2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Adaptive Planning with Generative Models under Uncertainty

Pascal Jutras-Dubé, Ruqi Zhang, and Aniket Bera
Department of Computer Science, Purdue University, USA
Implemention: https://pascaljd.github.io/ensemble-adaptive-policy/

***Abstract*— Planning with generative models has emerged as an effective decision-making paradigm across a wide range of domains, including reinforcement learning and autonomous navigation. While continuous replanning at each timestep might seem intuitive because it allows decisions to be made based on the most recent environmental observations, it results in substantial computational challenges, primarily due to the complexity of the generative model's underlying deep learning architecture. Our work addresses this challenge by introducing a simple adaptive planning policy that leverages the generative model's ability to predict long-horizon state trajectories, enabling the execution of multiple actions consecutively without the need for immediate replanning. We propose to use the predictive uncertainty derived from a Deep Ensemble of inverse dynamics models to dynamically adjust the intervals between planning sessions. In our experiments conducted on locomotion tasks within the OpenAI Gym framework, we demonstrate that our adaptive planning policy allows for a reduction in replanning frequency to only about 10% of the steps without compromising the performance. Our results underscore the potential of generative modeling as an efficient and effective tool for decision-making.**

I. Introduction

In recent years, the domain of generative modeling has witnessed transformative advancements, marked by the development of image synthesis models like DALL-E [1] and Stable Diffusion [2]. This technological progression has extended to the generation of high-quality videos from text prompts [3], [4]. Concurrently, language models like GPT [5] have achieved significant milestones in generating coherent text and engaging in conversations based on brief text prompts.

Recently, generative models have been applied to offline reinforcement learning (RL), where the goal is to derive optimal policies from previously collected datasets. The challenge of predicting future states and actions can be formulated as a sequence modeling task, which can be addressed through generative modeling [6], [7], [8].

However, the state prediction process incurs substantial computational costs due to the deep neural network architecture of the generative models [9], [8]. These computational demands can be a problem in real-time decision-making applications, where agents must rapidly take an action within a time-constrained control loop to plan or adjust their trajectory in response to new environmental observations.

Efforts to improve the sampling efficiency of generative models form a substantial body of work, but few strategies have been specifically developed for decision-making contexts. Most solutions are tailored to the specific architectural features of the generative models they use [9], [10]. Such model-specific methods, while effective, are constrained by their limited applicability across different models.

In this work, we introduce a novel approach that leverages the inherent structure of the decision-making problem to enhance the efficiency of the control process. We use a generative model to predict a trajectory of future environmental states, and a much smaller action model to determine the next actions based on this trajectory. Although planning with generative models is computationally intensive, it enables the prediction of long horizons of future states. Drawing on this observation, our approach executes multiple actions consecutively, thereby reducing the frequency of calls to the generative model. To determine the optimal times for updating the plan and invoking the generative model anew, we use the uncertainty in the action model's predictions as a guiding criterion. The proposed adaptive policy is illustrated in Figure 1.

Our work introduces the following contributions.

- We introduce a simple adaptive policy that enhances the planning process with generative models by leveraging the action model's confidence levels, enabling faster decision-making. Unlike existing solutions, our approach can be universally applied to different generative models without requiring any alterations. Our approach utilizes Deep Ensembles [11] for efficient and effective predictive uncertainty estimation, allowing for dynamic adjustments in the planning based on the model's confidence.
- We conduct a comprehensive evaluation of our method using the D4RL benchmark [12]. Our results demonstrate that our approach can achieve planning speeds of more then 50 times faster than the prior art. This improvement in speed is achieved with no or minimal impact on the rewards.

The remainder of this paper is structured as follows: Section II reviews related work, emphasizing generative modeling in decision-making and efforts to enhance the sampling speed, with a focus on diffusion models. Section III conceptualizes offline RL as a sequence modeling task, detailing how the problem can be split into two stages: generating a sequence of states over a long horizon, followed by action prediction. Section IV introduces our novel adaptive policy that leverages Deep Ensembles. Section V examines the empirical performance of our adaptive policy, highlighting its effective balance between speed and efficacy.

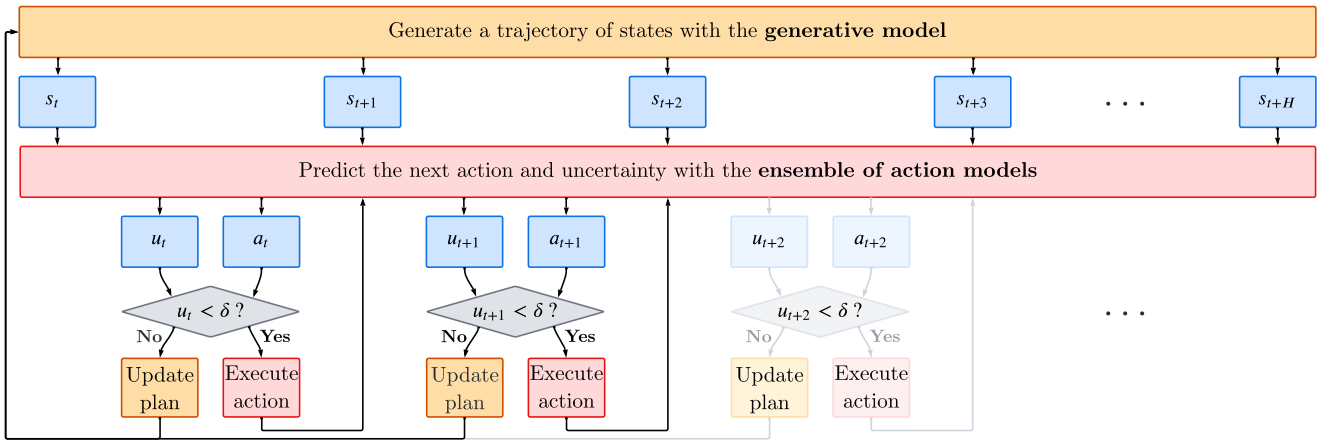

Fig. 1: *The generative model generates a trajectory of states and the action model computes the initial action. The policy continuously predicts and executes subsequent actions as long as the uncertainty remains below a predefined threshold.*

The paper concludes with Section VI, where we summarize our findings, discuss limitations, and outline directions for future research.

II. Related Work

A. Generative Modeling for Decision-Making

Janner et al. [6] and Chen et al. [7] reimagine reinforcement learning (RL) as a sequence modeling problem, deviating from the traditional approach that relies on estimating policies based on the Markov property. They implement a Transformer [13] to model distributions over states trajectories. This represents a conceptual shift away from conventional reinforcement learning techniques, which primarily concentrate on estimating value functions or determining policy gradients.

Diffusion models [14], [15], [16], [17] progressively perturb data towards noise via a Gaussian process and generate data by reversing that process. Diffuser [9] is a diffusion model designed for planning. It differs from traditional model-based planning by predicting entire trajectories, enhancing scalability for long horizon planning. While Diffuser diffuses over both states and actions, Decision Diffuser [8] diffuses only across states and models the action space with an inverse dynamics model. The choice to diffuse only over states is justified by the inherent challenges of modeling actions, which often represent complex, discrete, or force-based decisions. Other recent works have explored the use of diffusion models for planning, underscoring the potential of diffusion-based approaches in decision-making [18], [19], [20], [21].

B. Improving Sampling Speed

The iterative refinement process of diffusion models is computationally expensive, typically requiring tens to hundreds of calls to the underlying deep neural network. Consequently, improving the sampling speed of diffusion models has become an intensive domain of research. Recent works [22], [23] have demonstrated that using 2nd order stochastic differential equation solvers for denoising offers an excellent balance between sample quality and network evaluations, achieving impressive results in image generation with as few as 36 network function evaluations. Additionally, knowledge distillation techniques have significantly accelerated the sampling speed of diffusion models [24], [25], [26], [27]. Moreover, a new class of generative models known as consistency models [27], [28] are designed to overcome the low sampling speed inherent in diffusion models by directly mapping any point on the noise trajectory to the data space. This approach has shown promising results in image generation with as few as two and even one-step generation.

In planning, similar to sampling with consistency models, Janner et al. [9] propose warm-starting the generative process by adding partial noise to the previously generated trajectory and running the corresponding number of denoising steps. Concurrent with our work, Replanning with Diffusion Models [10] assesses when to replan based on the likelihood of existing plans. Their approach calculates trajectory feasibility by introducing Gaussian noise and evaluating the KL divergence during denoising, a process that is distinctly model-specific. The accuracy of the likelihood estimation is directly affected by how well the diffusion model has learned the distribution of successful trajectories. This means that any limitations in the diffusion model's training data or its capacity to capture the complexity of the environment could affect the reliability of the replanning criterion.

In our work, we leverage the inherent long-horizon prediction of generative models to execute multiple actions in a row, and use uncertainty associated with the action model's predictions as a criterion on when to resample.

C. Estimating Uncertainty in Neural Networks

Traditionally, quantifying uncertainty leans on a Bayesian framework, where a prior distribution is defined over the network's parameters. Given the training data, the posterior distribution over the parameters is computed, which is used to quantify predictive uncertainty. However, due to the in-

tractability of Bayesian inference in neural networks, various approximation methods have been proposed.

Markov Chain Monte Carlo (MCMC) methods [29] approximate sampling from the posterior distribution by constructing a Markov chain that has the desired distribution as its equilibrium distribution. However, MCMC is computationally expensive because it requires a large number of steps to achieve convergence, making it challenging for large-scale or real-time applications. Variational inference techniques [30], [31] provide a scalable alternative to MCMC by approximating the posterior distribution with a simpler, parameterized distribution. These techniques optimize the parameters of the simpler distribution to minimize the difference between the true posterior and the approximation, often measured by the Kullback-Leibler divergence.

Monte Carlo dropout [32] stands out for its simplicity in approximating Bayesian inference. By incorporating dropout during both training and inference, this method simulates sampling from the network's posterior distribution. The uncertainty estimation process entails multiple forward passes with dropout, culminating in an ensemble of outputs. The aggregation of these outputs yields an estimate of the predictive mean and variance, providing insights into the network's uncertainty regarding its predictions.

Deep ensembles [11] further simplify the uncertainty estimation. This method involves training several network instances from different initializations and combining their predictions. Such an approach not only captures the inherent and model-specific uncertainties but also does so without necessitating intricate changes to the network architecture or its training protocol.

III. Background

A. Problem Description

Let $\mathcal{S}$ and $\mathcal{A}$ be the state and action spaces, respectively, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ be a reward function. A trajectory is a sequence of T states, actions, and rewards

$$\boldsymbol{\tau} = (s, a, r)_t \in \mathcal{S} \times \mathcal{A} \times \mathbb{R}, 0 \leq t < T, \tag{1}$$

and its return is the sum of the time-steps rewards $R(\boldsymbol{\tau}) = \sum_t r_t$. The goal of the agent is to learn a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that predicts the next action to take given the current environment's state, such that it maximizes the expected return $\mathbb{E}\left[R(\boldsymbol{\tau})\right]$ over the trajectories.

In offline reinforcement learning, the agent learns from a dataset of trajectories that were collected through various, potentially suboptimal, policies. Unlike in online reinforcement learning, the agent does not have the opportunity to explore the environment or collect new data based on its current policy.

At test time, the environment is initialized with a state randomly selected from an initial state distribution, $s_0 \sim \rho_0$. Each action results in a new state, determined by the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$. This process repeats in a receding horizon control loop until a termination condition is met, ending the episode.

The task of policy learning can be divided into two phases: sequence modeling of future states followed by action generation based on those predictions.

B. Generative Modeling for States Prediction

The first phase involves learning the data distribution of state trajectories. Given a training dataset of trajectories $\mathcal{D} = \{\boldsymbol{\tau}^i\}_{0 \leq i < N}$, we extract sequences of states $\boldsymbol{s} = (s)_j, 0 \leq j < H$ of horizon H. The goal is to estimate the underlying conditional distribution $p_{\text{data}}(\cdot|s_t)$, where s_t is the initial state from which the prediction of future states begins. We create a model that represents a parameterized distribution p_θ, and we tune its parameters, θ, by minimizing a divergence between p_θ and p_{data}. Then we can generate new sequences of states by sampling from the model distribution $\hat{\boldsymbol{s}} = (s_t, \hat{s}_{t+1}, \ldots, \hat{s}_{t+H-1}) \sim p_\theta(\boldsymbol{s}|s_t)$. We use the hat notation to distinguish between predicted and observed states.

Following Janner et al. [9], we use a Denoising Diffusion Probabilistic Model (DDPM) [16] to generate state sequences. DDPM progressively adds noise to a data distribution until it becomes pure noise, and subsequently reverse that process through a Markov chain with learned transition kernels to generate trajectories from noise.

Given the data distribution $\boldsymbol{s}^0 \sim p_{\text{data}}(\boldsymbol{s}^0)$, the forward noising process produces a sequence of random vectors $\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^K$ with the transition kernel

$$q(\boldsymbol{s}^{k+1}|\boldsymbol{s}^k) = \mathcal{N}(\boldsymbol{s}^{k+1}; \sqrt{\alpha_k}\boldsymbol{s}^k, (1 - \alpha_k)\mathbf{I})$$

where α_k is the noise scale schedule. The number of diffusion steps K is chosen big enough such that $\boldsymbol{s}^K$ approximately follows a standard Normal distribution. The reverse denoising process is modeled by the learnable transition kernel

$$p_\theta(\boldsymbol{s}^{k-1}|\boldsymbol{s}^k) = \mathcal{N}(\boldsymbol{s}^{k-1}|\mu_\theta(\boldsymbol{s}^k, k), \Sigma_k),$$

starting with $\boldsymbol{s}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The reverse process is trained to match the actual time reversal of the forward process. The loss takes the form of

$$\mathbb{E}_{k \sim \mathcal{U}[\![1,K]\!], \boldsymbol{s}^0 \sim p_{\text{data}}(\boldsymbol{s}^0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{s}^k, k)\|^2\right]$$

where $\boldsymbol{s}^k$ is computed from $\boldsymbol{s}^0$ and $\boldsymbol{\epsilon}$, $\mathcal{U}[\![1, K]\!]$ is the discrete uniform distribution over $\{1, 2, \ldots, K\}$, and $\boldsymbol{\epsilon}_\theta$ is a deep neural network with parameters θ that predicts the noise $\boldsymbol{\epsilon}$ given $\boldsymbol{s}^k$ and k.

C. Action Prediction

Given the current state s_t, we first predict a horizon of future states $\hat{\boldsymbol{s}} \sim p_\theta(\boldsymbol{s}|s_t)$. In the second phase, the objective is to determine the action a_t that transitions s_t to the next predicted state $\hat{s}_{t+1}$ in the sequence $\hat{\boldsymbol{s}}$. To achieve this, we introduce an action model $f_\phi(s_t, s_{t+1}) = a_t$. The learning objective of f_ϕ, detailed in section IV-B, is to accurately predict a_t.

The overall generative modeling policy is the composition of the generative and action models.

Algorithm 1 Uncertainty-Based Adaptive Planning

Require: Generative distribution of states p_θ, ensemble of action models $E = \{f_{\phi_1}, f_{\phi_2}, \ldots, f_{\phi_M}\}$, uncertainty threshold δ

$t \leftarrow 0$
Observe initial state state s_0
while episode is not done **do**
  Predict futrue states $\hat{\boldsymbol{s}} \sim p_\theta(\boldsymbol{s}|s_t)$
  Predict action a_t and uncertainty u_t with E (Eqs 3, 4)
  Execute a_t, increment t, and observe new state s_t
  Update the state $\hat{s}_1 \leftarrow s_t$
  Predict a_t and u_t with E
  $i \leftarrow 1$
  while $i < H - 1$ and $u_t < \delta$ **do**
    Execute a_t, increment t, and observe new state s_t
    Update the state $\hat{s}_i \leftarrow s_t$
    Predict a_t and u_t with E
    $i \leftarrow i + 1$
  end while
end while

IV. Adaptive Decision-Making under Uncertainty

A. Adaptive Policy

Planning with generative models, though effective, incurs significant computational costs when performed at every iteration of the control loop. This is especially true for diffusion models because they require numerous passes through the underlying deep neural network to gradually produce data from random noise. In contrast, action models, designed with simpler architectures, require considerably less computational effort. This insight leads to a strategy whereby the agent leverages the generative model's capacity to predict a long horizon of future states to execute multiple actions consecutively.

This strategy naturally leads to a critical question: When should the agent update its planned trajectory of states or choose to execute an action? We propose to use the uncertainty of the action model's predictions as a criterion for this process. To this end, the action model f_ϕ is modified to not only predict the next action, but also to estimate the uncertainty of its prediction. This approach hinges on the premise that higher uncertainty signals a greater necessity for re-evaluation of the plan through the generative model, ensuring that subsequent decisions are made with the most current observations of the environment.

Specifically, we introduce the following adaptive policy. Starting from the current state, we generate a trajectory of states using the generative model and compute the initial action using the action model, which is then executed. We then continue to predict and execute actions, adjusting the plan with each new observation, as long as the uncertainty remains below a predefined threshold. The policy is detailed in Algorithm 1.

The threshold is a tunable test-time hyperparameter that balances the trade-off between computational efficiency and safety. This flexibility allows users to adjust the threshold according to their specific needs without the necessity for retraining, enabling a single model to adapt to varying demands on computational resources and accuracy levels.

B. Deep Ensembles for Predictive Uncertainty Estimation

We implement a Deep Ensemble [11] of action models to estimate the predictive uncertainty, which combines both aleatoric and epistemic uncertainties. Aleatoric uncertainty, which arises from inherent noise in the data, is quantified using the model's output variance, while epistemic uncertainty, stemming from the model's lack of knowledge, is captured by the variability among the different models in the ensemble.

Deep ensembles are straightforward to implement and require minimal or no modifications to a standard action model's architecture, which in our case is a neural network with parameters ϕ.

Let x denote the input features to the model, and y the actual action observed in the data, against which the model's predictions are compared. The network's final layer outputs two values:

$$f_\phi(x) = \left(\mu_\phi(x), \sigma^2_\phi(x)\right),$$

where the predicted mean $\mu_\phi(x)$ represents the model's expectation of the output and the variance $\sigma^2_\phi(x)$ quantifies the model's aleatoric uncertainty in its predictions. The model is trained by minimizing the Negative Log-Likelihood (NLL):

$$-\log p_\phi(y|x) = \frac{1}{2}\log(\sigma^2_\phi(x)) + \frac{(y - \mu_\phi(x))^2}{2\sigma^2_\phi(x)}. \quad (2)$$

The first term, penalizes large variances. The second term, is essentially a scaled mean squared error that becomes more penalizing when the variance is small but the prediction error is large. This modeling approach assigns higher variance for inputs where the model predicts outcomes with less certainty.

To ensure the variance $\sigma^2_\phi(x)$ remains positive, we apply the softplus function to the network's variance output: $\log(1 + \exp(\cdot))$, and introduce a minimum variance of 10^{-6} for numerical stability.

Each model in the ensemble is trained on the entire dataset but initialized with random parameters to introduce diversity in the predictions. The action at time t is determined by averaging the mean predictions across all M ensemble members:

$$a_t = \frac{1}{M}\sum_{m=1}^{M} \mu_{\phi_m}(x). \quad (3)$$

The predictive uncertainty, capturing both aleatoric and epistemic uncertainties, is computed as the sum of the models' average variance and the variance of the ensemble's mean predictions:

$$u_t = \frac{1}{M}\sum_{m=1}^{M} \sigma^2_{\phi_m}(x) + \text{Var}\left(\{\mu_{\phi_m}(x)\}_{m=1}^{M}\right). \quad (4)$$

This approach gives us a principled way to quantify the model's uncertainty in its action predictions.

V. Experiments

A. Experimental Setup

This section evaluates the efficacy of our proposed adaptive policy, which we name Ensemble Action, in offline reinforcement learning (RL) control tasks. We utilize the D4RL Hopper and Walker locomotion environments with different dataset settings:

- **Medium**: Generated from 1 million timesteps by a medium policy, achieving approximately one-third of an expert policy's score.
- **Medium-Replay**: Includes the replay buffer from an agent trained to a medium policy's performance level.
- **Medium-Expert**: Combines 1 million timesteps from the medium policy with an additional 1 million timesteps from an expert policy.

We assess Ensemble Action's performance in terms of time and accuracy. Our analysis focuses on potential reductions in network function evaluations (NFEs) [23] required by the diffusion model. Table I demonstrates the trade-off between the average normalized reward [12] and the percentage of saved NFEs, compared to a policy that samples from the diffusion model at every timestep.

Given its foundational role in decision-making with generative models, Decision Diffuser [8] serves as our primary benchmark. Sampling trajectories with Decision Diffuser involves 100 denoising iterations using DDPM whenever the agent take an action, significantly increasing the planning time. Additionally, we compare our approach to Static Plan Execution, which commits to a predetermined sequence of actions for the entire horizon without replanning. This method, while computationally efficient due to the absence of ongoing planning, assumes the initial plan remains optimal throughout its execution, potentially limiting effectiveness in dynamic or unpredictable conditions.

In our experiments, we set the planning horizon (H) to 100. We use identical diffusion models for state prediction in both Ensemble Action and Decision Diffuser. The action model is a simple 2-layer perceptron, with each layer consisting of 512 units. We create ensembles of 5 action models and conduct 50 random simulations for each task. We selected different uncertainty thresholds (δ) for each dataset, choosing values that achieved a favorable balance between prediction accuracy and time.

As anticipated, Static Plan Execution shows lower return, likely due to compounded errors from individual actions, underscoring the importance of adaptability in dynamic settings. In contrast, Ensemble Action maintains comparable rewards to the Decision Diffuser baseline while significantly reducing the need for network function evaluations by up to 93%. However, an outlier in this trend is observed in the Medium-Replay Hopper scenario for the NLL-trained Ensemble Action, where only 17% of NFEs were saved. This variability emphasizes the need for adaptive thresholds and possibly fine-tuning the decision criteria based on the characteristics of each dataset to optimize both performance and efficiency.

We also evaluate the computational efficiency of generating 100 steps on a Tesla V100-PCIe-32GB GPU by comparing the generation times of an ensemble of action models against those of Decision Diffuser and Decision Transformer [7]. The results, presented in Table II, indicate that the ensemble of action models achieves the fastest step generation times, recording 0.13 seconds for the Hopper and 0.16 seconds for the Walker environments. Decision Diffuser required significantly more time, approximately 20.19 seconds for Hopper and 20.81 seconds for Walker, demonstrating the superior efficiency of the ensemble approach in rapid step generation.

The plot in Figure 2 illustrates the time required to complete a 1000-step episode using Hopper environment for both Ensemble Action and Decision Diffuser. It is evident that our adaptive policy significantly outperforms Decision Diffuser in terms of computational efficiency. Specifically, our adaptive policy completes the 1000 steps in less than 25 seconds, while Decision Diffuser takes over 23 minutes. This difference highlights the efficiency of our approach in rapidly processing steps, thereby enabling quicker decision-making.

B. Ablation Study

Although training with NLL directly estimates aleatoric uncertainty by learning an additional output for variance, this introduces complexity and may not be necessary in environments with deterministic state transition functions. In contrast, while MSE loss does not inherently quantify aleatoric uncertainty, training action models with MSE is simpler and does not require any alterations to the original architecture or training procedures of the action model used in Decision Diffuser. In that case, the action model directly estimates the next action

$$f_\phi(x) = a_t$$

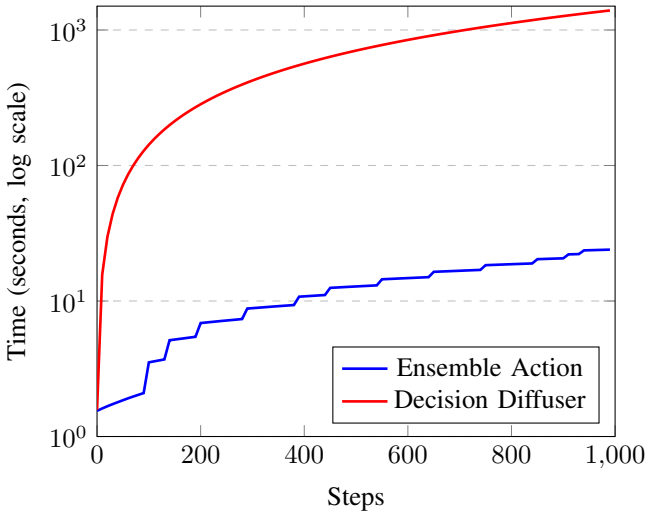

Fig. 2: *Ensemble Action completes 1000 steps in under 25 seconds, while the Decision Diffuser takes over 23 minutes, resulting in a 55x speedup.*

Dataset	Env.	DD		Static Plan Execution		EA MSE (Ours)		EA NLL (Ours)	
		Return	Saved NFE	Return	Saved NFE	Return	Saved NFE	Return	Saved NFE
Medium	Hopper	49.9	0%	5.3	98.7%	54.1	85.3%	62.1	91.1%
Medium	Walker	74.5	0%	3.9	98.7%	74.8	92.1%	52.5	76.8%
Med-Rep	Hopper	59.8	0%	7.1	98.7%	72.0	69.0%	69.7	17.0%
Med-Rep	Walker	62.7	0%	13.3	98.7%	66.4	91.3%	62.8	90.6%
Med-Exp	Hopper	110.0	0%	57.3	98.9%	109.0	84.6%	109.1	93.0%
Med-Exp	Walker	78.8	0%	15.6	98.8%	83.1	96.9%	80.4	89.7%

TABLE I: *This table presents the average normalized rewards achieved by Decision Diffuser (DD), Static Plan Execution, and Ensemble Action (EA) for both MSE and NLL training criteria. The table also reports the percentage of actions executed without sampling from the generative model.*

Env	DT	DD	EA (Ours)
Hopper	0.33	20.19	0.13
Walker	0.34	20.81	0.16

TABLE II: *Average time to generate 100 steps in seconds for Decision Transformer (DT), Decision Diffuser (DD), and Ensemble Action (EA)*

and the training objective is

$$\mathbb{E}_{a_t \in \mathcal{D}} \left[\|a_t - f_\phi(x)\|^2 \right].$$

The total uncertainty in Equation 4 is replaced by the epistemic uncertainty alone:

$$u_t = \text{Var}\left(\{ f_{\phi_m}(x) \}_{m=1}^{M} \right).$$

We present the results of the Ensemble Action trained with MSE in Table I. In our experiments, we find that training with MSE yields results comparable to those obtained with NLL, while being simpler.

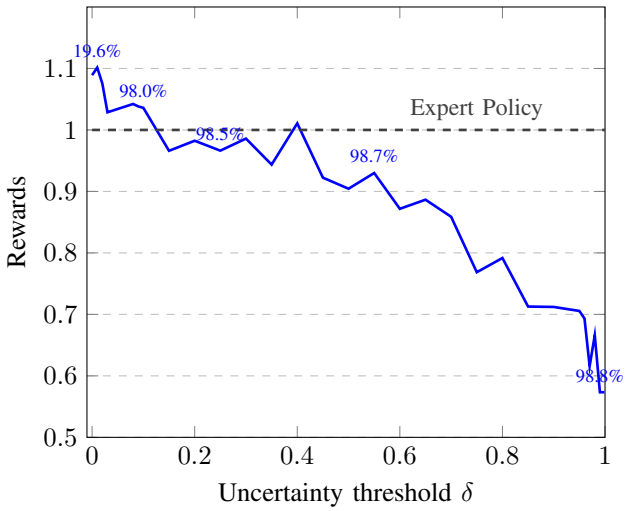

Fig. 3: *Impact of varying δ on rewards for the Ensemble Action model in the Hopper Medium-Expert dataset, with specific δ values showing saved NFEs.*

We also explored the impact of varying the size of the ensemble within the Hopper and Walker environments, using the Medium-Expert dataset. In these experiments, we train the ensemble members with MSE loss. Our findings indicate that reducing the ensemble size to four or three action models still yields robust performance. This reduction decreases computational overhead at test time and reduces training time, as fewer models require less time to train. Notably, when the ensemble size is reduced to one ($M = 1$), the Ensemble Action method essentially replicates the Decision Diffuser algorithm, which explains why the saved NFEs for $M = 1$ are 0%. The detailed outcomes of these experiments are presented in Table III.

Furthermore, we analyzed how the uncertainty threshold influences replanning decisions and overall reward outcomes, focusing on the Hopper Medium-Expert environment. Our findings, illustrated in Figure 3, demonstrate that fine-tuning the uncertainty threshold allows us to significantly reduce the need for NFEs by up to 98%, while maintaining expert-level rewards. Additionally, we observed, as expected, that the rewards increase as the uncertainty threshold decreases. These results underscore the potential of our approach to dramatically enhance computational efficiency without compromising the quality of decision-making.

VI. Discussion and Conclusion

In this study, we introduced an adaptive policy aimed at reducing planning time when using generative models. Our policy first produces a trajectory of future states with the generative model. Then, a Deep Ensemble of action models interprets this trajectory to predict the next actions, continuing to do so as long as its uncertainty stays below a threshold. This mechanism enables the policy to adaptively determine when to replan, optimizing computational efficiency while ensuring decisions are based on reliable predictions.

This adaptive strategy significantly reduces the need for frequent calls to the generative model–approximately 90% fewer calls in most of our experiments–without sacrificing decision quality. However, there are limitations and areas for future research that should be addressed. Exploring the applicability and scalability of our approach to more complex scenarios, such as real-world robotics and autonomous

Dataset	Environment	Number of Ensemble Members M				
		$M = 1$	$M = 2$	$M = 3$	$M = 4$	$M = 5$
Medium-Expert	Hopper	110.0 (0%)	106.6 (78.2%)	102.1 (97.7%)	107.7 (98.1%)	109.0 (84.6%)
Medium-Expert	Walker	78.8 (0%)	79.5 (89.1%)	78.3 (97.2%)	75.2 (96.9%)	83.1 (96.9%)

TABLE III: *Impact of the Number of Ensemble Members M on the Ensemble Action Performance. This table displays the average normalized reward and the percentage of saved NFEs.*

driving, presents exciting challenges. Additionally, it would be interesting to compare the computational effort of a non-generative state-of-the-art offline RL method, such as an actor-critic method, with our proposed method. Further research could also explore this strategy with other types of policies or generative models that decouple the state and action prediction phases.

In conclusion, our research lays the groundwork for more efficient use of generative models in decision-making, suggesting a path toward real-time decision-making systems.

REFERENCES

[1] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[3] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet *et al.*, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[4] OpenAI, "Video Generation Models as World Simulators," https://openai.com/research/video-generation-models-as-world-simulators, 2023, accessed: 2024-02-26.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[6] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 1273–1286.

[7] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[8] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" in *International Conference on Learning Representations*, 2023.

[9] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*, 2022.

[10] S. Zhou, Y. Du, S. Zhang, M. Xu, Y. Shen, W. Xiao, D.-Y. Yeung, and C. Gan, "Adaptive online replanning with diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[12] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[14] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[15] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[16] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[17] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[18] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[19] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[20] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, "AdaptDiffuser: Diffusion models as adaptive self-evolving planners," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 20 725–20 745.

[21] H. He, C. Bai, K. Xu, Z. Yang, W. Zhang, D. Wang, B. Zhao, and X. Li, "Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning," *Advances in neural information processing systems*, vol. 36, 2024.

[22] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman, and I. Mitliagkas, "Gotta go fast when generating data with score-based models," *arXiv preprint arXiv:2105.14080*, 2021.

[23] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," in *Proc. NeurIPS*, 2022.

[24] E. Luhman and T. Luhman, "Knowledge distillation in iterative generative models for improved sampling speed," *arXiv preprint arXiv:2101.02388*, 2021.

[25] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *International Conference on Learning Representations*, 2022.

[26] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 297–14 306.

[27] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," *arXiv preprint arXiv:2303.01469*, 2023.

[28] Y. Song and P. Dhariwal, "Improved techniques for training consistency models," *arXiv preprint arXiv:2310.14189*, 2023.

[29] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., 1996.

[30] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *International Conference on Machine Learning*, 2015, pp. 1613–1622.

[31] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, 2011.

[32] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International conference on machine learning*, 2016, pp. 1050–1059.