CLaP: Conditional Latent Planners for Offline Reinforcement Learning

Harry Donghyeop Shin
Stanford University
ds816@stanford.edu

Rose E. Wang
Stanford University
rewang@stanford.edu

Abstract

Recent work has formulated offline reinforcement learning (RL) as a sequence modeling problem, benefiting from the simplicity and scalability of the Transformer architecture. However, sequence models struggle to model trajectories that are long-horizon or involve complicated environment dynamics. We propose CLaP (Conditional Latent Planners) to learn a simple goal-conditioned latent space from offline agent behavior, and incrementally decode good actions from a latent plan. We evaluate our method on continuous control domains from the D4RL benchmark. Compared to non-sequential models and return-conditioned sequential models, CLaP shows competitive if not better performance across continuous control tasks. It particularly does better in environments with complex transition dynamics with up to $+149.8\%$ performance increase. Our results suggest that decision-making is easier with simplified latent dynamics that models behavior as being goal-conditioned.[1]

1 Introduction

Reinforcement learning (RL) has seen significant advances with the introduction of deep neural networks, achieving excellent results across a wide range of domains (Tesauro, 1994; Hafner and Riedmiller, 2011; Levine and Koltun, 2013; Mnih et al., 2013; Levine et al., 2016; Silver et al., 2017; Kalashnikov et al., 2018). These RL methods generally involve continued interactions with the environment, using experience to improve the learned policy. However, online interaction is expensive and potentially dangerous. This prompts the need for offline RL methods that allow for data to be collected in a safe way, and then have agents learn from this offline dataset (Haarnoja et al., 2018; Fujimoto et al., 2019; Wu et al., 2019; Kumar et al., 2020).

Learning from offline data is challenging because the pre-collected dataset of agent behavior may be limited in quantity and quality (Levine et al., 2020). Low data quantity leads to generalization issues, where a majority of states are out-of-distribution from the agent's training distribution. Low data quality leads to learning sub-optimal behavior, where the provided demonstrations are not generated by experts.

Recent works apply sequence models for decision making to overcome these issues (Janner et al., 2021; Chen et al., 2021). Sequence models streamline learning by not requiring additional algorithm constraints such as policy regularization (Fujimoto et al., 2019; Kumar et al., 2019) and conservatism (Yu et al., 2020; Kidambi et al., 2020). However, there are two limitations in prior sequence modeling approaches for RL. Chen et al. (2021) models trajectories as return-conditioned, which can cause issues when the optimal return is not known (Brandfonbrener et al., 2022). Janner et al. (2021)'s approach models trajectories as goal-conditioned by prepending the goal at the start of the sequence. However, sequence models struggle to attend to goals when the sequences are long or the environment

[1]Videos and code are available here:https://github.com/dh2shin/conditional_latent_planners

Foundation Models for Decision Making Workshop at Neural Information Processing Systems, 2022.

dynamics is complicated (Kiddon et al., 2016; Fan et al., 2019; Hua and Wang, 2020). Recent work in natural language processing (NLP) overcome these issues by modeling text as generated from a goal-conditioned latent process (Wang et al., 2021). The work assumes the latent dynamics is *simple* and *known*, which makes generating long coherent sequences easier.

Our work proposes Conditional Latent Planners (CLaP), a sequence-based model that decodes actions from a learned goal-conditioned latent space. We incorporate the ideas from Wang et al. (2021), which generates language from a goal-conditioned latent process, and apply them to the offline RL setting. In the offline RL setting, CLaP generates actions as sampled from a goal-conditioned latent process learned from offline data. Note that despite methodological similarities, the offline RL setting is very different from the NLP setting. For example, they assume the sequence model has been pretrained and can be further finetuned on large datasets, whereas we assume access to smaller and potentially low-quality data. We evaluate our method against a prior state-of-the art non-sequential method Conservative Q-Learning and the original Decision Transformer model on the D4RL benchmark (Fu et al., 2020; Kumar et al., 2020; Chen et al., 2021). We show that CLaP outperforms Conservative Q-Learning and Decision Transformer on tasks with particularly complex transition dynamics and matches their performance on other tasks.

2 Related Work

Latent planning methods Planning in unknown environments is difficult because it requires a faithful model of the world. Obtaining this model is often challenging and even impossible, especially in high-dimensional or image-based domains (Hafner et al., 2019; Moerland et al., 2020; Schrittwieser et al., 2021). Prior work circumvent the need to specify a world model for the raw state space with latent-based methods (Ha and Schmidhuber, 2018; Oord et al., 2018). They learn the latent space with a simple Variational Autoencoder or noise-based contrastive learning (Gutmann and Hyvärinen, 2010; Kingma and Welling, 2013; Rezende et al., 2014). However, without assuming temporal structure in the latent space, these methods often fail to predict future latents accurately. Accumulated prediction errors can result in decoding bad actions, leading to low returns. Our method overcomes these challenges by assuming a goal-conditioned structure for planning in latent space.

Wang et al. (2021) is a recent work in NLP which models language with goal-conditioned stochastic processes, namely the Brownian bridge process (Revuz and Yor, 2013). Their work assumes that each document is generated from a process that is pinned to the same latent start and goal. Intermediate latents are Gaussian distributions centered at interpolated points between the two, and with variances that are a function of time. Despite methodological similarities, we note key differences in the setting. For example, their work assumes that the sequence model has been pretrained on a large dataset (Radford et al., 2019) whereas ours is pretrained on an offline dataset that is much smaller in comparison. Their work assumes that the model has been pretrained and can be further finetuned on high-quality text, whereas we assume a varying quality of data. Finally, they assume that the documents are generated from a process pinned at the same start and goal, whereas we assume that the trajectories are generated from processes with different start and goal states. Our work focuses on addressing the challenges that arise from these differences.

Goal-conditioned RL A number of prior works have trained goal-conditioned RL policies using supervised learning and model-free RL (Kaelbling, 1993; Nair et al., 2018; Ghosh et al., 2019; Eysenbach et al., 2019). In the offline setting, many methods employ goal-conditioned Q-learning with hindsight relabeling techniques to incorporate goal information and to be sample efficient (Andrychowicz et al., 2017; Chebotar et al., 2021; Tian et al., 2021). These relabeling methods are related to how CLaP learns a goal-conditioned latent structure over agent trajectories. However, unlike relabeling methods, CLaP uses this latent structure to simplify dynamics and actively decode intermediate latents and actions needed to reach a desired goal state.

Sequence models for RL Our work builds off of recent sequence-modeling approaches for RL, such as Decision Transformer (Chen et al., 2021) and Trajectory Transformer (Janner et al., 2021). Decision Transformer is a causally masked Transformer that generates actions conditioned on past states, actions, and desired returns. Trajectory Transformer is a predictive dynamics model trained on discretized states, actions, and rewards, which behaves like a model-based RL method when paired with planning algorithms. Trajectory Transformer's goal-reaching RL scheme prepends goal

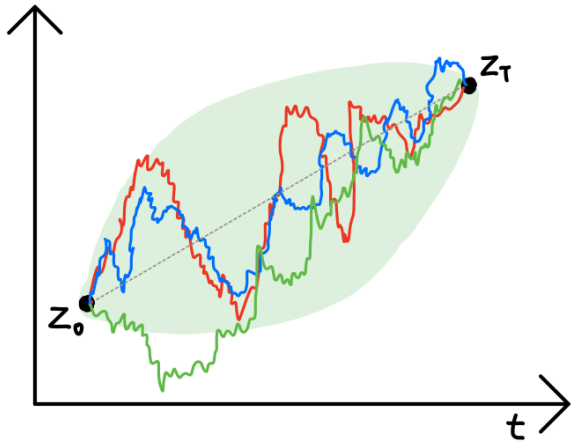


Figure 1: A Brownian bridge process pinned between start latent z_0 and goal latent z_T. The green oval indicates how uncertainty is greatest in the middle, and the three plots are possible latent paths that belong to the given process.

states to agent states in a raw, discretized format. However, such a basic form of goal-conditioning does not yield useful information on what goal-conditioning behavior actually looks like, especially for long-horizon tasks in complex environments. CLaP instead explicitly learns a latent space with goal-conditioned behavior and then uses it to decode good actions in a step-by-step, more granular fashion.

3 Offline RL Formalism

We formalize the offline RL setting described by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$, consisting of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $P(s' \mid s, a)$, and a reward function $r = \mathcal{R}(s, a)$. Similar to Chen et al. (2021), we use s_t, a_t, and $r_t = \mathcal{R}(s_t, a_t)$ to denote the state, action, and reward at timestep t. A trajectory can then be described by a sequence of states, actions, and rewards: $\tau = (s_0, a_0, r_0, s_1, s_2, s_3, \ldots, s_T, a_T, r_T)$. For the purposes of pretraining a Decision Transformer model, we represent trajectories using returns-to-go $\hat{R}_t = \sum_{t'=t}^{T} r'_t$ instead of rewards directly, resulting in the following representation for training and generation: $\tau = \left(\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \ldots, \hat{R}_T, s_T, a_T\right)$.

4 Conditional Latent Planners (CLaP)

In this section, we describe the training and inference procedures for CLaP. First, we discuss how CLaP is pretrained; this builds off of the Decision Transformer's training phase (Chen et al., 2021). We then discuss how CLaP's encoder and decoder are trained. The encoder maps states into latents, from which the decoder infers what actions to take. Finally, we explain how CLaP is used at inference time.

4.1 Pretraining

CLaP builds on the Decision Transformer architecture. We denote the policy as π_θ^{base}. We pretrain using the same objective as the Decision Transformer, namely to minimize error between the ground-truth action from the dataset a_t and the predicted action from the sequence-based policy $\hat{a}_t$ over a context length of C.

$$J(\theta) = \frac{1}{C}\left(\sum_{t=1}^{C} J^{(t)}(\theta)\right) = \frac{1}{C}\sum_{t=1}^{C}(a_t - \hat{a}_t)^2 \tag{1}$$

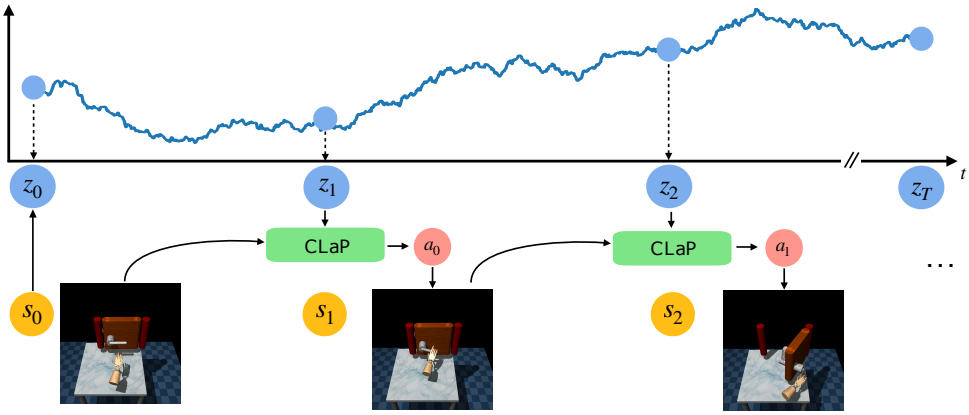


Figure 2: CLaP makes decisions conditioned on a latent plan. The learned encoder first maps the initial state s_0 and the goal state s_T to their latent representation z_0 and z_t. (top) The latent plan is sampled from a Brownian bridge process, which is a stochastic process pinned at the start latent z_0 and end latent z_T. CLaP samples from the process to produce a trajectory show in blue. (bottom) CLaP decodes actions a_t conditioned on the next latent z_{t+1} and the current state s_t.

The above mean-squared error is used for environments with continuous action spaces. In the discrete setting, the loss can be swapped with cross-entropy loss (Chen et al., 2021).

4.2 Encoder Training

The encoder maps states to low-dimensional latents, $f_\theta : \mathcal{S} \to \mathcal{Z}$. We model latent structure as in Wang et al. (2021), where the latents follow a Brownian bridge process. The Brownian bridge process has the following density pinned at a start point z_0 at $t = 0$ and end point z_T at $t = T$:

$$p(z_t|z_0, z_T) = \mathcal{N}\left((1 - \frac{t}{T})z_0 + \frac{t}{T}z_T, \frac{t(T-t)}{T}\right) \tag{2}$$

This density function denotes how z_t should be more like z_0 at the start, z_T at the end, and linearly interpolated in between but with greater uncertainty. Figure 1 illustrates this uncertainty in the Brownian bridge process. We use the same contrastive objective as in Wang et al. (2021). The objective maps a triplet of states s_1, s_2, s_3 to $f_\theta(s_1)$, $f_\theta(s_2)$, $f_\theta(s_3)$ and fits a Brownian bridge density in Equation 2.

Formally, given multiple sequences of agent states in trajectories, $S = \{s_1, \ldots, s_N\}$, we draw batch $\mathcal{B} : \{(s_0, s_t, s_T)\}$ of randomly sampled positive triplets s_0, s_t, s_T where $0 < t < T$. Then, our encoder is optimized by

$$\mathcal{L}_N = \mathbb{E}_X\left[-\log \frac{\exp\left(\mathrm{d}\left(s_0, s_t, s_T; f_\theta\right)\right)}{\sum_{t' \in B} \exp\left(\mathrm{d}\left(s_0, s_{t'}, s_T; f_\theta\right)\right)}\right] \quad \text{where} \tag{3}$$

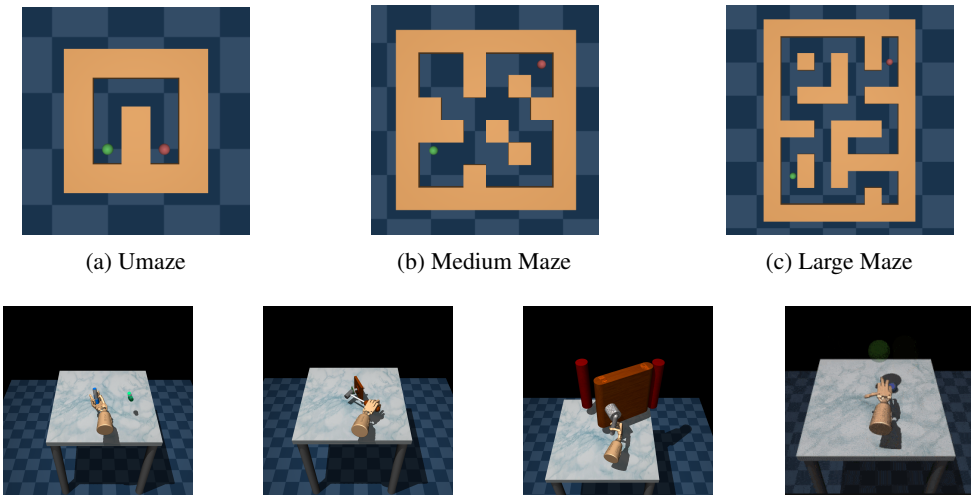
$$\mathrm{d}\left(s_0, s_t, s_T; f_\theta\right) = -\frac{1}{2\sigma^2}\|f_\theta\left(s_t\right) - \left(1 - \frac{t}{T}\right) f_\theta\left(s_0\right) - \frac{t}{T} f_\theta\left(s_T\right)\|_2^2 \tag{4}$$

This objective can be viewed as maximizing the extent to which true triplets from agent trajectories follow the Brownian bridge process while minimizing the extent to which alternate midpoints sampled from other trajectories do so.

One crucial difference between Wang et al. (2021) and CLaP is that Wang et al. (2021) maps all start points to a single latent, and all end points to another latent; their start points are centered at 0 and end points at 1. Our work does not pin start and end latents because we want to learn a policy that can generalize to any start and goal state.

4.3 Decoder Training with Latent Paths

After training the encoder, we train the decoder to produce actions using information from latent space. Let $\left(\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \ldots, \hat{R}_C, s_C\right)$ denote the original input sequence, where C is the



(a) Umaze (b) Medium Maze (c) Large Maze

(d) Pen (e) Hammer (f) Pen (g) Relocate

Figure 3: D4RL Maze2D & Adroit Environments

context length. We finetune the policy to produce an action similar to ground-truth action a_C. The finetuning process is as follows. We first edit the trajectory to incorporate future information for $t \in [1, C]$. We fetch the next state $s_{t+1} \in \mathbb{R}^d$ and encode the state with the trained encoder f_θ. We now have the latent embedding $z_{t+1} \in \mathbb{R}^{d'}$. To provide the policy with information about the immediate future latent, we add s_t and z_{t+1} together. If the two do not share dimension size, we learn a matrix $M \in \mathbb{R}^{d \times d'}$ which maps the latent to the state space dimensions. The new state is then $\hat{s}_t = s_t + M z_{t+1}$. We finetune our pretrained model from Section 4.1 with the new input sequence $(\hat{R}_1, \hat{s}_1, a_1, \ldots, \hat{R}_C, \hat{s}_C)$.

4.4 CLaP at inference time

Figure 2 illustrates the inference setup. Given the initial state s_0 and goal state s_T, we encode them to produce the latent codes z_0 and z_T. We sample a latent path with Brownian bridge dynamics in Equation 2. This produces a sequence $(z_0, z_1, \ldots, z_T)$. We then decode actions conditioned on the past input sequence of context-length C and the future latent target: at step t, CLaP takes in the sequence $(R_{t-C}, \hat{s}_{t-C}, a_{t-C}, \ldots, R_t, \hat{s}_t)$ which already incorporates z_{t+1} in $\hat{s}_t$, and generates a_t.

When the goal state is not known, we take it to be the average of the terminal states. Similarly, when T is unknown, we take it to be the average sequence length from our training data. This is a limitation of our method, particularly when dealing with completely new distributions at test time. In realistic scenarios, we can expect the goal to be known ahead of time. Despite these limitations, we find that CLaP still performs well in D4RL simulated environments, suggesting the learned latent dynamics is simple enough to generalize to unseen states.

5 Experiments

In this section, we investigate the performance of CLaP compared to two state-of-the-art methods. One is a non-sequential method, Conservative Q-Learning (CQL) (Kumar et al., 2020), and the other is a sequence-based method, Decision Transformer (DT) (Chen et al., 2021). We focus on how CLaP compares to DT, analyzing how much sequence models can benefit from latent goal-conditioning and incremental decoding from a latent path. We evaluate on two continuous control tasks from the D4RL benchmark, which require fine-grained continuous control (Fu et al., 2020). Maze2D is a navigation task where a 2D agent tries to reach a fixed goal location. The Adroit domain requires control of a 24 degree-of-freedom simulated robotic hand to hammer a nail, open a door, twirl a pen, or pick up and move a ball. For the four adroit domain tasks, we consider the human and expert data settings. Figure 3 illustrates all the environments used in our work. We use the default hyperparameters detailed in Chen et al. (2021) for training, which is done on a single Nvidia Titan Xp GPU.

Task Name	Base DT	CLaP	CQL
Umaze	**65.7** $\pm$ **1.4**	60.2 $\pm$ 1.7	5.7
Medium	**77.5** $\pm$ **5.1**	69.8 $\pm$ 2.5	5.0
Large	156.6 $\pm$ 1.9	**180.8** $\pm$ **1.7**	12.5
Pen-Expert	114.1 $\pm$ 6.2	**119.0** $\pm$ **5.8**	107.0
Hammer-Expert	**127.0** $\pm$ **0.2**	123.7 $\pm$ 1.3	86.7
Door-Expert	**103.4** $\pm$ **0.8**	101.4 $\pm$ 2.5	101.5
Relocate-Expert	**104.9** $\pm$ **1.0**	92.2 $\pm$ 1.2	95.0
Pen-Human	25.2/69.4 $\pm$ 8.4/21.9	**43.5/103.7** $\pm$ **3.8/8.3**	37.5/-
Hammer-Human	1.6/24.1 $\pm$ 0.0/2.7	**0.7/60.2** $\pm$ **0.0/7.7**	4.4/-
Door-Human	**1.5/19.2** $\pm$ **0.2/3.6**	0.1/-0.8 $\pm$ 0.0/0.0	9.9/-
Relocate-Human	0.0/0.0 $\pm$ 0.0/0.0	0.1/0.0 $\pm$ 0.0/0.1	**0.2/-**

Table 1: Normalized results comparing base DT, CQL, & CLaP. Average results are reported over 4 seeds, and normalized to a score between 0 (random) and 100 (expert). The highest scores are marked in gray cells.

We first compare sequential models DT and CLaP with the non-sequential model CQL. CQL represents the state-of-the-art non-sequential model-free offline RL method. CQL results are reported by the original paper (Kumar et al., 2020). Scores are normalized between 0 and 100, representing random and expert performance, respectively.[2]. Our results are shown in Table 1. We find that DT and CLaP almost always outperform CQL, demonstrating that we generally benefit using high-capacity sequential models for decision making.

We now analyze how goal-conditioning helps sequence models by comparing the two sequence models. CLaP achieves competitive performance in a majority of tasks and shows pronounced improvement over base DT in domains with complex transition dynamics: 15.5%, 49.4%, and 149.8% boosts in Large Maze, Pen, and Hammer Human.[3] This finding suggests that CLaP and its goal-conditioned latent paths effectively reduce the burden on the model from learning the complex transition dynamics of these environments. This enables the policy to more easily decode good actions, leading to better performance.

6 Conclusion

We proposed Conditional Latent Planners, seeking to incorporate goal-conditioned information into sequence models for decision making. On the D4RL benchmark, we showed CLaP matches the performance of DT, while outperforming on tasks that have particularly complex transition dynamics. This finding suggests that CLaP and its directed diffusion effectively reduce the burden on the model from learning the complex transition dynamics of these environments.

While our work empirically showed promising preliminary results with applying goal-conditioning even in domains with limited data, it is nevertheless challenging to learn good latent structures in high-variance, low-data scenarios. Future work can research how to learn better latent representations even in settings with limited data so that sequence models for decision making can benefit maximally from goal-conditioning.

Acknowledgments

Harry D. Shin's research was supported by the CURIS Summer Internship Program at Stanford University. Rose E. Wang is supported by the National Science Foundation Graduate Research Fellowship. The authors would like to give thanks to CoCoLab principal investigator Noah Goodman and the rest of CoCoLab for their support and helpful discussions.

[2]Due to D4RL environment constraints, the Adroit human tasks have timeouts set at maximum horizon of the environment t instead of when demonstrations end $\hat{t}$ (i.e. trajectories in the dataset are much longer than maximum episode length), so we include results for both $(t/\hat{t})$

[3]The pen and hammer tasks have 45, 46 dimensional observation spaces compared to 39 for door and relocate.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. (2017). Hindsight experience replay. *Advances in neural information processing systems*, 30.

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. (2022). When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv:2206.01079*.

Chebotar, Y., Hausman, K., Lu, Y., Xiao, T., Kalashnikov, D., Varley, J., Irpan, A., Eysenbach, B., Julian, R., Finn, C., et al. (2021). Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. (2019). Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Fan, A., Lewis, M., and Dauphin, Y. (2019). Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.

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

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. (2019). Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

Ha, D. and Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR.

Hafner, R. and Riedmiller, M. (2011). Reinforcement learning in feedback control. *Machine learning*, 84(1):137–169.

Hua, X. and Wang, L. (2020). Pair: Planning and iterative refinement in pre-trained transformers for long text generation. *arXiv preprint arXiv:2010.02301*.

Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286.

Kaelbling, L. P. (1993). Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823.

Kiddon, C., Zettlemoyer, L., and Choi, Y. (2016). Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 329–339.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. (2019). Stabilizing off-policy q-learning via bootstrapping error reduction. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative q-learning for offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

Levine, S. and Koltun, V. (2013). Guided policy search. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1–9, Atlanta, Georgia, USA. PMLR.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moerland, T. M., Broekens, J., and Jonker, C. M. (2020). Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Revuz, D. and Yor, M. (2013). *Continuous martingales and Brownian motion*, volume 293. Springer Science & Business Media.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR.

Schrittwieser, J., Hubert, T., Mandhane, A., Barekatain, M., Antonoglou, I., and Silver, D. (2021). Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219.

Tian, S., Nair, S., Ebert, F., Dasari, S., Eysenbach, B., Finn, C., and Levine, S. (2021). Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*.

Wang, R. E., Durmus, E., Goodman, N., and Hashimoto, T. (2021). Language modeling via stochastic processes. In *International Conference on Learning Representations*.

Wu, Y., Tucker, G., and Nachum, O. (2019). Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. (2020). Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142.