In-Context Decision-Making from Supervised Pretraining

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Large transformer models trained on diverse datasets have shown a remarkable ability to *learn in-context*, achieving high few-shot performance on tasks they were not explicitly trained to solve. In this paper, we study the in-context learning capabilities of transformers in decision-making problems, i.e., contextual bandits and Markov decision processes. We introduce a new pretraining method in which the transformer predicts an optimal action given a query state and an in-context dataset of interactions, across a diverse set of tasks. This pretraining procedure, while simple, produces an in-context algorithm with several surprising results. On a range of decision-making problems the learned algorithm simultaneously exhibits exploration online and conservatism offline, despite never being explicitly trained to do so. It also generalizes beyond the pretraining distribution to unseen tasks, in-context datasets, and reward noise. We show the learned algorithm can be viewed as an implementation of posterior sampling. We further leverage this connection to provide theoretical guarantees on its regret, and show that in situations where the training data is produced by an unknown, suboptimal regret-minimizing algorithm, the learned algorithm can achieve *provably lower regret* than the data-generating algorithm. These results suggest a promising yet simple path towards instilling strong in-context decision-making abilities in large transformer models.

1 Introduction

For supervised learning, transformer-based models trained at scale have shown impressive abilities to perform tasks given an input context, often referred to as few-shot prompting or in-context learning [Brown et al., 2020]: the model is presented with a small number of supervised input-output examples in its context, and is then asked to predict the most likely completion (i.e. output) of an unpaired input. In this work, we study in-context learning in sequential decision making settings, i.e., reinforcement learning (RL) settings. Here, the context takes the form of observation-action-reward tuples representing a dataset interactions with an unknown environment, and the agent must leverage these interactions when making decisions in the future. Notably, RL is considerably more varied and complex than supervised learning, and in-context RL inherits this complexity. For example, a hallmark of good decision-making in online RL algorithms is to select exploratory actions and improve with more interactions, whereas an RL agent trained from a static, suboptimal offline dataset should select actions conservatively. An in-context RL algorithm should do the same.

To enable this, we propose a new pretraining objective for decision-making, namely, to train a transformer to predict an optimal action[1] given a query state and an in-context dataset of interactions, across a diverse set of tasks. We refer to pretrained model as a Decision-Pretrained Transformer

---

[1] If not explicitly known, the optimal action can be determined by running any (potentially inefficient) minimax-optimal regret algorithm for each pretraining task.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

(DPT). Typically, the optimal action for a given query state is not completely determined by the context, and so our method effectively trains the transformer to perform posterior sampling [Osband et al., 2013]. For example, in the extreme case of an empty context, the transformer would learn the distribution of optimal actions with respect to the task distribution seen during pretraining. We demonstrate in theory and empirically that this simple pretraining produces an in-context algorithm, yielding surprising capabilities for learning in both online and offline decision-making tasks.

- **Pretraining to predict optimal actions gives rise to decision-making algorithms.** The pretraining objective is solely based on predicting optimal actions from in-context interactions. At the outset, it is not immediately apparent that these predictions at test-time would yield good decision-making behavior when the task is unknown and thus exploration is necessary. Intriguingly, this approach culminates in an algorithm that is capable of dealing with this uncertainty. For example, despite not being explicitly trained to explore, the model exhibits an exploration strategy on par with hand-designed algorithms, as a means to discover the optimal actions.
- **In-context algorithms generalize to new decision-making problems, offline and online.** The in-context algorithm can solve new tasks not seen during training, including unseen reward distributions on bandit problems and unseen goals, dynamics, and datasets in simple MDPs.
- **DPT can learn to leverage latent structure which prior algorithms required to be explicit**. We explore several instances of this behavior. For example, in parametric bandit problems, specialized algorithms can leverage structure (such as linear rewards) and offer provably better regret, but a representation must be known in advance. Perhaps surprisingly, we demonstrate that pretraining on linear bandit problems, even with unknown representations, leads DPT to select actions and explore in a way that matches an efficient linear bandit algorithm. This holds even when the source pretraining data comes from a suboptimal algorithm, demonstrating DPT's ability to learn improved strategies beyond what it was trained on.

2 Related Work

Meta-learning. Algorithmically, our work falls under the meta-learning framework [Schaul and Schmidhuber, 2010, Bengio et al., 1990]. At a high-level, these methods attempt to learn some underlying shared structure of the training distribution of tasks to accelerate learning of new tasks. While there is a choice in what shared 'structure' is specifically learned (e.g., the dynamics of the task [Fu et al., 2016, Nagabandi et al., 2018, Landolfi et al., 2019], a task context identifier [Rakelly et al., 2019, Humplik et al., 2019, Zintgraf et al., 2019, Liu et al., 2021], temporally extended skills and options [Perkins et al., 1999, Gupta et al., 2018, Jiang et al., 2022], or initialization of a neural network policy [Finn et al., 2017, Rothfuss et al., 2018]), we take a more agnostic approach by learning the learning algorithm itself [Duan et al., 2016, Wang et al., 2016, Mishra et al., 2017]. Algorithm Distillation (AD) [Laskin et al., 2022, Lu et al., 2023] also falls under this category, applying autoregressive supervised learning to distill (sub-sampled) traces of a single-task RL algorithm into a task-agnostic model. While our method DPT also leverages autoregressive SL, we take advantage of connections between posterior sampling and optimal regret-minimizing algorithms to derive an approach that is provably efficient.

Autoregressive transformers for decision-making. In decision-making fields such as RL and imitation learning, the use of transformer models trained using autoregressive supervised action prediction has proliferated in recent years [Yang et al., 2023], inspired by the successes of these techniques for large language models [Vaswani et al., 2017, Raffel et al., 2020, Brown et al., 2020]. For example, Decision Transformer (DT) [Chen et al., 2021, Janner et al., 2021] uses a transformer to autoregressively model sequences of actions from offline experience data, conditioned on the achieved return. During inference, one can then query the model with respect to a desired high return value. This approach has been shown to scale favorably to large models and multi-task settings [Lee et al., 2022], at times exceeding the performance of large-scale multi-task imitation learning with transformers [Reed et al., 2022, Brohan et al., 2022, Shafiullah et al., 2022]. However, DT is known to be provably (and unboundedly) sub-optimal in common scenarios [Brandfonbrener et al., 2022, Yang et al., 2022]. One common criticism of DT – and, in fact, autoregressive supervised learned transformers in general – is their inability to improve upon the dataset. For example, there is little reason for DT to output meaningful behavior if conditioned on a return higher than any of the returns observed in its training dataset, without excessively strong assumptions on the ability of the model to extrapolate [Brandfonbrener et al., 2022]. In contrast, a major contribution of our work is theoretical and empirical evidence for the ability of our method to improve over behaviors seen in the dataset in terms of regret, only relying on assumptions that parallel standard ones in offline RL settings.

Value and policy-based offline RL. Offline RL algorithms offer the opportunity to learn from existing datasets. To address distributional shift, many prior algorithms incorporate the principle of value pessimism [Kumar et al., 2020, Yu et al., 2021, Liu et al., 2020], or policy regularization [Fujimoto et al., 2019, Kumar et al., 2019, Siegel et al., 2020, Liu et al., 2019]. To reduce the amount of offline data required in a new task, methods for offline meta-RL can reuse interactions collected in a set of related tasks [Li et al., 2020, Mitchell et al., 2021, Dorfman et al., 2021]. However, they still must address distribution shift, requiring solutions such as policy regularization [Li et al., 2020] or additional online interactions [Pong et al., 2022]. Following the success of autoregressive models like DT and AD, our model avoids these issues. With our pretraining objective, our model also leverages offline datasets for new tasks more effectively than AD does.

3 In-Context Learning Model

Basic decision models. The basic decision model of our study is the finite-horizon Markov decision process (MDP). An MDP is specified by the tuple $\tau = \langle \mathcal{S}, \mathcal{A}, T, R, H, \rho \rangle$ to be solved, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward function, $H \in \mathbb{N}$ is the horizon, and $\rho \in \Delta(\mathcal{S})$ is the initial state distribution. A learner interacts with the environment through the following protocol: (1) an initial state s_1 is sampled from ρ; (2) at time step h, the learner chooses an action a_h and transitions to state $s_{h+1} \sim T(\cdot|s_h, a_h)$, and receives a reward $r_h \sim R(\cdot|s_h, a_h)$. The episode ends after H steps. A policy π maps states to distributions over actions and can be used to interact with the MDP. We denote the optimal policy as $\pi^\star$, which maximizes the value function $V(\pi^\star) = \max_\pi V(\pi) := \max_\pi \mathbb{E}_\pi \sum_h r_h$. Note this framework encompasses multi-armed bandit settings where the state space is a single point, e.g. $\mathcal{S} = \{1\}$, $H = 1$, and the optimal policy is $a^\star = \operatorname{argmax}_{a \in \mathcal{A}} \mathbb{E}\left[r_1 | a_1 = a\right]$.

Pretraining. Let $\mathcal{T}_{\text{pre}}$ be a distribution over tasks at the time of pretraining. A task $\tau \sim \mathcal{T}_{\text{pre}}$ can be viewed as a specification of an MDP, $\tau = \langle \mathcal{S}, \mathcal{A}, T, R, H, \rho \rangle$. The distribution $\mathcal{T}_{\text{pre}}$ can span different reward and transition functions and even different state and action spaces. We then sample a context (or a prompt) which consists of a dataset D of interactions between the learner and the MDP specified by τ. $D = \{s_j, a_j, s'_j, r_j\}_{j \in [n]}$ is a (potentially ordered) collection of transition tuples taken in τ We refer to D as the *in-context dataset* because it provides the contextual information about τ. D could be generated through variety of means, including but not limited to: (1) random interaction data within τ, (2) demonstrations from an expert, and (3) rollouts of an algorithm. Additionally, we independently sample a query state s_{query} from the distribution $\mathcal{D}_{\text{query}}$ over states and a label $a^\star$ is sampled from the optimal policy $\pi^\star_\tau(\cdot|s_{\text{query}})$ for task τ (see Section 5.3 for how to implement this in common practical scenarios). We denote the joint pretraining distribution over tasks, in-context datasets, query states, and action labels as P_{pre}:

$$P_{pre}(\tau, D, s_{\text{query}}, a^\star) = \mathcal{T}_{\text{pre}}(\tau)\mathcal{D}_{\text{pre}}(D;\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\pi^\star_\tau(a^\star|s_{\text{query}}) \tag{1}$$

Given the in-context dataset D and a query state s_{query}, we can train a model to predict the optimal action $a^\star$ in response. Let $D_j = \{(s_1, a_1, s'_1, r_1), \ldots, (s_j, a_j, s'_j, r_j)\}$ denote the partial dataset up to j samples. Formally, we aim to train a model M parameterized by θ, which outputs a distribution over actions $\mathcal{A}$, to minimize the expected loss over samples from the pretraining distribution:

$$\min_\theta \mathbb{E}_{P_{pre}} \textstyle\sum_{j \in [n]} \ell\left(M_\theta(\cdot \mid s_{\text{query}}, D_j), a^\star\right) \tag{2}$$

We will use a cross-entropy loss function: $\ell(M_\theta(\cdot\ |s_{\text{query}}, D_j), a^\star) := -\log M_\theta(a^\star \mid s_{\text{query}}, D_j)$ throughout, essentially treating this as a classification problem for finite $\mathcal{A}$. The resulting output model M_θ can be viewed as an algorithm that takes in a dataset of interactions D and can be queried for predictions of the optimal action via inputting a query state s_{query}.

Testing. After pretraining, a new task (MDP) τ is sampled from a test-task distribution $\mathcal{T}_{\text{test}}$. If the model is to be tested *offline*, then a dataset (prompt) is a sampled $D \sim \mathcal{D}_{\text{test}}(\,\cdot\,;\tau)$ and the policy that the model in-context learns is given conditionally as $M_\theta(\cdot \mid \cdot, D)$. Namely, we evaluate the policy by having it iteratively interact with the environment, selecting action $a_h \in \operatorname{argmax}_a M_\theta(a|s_h, D)$ when the learner visits state s_h. If the model is to be tested *online* through multiple episodes of

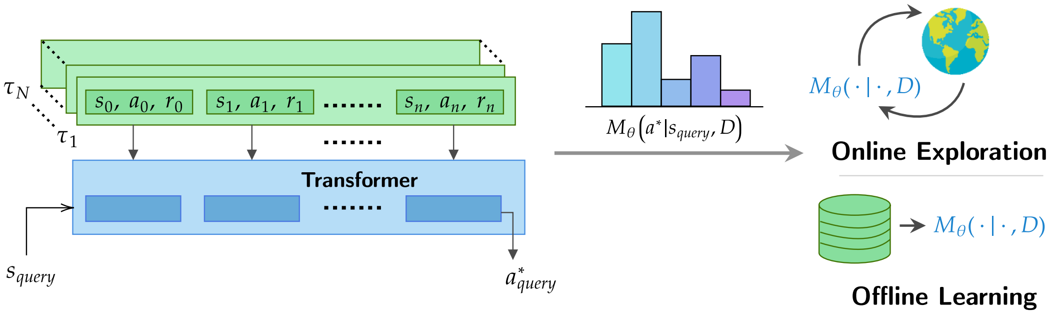


Figure 1: A transformer model M_θ is pretrained to predict an optimal action $a^\star_{\text{query}}$ from a state s_{query} in a task, given a dataset of interactions from that task. The resulting decision-pretrained transformer (DPT) learns a distribution over the optimal action conditioned on an in-context dataset. M_θ can be deployed in *new* tasks online by collecting data on the fly, or offline by immediately conditioning on a static dataset.

interaction, then the dataset is initialized as empty $D = \{\}$. At each episode, $M_\theta(\cdot \mid \cdot, D)$ is deployed where the model samples $a_h \sim M_\theta(\cdot|s_h, D)$ upon observing state s_h. Throughout a full episode, it collects interactions $\{s_1, a_1, r_1, \ldots, s_H, a_H, r_H\}$ which are appended to D. The model then repeats the process with another episode, initializing $D = \{s_1, a_1, r_1, \ldots, s_H, a_H, r_H\}$, and so on until a specified number of episodes has been reached.

A key distinction of the testing phase is that there are no updates to the parameters of the model M_θ. This is in contrast to hand-designed reinforcement learning algorithms that would perform parameter updates or maintain statistics using the dataset D to learn from scratch. Instead, the model M_θ performs a computation through its forward pass to generate a distribution over actions conditioned on the in-context D and query state s_h.

Sources of distribution mismatch. Inherent to this pre-train-then-test procedure, like nearly all foundation models, is distribution mismatch because a practitioner may be interested in solving specific downstream tasks that were not precisely known during pre-training. A model pre-trained on sufficiently diverse data should ideally be robust (to some extent) to these mismatches. Here, we briefly point out potential sources of distribution mismatch at test time that we have identified in the model and will investigate them throughout this work. (1) When deployed, M_θ will execute its learned policy which invariably induces a distribution over states different from $\mathcal{D}_{\text{query}}$. (2) Pretraining likely happens over a hugely diverse set of tasks $\mathcal{T}_{\text{pre}}$, but but a practitioner may be interested in applying M_θ to a specific subset. (3) For a similar reason, the datasets prompted at test-time can also be different, especially in the online case where they are collected by M_θ itself.

4 Learning in Bandits

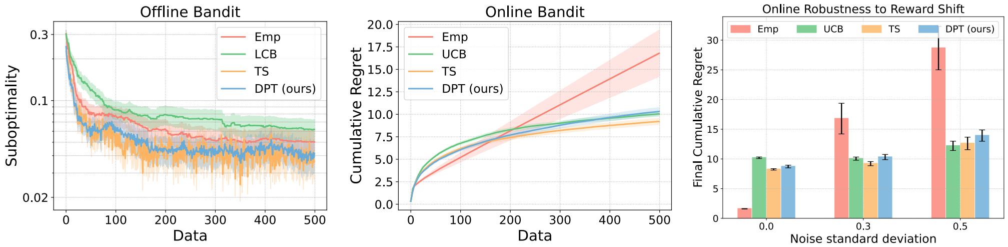
We begin with an empirical investigation of DPT in a multi-armed bandit, a well-studied special case of the MDP where the state space $\mathcal{S}$ is a singleton and the horizon $H = 1$ is a single step. We will examine the performance of DPT both when aiming to select a good action from offline historical data and for online learning where the goal is to maximize cumulative reward from scratch. Offline, it is critical to account for uncertainty to noise as certain actions may not be sampled well enough. Online, it is critical to judiciously balance exploration and exploitation to minimize overall regret.

Pretraining distribution. For the pretraining task distribution $\mathcal{T}_{\text{pre}}$, we sample 5-armed bandits ($|\mathcal{A}| = 5$). The reward function for arm a is a normal distribution $R(\cdot|s, a) = \mathcal{N}(\mu_a, \sigma^2)$ where $\mu_a \sim \text{Unif}[0, 1]$ independently and $\sigma = 0.3$. For the distribution over actions for the in-context datasets $\mathcal{D}_{\text{pre}}$, we randomly generate action frequencies by sampling probabilities from a Dirichlet distribution and mixing them with a point-mass distribution on one random arm. Then we sample the actions correspondingly. This encourages diversity of the in-context datasets. The optimal policy $\pi^\star$ for a particular bandit is simply the action with the highest mean: $\text{argmax}_a\, \mu_a$. We pretrain the model M_θ to predict $a^\star$ from D as described in Section 3 for datasets up to size $n = 500$.

Comparisons. We compare to several well-known algorithms for bandits[2]. All of the algorithms are designed to reason in a particular way about uncertainty based on their observations.

---

[2] See Appendix A.2 for additional details such as hyperparameters.



(a) (b) (c)

Figure 2: (a) Offline performance on in-distribution bandits, given random in-context datasets. (b) Online cumulative regret on bandits. (c) Final (after 500 steps) cumulative regret on out-of-distribution bandits with different Gaussian noise standard deviations. The mean and standard error are computed over 200 test tasks.

- Empirical mean algorithm (Emp) selects the action with the highest empirical mean reward naively.
- Upper Confidence Bound (UCB) selects the action with the highest upper confidence bound.
- Lower Confidence Bound (LCB) selects the action with the highest lower confidence bound.
- Thompson Sampling (TS) selects the action with the highest sampled mean from a posterior distribution over reward models.

Emp and TS [Russo et al., 2018, Thompson, 1933] can both be used for offline or online learning; UCB [Auer et al., 2002] is known to be provably optimal online by ensuring exploration through optimism under uncertainty; and LCB [Xiao et al., 2021, Jin et al., 2021] is used to minimize suboptimality given an offline dataset by selecting actions pessimistically. We evaluate algorithms with standard bandit metrics. Offline, we use the suboptimality $\mu_{a^\star} - \mu_{\hat{a}}$ where $\hat{a}$ is the chosen action. Online, we use cumulative regret: $\sum_k \mu_{a^\star} - \mu_{\hat{a}_k}$ where $\hat{a}_k$ is the kth action chosen.

DPT learns to reason through uncertainty. As shown in Figure 2a, in the offline setting, DPT significantly exceeds the performance of Emp and LCB while matching the performance of TS, when the in-context datasets are sampled from the same distribution as during pretraining. The results suggest that the transformer is capable of reasoning through uncertainty caused by the noisy rewards in the dataset. Unlike Emp which can be fooled by noisy, undersampled actions, the transformer has learned to *hedge* to a degree. However, it also suggests that this hedging is fundamentally different from what LCB does, at least on this specific distribution[3].

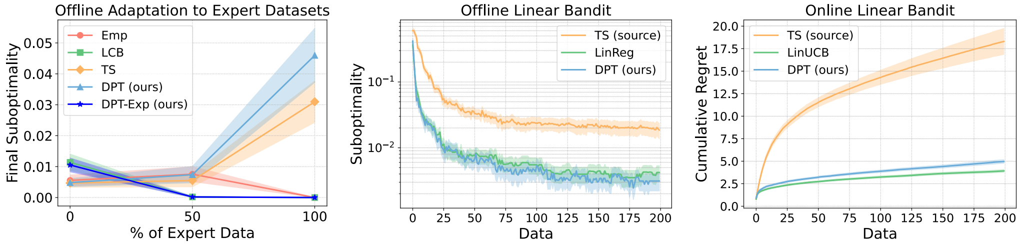
Interestingly, the same transformer produces an extremely effective online bandit algorithm when sampling actions instead of taking an argmax. As shown in Figure 2b, our method matches the performance of classical optimal algorithms, UCB and TS, which are specifically designed for exploration. This is notable because DPT was not explicitly trained to explore, but its emergent strategy is on par with some of the best. In Figure 2c, we show this property is robust to noise in the rewards not seen during pretraining by varying the standard deviation. In Appendix B, we show this generalization happens offline too and even with unseen Bernoulli rewards.

Adapting to expert-biased datasets. A common assumption in offline RL is that datasets tend to be a mixture between optimal data (e.g. expert demonstrations) and suboptimal data (e.g. random interactions) [Rashidinejad et al., 2021]. Hence, LCB is generally effective in practice. Motivated by this, we pretrain a second model where the in-context dataset is mixed with varying fractions of expert data. We denote this model by DPT-Exp. In Figure 3a, we plot the performance of both pretrained models when evaluated on new datasets with different fractions of expert data. Our results suggest that when the pretraining distribution is similarly trained with expert-suboptimal data, DPT-Exp behaves similarly to LCB, while DPT continues to resemble TS. This is quite interesting as for other methods, such as TS, it is less clear how to automatically incorporate the right amount of expert bias to yield the same effect.

Leveraging structure from suboptimal data. We next demonstrate that the model can learn to leverage the inherent structure of a problem class, even without prior knowledge of this structure, and even when learning from in-context datasets that do not explicitly utilize it. More precisely, we consider tasks sampled from linear bandit tasks τ, where the reward function is given by $\mathbb{E}\left[r \mid a, \tau\right] = \langle \theta_\tau, \phi(a) \rangle$ and $\theta_\tau \in \mathbb{R}^d$ is a task-specific parameter vector and $\phi : \mathcal{A} \to \mathbb{R}^d$ is fixed feature vector

---

[3] Note our randomly generated environments are equally likely to have expert-biased datasets and adversarial datasets, so LCB is not expected to outperform here [Xiao et al., 2021].



(a) (b) (c)

Figure 3: (a) Offline performance of DPT and DPT-Exp on expert-biased datasets. (b) Offline performance of DPT trained on linear bandits from TS source data. LinReg does linear regression and outputs the greedy action. (c) Online cumulative regret of the same model. The mean and standard error are computed over 200 test tasks.

that is the same for all tasks τ. Given the feature representation ϕ, LinUCB [Abbasi-Yadkori et al., 2011], a UCB-style algorithm that leverages ϕ, achieves regret $\widetilde{\mathcal{O}}(\sqrt{dK})$ over K steps, a substantial gain when $d \ll |\mathcal{A}|$. Here, we pretrain a DPT model with datasets gathered by Thompson Sampling (TS), which treats the arms as if they are independent and is suboptimal for this problem.

Figures 3b and 3c show that the model can learn to take advantage of the unknown linear structure, essentially learning an effective surrogate for ϕ, and can use this structure to do more informed exploration online and decision-making offline. It is nearly on par with LinUCB (which is given ϕ) and significantly outperforms the dataset source, TS, which does not know the structure. In addition, this presents evidence that supervised learning-based approaches to decision-making *can* learn novel exploration strategies that transcend the quality of their original data.

5 Learning in Markov Decision Processes

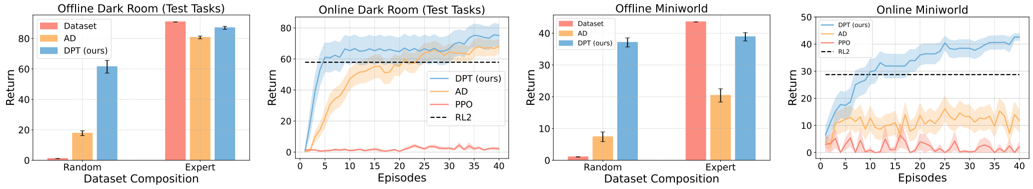
We next study whether DPT can tackle Markov decision processes by testing its ability to perform exploration and credit assignment. In the following experiments, the in-context algorithm demonstrates generalization to new tasks, scalability to image-based observations, and capability to stitch in-context behaviors (Section 5.2). This section also examines whether DPT can be pretrained with datasets and action labels generated by a different RL algorithm (Section 5.3).

5.1 Experimental Setup

Environments. We consider environments that require targeted exploration to solve the task. The first is Dark Room [Zintgraf et al., 2019], a 2D discrete environment where the agent must locate the unknown goal location in a 10×10 room, and only receives a reward of 1 when at the goal. We hold out a set of goals for generalization evaluation. Our second environment is Miniworld [Chevalier-Boisvert, 2018], a 3D visual navigation problem to test the scalability of our method to image observations. The agent is in a room with four boxes of different colors, and must find the target box, the color of which is unknown to the agent initially. It receives a reward of 1 only when near the correct box. Details on these environments and the pre-training datasets are in App. A.3 and A.4.

Comparisons. Our experiments aim to understand the effectiveness of our pretraining procedure in comparison to that of other context-based meta-RL algorithms. To that end, we compare to meta-RL algorithms based on supervised and on RL objectives.

- Proximal Policy Optimization (PPO) [Schulman et al., 2017]: We compare to this single-task RL algorithm, which trains from scratch without any pretraining data, to contextualize the performance of our model and other meta-RL algorithms.
- Algorithm Distillation (AD) [Laskin et al., 2022]: AD first generates a dataset of learning histories by running an RL algorithm in each training task. Then, given a sampled subsequence $h_j = (o_j, a_j, r_j, \ldots, o_{j+c})$ from a learning history, a sequence model is trained to predict the next action a_{j+c} from the learning history.
- RL^2 [Duan et al., 2016]: This online meta-RL comparison uses a recurrent neural network to adapt the agent's policy from the given context. Unlike AD and our method, which is trained with a supervised objective, the RL^2 agent is trained to maximize the expected return with PPO.



(a) (b) (c) (d)

Figure 4: (a) Offline performance on held-out Dark Room goals, given random and expert datasets. (b) Online performance on held-out Dark Room goals. (c) Offline performance on Miniworld, given random and expert datasets. (d) Online performance on Miniworld after 40 episodes. We report the average and standard error of the mean over 100 different offline datasets in (a) and (c) and 20 online trials in (b) and (d).

PPO and RL^2 are online algorithms, while AD is capable of learning both offline and online. Details on the implementation of these algorithms can be found in Appendix A.2.

5.2 Main Results

Generalizing to new offline datasets and tasks. To study the generalization capabilities of DPT, we evaluate the model in Dark Room on a set of 20 held-out goals not in the pretraining dataset. When given an expert dataset, DPT achieves near-optimal performance. Even when given a random dataset, which has an average total reward of 1.1, DPT obtains a much higher average return of 61.5 (see Fig. 4a). Qualitatively, we observe that when the in-context dataset contains a transition to the goal, DPT immediately exploits this and takes a direct path to the goal. In contrast, while AD demonstrates strong offline performance with expert data, it performs worse in-context learning with random data compared to DPT. The difference arises because AD is trained to infer a better policy than the in-context data, but not necessarily the optimal one.

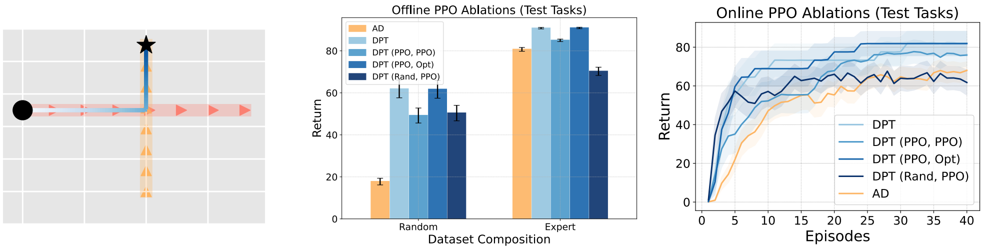
We next evaluate DPT, AD, RL^2, and PPO online without any prior data from the 20 test-time Dark Room tasks, shown in Fig. 4b. After 40 episodes, PPO does not make significant progress towards the goal, highlighting the difficulty of learning from such few interactions alone. RL^2 is trained to perform adaptation within four episodes each of length 100, and we report the performance after the four adaptation episodes. Notably, DPT on average solves each task faster than AD and reaches a higher final return than RL^2, demonstrating its strong ability to explore effectively online. In Appendix B, we also present results on generalization to new dynamics.

Learning from image-based observations. In Miniworld, the agent receives RBG image observations of 25×25 pixels. As shown in Fig. 4d, DPT can solve this high-dimensional task offline from both random and expert datasets. Compared to AD and RL^2, DPT also learns online more efficiently.

Stitching novel trajectories from in-context subsequences. A desirable property of some offline RL algorithms is the ability to stitch suboptimal subsequences from the offline dataset into new trajectories with higher return. To test whether DPT exhibits stitching, we design the *Dark Room (Three Tasks)* environment in which there are three possible tasks. The pretraining data consists only of expert demonstrations of two of them. At test-time DPT is evaluated on third unseen task, but its offline dataset is only expert demonstrations of the original two. Despite this, it leverages the data to infer a path solving the third task (see Fig. 5a).

5.3 Learning from Algorithm-Generated Policies and Rollouts

So far, we have only considered in-context interactions collected by a random policy and action labels provided by an optimal policy. However, most realistic tasks do not have an optimal policy readily available. In the pretraining phase of this experiment, we instead use action labels given by the best policy a PPO agent learns. We also experiment with rollouts from the PPO agent's replay buffer as the pretraining in-context datasets, which AD trains on and thus allows us to compare more directly to AD. We evaluate three variants: DPT (PPO, Opt) is with PPO contexts and optimal policy labels, DPT (Rand, PPO) is with random contexts and PPO policy labels, and DPT (PPO, PPO) is with PPO contexts and PPO policy labels. The final variant DPT (PPO, PPO) can be viewed as a direct comparison between our pretraining objective and that of AD, given the same pretraining data. In Figs. 5b and 5c, we see that DPT (Rand, PPO) online performs on par with and offline worse than



(a) (b) (c)

Figure 5: (a) In *Dark Room (Three Tasks)*, DPT stitches a new, optimal trajectory to the goal (blue) given two in-context demonstrations of other tasks (pink and orange). (b) Offline Dark Room performance of DPT trained on PPO data. (c) Online Dark Room performance of DPT trained on PPO data.

AD, but all the other variants outperform AD, including DPT (PPO, PPO). As we might expect, the variants of DPT that use the PPO policy to generate action labels perform worse compared to their counterparts with the optimal action labels. In Appendix B, we analyze the sensitivity of DPT to other hyperparameters, such as the context size and amount of pretraining data.

6 Theory

Under some assumptions on data and model parameterization, we prove we can formally relate the pretraining procedure to posterior sampling [Osband et al., 2013], a theoretical generalization of TS that samples tasks from a posterior distribution and computes optimal policies given data. This allows us to obtain regret bounds for in-context learning in DPT under the following pretraining conditions.

History-dependent pretraining. For our theoretical analysis, we assume pretraining also conditions on a queried optimal history. Given on a task τ, let ξ_h for $h \in [H]$ denote a partial history of state-actions $\xi_h^\tau = (s_1, a_1, \ldots, s_{h-1}, a_{h-1})$ generated by following the optimal policy $\pi_\tau^\star(\cdot|\cdot)$ in τ. Accounting for ξ, we denote the joint pretraining distribution as $P_{pre}(\tau, h, \xi_h, D, s_{\text{query}}, a^\star)$, defined additionally by sampling $h \sim \text{Unif}[H-1]$ independently and then rolling out $\pi_\tau^\star$ for h steps in τ to record ξ_h. The model M_θ can then be trained identically to the description in Section 3, but also conditioning on h and ξ_h to product the conditional distribution $M_\theta(a^\star|s_{\text{query}}, D, \xi_h)$. Intuitively, the difference here is that we are not only utilizing the optimal policy $\pi_\tau^\star$ to label s_{query}, but we are also using it to generate a roll-in trajectory ξ_h. The test-time procedure remains the same as before when deploying M_θ except that when sampling a_h in s_h, we also condition on the partial history ξ_{h-1} and update it with each time step. Note for bandits and contextual bandits (i.e. $H = 1$), there is no difference between the pretraining procedures in prior sections, and the history-dependent pretraining, since ξ is empty. For MDPs, the original pretraining method can be seen as a simpler approximation of this modified method that still exhibits many of the desired characteristics in practice. In our proofs we will be clear where this modification is important.

Assumption 1. *(Learned model is consistent). Let M_θ denote the pretrained model. For all (s_{query}, D, ξ_h), we have $P_{pre}(a|s_{query}, D, \xi_h) = M_\theta(a|s_{query}, D, \xi_h)$ for all $a \in \mathcal{A}$.*

To provide some cursory justification for this assumption, we assume we have a very large amount of pretraining data, and note that optimizing the log-likelihood objective guarantees $\mathbb{E}_{P_{pre}} \|P_{pre}(\cdot|s_{\text{query}}, D, \xi_h) - M_\theta(\cdot|s_{\text{query}}, D, \xi_h)\|_1^2 \xrightarrow{N} 0$, with high probability for transformer model classes of bounded complexity. Given the above conditions, trajectories generated online by sampling from the pretrained model M_θ follow the same distribution as those generated by a policy from well-specified posterior sampling (see Appendix C for a definition):

Theorem 1. *Let the current task τ_c and dataset D be fixed. Let $P_{ps}(\xi)$ and $P_{M_\theta}(\xi)$ denote the distributions over trajectories under posterior sampling and M_θ in τ_c, respectively, conditioned on D and with a shared prior distribution $\mathcal{T}_{pre}$. Then, $P_{ps}(\xi) = P_{M_\theta}(\xi)$ for all ξ.*

The equivalence immediately implies a cumulative online regret guarantee for M_θ given prior results on posterior sampling[Osband et al., 2013]. Specifically, let $\mathcal{T}_{\text{pre}} = \mathcal{T}_{\text{test}}$ be distributions over finite MDPs with shared $S := |\mathcal{S}|$ and $A := |\mathcal{A}|$ and horizon H. Furthermore, we let s_{query} be uniform over $\mathcal{S}$ during pretraining and D of length $i \in [KH]$ be constructed by sampling s_i and a_i uniformly from

$\mathcal{S}$ and $\mathcal{A}$ and observing the resulting r_i and s_i' from τ. For a task τ define the cumulative regret over K episodes as $\mathrm{Reg}_\tau(M_\theta) := \sum_{k\in[K]} V_\tau(\pi_\tau^\star) - V_\tau(\hat{\pi}_k)$ where $\hat{\pi}_k(\cdot|s_h) = M_\theta(\cdot|s_h, D_{H(k-1)}, \xi_{h-1})$.

Corollary 6.1 (Regret guarantee of learned algorithm)**.** *Suppose that* $\sup_\tau \frac{\mathcal{T}_{\mathit{test}}(\tau)}{\mathcal{T}_{\mathit{pre}}(\tau)} \le \mathcal{C}$ *for some* $\mathcal{C} > 0$. *For the above MDP setting, the pretrained model* M_θ *satisfies* $\mathbb{E}_{\mathcal{T}_{\mathit{test}}}\left[\mathit{Reg}_\tau(M_\theta)\right] \le \widetilde{\mathcal{O}}\left(\mathcal{C}HS\sqrt{AK}\right)$.

A similar analysis allows us to prove why training over (latently linear) bandit tasks using our pretraining procedure, even when using data generated by algorithms unaware of this structure, can lead to substantial empirical gains. We observed this empirically in Section 4.

Corollary 6.2 (Latent representation learning in linear bandits)**.** *Consider* d*-dimensional linear bandits with a fixed feature map* $\phi : \mathcal{A} \to \mathbb{R}^d$*, and datasets* D *gathered using Gaussian Thompson sampling for* K *steps per task (using one-hot encodings for each unique action). Let* $\sup_{a\in\mathcal{A}} \|\phi(a)\|_2 \le 1$. *Then the pretrained model* M_θ *satisfies* $\mathbb{E}_{\mathcal{T}_{\mathit{test}}}\left[\mathit{Reg}(M_\theta)\right] \le \widetilde{\mathcal{O}}(\sqrt{dK})$.

This significantly improves over the $\mathcal{O}(\sqrt{|\mathcal{A}|K})$ regret bound for Thompson Sampling that does not leverage the linear structure. Note that such a TS algorithm is also used to gather the datasets for training, highlighting how our approach can provably learn better cumulative regret algorithms than those observed in the provided data. Indeed, the impact of the algorithms used, and datasets generated, for pretraining is an important issue. We now present a sufficient condition, under which the learned model will be identical for different sets of algorithms and datasets used for pretraining:

Proposition 6.3. *Let* P_{pre}^1 *and* P_{pre}^2 *be pretraining distributions that differ only in* $\mathcal{D}_{pre}^1$ *and* $\mathcal{D}_{pre}^2$*: both are over the same underlying task distribution, and have the same dataset size* n*. If* $\mathcal{D}_{pre}^1(a_j|s_j, D_j; \tau)$ *and* $\mathcal{D}_{pre}^2(a_j|s_j, D_j; \tau)$ *are invariant to* τ *for all* $j \in [n]$ *(e.g.* $\mathcal{D}_{pre}^1(a_j|s_j, D_j; \tau) = \mathcal{D}_{pre}^1(a_j|s_j, D_j)$*), then* $P_{pre}^1(a^\star|s_{query}, D, \xi_h) = P_{pre}^2(a^\star|s_{query}, D, \xi_h)$.

This condition is similiar to the sequential ignorability condition often used in offline RL analysis, which assumes the selected actions depend only on the observed history and not additional confounders or potential outcomes. Note this condition implies that if we generate in-context datasets D by running various algorithms that depend only on the observed data in the current task, we will end up with the same posterior distribution for downstream tasks: for example, Thompson Sampling could be used for $\mathcal{D}_{\text{pre}}^1$ and Proximal Policy Optimization for $\mathcal{D}_{\text{pre}}^2$. Expert-generated trajectories violate the precondition of Proposition 6.3, since knowledge of the task, beyond that available by the observed data so far, is being used. This helps to explain our empirical results that pretraining on expert-biased datasets leads to a qualitatively different learned model at test-time.

7 Discussion

In this paper, we studied the problem of in-context decision-making and introduced a new pretraining method and transformer model, DPT, which is trained to predict optimal actions given an in-context dataset of interactions. Through an in-depth evaluations in classic decision problems in bandits and MDPs, we showed that this simple objective naturally gives rise to an in-context algorithm that is capable of online exploration and offline decision-making, unlike other algorithms that are explicitly trained or designed to do these. Our empirical and theoretical results provide some first steps towards understanding what factors are key for this to succeed such as: (1) classes of problems and distribution shifts where we can guarantee good performance (2) how the in-context algorithm can automatically leverage the structure in $\mathcal{T}_{\text{pre}}$; (3) how the dataset distribution $\mathcal{D}_{\text{pre}}$ can affect the in-context algorithm.

Limitations and future work. One limitation of DPT is the requirement of optimal actions at pretraining. Empirically, we find that this requirement can be relaxed by using actions generated by another RL-trained agent, which only leads to a slight loss in performance. However, fully understanding this problem and how best to leverage multi-task decision-making datasets remains a key open problem. We also discussed that the practical implementation for MDPs differs from true posterior sampling. It would be interesting to further understand and bridge this empirical-theoretical gap in the future. Finally, we remark that our preliminary analysis suggests promise for DPT to generalize to new tasks beyond its pretraining distribution. This suggests that diversifying the task distributions during pretraining could significantly enhance the model's ability to generalize to new tasks. This possibility holds an exciting avenue for future work.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems*, 33:20095–20107, 2020.

Shipra Agrawal and Navin Goyal. Near-optimal regret bounds for thompson sampling. *Journal of the ACM (JACM)*, 64(5):1–24, 2017.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv:2206.01079*, 2022.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Maxime Chevalier-Boisvert. Miniworld: Minimalistic 3d environment for rl & robotics research. `https://github.com/maximecb/gym-miniworld`, 2018.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34:4607–4618, 2021.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Yiding Jiang, Evan Liu, Benjamin Eysenbach, J Zico Kolter, and Chelsea Finn. Learning options via compression. *Advances in Neural Information Processing Systems*, 35:21184–21199, 2022.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Nicholas C Landolfi, Garrett Thomas, and Tengyu Ma. A model-based approach for sample-efficient multi-task reinforcement learning. *arXiv preprint arXiv:1907.04964*, 2019.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

Lanqing Li, Rui Yang, and Dijun Luo. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*, 2020.

Yingcong Li, M Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and implicit model selection in in-context learning. *arXiv preprint arXiv:2301.07067*, 2023.

Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International conference on machine learning*, pages 6925–6935. PMLR, 2021.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *UAI*, 2019.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch off-policy reinforcement learning without great exploration. *Advances in neural information processing systems*, 33:1264–1274, 2020.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *arXiv preprint arXiv:2303.03982*, 2023.

Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. *Advances in neural information processing systems*, 30, 2017.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pages 7780–7791. PMLR, 2021.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Theodore J Perkins, Doina Precup, et al. Using options for knowledge transfer in reinforcement learning. Technical report, Citeseer, 1999.

Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pages 17811–17829. PMLR, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716, 2021.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*, 2022.

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

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Tom Schaul and Jürgen Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*, 2022.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *arXiv preprint arXiv:2303.07895*, 2023.

Chenjun Xiao, Yifan Wu, Jincheng Mei, Bo Dai, Tor Lattimore, Lihong Li, Csaba Szepesvari, and Dale Schuurmans. On the optimality of batch policy optimization algorithms. In *International Conference on Machine Learning*, pages 11362–11371. PMLR, 2021.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Mengjiao Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. Dichotomy of control: Separating what you can control from what you cannot. *arXiv preprint arXiv:2210.13435*, 2022.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Tong Zhang. From e-entropy to kl-entropy: Analysis of minimum information complexity density estimation. 2006.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representation (ICLR)*, 2020.

Additional Related Work

In-context learning. Beyond decision-making and reinforcement learning, our approach takes inspiration from general in-context learning, a phenomenon observed most prominently in large language models in which large-scale autoregressive modelling can surprisingly lead to a model that exhibits meta-learning capabilities [Brown et al., 2020]. Recently, there has been great interest in understanding the capabilities and properties of in-context learning [Garg et al., 2022, von Oswald et al., 2022, Razeghi et al., 2022, Olsson et al., 2022, Kirsch et al., 2022, Shin et al., 2022, Li et al., 2023, Wies et al., 2023, Akyürek et al., 2022]. While a common hypothesis suggests that this phenomenon is due to properties of the data used to train large language models [Chan et al., 2022], our work suggests that this phenomenon can also be encouraged in general settings via adjustments to the pre-training objective. In fact, our method could be interpreted as explicitly encouraging the ability to perform Bayesian inference, which is a popular explanation for the mechanism behind in-context learning for large language models [Xie et al., 2021].

Posterior Sampling. Posterior sampling originates from the seminal work of Thompson [1933], and has been popularized and thoroughly investigated in recent years by a number of authors Russo et al. [2018], Agrawal and Goyal [2017], Strens [2000], Osband et al. [2013], Russo and Van Roy [2014]. For bandits, it is often referred to as Thompson Sampling, but the framework is easily generalizable to RL. The principle is as follows: begin with a prior over possible models (i.e. reward and transition functions), and maintain a posterior distribution over models by updating as new interactions are made. At decision-time, sample a model from the posterior and execute its optimal policy. The aforementioned prior works have developed strong theoretical guarantees on Bayesian and frequentist regret for posterior sampling. Despite its desirable theoretical characteristics, a major limitation is that computing the posterior is often computationally intractable, leading practitioners to rely on approximation-based solutions [Lu and Van Roy, 2017, Osband et al., 2016, 2018]. In Section 6, we show that a version of the DPT model learned from pretraining can be viewed as implementing posterior sampling as it should be without resorting to approximations or deriving complicated posterior updates. Instead, the posterior update is implicitly learned through pretraining. This suggests that in-context learning (or meta-learning more generally) could be a key in unlocking practically applicable posterior sampling for RL.

A Implementation and Experiment Details

Algorithm 1 Pretraining

```
// Collecting pretraining dataset
Initialize empty dataset $\mathcal{B}$
for $i$ in $[N]$ do
    Sample task $\tau \sim \mathcal{T}_{\text{pre}}$
    Sample interaction dataset $D \sim \mathcal{D}_{\text{pre}}(\cdot; \tau)$ of length $n$
    Sample $s_{\text{query}} \sim \mathcal{D}_{\text{query}}$ and $a^\star \sim \pi^\star_\tau(\cdot|s_{\text{query}})$
    Add $(s_{\text{query}}, D, a^\star)$ to $\mathcal{B}$
end for
// Training model on dataset
Initialize model $M_\theta$ with parameters $\theta$
while not converged do
    Sample $(s_{\text{query}}, D, a^\star)$ from $\mathcal{B}$
    Predict $\hat{p}_j(\cdot) = M_\theta(\cdot|s_{\text{query}}, D_j)$ for all $j \in [n]$.
    Compute cross entropy loss in (5) with respect to $a^\star$ and backpropagate to update $\theta$.
end while
```

A.1 DPT Architecture: Formal Description

In this section, we provide a detailed description of the architecture alluded to in Section 3 and Figure 1. See hyperparameter details for models in their respective sections. The model is implemented in Python with PyTorch [Paszke et al., 2019]. The backbone of the transformer architecture we use is an autoregressive GPT-2 model from the HuggingFace `transformers` library.

Algorithm 2 Offline test-time deployment

```
// Task and offline dataset are generated without learner's control
Sample unknown task $\tau \sim \mathcal{T}_{\text{test}}$
Sample dataset $D \sim \mathcal{D}_{\text{test}}(\cdot; \tau)$
// Deploying offline policy $M_\theta(\cdot|\cdot, D)$
$s_1 = \texttt{reset}(\tau)$
for $h$ in $[H]$ do
    $a_h = \text{argmax}_{a \in \mathcal{A}} M_\theta(\cdot|s_h, D)$ // Most likely action
    $s_{h+1}, r_h = \texttt{step}(\tau, a_h)$
end for
```

Algorithm 3 Online test-time deployment

```
// Online, dataset is empty as learning is from scratch
Initialize $D = \{\}$
Sample unknown task $\tau \sim \mathcal{T}_{\text{test}}$
for ep in max_eps do
    $s_1 = \texttt{reset}(\tau)$
    for $h$ in $[H]$ do
        $a_h \sim M_\theta(\cdot|s_h, D)$ // Sample action from predicted distribution
        $s_{h+1}, r_h = \texttt{step}(\tau, a_h)$
    end for
    // Experience from previous episode added to dataset
    Add $(s_1, a_1, r_1, \ldots)$ to $D$
end for
```

For the sake of exposition, we suppose that $\mathcal{S}$ and $\mathcal{A}$ are subsets of $\mathbb{R}^{d_S}$ and $\mathbb{R}^{d_A}$ respectively. We handle discrete state and action spaces with one-hot encoding. Consider a single training datapoint derived from an (potentially unknown) task τ: we have a dataset D of interactions within τ, a query state s_{query}, and its corresponding optimal action $a^\star = \pi^\star_\tau(s_{\text{query}})$. We construct the embeddings to be passed to the backbone in the following way. From the dataset $D = \{(s_j, a_j, s'_j, r_j)\}_{j \in [n]}$, we construct vectors $\xi_j = (s_j, a_j, s'_j, r_j)$ by stacking the elements of the transition tuple into dimension $d_\xi := 2d_S + d_A + 1$ for each j in the sequence. This sequence of n elements is concatenated with another vector $v := (s_{\text{query}}, \mathbf{0})$ where the $\mathbf{0}$ vector is a vector of zeros of sufficient length to make the entire element dimension d_ξ. The $(n+1)$-length sequence is given by $X = (v, \xi_1, \ldots, \xi_n)$. As order does not matter for the dataset D[4], we do not use positional encoding in order to take advantage of this invariance. We first apply a linear layer $\texttt{Linear}(X)$ and pass the result to the transformer, which outputs the sequence $Y = (\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_n)$. In the continuous action case, these can be used as is for predictions of $a^\star$. For the discrete action case, we use them as logits to be converted to either a distribution over actions in $\mathcal{A}$ or one-hot vector predictions of $a^\star$. Here, we compute action probabilities

$$\hat{p}_j = \texttt{softmax}(\hat{y}_j) \in \Delta(\mathcal{A}) \tag{3}$$

Because of the GPT-2 causal architecture (we defer details to the original papers [Radford et al., 2019, Brown et al., 2020]), we note that $\hat{p}_j$ depends only on s_{query} and the partial dataset $D_j = \{(s_k, a_k, s'_k, r_k)\}_{k \in [j]}$, which is why we write the model notation,

$$M_\theta(\cdot|s_{\text{query}}, D_j) = \hat{p}_j(\cdot), \tag{4}$$

to denote that the predicted probabilities of the jth element only depend on D_j and not the entire D for the model M with parameters $\theta \in \Theta$. For example, with $j = 0$, the prediction of $a^\star$ is made without any contextual information about the task τ except for s_{query}, which can be interpreted as the prior over $a^\star$. We measure loss of this training example via the cross entropy for each $j \in [n]$:

$$-\sum_{j \in [n]} \log \hat{p}_j(a^\star) \tag{5}$$

---

[4] There are caveats to this when the data comes from an algorithm such as PPO or Thompson Sampling.

Intuition. Elements of the inputs sequence X represent transitions in the environment. When passed through the GPT-2 transformer, the model learns to associate elements of the sequence via the standard query-key-value mechanism of the attention model. The query state s_{query} is demarcated by its zeros vector (which also acts as padding). Unlike other examples of transformers used for decision-making such as the Decision Transformer [Chen et al., 2021] and Algorithm Distillation [Laskin et al., 2022], DPT does not separate the individual (s, a, s', r) into their own embeddings to be made into one long sequence. This is because we view the transition tuples in the dataset as their own singletons, to be related with other singletons in the dataset through the attention mechanism. We note that there are various other implementation variations one could take, but we found success and robustness with this one.

A.2 Implementation Details

A.2.1 Bandit algorithms

First, we describe the comparisons from the bandit experiments with hyperparameters.

Empirical Mean (Emp). Emp has no hyperparameters, but we give it some mechanism to avoid degenerate scenarios. In the offline setting, Emp will only choose from actions that have at least one example in the dataset. This gives Emp and LCB-style effect when actions are missing. Similarly, online, Emp will sample each action at least once before defaulting to its real strategy. These changes only improve Emp.

Upper Confidence Bound (UCB). According to the Hoeffding bound, we choose actions as $\hat{a} \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ \hat{\mu}_a + \sqrt{1/n_a} \right\}$ where $\hat{\mu}_a$ is the empirical mean so far for action a and n_a is the number of times a has been chosen so far. To arrive at this constant for the bonus, we coarsely tried a set of plausible values given the noise and found this to perform the best.

Lower Confidence Bound (LCB). We choose actions as $\hat{a} \in \operatorname{argmax}_{a \in \mathcal{A}} \left\{ \hat{\mu}_a - \sqrt{1/n_a} \right\}$ where $\hat{\mu}_a$ is the empirical mean so far for action a and n_a is the number of times a has been chosen so far.

Thompson Sampling (TS). Since the means are sampled uniformly from $[0, 1]$, Gaussian TS is partially misspecified; however, we set prior mean and variance to $\frac{1}{2}$ and $\frac{1}{12}$ to match the true ones. The noise model was well-specified with the correct variance. In the linear experiments of Figure 3b and Figure 3c, we set the prior mean and variance to 0 and 1 to fit the true ones better.

LinUCB. We choose $\hat{a}_t \in \operatorname{argmax}_{a \in \mathcal{A}} \langle \hat{\theta}_t, \phi(a) \rangle + \beta \|\phi(a)\|_{\hat{\Sigma}_t^{-1}}$ where $\beta = 1$ and $\hat{\Sigma}_t = I + \sum_{s \in [t-1]} \phi(a_s)\phi(a_s)^\top$ and $\hat{\theta}_t = \hat{\Sigma}_t^{-1} \sum_{s \in [t-1]} r_s \phi(a_s)$. Here, r_s and a_s are the reward and action observed at time s.

LinReg. LinReg (offline) is the same as LinUCB except we set $\beta = 0$ to greedily choose actions.

DPT. The transformer for DPT has an embedding size of 32, context length of 500 for basic bandits and 200 for linear bandits, 4 hidden layers, and 4 attention heads per attention layer for all bandits. We use the AdamW optimizer with weight decay 1e-4, learning rate 1e-4, and batch-size 64. For all experiments, we shuffle the in-context dataset D since order does not matter except in the linear bandit.

A.2.2 RL Algorithms

Below, we describe the comparisons from the MDP experiments and their hyperparameters.

Proximal Policy Optimization (PPO). The reported results for PPO use the Stable Baselines3 implementation [Raffin et al., 2021] with the default hyperparameters, which successfully learns each task given 100K environment steps in Dark Room and 125K environment steps in Miniworld. In Dark Room, the policy is implemented as a multi-layer perceptron with two hidden layers of 64 units each. In Miniworld, the policy is a convolutional neural network with two convolutional layers with 16 3×3 kernels each, followed by a linear layer with output dimension of 8.

Algorithm Distillation (AD). We first collect learning histories with PPO for each of the training tasks. Then, given a cross-episodic context of length H, where H is the task horizon, the model is trained to predict the actions taken K episodes later (given the states visited in that episode). In our experiments, we evaluate AD across different values of K. Between $K = 10, 50$, and 100, we found $K = 100$ to be most performant in the Dark Room environment. In Miniworld, we sampled the latter episode from the final 500 episodes of the task's replay buffer, which performed better than other choices of K we experimented with, including $K = 10, 50$, and 100. In Dark Room, the transformer has similar hyperparameters as DPT: an embedding size of 32, context length of 100 steps, 4 hidden layers, and 4 attention heads per attention layer. In Miniworld, we first encode the image with a convolutional network with two convolutional layers with 16 3×3 kernels each, followed by a linear layer with output dimension of 8.

RL^2. The reported results for RL^2 use an open-sourced implementation from Zintgraf et al.. The implementation uses PPO as the RL algorithm and defines a single trial as four consecutive episodes. The policy is implemented with one hidden layer of 32 units in Dark Room. In Miniworld, the policy is parameterized with a convolutional neural network with two convolutional layers with 16 3×3 kernels each, followed by a linear layer with output dimension of 8.

DPT. The transformer for DPT has an embedding size of 32, context length of 100 steps, 4 hidden layers, and 4 attention heads per attention layer in Dark Room. In Miniworld, the image is first passed through a convolutional network with two convolutional layers 16 3×3 kernels each, followed by a linear layer with output dimension of 8. The transformer model that processes these image embeddings otherwise has the same hyperparameters as in Dark Room. We use the AdamW optimizer with weight decay 1e-4, learning rate 1e-3, and batch-size 128.

A.3 Bandit Pretraining and Testing

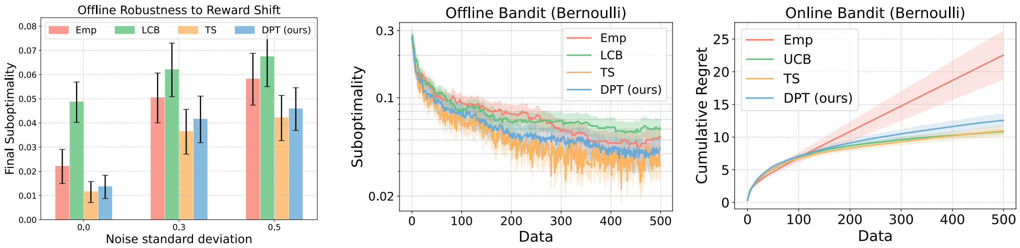
Basic Bandit. Offline, to generate the in-context datasets for pretraining, we used a Dirichlet distribution to sample action frequencies in order to generate datasets with diverse compositions (i.e. some more uniform, some that only choose a few actions, etc.): $p_1 \sim \text{Dir}(\mathbb{1})$ where $p_1 \in \Delta(\mathcal{A})$ and $\mathbb{1} \in \mathbb{R}^{|\mathcal{A}|}$. We also mixed this with a distribution that has all mass on one action: $\hat{a} \sim \text{Unif}(\mathcal{A})$ and $p_2(\hat{a}) = 1$ and $p_2(a) = 0$ for all $a \neq \hat{a}$. The final action distribution is $p = (1 - \omega)p_1 + \omega p_2$ where $\omega \sim \text{Unif}(0.1[10])$. We train on 100,000 pretraining samples for 300 epochs. In Figure 2a, $\mathcal{D}_{\text{test}}$ is generated in the same way.

Expert-Biased Bandit. To generate expert-biased datasets for pretraining, we compute the action frequencies to bias the dataset towards the optimal action. Let $a^\star$ be the optimal one. As before, we take $p_1 \sim \text{Dir}(\mathbb{1})$. Then, $p_2(a^\star) = 1$ and $p_2(a) = 0$ for all $a \neq a^\star$. For of bias of ω, we take $p = (1 - \omega)p_1 + \omega p_2$ with $\omega \sim \text{Unif}(0.1[10])$. We use the same pretraining sample size and epochs as before. For testing, $\mathcal{D}_{\text{test}}$ is generated the same way except we fix a particular $\omega \in \{0, 0.5, 1\}$ to test on.

Linear Bandit. We consider the case where $|\mathcal{A}| = 10$ and $d = 2$. To generate environments from $\mathcal{T}_{\text{pre}}$, we first sampled a fixed set of actions from $\mathcal{N}(\mathbf{0}, I_d/d)$ in $\mathbb{R}^d$ to represent the features. Then, for each τ, we sampled $\theta_\tau \sim \mathcal{N}(\mathbf{0}, I_d/d)$ to produce the means $\mu_a = \langle \theta_\tau, \phi(a) \rangle$ for $a \in \mathcal{A}$. To generate the in-context dataset, we ran Gaussian TS (which does not leverage ϕ) over $n = 200$ steps (see hyperparameters in previous section). Because order matters, we did not shuffle and used $1,000,000$ pretraining samples over 200 epochs. At test time, we set $\mathcal{T}_{\text{test}} = \mathcal{T}_{\text{pre}}$ and $\mathcal{D}_{\text{test}} = \mathcal{D}_{\text{pre}}$. Note that ϕ is fixed over all τ, as is standard for a linear bandit.

A.4 MDP Environment Details

Dark Room. The agent must navigate a 10×10 grid to find the goal within $H = 100$ steps. The agent's observation is its xy-position, the allowed actions are left, right, up, down, and stay, and the reward is only $r = 1$ when the agent is at the goal, and $r = 0$ otherwise. At test time, the agent



(a) (b) (c)

Figure 6: (a) Final (after 500 steps) offline suboptimality on out-of-distribution bandits with different Gaussian noise standard deviations. (b) Offline performance on out-of-distribution Bernoulli bandits, given random in-context datasets. (c) Online cumulative regret on Bernoulli bandits. The mean and standard error are computed over 200 test tasks.

begins at the $(0, 0)$ position. We randomly designate 80 of the 100 grid squares to be goals for the training tasks, and hold out the remaining 20 for evaluation.

Miniworld. The agent must navigate to the correct box, which is initially unknown, from 25×25 RGB image observations. The agent is additionally conditioned on its own position and direction vectors. In each episode, the environment is initialized with four boxes of different colors, one in each corner of the square room. The agent can turn left, turn right, or move forward. The reward is only $r = 1$ when the agent is near the correct box and $r = 0$ otherwise, and each episode is 50 time-steps long. At test time, the agent begins in the middle of the room.

A.5 MDP Pretraining Datasets

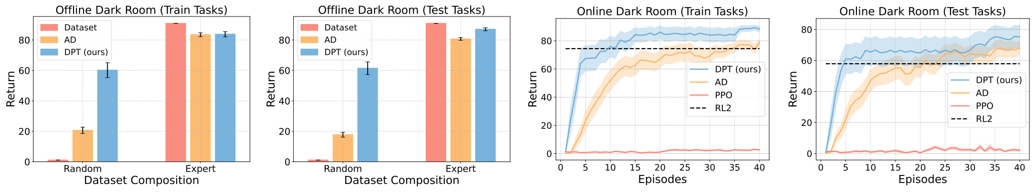
Dark Room. In Dark Room, we collect 100K in-context datasets, each of length $H = 100$ steps, with a uniform-random policy. The 100K datasets are evenly collected across the 100 goals. The query states are uniformly sampled from the state space, and the optimal actions are computed as follows: move up/down until the agent is on the same y-position as the goal, then move left/right until the agent is on the x-position as the goal. Of the 100K collections of datasets, query states, and optimal actions, we use the first 80K (corresponding to the first 80 goals) for training and the remaining 20K for validation.

Miniworld. While this task is solved from image-based observations, we also note that there are only four distinct tasks (one for each colored box), and the agent does not need to handle new tasks at test time. Hence, the number of in-context datasets required in pretraining is fewer – we use 20K datasets each of length $H = 50$ steps. Of this total, 18K are collected with a uniform-random policy, while the remaining 2K are collected with a hand-designed optimal policy. These 2K demonstrations are notably provided to both DPT and AD, which are capable of learning from these demonstrations at pretraining. The in-context datasets only contain the position and direction vectors, and not the images, as the observation. The query states, which consist of the position, direction, and image, are sampled uniformly from the entire state space, i.e., the position is first sampled uniformly in the room, then the orientation is sampled from $[0, 360)$ degrees. The optimal actions are computed as follows: turn towards the correct box if the agent is not yet facing it (within ± 15 degrees), otherwise move forward. Of the 20K collections of datasets, query states, and optimal actions, we use 16K for training and the remaining 4K for validation.

B Additional Experimental Results

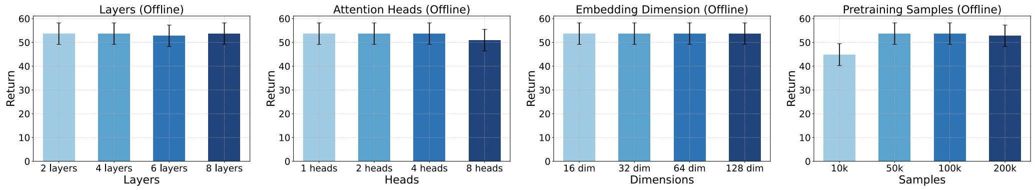
B.1 Bandits

This section reports additional experimental results in bandit environments.



(a) (b) (c) (d)

Figure 8: All comparisons in Dark Room evaluated on the tasks that were seen during pretraining, displayed next to their evaluations on test task counterparts from the main text.



(a) (b) (c) (d)

Figure 9: Sensitivity analysis of the offline Dark Rook task over the GPT-2 transformer's hyperparameters: (a) layers (b) attention heads (c) embedding dimensions (d) pretraining samples.

Out-of-distribution reward variances. In Figures 2c and 6a, we demonstrate the robustness of the basic pretrained model under shifts in the reward distribution at test time by varying the amount of noise observed in the rewards. DPT maintains robustness to these shifts similar to TS.

Bernoulli rewards. We test the out-of-distribution ability of DPT further by completely changing the reward distribution from Gaussian to Bernoulli bandits. Despite being trained only on Gaussian tasks during pretraining, DPT maintains strong performance both offline and online in Figures 6b and 6c.

B.2 Markov Decision Processes

This section reports additional experimental results in the Dark Room and Miniworld environments.

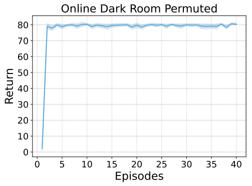


Figure 7: Online evaluation of DPT on Dark Room when tested on novel actions set permutations.

Performance on training tasks. In Fig. 8, we show the performance of each method on the training tasks in Dark Room. Offline, DPT and AD demonstrate comparable performance as on the training tasks, indicating a minimal generalization gap to new goals. Online, DPT, AD, and RL^2 also achieve performance on the training tasks similar to that on the test tasks.

Generalization to new dynamics. In this experiment, we study generalization to variations in a different aspect of the MDP, namely the dynamics. We design *Dark Room (Permuted)*, a variant of Dark Room in which the goal is fixed to a corner but the action space is randomly permuted. Hence, the agent must leverage its historical context to infer the effect of each action. On a held-out set of 20 permutations, DPT infers the optimal policy correctly every time offline, given only 100 offline samples, matching the optimal policy at 83 return. Similarly, the online performance immediately snaps to a near optimal policy in one episode once it identifies the novel permutation in Figure 7.

B.3 Sensitivity Analysis

We next seek to understand the sensitivity of DPT to different hyperparameters, including the model size and size of the pretraining dataset. These experiments are performed in the Dark Room environment. As shown in Fig. 9, the performance of DPT is robust to the model size; it is the same across different embedding sizes, number of layers, and number of attention heads. Notably, the performance is slightly worse with 8 attention heads, which may be attributed to slight overfitting. We do see that when the pretraining dataset is reduced to 10% of its original size (10000 samples) the performance degrades, but otherwise has similar performance with larger pretraining datasets.

C Additional Theory and Omitted Proofs

We start with a well-known concentration inequality for the maximum-likelihood estimate (MLE) to provide some more justification for the approximation made in Assumption 1. An early version of this result can be found in Zhang [2006]. We state a version from Agarwal et al. [2020]. Let $\mathcal{F}$ be a finite function class used to model a conditional distribution $p_{Y|X}(y|x)$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Assume there is $f^\star \in \mathcal{F}$ such that $p(y|x) = f^\star(y|x)$ (realizable), and $f(\cdot|x) \in \Delta(\mathcal{Y})$ for all $x \in \mathcal{X}$ and $f \in \mathcal{F}$ (proper). Let $D = \{x_i, y_i\}_{i \in [N]}$ denote a dataset of i.i.d samples where $x_i \sim p_X$ and $y_i \sim p_{Y|X}(\cdot|x_i)$. Let

$$\hat{f} = \operatorname*{argmax}_{f \in \mathcal{F}} \sum_{i \in [N]} \log f(y_i|x_i) \tag{6}$$

Proposition C.1 (Theorem 21 of Agarwal et al. [2020]). *Let D and $\hat{f}$ be given as above under the aforementioned conditions. Then, with probability at least $1 - \delta$,*

$$\mathbb{E}_{x \sim p_X} \|\hat{f}(\cdot|x) - p_{Y|X}(\cdot|x)\|_1^2 \leq \frac{8 \log\left(|\mathcal{F}|/\delta\right)}{N} \tag{7}$$

The finiteness of $\mathcal{F}$ is done for simplicity, but we can see that this yields dependence on the log-cardinality, a common measure of complexity. Extensions to infinite $\mathcal{F}$ of bounded statistical complexity can be readily made to replace this. For our setting, the bound suggests that $\mathbb{E}_{P_{pre}} \|P_{pre}(\cdot|s_{\text{query}}, D, \xi_h) - M_\theta(\cdot|s_{\text{query}}, D, \xi_h)\|_1^2 \to 0$ as $N \to \infty$ with high probability, provided the function class of M_θ has bounded statistical complexity.

C.1 Posterior Sampling

Posterior sampling is most generally described with the following procedure [Osband et al., 2013]. Initialize a prior distribution $\mathcal{T}_1 = \mathcal{T}_{\text{pre}}$ and dataset $D = \{\}$. For $k \in [K]$

1. Sample $\tau_k \sim \mathcal{T}_k$ and compute $\hat{\pi}_k$
2. Execute $\pi^\star_{\tau_k}$ and add interactions to D
3. Update posterior distribution $\mathcal{T}_{k+1}(\tau) = P_{pre}(\tau|D)$.

The prior and posteriors are typically over models such as reward functions in bandits or transition dynamics in MDPs.

C.2 Proof of Theorem 1

Proof. Without loss of generality, for a task τ, we take $\pi^\star_\tau(\cdot|s)$ to be deterministic and denote the optimal action in state s as $\pi^\star_\tau(s)$. Recall that we consider a fixed current task τ_c and a fixed in-context dataset D. We denote by P_{ps} and P_{M_θ} the joint distributions over ξ_h on τ_c for posterior sampling and M_θ under Assumption 1, both conditioned on D. Because of this, with some abuse of notation, we will also use P_{M_θ} to more generally refer to the pretraining distribution. We will prove this statement via induction over the length of the trajectory ξ_1 through ξ_H. First, consider the base case for a trajectory of length $h = 1$. We have that

$$\begin{aligned}
P_{ps}(s_1, a_1) &= \rho(s_1) P_{ps}(a_1|s_1) \\
&= \rho(s_1) \int_\tau P(\tau|D) \pi^\star_\tau(a_1|s_1) d\tau \\
&= \rho(s_1) M_\theta(a_1|s_1, D) \\
&= P_{M_\theta}(s_1, a_1)
\end{aligned}$$

where the second line uses the definition of the posterior sampling algorithm and the third line uses the fact that $\int_\tau P(\tau|D)\pi^\star_\tau(a_1|s_1)$ is equivalent to the posterior distribution over the optimal first action in the pretraining distribution. Now consider the inductive hypothesis at step $h-1$ that

$$P_{ps}(\xi_{h-1}) = P_{M_\theta}(\xi_{h-1}). \tag{8}$$

We aim to show that this also holds for ξ_h. By the inductive hypothesis, we have

$$\begin{aligned} P_{ps}(\xi_h) &= P_{ps}(\xi_{h-1})P_{ps}(s_h, a_h|\xi_{h-1}) \\ &= P_{M_\theta}(\xi_{h-1})P_{ps}(s_h, a_h|\xi_{h-1}). \end{aligned}$$

It remains to show that $P_{ps}(s_h, a_h|\xi_{h-1}) = P_{M_\theta}(s_h, a_h|\xi_{h-1})$. Note that this distribution is similar to the posterior but conditioned on the information that the policy used up until step h has generated the sequence ξ_{h-1} which collapses the probability over a_h. We can verify this claim by direct inspection. Due to the definition of the posterior sampler we have

$$P_{ps}(s_h, a_h|\xi_{h-1}) = T_{\tau_c}(s_h|s_{h-1}, a_{h-1})P_{ps}(a_h|s_h, \xi_{h-1}), \tag{9}$$

where we use $T_{\tau_c}(s_h|s_{h-1}, a_{h-1})$ to denote the Markov transition probability of the (unknown to the algorithm) true current task τ_c. Furthermore,

$$\begin{aligned} P_{ps}(a_h|s_h, \xi_{h-1}) &= \int_\tau P_{ps}(a_h, \tau|s_h, \xi_{h-1})d\tau & (10) \\ &= \int_\tau P_{ps}(\tau|s_h, \xi_{h-1})\pi^\star_\tau(a_h|s_h)d\tau & (11) \end{aligned}$$

where $P_{ps}(\tau|s_h, \xi_{h-1})$ is the probability that the posterior sampling algorithm sampled task τ at the start of the episode, given dataset D and the observed trajectory ξ_{h-1}, s_h.[5]

Using Bayes rule, we can re-express $P_{ps}(\tau|s_h, \xi_{h-1})$ as:

$$\begin{aligned} P_{ps}(\tau|s_h, \xi_{h-1}) &= \frac{P_{ps}(s_h, \xi_{h-1}|\tau)P(\tau|D)}{P_{ps}(s_h, \xi_{h-1})} & (12) \\ &= \frac{P_{ps}(s_h, \xi_{h-1}|\tau)P(\tau|D)}{T_{\tau_c}(s_h|s_{h-1}, a_{h-1})P_{ps}(\xi_{h-1})} & (13) \\ &= \frac{\left[\prod_{i=1}^{h-1} T_{\tau_c}(s_{i+1}|s_i, a_i)\pi^\star_\tau(a_i|s_i)\right] P(\tau|D)}{T_{\tau_c}(s_h|s_{h-1}, a_{h-1})P_{ps}(\xi_{h-1})} & (14) \end{aligned}$$

where the second line follows by the Markov property, and the third line follows since for a given task τ, posterior sampling executes the optimal policy for τ. Note that in the pretraining procedure, we train the model only on partial histories ξ_h labeled by the optimal policy. Therefore, the pretraining distribution is also decomposed as

$$\begin{aligned} P_{M_\theta}(\tau|s_h, \xi_{h-1}) &= \frac{P_{M_\theta}(s_h, \xi_{h-1}|\tau)P(\tau|D)}{P_{M_\theta}(s_h, \xi_{h-1})} & (15) \\ &\propto \frac{\left[\prod_{i=1}^{h-1} \pi^\star_\tau(a_i|s_i)\right] P(\tau|D)}{P_{M_\theta}(s_h, \xi_{h-1})}. & (16) \end{aligned}$$

Since the numerators between the two posteriors are proportional in τ and the denominators do not depend on τ, we have that $P_{ps}(\tau|s_h, \xi_{h-1}) = P_{M_\theta}(\tau|s_h, \xi_{h-1})$. Therefore,

$$\begin{aligned} \int_\tau P_{ps}(\tau|s_h, \xi_{h-1})\pi^\star_\tau(a_h|s_h)d\tau &= \int_\tau P_{M_\theta}(\tau|s_h, \xi_{h-1})\pi^\star_\tau(a_h|s_h)d\tau & (17) \\ &= M_\theta(a_h|s_h, \xi_{h-1}, D). & (18) \end{aligned}$$

This concludes the proof.

□

---

[5] Note, posterior sampling is always assumed to have computed the optimal policy π^*_τ for the task it samples τ. However, ambiguity in which task was sampled by PS can arise when the actions taken so far, and states visited, could be trajectories from optimal policies from two or more tasks.

C.3 Proof of Corollary 6.1

Proof. We use the equivalence between M_θ and posterior sampling established in Theorem 1. The proof then follows immediately from Theorem 1 of Osband et al. [2013] to guarantee that

$$\mathbb{E}_{\mathcal{T}_{\text{pre}}}\left[\text{Reg}_\tau(M_\theta)\right] \le \widetilde{\mathcal{O}}\left(HS\sqrt{AK}\right) \tag{19}$$

where the notation $\widetilde{\mathcal{O}}$ omits polylogarithmic dependence. The bound on the test task distribution follows from the assumed bound on the likelihood ratio under the priors:

$$\int \mathcal{T}_{\text{test}}(\tau)\text{Reg}_\tau(M_\theta)d\tau \le \mathcal{C}\int \mathcal{T}_{\text{pre}}(\tau)\text{Reg}_\tau(M_\theta)d\tau. \tag{20}$$

☐

C.4 Proof of Corollary 6.2

Erratum: In the main text, we erroneously stated that the regret is $\widetilde{\mathcal{O}}(\sqrt{dK})$. The correct regret is $\widetilde{\mathcal{O}}(d\sqrt{K})$.

Proof. The proof once again follows by immediately deferring to the established result of Russo and Van Roy [2014] (Proposition 3) for linear bandits by the posterior sampling equivalence of Theorem 1. This ensures that posterior sampling achieves regret $\widetilde{\mathcal{O}}(d\sqrt{K})$. It remains, however, to justify that $P_{pre}(\cdot|D_k)$ will be covered by Gaussian Thompson Sampling for all D_k with $k \in [K]$. This is verified by noting that $P_{ps}(a|D_k) > 0$ for non-generate Gaussian Thompson Sampling (positive variances of the prior and noise distribution) and finite K. This guarantees that any D_k will have support. ☐

C.5 Proof of Proposition 6.3

Proof. The proof follows by direct inspection of the pretraining distributions. For P^1_{pre}, we have

$$P^1_{pre}(a^\star|s_{\text{query}}, D, \xi) = \int_\tau \pi^\star_\tau(a^\star|s_{\text{query}})P^1_{pre}(\tau|s_{\text{query}}, D, \xi)d\tau \tag{21}$$

The posterior distribution over tasks is simply

$$\begin{aligned}
P^1_{pre}(\tau|s_{\text{query}}, D, \xi) &= \frac{P^1_{pre}(\tau, s_{\text{query}}, D, \xi)}{P^1_{pre}(s_{\text{query}}, D, \xi)} && (22)\\
&\propto P^1_{pre}(\tau)P^1_{pre}(\xi|\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\mathcal{D}^1_{\text{pre}}(D;\tau) && (23)\\
&= P^2_{pre}(\tau)P^2_{pre}(\xi|\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\mathcal{D}^1_{\text{pre}}(D;\tau) && (24)
\end{aligned}$$

Then, the distribution over the in-context dataset can be decomposed as

$$\begin{aligned}
\mathcal{D}^1_{pre}(D;\tau) &= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1};\tau) && (25)\\
&= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1}) && (26)\\
&= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1})\frac{\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1})}{\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1})} && (27)\\
&= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1})\prod_{i\in[n]}\frac{\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1})}{\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1})} && (28)\\
&= \prod_{i\in[n]} R_\tau(r_i|s_i,a_i)T_\tau(s'_i|s_i,a_i)\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1},\tau)\prod_{i\in[n]}\frac{\mathcal{D}^1_{\text{pre}}(a_i|s_i,D_{i-1})}{\mathcal{D}^2_{\text{pre}}(a_i|s_i,D_{i-1})} && (29)\\
&\propto \mathcal{D}^2_{pre}(D;\tau), && (30)
\end{aligned}$$

where the second equality holds because $\mathcal{D}^1_{\text{pre}}(a_j|s_j, D_j; \tau)$ is assumed to be invariant to τ, and the fifth equality holds because $\mathcal{D}^2_{\text{pre}}(a_j|s_j, D_j; \tau)$ is assumed to be invariant to τ.

Therefore, we conclude that

$$\begin{aligned} P^1_{pre}(\tau|s, D, \xi) &\propto P^2_{pre}(\tau)P^2_{pre}(\xi|\tau)\mathcal{D}_{\text{query}}(s_{\text{query}})\mathcal{D}^2_{\text{pre}}(D; \tau) && (31) \\ &\propto P^2_{pre}(\tau|s, D, \xi). && (32) \end{aligned}$$

Since also $\sum_\tau P^1_{pre}(\tau|s, D, \xi) = 1 = \sum_\tau P^2_{pre}(\tau|s, D, \xi)$, then

$$P^1_{pre}(\tau|s, D, \xi) = P^2_{pre}(\tau|s, D, \xi). \tag{33}$$

Substituting this back into Equation 21 yields $P^1_{pre}(a^\star|s_{\text{query}}, D, \xi) = P^1_{pre}(a^\star|s_{\text{query}}, D, \xi)$. ☐