Robotics*, 7(9):468–492, 2008. 3

[38] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple amp; efficient attention networks, 2022. 3

[39] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 3

[40] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models, 2023. 2, 3

[41] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE intelligent vehicles symposium (IV)*, pages 812–818. IEEE, 2017. 6

[42] Dean A. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems 1*, pages 305–313. San Francisco, CA: Morgan Kaufmann, 1989. 3

[43] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. *CoRR*, abs/2104.09224, 2021. 3

[44] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022. 3

[45] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, A. Sophia Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. In *6th Annual Conference on Robot Learning*, 2022. 2, 3, 6, 7

[46] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023. 3

[47] Nicholas Rhinehart, Jeff He, Charles Packer, Matthew A Wright, Rowan McAllister, Joseph E Gonzalez, and Sergey Levine. Contingencies from observations: Tractable contingency planning with learned behavior models. In *2021 IEEE International Con-*

ference on Robotics and Automation (ICRA), pages 13663–13669. IEEE, 2021. 3, 8

[48] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2022. 3

[49] Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. 1, 7, 8

[50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *ArXiv*, abs/2205.11487, 2022. 3

[51] Oliver Scheel, Luca Bergamini, Maciej Wołczyk, Bla.zej Osi'nski, and Peter Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, 2021. 6

[52] Oliver Scheel, Luca Bergamini, Maciej Wolczyk, Błażej Osiński, and Peter Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, pages 718–728. PMLR, 2022. 2, 6, 7

[53] Hao Sha, Yao Mu, Yuxuan Jiang, Li Chen, Chenfeng Xu, Ping Luo, Shengbo Eben Li, Masayoshi Tomizuka, Wei Zhan, and Mingyu Ding. Languagempc: Large language models as decision makers for autonomous driving. *arXiv preprint arXiv:2310.03026*, 2023. 3

[54] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone, 2022. 3

[55] Jascha Narain Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *ArXiv*, abs/1503.03585, 2015. 3

[56] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 4

[57] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning, 2023. 3

[58] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. *CoRR*, abs/1911.00997, 2019. 3

[59] Julen Urain, Niklas Funk, Georgia Chalvatzaki, and Jan Peters. Se (3)-diffusionfields: Learning cost functions for joint grasp and motion optimization through diffusion. *ICRA*, 2023. 1

[60] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. 2

[61] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. 3

[62] Licheng Wen, Xuemeng Yang, Daocheng Fu, Xiaofeng Wang, Pinlong Cai, Xin Li, Tao Ma, Yingxuan Li, Linran Xu, Dengke Shang, et al. On the road with gpt-4v (ision): Early explorations of visual-language model on autonomous driving. *arXiv preprint arXiv:2311.05332*, 2023. 3

[63] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015. 1, 5, 8

[64] Zhou Xian, Nikolaos Gkanatsios, Théophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chained-diffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *CoRL 2023*. 3

[65] Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Regression planning networks. *CoRR*, abs/1909.13072, 2019. 3

[66] Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kenneth KY Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *arXiv preprint arXiv:2310.01412*, 2023. 3

[67] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *ArXiv*, abs/2310.06114, 2023. 3

[68] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023. 5

[69] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 3

[70] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[71] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. *ArXiv*, abs/2210.17366, 2022. 3

[72] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. *ArXiv*, abs/2210.17366, 2022. 3

[73] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *CoRR*, abs/2012.07277, 2020. 3

[74] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for bertha—a local, continuous method. In *2014 IEEE intelligent vehicles symposium proceedings*, pages 450–457. IEEE, 2014. 3