LANGUAGE MODEL AUGMENTED DECISION TRANSFORMER FOR MULTI-TASK OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline multi-task reinforcement learning (MTRL) aims to learn a unified policy from heterogeneous offline datasets that generalizes across tasks. However, existing offline MTRL methods often struggle with two fundamental issues: severe per-task data scarcity and limited cross-task generalization. In this paper, we propose Language-Model Augmented Decision Transformer (LaMA-DT), a unified sequence-modeling framework that leverages a single large language model (LLM) for both trajectory augmentation and multi-task policy learning. LaMA-DT uses natural-language task descriptions to guide task-aligned trajectory augmentation through a set of supervised generation and alignment objectives, and incorporates a scoring-and-filtering mechanism to retain high-quality augmented trajectories. The same LLM is then repurposed as a language-conditioned Decision Transformer and fine-tuned on a mixture of real and augmented data. Experiments on Meta-World and MuJoCo multi-task benchmarks demonstrate that LaMA-DT consistently outperforms competitive baselines, particularly in low-data regimes, showing that language-guided data augmentation substantially improves cross-task generalization.

1 INTRODUCTION

Offline reinforcement learning (RL) (Levine et al., 2020) enables agents to learn high-quality decision policies directly from static datasets. To extend offline RL to more realistic applications, it is important to handle multi-task settings, where an agent must solve multiple related tasks that share state and action spaces but differ in goals, rewards, or context. This motivates offline multi-task RL (MTRL) (Yu et al., 2020b), which learns from heterogeneous offline datasets and aims to produce a single policy that can perform well across tasks. In practice, offline MTRL is limited by two tightly coupled challenges. First, *per-task data is often scarce:* each task may have only a small and fragmented dataset, since collecting trajectories for many tasks is expensive and often infeasible. Second, *cross-task generalization is difficult:* tasks vary in objectives, dynamics, and observation distributions, leading to substantial distribution shifts that hinder learning a single policy across tasks, especially when limited state–action coverage further amplifies these shifts.

Existing offline MTRL methods offer partial solutions, but typically target only one challenge or rely on strong data assumptions. For improving data use across tasks, CDS (Yu et al., 2021) transfers transitions from auxiliary tasks to a target task based on estimated usefulness. While effective when abundant and closely related auxiliary data are available, CDS struggles with cross-task generalization, as it requires training a separate model for each target task and degrades when auxiliary data are scarce or poorly matched. SRTD (Yoo et al., 2022) learns a shared latent skill space and performs data augmentation via imagined rollouts; however, learning a stable and transferable latent representation generally requires sufficient task-specific data. When per-task datasets are limited, the learned latent space becomes unreliable, reducing the effectiveness of the generated rollouts. A separate line of work builds on Decision Transformers (DTs) (Chen et al., 2021) and adapts them to multi-task learning to improve generalization through shared sequence modeling. For example, HarmoDT (Hu et al., 2024) improves parameter sharing by reducing gradient conflicts across tasks, and SoCo-DT (Wang et al., 2025) refines sharing via Fisher-information–guided soft masks. However, these DT-based MTRL methods still typically rely on having enough trajectories per task to learn transferable sequence

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

representations; when each task has limited data, performance can drop sharply due to insufficient coverage and overfitting to narrow task-specific patterns. These gaps naturally raise a question: *In offline MTRL under **low-data regimes** (i.e., with scarce task-specific data), how can we expand effective data coverage while achieving robust cross-task generalization?*

A key observation is that, although per-task trajectories are limited, high-level task descriptions are almost always available. Such descriptions convey information such as the task name, goal, object identity, or expertise level, which can be succinctly expressed in natural language. This motivates our approach: use natural language as a task-specification medium to guide trajectory augmentation via large language models (LLMs), thereby improving multi-task policy learning in data-scarce scenarios. However, since LLMs are not pretrained on trajectory data, realizing this idea requires ensuring that augmented trajectories both remain faithful to limited real data while expanding state–action coverage, and stay aligned with task descriptions to prevent cross-task drift.

In this paper, we propose Language-Model Augmented Decision Transformer (LaMA-DT) for offline MTRL, which employs a single LLM for both trajectory augmentation and multi-task policy learning. To enable high-quality, task-aligned trajectory generation, the LLM is first fine-tuned across tasks using four carefully designed supervised objectives that jointly capture environmental dynamics and temporal dependencies in sequential decision-making. To further ensure semantic alignment between natural-language task descriptions and generated trajectories, LaMA-DT incorporates two additional task–trajectory alignment objectives. All six objectives are jointly optimized under a dedicated multi-objective fine-tuning strategy tailored for LLM-based trajectory learning. To ensure that only high-quality augmented trajectories are used for policy learning, we introduce a likelihood-based scoring-and-filtering mechanism that evaluates generated trajectories across all objectives and discards low-quality or task-misaligned samples. Finally, the same LLM is repurposed as a DT—a natural choice for sequence modeling—conditioned on natural-language task descriptions and fine-tuned on a mixture of real and high-quality augmented trajectories, resulting in robust cross-task generalization. Key contributions are summarized as follows:

- We propose LaMA-DT, an LLM-based offline MTRL framework that unifies trajectory augmentation and multi-task policy learning. It addresses data scarcity by leveraging natural-language task descriptions to guide task-aligned trajectory augmentation via a set of objectives that capture task-specific data distributions and enforce semantic alignment.
- To enable cross-task generalization in offline MTRL, LaMA-DT introduces an LLM-based policy learning mechanism that combines scored and filtered high-quality augmented trajectories with original data, conditioned on task descriptions, to fine-tune a DT initialized from the same LLM, yielding improved cross-task generalization.
- LaMA-DT demonstrates consistent improvements over competitive baselines on Meta-World and MuJoCo multi-task benchmarks, with particularly strong gains in **low-data regimes**, highlighting that effective data augmentation substantially enhances multi-task learning performance[1].

2 PRELIMINARIES

Offline Multi-Task Reinforcement Learning (MTRL). An RL problem is typically modeled as a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ is the transition distribution, $r(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The goal is to learn a policy $\pi(a \mid s)$ that maximizes the expected discounted return: $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. In offline RL, the agent must learn solely from a fixed dataset of trajectories $\mathcal{D} = \{\tau_j\}_{j=1}^{N}$ collected by an unknown behavior policy, where each trajectory $\tau_j = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$. Offline MTRL extends the offline RL setting to a collection of tasks $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^{M}$, where each task $\mathcal{T}_i$ corresponds to its own MDP $(\mathcal{S}, \mathcal{A}, P^{\mathcal{T}_i}, r^{\mathcal{T}_i}, \gamma)$ and is associated with an offline dataset $\mathcal{D}_i = \{\tau_j^i\}$. Tasks may differ in their reward functions and transition dynamics, but share a common state and action space. The objective is to learn a single policy that performs well across this fixed set of tasks. Formally, the optimal policy is defined as: $\pi^* = \arg\max_\pi \frac{1}{M} \sum_{i=1}^{M} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t^{\mathcal{T}_i}\right]$.

[1]Code available at `https://anonymous.4open.science/r/o33zlamadt-6587`.

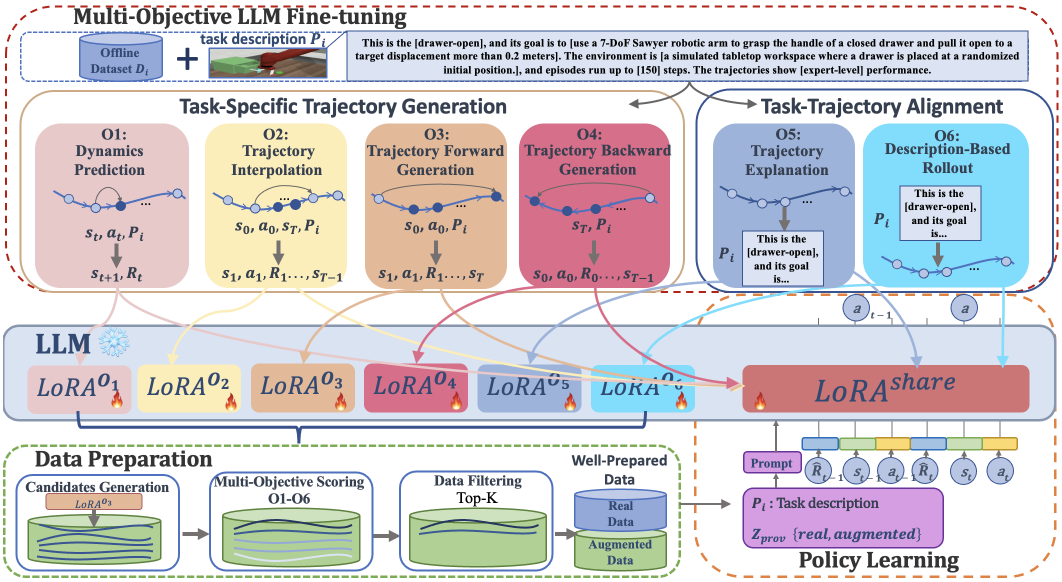


Figure 1: LaMA-DT consists of three components: (i) Multi-Objective Fine-Tuning. We adapt the LLM to generate meaningful, task-specific trajectories from limited offline data and natural-language task descriptions by training objective-specific LoRA adapters together with a shared LoRA adapter, enabling both trajectory generation and task–trajectory alignment. (ii) Data Preparation. Candidate trajectories are generated per task using O_3, scored using aggregated multi-objective likelihoods, and the top-K trajectories are retained and merged with the original offline data. (iii) Multi-Task Policy Learning. We repurpose the LLM and shared LoRA adapter as a prompt-conditioned DT, trained on the combined real and augmented dataset, with task descriptions $\mathbf{p}_i$ and a provenance token z_{prov}.

Decision Transformer (DT). DT (Chen et al., 2021) formulates offline RL as a return-conditioned sequence modeling problem using a transformer architecture, learning policies directly from offline trajectories by autoregressively predicting actions. In DT, each trajectory is represented as a sequence of tokens $(\hat{R}_t, s_t, a_t)$, where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ is the return-to-go (RTG) from timestep t. The RTG serves as a conditioning signal, encouraging the model to imitate high-return behaviors present in the dataset. The model is trained to predict the next action a'_t given the history and current state, with the ground-truth action a_t as supervision.

3 Methodology

In this section, we present Language-Model Augmented Decision Transformer (LaMA-DT) for offline MTRL, which employs a single LLM for both trajectory augmentation and multi-task policy learning. Fig. 1 illustrates the framework overview.

3.1 Task Descriptions in Offline MTRL

A central difficulty in offline MTRL is data scarcity: per-task datasets are often small, making it difficult to learn reliable policies that generalize across tasks. Although trajectory data are scarce, rich task-level metadata are almost always available. Such metadata—including task names, goals, relevant objects, environmental conditions, and typical performance—can be naturally expressed in natural language. This observation motivates the use of natural-language task descriptions to guide an LLM in generating additional, task-aligned trajectories, leveraging its strong sequence-modeling and reasoning capabilities to expand effective data coverage and thus mitigate data scarcity. To operationalize this idea, we construct a standardized natural-language description for each task $\mathcal{T}_i \in \mathcal{T}$ using a simple template: "`This is the [task_name], and its goal is to [task_goal]. The environment is [environment_description]. and episodes run up to [max_episode_length] steps. The trajectories show [expertise_level] performance.`"

Each placeholder is deterministically filled by metadata extracted from task $\mathcal{T}_i$. Formally, our training data consist of pairs of task descriptions and their associated trajectories, $\left\{ (\mathbf{p}_i,\ \mathcal{D}_i = \{\tau_j^i\}) \right\}_{i=1}^{M}$, where $\mathbf{p}_i$ denotes the natural-language description of task $\mathcal{T}_i$.

3.2 Task-Aligned Trajectory Augmentation with LLMs

3.2.1 Task-Specific Trajectory Generation

We design four trajectory-generation objectives (O_1–O_4) to fine-tune the LLM, enabling it to capture environmental dynamics, temporal dependencies in sequential decision-making, and task-specific data distributions, thereby generating plausible trajectories that resemble the available data while effectively expanding state–action coverage.

Dynamics Learning Objective. We first equip the LLM with an explicit understanding of environment dynamics to ensure physically grounded trajectory generation via O_1:

Objective 1: Dynamics Prediction (O_1). Accurate modeling of dynamics is essential to prevent physically implausible trajectories (*e.g.*, a robot arm violating joint limits or a locomotion agent defying gravity). O_1 fine-tunes the LLM to predict the next state s' given the current state s and action a in a supervised manner, enabling it to capture the transition dynamics. For each task $\mathcal{T}_i$, the LLM receives its task description $\mathbf{p}_i$ along with the prompt: "`Predict the next state given the current state and action` (s, a)`.`"

Sequential Decision Learning Objectives. Beyond modeling environmental dynamics, we design O_2–O_4 to capture temporal dependencies in sequential decision-making across different tasks from complementary perspectives, while also reinforcing the dynamics learned in O_1:

Objective 2: Trajectory Interpolation (O_2). O_2 is designed to generate plausible intermediate states, actions, and rewards between two observed points along a trajectory. This objective encourages the LLM to preserve local temporal consistency, ensuring that short-horizon transitions remain coherent within the overall trajectory and thereby reducing error accumulation in long-horizon rollouts. In practice, we randomly select two states from a real trajectory as the start and end states, and use the ground-truth subtrajectory between them as supervision. For O_2, the LLM is provided with the task description $\mathbf{p}_i$ for task $\mathcal{T}_i$ along with the prompt: "`Fill in the missing part of the trajectory given the start state and action` (s_0, a_0) `and the final state` s_T`.`"

Objective 3: Trajectory Forward Generation (O_3). Forward generation is a core capability for trajectory modeling: given an initial state–action pair, the model should be able to generate a plausible rollout over time. O_3 aims to capture long-horizon temporal dependencies by encouraging the LLM to maintain consistency throughout the trajectory in the forward direction. Specifically, O_3 fine-tunes the LLM in a supervised manner to generate complete trajectories starting from (s_0, a_0), using trajectories from $\mathcal{D}_i$ as ground-truth supervision. For O_3, the LLM is provided with the task description $\mathbf{p}_i$ for task $\mathcal{T}_i$ along with the prompt: "`Generate the whole trajectory until termination given the start state and action` (s_0, a_0)`.`"

Objective 4: Trajectory Backward Generation (O_4). Human learning often relies on bidirectional reasoning: we not only predict future consequences from a current state but also reason backward from a desired outcome. A similar capability is valuable for trajectory generation. Relying solely on forward prediction may lead to error accumulation or trajectories that diverge from the intended task goal—especially in multi-task settings where tasks may share similar local dynamics but differ substantially in their global objectives. O_4 is designed to address this issue by fine-tuning the LLM to reconstruct trajectories backward from a given final state with supervision. For O_4, the LLM is provided with the task description $\mathbf{p}_i$ for task $\mathcal{T}_i$ along with the prompt: "`Reconstruct the trajectory backwards from the final state` s_T`.`"

3.2.2 Task–Trajectory Alignment

However, O_1–O_4 rely primarily on limited task-specific trajectories as supervision and do not explicitly enforce alignment between generated trajectories and the intended task description $\mathbf{p}_i$. As a result, the LLM may learn to generate coherent and physically plausible trajectories driven mainly by data supervision, while failing to respect the specified task descriptions. Such task–trajectory

drift is particularly problematic in multi-task settings, where different tasks often share similar state–action distributions yet differ fundamentally in their goals. To address this gap, we introduce two task–trajectory alignment objectives (O_5–O_6) that explicitly couple generated trajectories with their natural-language task descriptions:

Objective 5: Trajectory Explanation (O_5). O_5 trains the LLM to map trajectories to natural-language explanations in a supervised manner, requiring it to interpret behavioral patterns and infer the underlying task intent. By learning to articulate what a trajectory accomplishes, the LLM develops a semantic understanding of task goals, allowing it to distinguish trajectories based on purpose rather than merely imitating plausible motion patterns. Given a specific trajectory τ and an explanatory prompt (*i.e.*, "`Explain the following trajectory in natural language.`"), the model is trained to produce a description summarizing the task goal and the executed behavior.

Objective 6: Description-Based Rollout (O_6). O_6 trains the LLM to generate trajectories directly from natural-language task descriptions, enabling conditional planning across diverse tasks based on semantic understanding. Given a task description $\mathbf{p}_i$, an initial state s_0, and an expected return r, the model is trained to produce a trajectory $\tilde{\tau}$ that fulfills $\mathbf{p}_i$. By explicitly grounding trajectory generation in the semantics of $\mathbf{p}_i$, O_6 ensures that the LLM produces task-aligned behaviors—an ability that is critical in multi-task settings where different tasks may share similar initial states or final rewards yet require fundamentally different goals and behaviors.

O_5 and O_6 form a bidirectional pair that establishes a semantic feedback loop: O_5 maps trajectories to task descriptions ($\tau \rightarrow \mathbf{p}_i$), while O_6 performs the inverse mapping from task descriptions to trajectories ($\mathbf{p}_i \rightarrow \tau$). When interleaved with O_1–O_4, these alignment objectives prevent the model from drifting toward goal-agnostic generation and ensure that augmented trajectories are not only physically plausible but also semantically consistent with their intended task descriptions.

3.2.3 Multi-Objective LLM Fine-Tuning.

To mitigate gradient interference among the heterogeneous supervision objectives O_1–O_6, we introduce an objective-aware fine-tuning strategy utilizing LoRA (Hu et al., 2022). Specifically, each objective O_j ($j = 1, \ldots, 6$) is associated with an objective-specific LoRA adapter $\boldsymbol{\theta}^{O_j}_{\text{LoRA}}$, while the base LLM parameters $\boldsymbol{\theta}_{\text{base}}$ remain frozen (as shown in Fig. 1). During training, each objective updates only its corresponding adapter, effectively isolating task-specific gradients and stabilizing multi-objective optimization. However, isolating objectives via separate adapters does not explicitly promote knowledge sharing across objectives. To facilitate cross-objective knowledge transfer, we introduce an additional shared LoRA adapter $\boldsymbol{\theta}^{\text{sh}}_{\text{LoRA}}$, which is updated jointly by all objectives. Concretely, when optimizing objective O_j, we activate both the objective-specific adapter $\boldsymbol{\theta}^{O_j}_{\text{LoRA}}$ and the shared adapter $\boldsymbol{\theta}^{\text{sh}}_{\text{LoRA}}$, and minimize: $\mathcal{L}_{O_j}\left(\theta_{\text{base}} + \theta^{O_j}_{\text{LoRA}} + \theta^{\text{sh}}_{\text{LoRA}}\right), j \in \{1, \ldots, 6\}$. In this design, the objective-specific adapters capture specialized capabilities required by objective, while the shared adapter aggregates transferable knowledge across objectives.

3.3 Multi-Task Policy Learning with LaMA-DT

3.3.1 Data Preparation for LaMA-DT.

After fine-tuning the LLM with O_1–O_6, we perform task-wise trajectory augmentation using the forward generation objective O_3. For each task $\mathcal{T}_i$, we sample initial state–action pairs (s_0, a_0) from the real offline dataset $\mathcal{D}_i$ and generate N candidate trajectories: $\{\tilde{\tau}^k\}_{k=1}^{N} \sim p_{O_3}(\tau \mid \mathbf{p}_i, (s_0, a_0))$.

Augmented Trajectory Scoring and Filtering. Not all generated trajectories are equally reliable or task-consistent. To filter out low-quality or misaligned generations, we score each candidate trajectory $\tilde{\tau}$ using all six objectives (O_1–O_6) via likelihood-based evaluation without parameter updates, reusing the previously trained objective-specific LoRA adapters ($\boldsymbol{\theta}^{O_1}_{\text{LoRA}}$-$\boldsymbol{\theta}^{O_6}_{\text{LoRA}}$). Concretely, for O_1, we measure transition consistency as

$$S_{O_1}(\tilde{\tau}) = \frac{1}{T}\sum_{t=0}^{T-1} \log p_{O_1}(s_{t+1} \mid \mathbf{p}_i, s_t, a_t). \tag{1}$$

For O_2, we sample a subtrajectory from $\tilde{\tau}$ by selecting a start index u and fixing the terminal state s_T, and score the likelihood of reconstructing the intermediate segment as:

$$S_{O_2}(\tilde{\tau}) = \frac{1}{|\tilde{\tau}_{u:T} - 1|} \sum_{t=u+1}^{T-1} \log p_{O_2}(s_t, a_t, r_t \,|\, \mathcal{C}_2(t)) \,, \tag{2}$$

where $\tilde{\tau}_{u:T} := \{(s_t, a_t, r_t)\}_{t=u}^{T-1}$ denotes the intermediate portion of $\tilde{\tau}$, and the conditioning context is $\mathcal{C}_2(t) = (\mathbf{p}_i,\ s_T,\ \tilde{\tau}_{u:t})$.

The rollout-based objectives—forward generation (O_3), backward generation (O_4), and description-based rollout (O_6)—share the same autoregressive likelihood form:

$$S_{O_j}(\tilde{\tau}) = \frac{1}{T} \sum_{t=0}^{T-1} \log p_{O_j}(s_{t+1}, a_{t+1}, r_{t+1} \mid \mathcal{C}_j, \tilde{\tau}_{\leq t}), \tag{3}$$

where the conditioning context $\mathcal{C}_j$ differs across objectives: O_3 conditions on the task description and initial state–action pair $(\mathbf{p}_i, s_0, a_0)$; O_4 conditions on the task description and terminal state s_T; and O_6 conditions on the task description $\mathbf{p}_i$ together with (s_0, r).

For O_5, we evaluate alignment by scoring the task description under the trajectory-to-text model,

$$S_{O_5}(\tilde{\tau}) = \frac{1}{|\mathbf{p}_i|} \sum_{\ell=1}^{|\mathbf{p}_i|} \log p_{O_5}(p_{i,\ell} \mid \tilde{\tau}, p_{i,<\ell}). \tag{4}$$

All scores are length-normalized. We aggregate the six scores with the mean $S(\tilde{\tau}) \triangleq \frac{1}{6} \sum_{j=1}^{6} S_{O_j}(\tilde{\tau})$.

For each task, we retain the top-K candidates with the highest overall scores and add them to the augmented dataset: $\mathcal{D}_i^{\text{aug}} = \text{Top-}K\Big(\{\tilde{\tau}^k\}_{k=1}^{N}\ ;\ S(\cdot)\Big)$, and train the final policy on the combined dataset $\mathcal{D}_i \cup \mathcal{D}_i^{\text{aug}}$.

3.3.2 Policy Learning with LaMA-DT

LaMA-DT Initialization. After data preparation, we fine-tune LaMA-DT by repurposing the same LLM as a language-conditioned DT and attaching a policy-specific LoRA adapter $\boldsymbol{\theta}_{\text{LoRA}}^{\pi}$. The policy adapter is initialized from the shared adapter learned during trajectory augmentation: $\boldsymbol{\theta}_{\text{LoRA}}^{\pi} \leftarrow \theta_{\text{LoRA}}^{\text{sh}}$. This initialization explicitly transfers shared knowledge acquired from O_1–O_6 into the policy learning stage, ensuring that the augmentation objectives directly benefit multi-task policy learning.

LaMA-DT Policy Learning. Using the prepared dataset $\mathcal{D}_i \cup \mathcal{D}_i^{\text{aug}}$, each trajectory τ—real or augmented—is serialized into a sequence of triples $\tau = \{(\hat{R}_t, s_t, a_t)\}_{t=0}^{T}$, where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ denotes the return-to-go. Each trajectory is prepended with two forms of conditioning information: (i) a provenance token $z_{\text{prov}} \in \{\texttt{real}, \texttt{augmented}\}$ indicating whether the trajectory is real or augmented, and (ii) the natural-language task description $\mathbf{p}_i$. The provenance token explicitly informs the LLM of the data source, enabling LaMA-DT to perform source-aware behavior cloning. This allows the policy to benefit from the diversity of augmented trajectories while remaining anchored to the empirical distribution of real data. We finally optimize LaMA-DT using Equation 5. The loss minimizes the mean squared error between ground-truth and predicted actions:

$$\mathcal{L}_{\text{LaMA-DT}} \triangleq \mathbb{E}_{\substack{\mathcal{T}_i \sim \mathcal{T} \\ \tau \sim \mathcal{D}_i \cup \mathcal{D}_i^{\text{aug}}}} \left[\sum_{t=1}^{T} \| a_t - \pi(\tau_{0:t-1},\ \mathbf{p}_i,\ z_{\text{prov}}) \|_2^2 \right]. \tag{5}$$

Testing and Generalization. At test time, given a task description $\mathbf{p}_i$, we set $z_{\text{prov}} = \texttt{real}$ and follow the standard DT rollout procedure. Starting from the initial observation, the model autoregressively predicts actions conditioned on the history $(\hat{R}_{\leq t}, s_{\leq t}, a_{<t})$ and executes them in the environment. Notably, LaMA-DT can be readily adapted to unseen tasks via few-shot learning: a small set of task-specific trajectories from the new task is used to quickly adapt the objective-specific LoRA adapters for data augmentation, after which scoring-and-filtering and policy learning are performed to obtain an adapted policy. The detailed algorithm for LaMA-DT is in Alg 1.

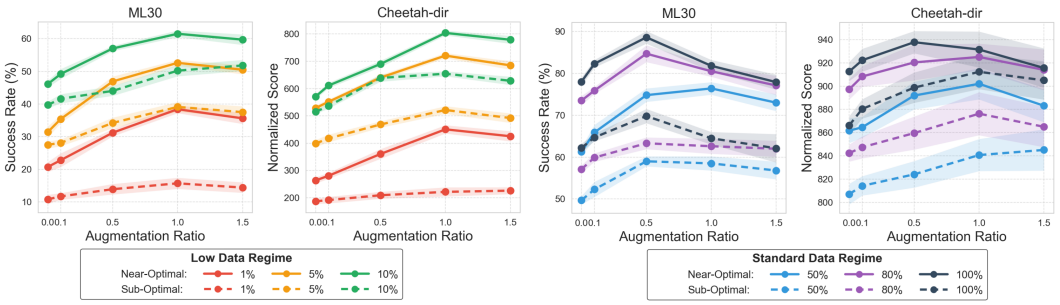


Figure 2: Performance of LaMA-DT with different augmentation ratios $r = |\mathcal{D}_i^{\text{aug}}|/|\mathcal{D}_i|$ on Meta-World ML30 and MuJoCo Cheetah-dir, across low and standard data regimes, and under sub-optimal and near-optimal data qualities.

Table 1: The average returns of LaMA-DT vs. baselines on Meta-World (Sub-Optimal) and MuJoCo (Near-Optimal) datasets under standard data regime.

	Task	CDS	MTBC	MTIQL	MTDIFF	MTDT	Prompt-DT	HarmoDT	LaMA-DT
MW	ML5	50.7±3.3	46.2±2.8	61.1±4.5	76.0±2.8	74.5±4.4	76.8±3.0	78.9±3.8	**83.4±2.9**
	ML30	32.9±2.5	34.5±1.7	43.3±2.2	57.5±1.5	55.2±2.1	58.7±1.3	63.4±1.6	**69.8±1.7**
MuJoCo	Cheetah-vel	-134.5±15.8	-148.4±11.2	-142.0±14.4	-186.2±8.2	-170.1±17.9	-127.6±10.9	-86.6±7.7	**-73.5±12.5**
	Cheetah-dir	428.9±17.5	429.7±6.6	454.1±21.3	536.8±10.7	678.5±24.3	705.6±8.3	926.3±10.4	**937.8±9.6**
	Ant-dir	131.0±10.6	153.8±15.9	120.6±9.9	248.0±16.2	235.7±15.4	302.5±17.1	315.6±5.5	**328.5±8.4**

4 EXPERIMENTS

Our experiments are designed to answer the questions: **(Q1)** How does LaMA-DT compare with strong offline MTRL baselines on benchmarks? **(Q2)** In low-data regimes, does LaMA-DT retain advantages over data-hungry multi-task DT variants? **(Q3)** Does LaMA-DT exhibit consistent performance gains across offline datasets of varying quality? **(Q4)** How does the amount of augmented data influence performance in both standard- and low-data regimes? **(Q5)** Are the individual components of LaMA-DT critical? **(Q6)** Are the augmented data in LaMA-DT high-quality?

Experimental Setup. We evaluate LaMA-DT on two widely used offline MTRL benchmarks: MuJoCo (Todorov et al., 2012) and Meta-World (Yu et al., 2020b). For MuJoCo locomotion, we follow prior prompt-based Decision Transformer protocols (Xu et al., 2022) and report results on Cheetah-dir, Cheetah-vel, and Ant-dir. For Meta-World, we use the ML5 and ML30 task suites with shared Sawyer robot dynamics, and evaluate performance using the average *success rate* across tasks. Offline datasets are collected using SAC (Haarnoja et al., 2018). We consider two dataset settings: (i) a *near-optimal* setting consisting primarily of expert trajectories, and (ii) a *sub-optimal* setting comprising 50% expert and 50% non-expert trajectories. For low data regimes, we uniformly subsample the per-task training data to {1%, 5%, 10%} of the full dataset. All results are averaged over 3 seeds, and each task is evaluated for 50 episodes.

Baselines. We compare LaMA-DT against representative offline MTRL baselines: (i) **CDS** (Yu et al., 2021), which applies a conservative transition-sharing strategy across tasks; (ii) **MTBC** (Hu et al., 2024), a multi-task behavior cloning approach using network scaling and task-ID–conditioned actors; (iii) **MTIQL** (Kostrikov et al., 2021a), which extends IQL to the multi-task setting with task-conditioned actors and critics; (iv) **MTDIFF** (He et al., 2023), a diffusion-based generative planner for multi-task RL; (v) **MTDT** (Chen et al., 2021), a multi-task DT conditioned on task IDs; (vi) **Prompt-DT** (Xu et al., 2022), which leverages trajectory prompts and return-to-go conditioning on DT; and (vii) **HarmoDT** (Hu et al., 2024), a prompt-based DT with task-specific masking.

Main Results (Q1). Table 1 summarizes average performance on Meta-World (ML5/ML30) and MuJoCo locomotion tasks (Cheetah-vel, Cheetah-dir, Ant-dir) under standard data regimes. LaMA-DT consistently outperforms all baselines across benchmarks, achieving an average improvement of 5.5% on Meta-World and 6.8% on MuJoCo relative to the strongest baseline. On Meta-World, gains are most pronounced on ML30, a highly heterogeneous task suite prone to negative transfer, where LaMA-DT improves over the best baseline by 6.4%. On MuJoCo, LaMA-DT performs strongly on both direction- and velocity-conditioned control, achieving the largest improvements on Cheetah-vel

Table 2: Performance under low-data regimes (1%, 5%, and 10%) across different data qualities (sub-optimal vs. near-optimal).

Method	ML30						Cheetah-dir					
	Sub-Optimal			Near-Optimal			Sub-Optimal			Near-Optimal		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
MTDT	10.9±3.0	19.2±2.4	25.6±2.3	12.5±2.0	24.1±2.3	35.6±1.9	113.4±20.2	274.5±18.0	386.3±21.9	145.2±23.1	320.5±19.6	435.6±20.4
Prompt-DT	12.8±1.4	24.3±0.9	36.8±1.1	15.3±1.5	28.5±0.7	41.2±0.8	191.3±17.1	378.9±18.3	507.4±15.3	256.3±16.2	474.0±12.8	544.7±15.7
HarmoDT	12.1±1.6	27.0±1.4	42.6±1.5	22.4±1.1	36.1±0.9	46.8±0.9	185.2±12.3	452.2±17.5	526.8±13.9	260.7±11.2	540.3±9.8	592.0±10.3
LaMA-DT	**15.7±1.7**	**39.1±1.5**	**51.8±1.7**	**38.4±1.2**	**52.6±0.9**	**61.5±1.0**	**225.9±12.7**	**521.5±14.0**	**654.4±9.2**	**450.6±15.1**	**720.8±9.4**	**803.9±8.9**

(15.1%) and Ant-dir (4.1%), while remaining competitive on Cheetah-dir. These results indicate that even under standard data regimes, the proposed data augmentation strategy in LaMA-DT continues to provide consistent performance gains.

Performance in Low-Data Regimes (Q2). In Table 2, across all benchmarks and data fractions, LaMA-DT consistently outperforms DT-based baselines, demonstrating substantial advantages in low-data regimes. It achieves consistent gains on both ML30 and Cheetah-dir across all data levels (1%, 5%, and 10%), with performance improving steadily as more data become available. For example, on ML30 with suboptimal offline data, LaMA-DT improves from 15.7 at 1% data to 51.8 at 10%, significantly outperforming Prompt-DT (by 22.7% at 1% and 40.8% at 10%) and HarmoDT (by 29.8% at 1% and 21.6% at 10%), indicating superior data efficiency. A similar trend is observed on Cheetah-dir: under near-optimal data, it outperforms HarmoDT by 47.3% and Prompt-DT by 58.5% on average across the three data fractions. Notably, even in the extreme 1% regime, LaMA-DT maintains a clear margin over all baselines, highlighting its robustness under severe data scarcity.

Robustness to Data Quality (Q3). Table 2 shows that LaMA-DT consistently maintains a clear performance advantage over competing methods under both *sub-optimal* and *near-optimal* datasets. Notably, LaMA-DT substantially outperforms DT-based baselines in the *sub-optimal* setting, demonstrating strong robustness to imperfect offline data. As data quality improves, LaMA-DT exhibits larger absolute performance gains than competing approaches: on Meta-World ML30 at the 5% data level, performance increases from 39.1 (sub-optimal) to 52.6 (near-optimal), a gain that is significantly larger than those achieved by Prompt-DT or HarmoDT. These results indicate that LaMA-DT can handle low-quality data effectively, and scale favorably with improved data quality, suggesting the augmented data is of high quality and successfully mitigates the data imperfection issues.

Effect of Augmentation Ratio (Q4). We examine how the augmentation ratio $r = |\mathcal{D}_i^{\text{aug}}|/|\mathcal{D}_i|$ affects LaMA-DT performance under different data quantities (low-data: 1%, 5%, 10%; standard-data: 50%, 80%, 100%) and data qualities (sub-optimal vs. near-optimal). In low-data regimes (Fig. 2(a,c)), increasing r substantially improves performance, confirming that trajectory augmentation effectively mitigates data scarcity. However, the gains are non-monotonic: overly large ratios (*e.g.*, $r = 1.5$) often lead to saturation or degradation, suggesting that excessive augmented data may introduce distribution shift and error accumulation. In standard-data regimes (Fig. 2(b,d)), the optimal ratio shifts to smaller values (typically $r \approx 0.5$), as abundant real data already provides strong supervision. Across all subfigures, LaMA-DT trained on near-optimal data (solid lines) exhibit more stable improvements than those trained on sub-optimal data (dashed lines), indicating that augmentation cannot fully compensate for low-quality data and may amplify suboptimal patterns when overused.

Ablation Studies (Q5). As shown in Table 3, removing any component results in performance degradation. Among single-objective ablations, O_3 contributes the most, confirming the importance of long-horizon trajectory structure. More importantly, group ablations reveal strong synergy: jointly removing the Sequential Decision Learning objectives (w/o O_2, O_3, and O_4) results in the largest performance drop across all benchmarks (50.2 vs.

Table 3: Ablation studies on LaMA-DT.

Task	w/o O_1	w/o O_2	w/o O_3	w/o O_4	w/o O_5	w/o O_6
ML30	61.7±1.3	60.5±2.1	56.4±2.8	66.3±1.6	63.8±2.0	65.0±2.3
Cheetah-dir	842.3±14.3	862.4±10.2	765.9±11.7	910.2±16.8	902.6±15.4	922.5±26.3
Ant-dir	259.0±7.0	281.3±8.5	247.3±9.6	310.8±12.3	313.9±10.4	322.4±16.2

Task	w/o O_5&O_6	w/o O_2-O_4	w/o $\theta^{\text{sh}}_{\text{LoRA}}$	w/o Filtering	**LaMA-DT**
ML30	57.8±2.0	50.2±2.7	63.1±1.4	62.1±3.4	**69.8±1.7**
Cheetah-dir	835.1±13.5	636.9±22.1	878.0±8.4	883.1±15.9	**937.8±9.6**
Ant-dir	274.7±8.9	216.6±8.7	313.2±9.8	285.8±12.6	**328.5±8.4**

Table 4: Quality of augmented data on Cheetah-dir. We compare against real expert data (reference) and MTDIFF (He et al., 2023) as a generator baseline. Lower values are better for Dyn. MSE, State KL, and L2 Dist., while higher values are better for Valid Coverage. Criteria details are in Appendix E.6.

Source	Dyn. MSE↓	State KL vs. real↓	L2 Dist. from Dataset	Valid Coverage↑
Real expert (reference)	0.0000	0.00	—	100%
MTDIFF (He et al., 2023)	0.0107	0.51	0.4552	102%
LaMA-DT (unfiltered)	0.0089	0.42	0.4831	99%
LaMA-DT (filtered)	**0.0031**	**0.18**	**0.4376**	**109%**

69.8 on ML30; 636.9 vs. 937.8 on Cheetah-dir), substantially greater than any single-objective ablation. This highlights the effectiveness of trajectory-level supervision as a unified group. Similarly, removing both task-level objectives (w/o O_5&O_6) causes a sharper drop than either alone, showing their complementary role in mitigating task drift.

Augmentation Quality (Q6). We assess augmentation quality on Cheetah-dir using four criteria reported in Tab. 4: Dyn. MSE for dynamics fidelity; State KL for distributional alignment with real data; L2 Dist. (to real states) for training-set proximity; and Valid Coverage, which measures the fraction of physically valid generated states relative to real states (values above 100% indicate out-of-distribution yet effective exploratory states validated by the environment dynamics). Against MTDIFF (He et al., 2023), our filtered augmentations achieve $\sim 3.5\times$ lower Dyn. MSE and $\sim 2.8\times$ lower State KL, indicating stronger physical faithfulness and tighter distributional alignment. Crucially, the filter in LaMA-DT does not simply prune novel states: comparing unfiltered to filtered, it lowers Dyn. MSE ($0.0089 \rightarrow 0.0031$), halves State KL ($0.42 \rightarrow 0.18$), *and* raises Valid Coverage from 99% to 109%, indicating selectively removing off-manifold rollouts while retaining 9% of dynamics-validated novel states beyond the offline data. Fig. 3 corroborates this: the unfiltered panel scatters exploration crosses far outside the 90% real-data envelope, while the filtered panel keeps the in-distribution cloud tightly overlapping the real clusters and an exploration set that hugs the envelope from just outside. These quality gains translate to clear downstream policy improvements over the unfiltered variant (+54.7 on Cheetah-dir) shown in Table 3.

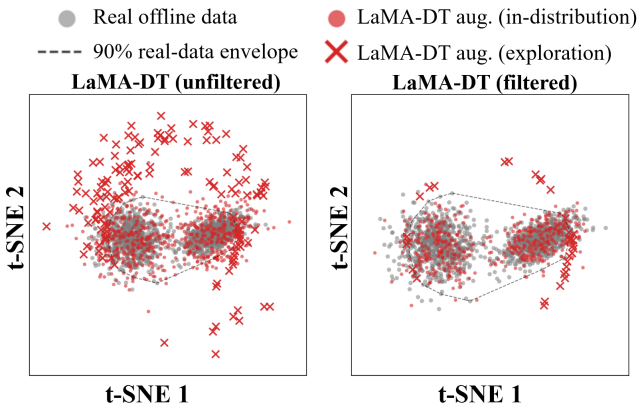


Figure 3: t-SNE projection of Cheetah-dir states: **unfiltered** (left) vs. **filtered** (right). Gray dots are real states from offline data, red dots are augmented states overlapping the real manifold (in-distribution), red crosses are augmented states lying outside the dataset for exploration, and the dashed contour is the 90% real-data envelope.

5 Conclusion

We presented LaMA-DT, an LLM-based offline MTRL framework designed to address data scarcity and cross-task generalization. LaMA-DT leverages specialized fine-tuning objectives to enable a single LLM to generate high-fidelity, task-aligned trajectories for augmentation. The same LLM is then repurposed as a DT and fine-tuned on both augmented and real trajectories. Experiments demonstrate that LaMA-DT consistently outperforms state-of-the-art offline MTRL baselines, highlighting the effectiveness of LLM-driven trajectory augmentation and policy learning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting and polishing the manuscript. As part of the proposed LaMA-DT method, a pretrained language model generates task-conditioned trajectories for offline data augmentation and is reused for policy learning, as described in Sections 3.2–3.3 and Appendices B and E. The authors reviewed and revised the AI-assisted writing. We take responsibility for the final content of this work, including its text, claims, and AI-generated artifacts.

REPRODUCIBILITY STATEMENT

We provide pseudocode for the three stages of LaMA-DT in Appendix B, benchmark and dataset details in Appendix C, baseline descriptions in Appendix D, and implementation settings, hyperparameters, and evaluation protocols in Appendix E. An anonymized code repository is available at `https://anonymous.4open.science/r/o33zlamadt-6587`.

REFERENCES

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.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.*, 6:503–556, dec 2005. ISSN 1532-4435.

Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse RL: hindsight inference for policy improvement. *CoRR*, abs/2002.11089, 2020. URL `https://arxiv.org/abs/2002.11089`.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 1126–1135, 2017.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *CoRR*, abs/1812.02900, 2018. URL `http://arxiv.org/abs/1812.02900`.

Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *ArXiv*, abs/2111.10364, 2021. URL `https://api.semanticscholar.org/CorpusID:244463196`.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36:64896–64917, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*. OpenReview.net, 2022.

Shengchao Hu, Ziqing Fan, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. In *International Conference on Machine Learning*, 2024.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR*, abs/1907.00456, 2019. URL `http://arxiv.org/abs/1907.00456`.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018. URL http://arxiv.org/abs/1806.10293.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *CoRR*, abs/2104.08212, 2021. URL https://arxiv.org/abs/2104.08212.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Yilun Kong, Guozheng Ma, Qi Zhao, Haoyu Wang, Li Shen, Xueqian Wang, and Dacheng Tao. Mastering massive multi-task reinforcement learning via mixture-of-expert decision transformer. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021a. URL https://arxiv.org/abs/2110.06169.

Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *CoRR*, abs/2103.08050, 2021b. URL https://arxiv.org/abs/2103.08050.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *CoRR*, abs/1906.00949, 2019. URL http://arxiv.org/abs/1906.00949.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pp. 45–73. Springer, 2012.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020. URL https://api.semanticscholar.org/CorpusID:218486979.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021.

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*, pp. 7780–7791. PMLR, 2021.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019. URL http://arxiv.org/abs/1910.00177.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. *CoRR*, abs/2012.11547, 2020. URL https://arxiv.org/abs/2012.11547.

Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon S. Du, and Huazhe Xu. Unleashing the power of pre-trained language models for offline reinforcement learning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin A. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *CoRR*, abs/2002.08396, 2020. URL https://arxiv.org/abs/2002.08396.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. *CoRR*, abs/1807.01473, 2018. URL http://arxiv.org/abs/1807.01473.

Shudong Wang, Xinfei Wang, Chenhao Zhang, Shanchen Pang, Haiyuan Gui, Wenhao Ji, and Xiaojian Liao. Soft conflict-resolution decision transformer for offline multi-task reinforcement learning. *arXiv preprint arXiv:2511.13133*, 2025.

Zhi Wang, Li Zhang, Wenhao Wu, Yuanheng Zhu, Dongbin Zhao, and Chunlin Chen. Meta-dt: offline meta-rl as conditional sequence modeling with world model disentanglement. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *CoRR*, abs/1911.11361, 2019. URL http://arxiv.org/abs/1911.11361.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 24631–24645. PMLR, 2022.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 2402–2413, Red Hook, NY, USA, 2018. Curran Associates Inc.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Yu Yang and Pan Xu. Pre-trained language models improve the few-shot prompt ability of decision transformer. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=k520i3XEMK.

Minjong Yoo, SangWoo Cho, and Honguk Woo. Skills regularized task decomposition for multi-task offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020a.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020b.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142. Curran Associates, Inc., 2020c.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:11501–11516, 2021.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. *CoRR*, abs/2202.01741, 2022. URL https://arxiv.org/abs/2202.01741.

Shilin Zhang, Zican Hu, Wenhao Wu, Xinyi Xie, Jianxiang Tang, Chunlin Chen, Daoyi Dong, Yu Cheng, Zhenhong Sun, and Zhi Wang. Text-to-decision agent: Offline meta-reinforcement learning from natural language supervision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Hongling Zheng, Li Shen, Yong Luo, Tongliang Liu, Jialie Shen, and Dacheng Tao. Decomposed prompt decision transformer for efficient unseen task generalization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

A Related Work

Offline RL. Offline reinforcement learning learns policies from fixed datasets without additional environment interaction (Levine et al., 2020; Ernst et al., 2005; Lange et al., 2012), and has shown promise in robotics (Kalashnikov et al., 2018; Rafailov et al., 2020), NLP (Jaques et al., 2019), and healthcare (Wang et al., 2018). A central challenge is *distribution shift* between the learned policy and the behavior policy, which can lead to erroneous value backups and unstable learning (Fujimoto et al., 2018; Levine et al., 2020). To address this, prior work constrains the policy to stay close to the behavior distribution (Kumar et al., 2019; Fujimoto et al., 2018; Wu et al., 2019; Jaques et al., 2019; Peng et al., 2019; Siegel et al., 2020) or regularizes the learned value/Q functions (Kumar et al., 2020; Kostrikov et al., 2021b), while model-based approaches build conservative dynamics/reward models to reduce out-of-distribution errors (Kidambi et al., 2020; Yu et al., 2020c). In addition, Decision Transformer (Chen et al., 2021) casts offline RL as conditional sequence modeling.

Multi-task RL and offline MTRL. Multi-task RL studies learning a shared policy across tasks in RL settings. A key obstacle is negative transfer caused by conflicting gradients across tasks, motivating gradient-surgery or conflict-aware optimization methods (Yu et al., 2020a; Chen et al., 2020; Liu et al., 2021). Multi-task offline RL aims to learn a shared policy from offline datasets collected from a set of tasks. A common formulation is to train a task-conditioned policy under a unified objective, but such shared training often suffers from negative transfer caused by conflicting gradients across tasks and heterogeneous data sources. To mitigate this issue, prior work has explored selective sharing and conflict-aware optimization mechanisms. HarmoDT (Hu et al., 2024) addresses this challenge from a parameter-selection perspective by learning task-specific binary masks over shared parameters, enabling conflict-aware parameter sharing in multi-task offline Decision Transformer training. Complementary to gradient conflict handling, data sharing methods selectively reuse transitions across tasks while remaining conservative under distribution shift, such as CDS (Yu et al., 2021) and related value-based sharing approaches (Eysenbach et al., 2020; Kalashnikov et al., 2021; Yu et al., 2022). More recently, M3DT (Kong et al., 2025) introduces a Mixture-of-Experts Decision Transformer to scale offline MTRL to massive task suites by decoupling task load across expert subsets. However, M3DT targets *task scalability under full per-task datasets*, an axis orthogonal to our focus on *per-task data scarcity*: LaMA-DT addresses scarcity through language-guided trajectory augmentation rather than through architectural expansion. In contrast to these baselines, LaMA-DT aligns text and behavior, and then uses the resulting objectives to score and filter generated trajectories before decision-transformer policy learning, leading to more robust multi-task offline RL.

Offline Meta-RL. Offline meta-RL aims to learn policies that generalize to unseen tasks from fixed offline datasets, without any online interaction. A large body of prior work follows the temporal-difference (TD) paradigm, using either optimization-based meta-learning to find fast-adaptable initializations (Finn et al., 2017; Mitchell et al., 2021; Xu et al., 2018). However, TD-based offline meta-RL can be prone to instability under off-policy bootstrapping (Levine et al., 2020), motivating alternative approaches that avoid value backups. Along this line, decision-transformer-style methods treat offline meta-RL as conditional sequence modeling over trajectories (Chen et al., 2021). Prompt-based approaches such as Prompt-DT (Xu et al., 2022) and DPDT (Zheng et al., 2024) adapt to new tasks using soft or hand-crafted prompts, while GDT (Furuta et al., 2021) leverages hindsight rewards as an adaptation signal. Meta-DT (Wang et al., 2024) further introduces a meta-policy to select informative trajectories for adaptation. Two contemporaneous approaches further inject language signals into this paradigm: LPDT (Yang & Xu, 2025) initializes Prompt-DT with a pre-trained language model and adds prompt regularization to improve few-shot adaptation, while T2DA (Zhang et al., 2025) learns CLIP-style contrastive alignment between task descriptions and trajectory embeddings to enable zero-shot transfer. Both methods leverage language to ground task semantics, but they differ from LaMA-DT in two important ways: (i) neither performs *trajectory-level data augmentation*—LPDT relies on LM initialization while T2DA relies on contrastive alignment—and (ii) both are evaluated primarily under the meta-RL zero/few-shot transfer protocol, whereas LaMA-DT addresses the multi-task offline setting under varying per-task data fractions. We accordingly compare against multi-task offline RL baselines that share our evaluation protocol, and view direct comparison with LPDT and T2DA on their meta-RL setups as a complementary direction. Our work builds on the sequence-modeling perspective, but explicitly leverages natural language task descriptions to ground task semantics in behavioral data, also enabling generalization ability in offline meta-RL settings.

LLM-based offline RL. A complementary line of work directly leverages pre-trained language models for offline RL. LaMo (Shi et al., 2024) initializes the DT backbone with a frozen pre-trained language model and fine-tunes via LoRA on a small fraction of parameters, demonstrating substantial gains in *single-task* offline RL—especially under sparse rewards and limited data. LaMA-DT extends this direction along two axes: (i) we operate in the multi-task offline setting rather than per-task training, requiring explicit mechanisms to disambiguate similar state-action distributions across tasks; and (ii) we reuse the same LM not only as the policy backbone but also as a *trajectory generator* that produces task-aligned augmented data via six supervised objectives, addressing the data scarcity bottleneck that initialization alone cannot resolve.

B Detailed Training Algorithm

We give the full procedure of LaMA-DT in two complementary forms. Algorithm 1 provides a compact high-level overview of the three stages, and Algorithms 2–4 restate each stage with the full set of algorithmic details (sampling, adapter activation, loss instantiation), following the same notation as Sec. 3.2–Sec. 3.3. We use θ_{base} for the frozen LLM weights, $\theta_{\text{LoRA}}^{\text{O}_j}$ for the objective-specific LoRA adapter of objective O_j, $\theta_{\text{LoRA}}^{\text{sh}}$ for the shared LoRA adapter, and $\theta_{\text{LoRA}}^{\pi}$ for the policy LoRA used at Stage III. We write $f_{\boldsymbol{\theta}}(\cdot)$ for a forward pass with the active LoRA set $\boldsymbol{\theta}$ added on top of θ_{base}.

Algorithm 1 LaMA-DT Training (Compact Overview)

Input: $\{(\mathbf{p}_i, \mathcal{D}_i)\}_{i=1}^{M}$; frozen base LLM params $\boldsymbol{\theta}_{\text{base}}$.
Output: Unified multi-task policy $\pi_{\boldsymbol{\theta}}$.
// Stage I: Multi-objective LLM fine-tuning (O_1–O_6)
Initialize objective adapters $\{\boldsymbol{\theta}_{\text{LoRA}}^{\text{O}_j}\}_{j=1}^{6}$ and shared adapter $\boldsymbol{\theta}_{\text{LoRA}}^{\text{sh}}$; freeze $\boldsymbol{\theta}_{\text{base}}$.
while not converged **do**
  Sample task i and a batch $\mathcal{B} \subset \mathcal{D}_i$; sample O_j.
  Update $\boldsymbol{\theta}_{\text{LoRA}}^{\text{O}_j}$ and $\boldsymbol{\theta}_{\text{LoRA}}^{\text{sh}}$ with the loss of O_j.
end while
// Stage II: Data preparation (O_3 generation + O_1–O_6 scoring)
for each task $i = 1$ to M **do**
  Initialize $\mathcal{D}_i^{\text{aug}} \leftarrow \emptyset$.
  for each sampled $(s_0, a_0) \sim \mathcal{D}_i$ **do**
    Generate $\{\tilde{\tau}^k\}_{k=1}^{N}$ using O_3.
    Score each $\tilde{\tau}^k$ by $S(\tilde{\tau}^k) = \frac{1}{6}\sum_{j=1}^{6} S_{\text{O}_j}(\tilde{\tau}^k)$.
    Add the top-K candidates $(\tilde{\tau}, \mathbf{p}_i, z_{\text{prov}} = \texttt{augmented})$ to $\mathcal{D}_i^{\text{aug}}$.
  end for
end for
// Stage III: Policy learning (LaMA-DT)
Initialize policy adapter $\boldsymbol{\theta}_{\text{LoRA}}^{\pi} \leftarrow \boldsymbol{\theta}_{\text{LoRA}}^{\text{sh}}$; freeze $\boldsymbol{\theta}_{\text{base}}$.
Train LaMA-DT on $\{\mathcal{D}_i \cup \mathcal{D}_i^{\text{aug}}\}$ using Eq. 5.
return $\pi_{\boldsymbol{\theta}}$.

Trajectory serialization. For each task $\mathcal{T}_i$, every trajectory $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T}$ is mapped into the LLM hidden space via three numeric MLPs (one each for s, a, the per-step return-to-go $\hat{R}_t = \sum_{t' \geq t} r_{t'}$). The natural-language task description $\mathbf{p}_i$ is tokenized and embedded by the frozen LLM tokenizer; for Stage III an extra provenance marker $z_{\text{prov}} \in \{\texttt{real}, \texttt{augmented}\}$ is appended to the text prefix. The full input is

$$[\,\texttt{Emb}(\mathbf{p}_i,\, z_{\text{prov}}) \,\|\, \hat{R}_0,\, s_0,\, a_0,\, \hat{R}_1,\, s_1,\, a_1,\, \ldots\,]$$

and is processed by $f_{\boldsymbol{\theta}}$ as a single sequence; the numeric heads read out the corresponding per-step continuous predictions.

B.1 Stage I: Multi-Objective LoRA Fine-Tuning

Algorithm 2 Stage I: Multi-Objective LoRA Fine-Tuning

Input: task pool $\{(\mathbf{p}_i, \mathcal{D}_i)\}_{i=1}^{M}$; frozen θ_{base}; iteration budget T_{I}; window length L.
Initialize objective adapters $\{\theta_{\text{LoRA}}^{\text{O}_j}\}_{j=1}^{6}$ and shared adapter $\theta_{\text{LoRA}}^{\text{sh}}$.
for step $= 1, \ldots, T_{\text{I}}$ **do**
  Sample task $i \sim \text{Uniform}(\{1, \ldots, M\})$ and trajectory $\tau \sim \mathcal{D}_i$.
  Sample window of length L from τ and an objective $j \sim \text{Uniform}(\{1, \ldots, 6\})$.
  Construct (x_j, y_j) from $(\mathbf{p}_i, \tau_{\text{window}})$ as in Sec. 3.2.1–3.2.2.
  Activate the active set $\boldsymbol{\theta}_j = \theta_{\text{base}} + \theta_{\text{LoRA}}^{\text{O}_j} + \theta_{\text{LoRA}}^{\text{sh}}$.
  Compute the loss $\mathcal{L}_{\text{O}_j}\big(f_{\boldsymbol{\theta}_j}(x_j),\, y_j\big)$ (MSE for $j \in \{1, 2, 3, 4, 6\}$; CE for $j = 5$).
  Update only $\theta_{\text{LoRA}}^{\text{O}_j}$ and $\theta_{\text{LoRA}}^{\text{sh}}$ by AdamW; θ_{base} stays frozen.
end for
Output: adapters $\{\theta_{\text{LoRA}}^{\text{O}_j}\}_{j=1}^{6}$ and shared adapter $\theta_{\text{LoRA}}^{\text{sh}}$.

B.2 Stage II: Augmented Trajectory Generation, Scoring, and Filtering

Algorithm 3 Stage II: Augmented Trajectory Generation, Scoring, and Filtering

Input: adapters from Stage I; $\{\mathcal{D}_i\}_{i=1}^{M}$; #candidates N, top-K, sampling temperature $\mathcal{T}$.
Initialize $\mathcal{D}_i^{\text{aug}} \leftarrow \emptyset$ for all i.
for each task $i = 1, \ldots, M$ **do**
  for each sampled $(s_0, a_0) \in \mathcal{D}_i$ **do**
    Activate $\theta_{\text{base}} + \theta_{\text{LoRA}}^{\text{O}_3} + \theta_{\text{LoRA}}^{\text{sh}}$ and autoregressively roll out N candidate trajectories at temperature $\mathcal{T}$:
      $\{\tilde{\tau}^k\}_{k=1}^{N} \sim p_{\text{O}_3}(\tau \mid \mathbf{p}_i,\, (s_0, a_0))$.
    for $k = 1, \ldots, N$ **do**
      for $j = 1, \ldots, 6$ **do**
        Activate $\theta_{\text{base}} + \theta_{\text{LoRA}}^{\text{O}_j}$ and compute the length-normalized teacher-forcing log-likelihood $S_{\text{O}_j}(\tilde{\tau}^k)$ from Sec. 3.3.
      end for
      Aggregate score $S(\tilde{\tau}^k) \triangleq \frac{1}{6}\sum_{j=1}^{6} S_{\text{O}_j}(\tilde{\tau}^k)$.
    end for
    Add the top-K candidates by S, each tagged with $z_{\text{prov}} = \texttt{augmented}$, to $\mathcal{D}_i^{\text{aug}}$.
  end for
end for
Output: augmented per-task datasets $\{\mathcal{D}_i^{\text{aug}}\}_{i=1}^{M}$.

Stage II scoring as a budget-cheap reranker. Generation calls $\theta_{\text{LoRA}}^{\text{O}_3}$ once per candidate; scoring then loops over the six objective adapters, but each scoring pass is forward-only, length-normalized, and shares the same input encoding. Because filtering keeps top-K candidates per (s_0, a_0) rather than per task, the augmentation budget grows linearly with the number of seed pairs and is independent of M.

B.3 Stage III: Multi-Task Policy Learning

Algorithm 4 Stage III: Multi-Task Policy Learning

1: **Input:** $\{\mathcal{D}_i \cup \mathcal{D}_i^{\text{aug}}\}_{i=1}^{M}$; shared adapter $\theta_{\text{LoRA}}^{\text{sh}}$ from Stage I; iteration budget T_{III}.
2: Initialize the policy adapter from the shared adapter: $\theta_{\text{LoRA}}^{\pi} \leftarrow \theta_{\text{LoRA}}^{\text{sh}}$; freeze θ_{base}.
3: **for** step $= 1, \ldots, T_{\text{III}}$ **do**
4: Sample task i (uniform over tasks; optional balanced re-weighting); sample $\tau \sim \mathcal{D}_i \cup \mathcal{D}_i^{\text{aug}}$ with augmentation ratio r.
5: Sample a window of length L from τ and form the DT-style input with $\mathbf{p}_i$ and $z_{\text{prov}} \in \{\texttt{real}, \texttt{augmented}\}$.
6: Predict $\hat{a}_t = f_{\theta_{\text{base}}+\theta_{\text{LoRA}}^{\pi}}(\tau_{0:t-1}, \mathbf{p}_i, z_{\text{prov}})$ for every action position.
7: Update $\theta_{\text{LoRA}}^{\pi}$ (and the numeric heads) by AdamW on $\mathcal{L}_{\text{LaMA-DT}}$ in Eq. 5; θ_{base} stays frozen.
8: **end for**
9: **Deployment:** given $\mathbf{p}_i$, set $z_{\text{prov}} = \texttt{real}$ and roll out autoregressively; update $\hat{R}_{t+1} = \hat{R}_t - r_t$.
10: **Output:** unified multi-task policy $\pi_{\theta_{\text{base}}+\theta_{\text{LoRA}}^{\pi}}$.

B.4 Per-Objective Loss Instantiation

Stage I uses a Gaussian log-likelihood (equivalently, length-normalized MSE) for the five numeric objectives and a token-level cross-entropy for the text-output objective O_5:

$$\mathcal{L}_{\text{O}_j}(\boldsymbol{\theta}_j) = \frac{1}{|y_j|} \sum_{t=1}^{|y_j|} \left\| y_{j,t} - \mu_{j,t}(x_j; \boldsymbol{\theta}_j) \right\|_2^2 \qquad j \in \{1, 2, 3, 4, 6\}, \tag{6}$$

$$\mathcal{L}_{\text{O}_5}(\boldsymbol{\theta}_5) = -\frac{1}{|y_5|} \sum_{t=1}^{|y_5|} \log p_{\boldsymbol{\theta}_5}(y_{5,t} \mid x_5,\, y_{5,<t}) \,. \tag{7}$$

The same (x_j, y_j) construction is reused at scoring time (Algorithm 3, line 9), so the resulting $S_{\text{O}_j}(\tilde{\tau})$ are interpretable as conditional log-likelihoods evaluated on the candidate.

B.5 Few-Shot Adaptation to Unseen Tasks

For unseen-task generalization we treat the small support set $\mathcal{D}_i^{\text{few}}$ as a one-task version of the Stage I data, and run a three-step adaptation procedure that exactly mirrors the standard Stage I/II/III pipeline.

(i) Lightweight objective adaptation. Starting from the Stage I adapters, we run a short adapt phase that updates the objective-specific adapters $\{\theta_{\text{LoRA}}^{\text{O}_j}\}_{j=1}^{6}$ together with the shared adapter $\theta_{\text{LoRA}}^{\text{sh}}$ on $\mathcal{D}_i^{\text{few}}$, while keeping the LLM backbone θ_{base} frozen. Each objective head therefore better reflects the dynamics, rewards, and semantics of the new task i.

(ii) Multi-objective generation and filtering. We then use the adapted O_3 adapter to generate synthetic rollouts. For each initial state–action pair (s_0, a_0) sampled from $\mathcal{D}_i^{\text{few}}$, we draw N candidate trajectories $\{\tilde{\tau}^{(m)}\}_{m=1}^{N}$ by autoregressively rolling out the policy conditioned on the task description. Each candidate is scored under all six adapted objectives (O_1–O_6) and ranked by the aggregate score $S(\tilde{\tau}^{(m)})$; we retain the top-K candidates per seed pair to form a high-quality synthetic set $\mathcal{D}_i^{\text{syn}}$.

(iii) Policy learning on the union. Finally, we run Stage III on the union $\mathcal{D}_i^{\text{few}} \cup \mathcal{D}_i^{\text{syn}}$, exactly as in Algorithm 4: we initialize the policy adapter $\theta_{\text{LoRA}}^{\pi} \leftarrow \theta_{\text{LoRA}}^{\text{sh}}$ (using the just-adapted shared adapter), and update $\theta_{\text{LoRA}}^{\pi}$ on this union with θ_{base} frozen. This yields an adapted policy for task i that has never interacted with the environment online; we evaluate it directly under the standard 50-episode protocol.

C Environment Details

C.1 Meta-World

The Meta-World benchmark (Yu et al., 2020b) is a comprehensive suite of robotic manipulation tasks designed to evaluate generalization and multi-task learning in continuous control. It involves a Sawyer robotic arm performing 50 distinct manipulation tasks (MT50), each characterized by unique object geometries, joint configurations, and goal dynamics. The robot operates in a 4-dimensional fine-grained continuous action space, controlling the 3D end-effector position and gripper openness. Following recent advances in multi-task RL (Yang et al., 2020; Hu et al., 2024), we employ the random-goal variant (MT50-rand), where goal positions are randomized per episode to promote adaptability and discourage overfitting to fixed targets. In our experiments, we focus on a subset of 30 tasks (ML30) to balance computational efficiency with comprehensive evaluation across diverse manipulation skills. The primary evaluation metric is the average success rate across tasks, reflecting the robot's capacity to generalize under varied and unseen goal configurations.

For offline data collection, we adopt the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018) to train individual policies for each task until convergence. For every task, we record 1M transitions during training, resulting in a comprehensive offline dataset composed of both successful and failed attempts. Based on the convergence stage, we curate two datasets matching the main-text definitions:

- **Near-Optimal (NO):** the complete 1M transitions per task collected from the converged SAC policy, dominated by expert behavior.
- **Sub-Optimal (SO):** a 1M-transition mixture of (i) 0.5M expert transitions sampled from the converged SAC policy and (ii) 0.5M non-expert transitions collected from a mid-performance SAC policy, yielding a 50% expert / 50% non-expert composition consistent with the main text.

These datasets jointly provide a rich distribution of state–action pairs spanning skill acquisition stages, supporting both imitation and reinforcement learning settings under low-data constraints.

C.2 MuJoCo

We further evaluate on MuJoCo continuous-control benchmarks (Todorov et al., 2012) that require generalization across goal-conditioned tasks, including Cheetah-dir, Cheetah-vel, and Ant-dir.

Dataset construction. For MuJoCo tasks, we follow the standard offline data generation protocol in prior offline RL work (Mitchell et al., 2021), where trajectories are collected by training a policy with SAC (Haarnoja et al., 2018) and applying action penalties to discourage excessively large control inputs. We consider two dataset expertise levels matching the main-text definitions:

- **Near-Optimal (NO):** trajectories generated by a converged expert SAC/TD3 policy, dominated by expert behavior.
- **Sub-Optimal (SO):** a mixture of (i) 50% trajectories sampled from the NO dataset and (ii) 50% trajectories collected from mid-performance SAC policies, resulting in a 50% expert / 50% non-expert composition with comparable data volume.

These two regimes allow us to systematically evaluate robustness to trajectory quality while keeping the overall data protocol consistent.

Task families. We evaluate across three MuJoCo benchmarks:

- **Cheetah-dir:** a HalfCheetah agent is conditioned on a discrete direction variable indicating whether it should run forward or backward, with a reward encouraging movement along the specified direction. In the *multi-task offline RL* setting, training and evaluation are performed on the two direction tasks. In the *meta offline RL* setting, the agent is trained on one direction and evaluated on the held-out opposite direction.
- **Cheetah-vel:** the agent is conditioned on a target forward velocity and receives a velocity-tracking reward that penalizes deviation from the specified speed. In the *multi-task offline*

RL setting, the agent is trained and evaluated on all 40 velocity tasks. In the *meta offline RL* setting, training is conducted on 35 target velocities and evaluation is performed on the remaining 5 unseen velocities.

- **Ant-dir:** the Ant agent is conditioned on a target movement direction in the 2D plane and rewarded for moving toward that direction. In the *multi-task offline RL* setting, all 50 direction tasks are used for both training and evaluation. In the *meta offline RL* setting, the agent is trained on 45 directions and evaluated on 5 held-out unseen directions.

Overall, these MuJoCo benchmarks complement Meta-World by emphasizing locomotion control and goal-conditioned dynamics, enabling a comprehensive evaluation of LaMA-DT across heterogeneous offline control domains under both NO and SO dataset expertise.

D BASELINES

We compare **LaMA-DT** against representative *multi-task offline RL* baselines spanning value-based, behavior-cloning, generative, and Decision-Transformer paradigms. Across all baselines, we use the same task splits and dataset preprocessing as in our main experiments, and we follow the original papers' recommended architectures and hyperparameters whenever available.

CDS (Conservative Data Sharing) (Yu et al., 2021). CDS is a representative *data sharing* baseline for multi-task offline RL. Its key idea is to selectively share transitions across tasks while remaining conservative under distribution shift. In our implementation, we train a single task-conditioned actor $\pi_\theta(a \mid s, i)$ that takes the state s and a task identifier i as input, and outputs the action.

MTBC (Multi-Task Behavior Cloning). We extend behavior cloning to multi-task offline policy learning by using a task-ID conditioned actor that learns a single policy across all tasks, minimizing the MSE for continuous actions.

MTIQL (Multi-Task Implicit Q-Learning). IQL (Kostrikov et al., 2021a) learns the value function via expectile regression and trains the actor via advantage-weighted regression, which makes it effective for offline settings without behavior policy constraints as strict as CQL-style methods. MTIQL extends IQL to multi-task learning by using a task-ID conditioned actor and multi-head critic networks.

MTDIFF (He et al., 2023). MTDIFF is a diffusion-based approach for multi-task RL that uses conditional generative modeling for planning and data synthesis. We use MTDIFF-P as our baseline.

MTDT (Multi-Task Decision Transformer) (Chen et al., 2021). MTDT extends Decision Transformer to multi-task data by conditioning the Transformer policy on a task identifier embedding. Trajectories are serialized into token sequences and the model is trained with an autoregressive action prediction objective. At evaluation, MTDT is provided with the task ID and desired return-to-go to generate task-specific actions.

Prompt-DT (PDT) (Xu et al., 2022). Prompt-DT conditions action generation on *trajectory prompts* rather than explicit task IDs. A small set of prompt trajectories is prepended to the input sequence, serving as in-context demonstrations of task behavior. The model then predicts actions for the query trajectory conditioned on both the prompt tokens and return-to-go.

HarmoDT (Hu et al., 2024). HarmoDT is a prompt-based Decision Transformer baseline for multi-task offline RL that mitigates *gradient conflicts* by learning *task-specific parameter masks*, enabling conflict-aware sharing across tasks. We use **HarmoDT-F** as the HarmoDT baseline.

Low-data setting. To evaluate data efficiency, we construct low-data regimes by subsampling each task's offline dataset to a fixed percentage of the original size (e.g., 1%, 5%, 10%). Crucially, we preserve trajectory boundaries during subsampling: instead of randomly shuffling individual transitions across episodes, we sample *complete trajectories* from the original dataset. This ensures that all sequence-modeling baselines receive consistent temporal structure, and that our six objectives

can be trained properly. While the subsampling percentage is defined with respect to the *number of transitions*, the actual sampling unit remains a *full trajectory* (i.e., we include all transitions within each selected episode). For each task, we randomly sample trajectories without replacement until the target percentage is reached, maintaining the same trajectory-level statistics as the full dataset, only scaled down in volume.

E Implementation Details and Hyper-parameters

We document the implementation that produced the headline results in Tab. 1–Tab. 2. Wherever an implementation choice could deviate from the description in Sec. 3.2–Sec. 3.3, we follow the main paper. Anything below should therefore be read as concrete realizations of the abstractions in the method, not as additional design.

E.1 Backbone, LoRA, and Numeric Tokenization

Backbone. The frozen LLM is Llama-3.2-1B with 16 transformer blocks, hidden size 2048, 32 attention heads, and SiLU activations in the feedforward blocks. All base parameters have `requires_grad=False` throughout.

LoRA placement. Each LoRA adapter is attached only to the attention output projection (`o_proj`) and the MLP down-projection (`down_proj`) of every transformer block, with rank $r = 16$, scaling $\alpha = 32$ (effective $\alpha/r = 2$), and dropout 0.05 for Cheetah-dir / Cheetah-vel / Ant-dir, 0.10 for Meta-World ML30. This placement keeps the trainable footprint small while still injecting capacity into both the attention and MLP paths.[2]

Numeric tokenization. Continuous states, actions, and per-step return-to-go are mapped into the LLM hidden space by three independent one-layer MLPs:

$$\begin{aligned}\texttt{state_mlp}(s) &= \tanh(W_s s + b_s),\\ \texttt{action_mlp}(a) &= \tanh(W_a a + b_a),\\ \texttt{return_mlp}(\hat{R}) &= \tanh(W_r \hat{R} + b_r).\end{aligned}$$

each producing a 2048-dim token. Continuous predictions are read out by symmetric one-layer linear heads (`state_head`, `action_head`, `reward_head`) without an output non-linearity.

Sequence layout. A training input is $[\,\texttt{Emb}(\mathbf{p}_i, z_{\text{prov}});\ \hat{R}_0, s_0, a_0;\ \hat{R}_1, s_1, a_1;\ \dots\,]$ with a context window of $L = 20$ time steps (i.e., 60 numeric positions plus the text prefix). Trajectories shorter than L are right-padded by repeating the last state and zero-padding the action and return tokens. State and return-to-go are normalized per task using statistics computed on the real offline split.

E.2 Stage I Setup (Multi-Objective LoRA Fine-Tuning)

At every step, one objective $j \sim \text{Uniform}(\{1, \dots, 6\})$ and one task $i \sim \text{Uniform}(\{1, \dots, M\})$ are drawn; the corresponding (x_j, y_j) is constructed from a randomly sliced window. Only $\theta^{\text{O}_j}_{\text{LoRA}}$ and $\theta^{\text{sh}}_{\text{LoRA}}$ are updated. We use AdamW (learning rate 1×10^{-4}, weight decay 0.01), gradient accumulation over four micro-batches, gradient clipping at $\|\nabla\|_2 \le 1.0$, and a linear warmup (1,000 steps for MuJoCo, 2,000 for Meta-World) followed by a cosine decay. The total number of optimization steps is 50,000 for the MuJoCo configurations and 100,000 for Meta-World ML30. Stage I uses bf16 mixed precision with gradient checkpointing, and is run under DDP across two NVIDIA A6000 GPUs.

E.3 Stage II Setup (Augmentation Generation and Filtering)

We sample one initial (s_0, a_0) per real trajectory and generate $N = 4$ candidate rollouts under the O_3 adapter at sampling temperature $\mathcal{T} = 0.4$ (chosen by the ablation in Tab. 7). Long horizons

[2]The method only requires that $\theta^{\text{O}_j}_{\text{LoRA}}$ and $\theta^{\text{sh}}_{\text{LoRA}}$ are objective-specific / shared adapters; the choice of which two projections to attach LoRA to is an implementation detail and does not change Sec. 3.2.3.

are produced via chunked rollouts of length 50 that re-anchor on intermediate teacher-forced states; this preserves the seed and final state and reduces wall-clock time by roughly $3\times$ relative to fully autoregressive generation. Each candidate is then scored under all six objectives by reactivating the relevant LoRA and running a single forward teacher-forcing pass; per-objective scores are length-normalized and combined by an unweighted mean as in Sec. 3.3. We retain the top-1 candidate per seed pair (i.e., $K = 1$), tag it with $z_{\text{prov}} = \texttt{augmented}$, and write the resulting per-task augmented set to disk.

E.4 Stage III Setup (Multi-Task Policy Learning)

The policy adapter $\theta^{\pi}_{\text{LoRA}}$ is initialized from $\theta^{\text{sh}}_{\text{LoRA}}$; the numeric heads are initialized from scratch. Each batch mixes real and augmented trajectories at the augmentation ratio $r = |\mathcal{D}_i^{\text{aug}}|/|\mathcal{D}_i|$ (default $r = 0.5$, swept in Fig. 2). For Meta-World ML30 we additionally enable balanced sampling (a `WeightedRandomSampler` that gives every task equal frequency in expectation) to compensate for unequal per-task trajectory counts. The loss is the action MSE in Eq. 5; reward and state heads are not supervised at this stage. Optimization uses AdamW with learning rate 5×10^{-5} (MuJoCo) or 1×10^{-4} (Meta-World ML30), weight decay 0.01 (MuJoCo) or 1×10^{-4} (Meta-World), warmup over 2,000–3,000 steps, and a cosine decay over the full schedule of 100,000 (MuJoCo) and (Meta-World) steps. Gradient clipping is $\|\nabla\|_2 \leq 1.0$ for MuJoCo and the tighter $\|\nabla\|_2 \leq 0.25$ for Meta-World ML30, which we found necessary for stable late-stage optimization on the heterogeneous task suite. The training batch size is 32.

E.5 Evaluation Protocol

Final policies are evaluated for 50 episodes per task, averaged over three random seeds for Tab. 1 and Tab. 2. The provenance token is fixed to `real` at deployment. The return-to-go is initialized to the target value listed in Tab. 5 and decremented by the actual environment reward at each step. We report average normalized return for MuJoCo and per-task average success rate for Meta-World, following the conventions of prior prompt-based DT work (Xu et al., 2022; Hu et al., 2024).

Table 5: Per-environment evaluation settings used to obtain Tab. 1 and Tab. 2.

	Cheetah-dir	**Cheetah-vel**	**Ant-dir**	**Meta-World ML30**
Episodes / task	50	50	50	50
RTG target (`forward` task)	760	per-task ref. velocity	per-task ref. dir.	4,500
RTG normalization	yes	yes	yes	yes
Provenance at test time	`real`	`real`	`real`	`real`

E.6 Augmentation Quality Metrics: Definitions and Computation

We define the four columns of Tab. 4. Each trajectory is flattened into one-step transitions (s_t, a_t, s_{t+1}), and we work in z-score-normalized state space, with per-dimension mean and standard deviation estimated on $\mathcal{D}^{\text{real}}$ and applied to both sets.

Dyn. MSE (one-step dynamics fidelity, ↓). For every augmented transition we reset the MuJoCo simulator to s_t, apply a_t, and read off the true next state $f^\star(s_t, a_t)$. Dyn. MSE is the per-dimension squared error between the augmented next state and $f^\star(s_t, a_t)$ in normalized space, averaged over all transitions in $\mathcal{D}^{\text{aug}}$. Real data is generated by $f^\star$ itself, so its residual is floating-point noise (rounded to 0.0000).

State KL vs. real (↓). We estimate $\mathrm{KL}(P_{\text{aug}} \,\|\, P_{\text{real}})$ between the empirical state distributions using a non-parametric k-NN estimator with $k = 5$ and a `KDTree` per set, which avoids density assumptions in high-dimensional state space. Real-vs-real is 0 by construction.

L2 Dist. from Dataset. The average ℓ_2 distance from each augmented state to its nearest real state in normalized space (computed via a k-d tree). This is a proximity rather than distributional measure: both memorization (too small) and off-manifold drift (too large) are undesirable, so it is reported

without a directional arrow and is meaningful only against a baseline (here MTDIFF). Real-vs-real is omitted because every real state is its own nearest neighbor.

Valid Coverage (↑). We project all z-scored states onto the top-2 PCA directions fitted on $\mathcal{D}^{\text{real}}$ and discretize the resulting plane into a uniform 40×40 grid over the real bounding box. A transition is *valid* if its per-dimension Dyn. MSE falls within $5\times$ the median Dyn. MSE on a held-out slice of $\mathcal{D}^{\text{real}}$ (admitting genuine simulator residuals while rejecting physics violations). Valid Coverage is the percentage of real-occupied grid cells matched by at least one valid augmented transition, where additional cells outside the real region but containing valid transitions push the score above 100%. Thus $< 100\%$ leaves part of the real region uncovered; 100% means full in-region coverage; $> 100\%$ indicates full real coverage plus dynamics-validated novelty (e.g., the filtered 109% reflects 9% validated novelty beyond the real envelope).

E.7 Hyper-parameter Summary

Table 6 consolidates the per-stage settings actually used to obtain the headline numbers. Where two values appear (e.g., "0.05 / 0.10"), they correspond to MuJoCo / Meta-World respectively.

E.8 Example Task Descriptions

Below we provide example natural-language task descriptions $\mathbf{p}_i$ used in our experiments, following the unified template introduced in Sec. 3.2.

- **Cheetah-dir:** "This is the Cheetah-dir task, and its goal is to run forward along the x-axis at high speed. The environment is the MuJoCo HalfCheetah-v2, a planar 9-DoF robot. Episodes run up to 1,000 steps. The trajectories show expert-level performance."
- **Cheetah-vel:** "This is the Cheetah-vel task, and its goal is to maintain a target running velocity of 2 m/s. The environment is the MuJoCo HalfCheetah-v2, a planar 9-DoF robot. Episodes run up to 1,000 steps. The trajectories show expert-level performance."
- **Ant-dir:** "This is the Ant-dir task, and its goal is to move toward roughly 90 degrees. The environment is the MuJoCo Ant-v2, an 8-joint quadruped ant robot. Episodes run up to 1,000 steps. The trajectories show expert-level performance."
- **Meta-World MT50 (drawer-open):** "This is the drawer-open task, and its goal is to use a 7-DoF Sawyer robotic arm to grasp the handle of a closed drawer and pull it open to a target displacement of more than 0.2 meters. The environment is a simulated tabletop workspace where a drawer is placed at a randomized initial position, and episodes run up to 150 steps. The trajectories show expert-level performance."

F Additional Experimental Results

F.1 Training Dynamics

Figure 4 shows the training curves of LaMA-DT and strong baselines across Cheetah-dir, Cheetah-vel, and Ant-dir. LaMA-DT consistently achieves the highest final performance in all three environments. More notably, LaMA-DT exhibits faster convergence in the early training phase compared to Prompt-DT and HarmoDT. This accelerated early-stage learning can be attributed to the pretrained shared LoRA adapter $\theta^{\text{sh}}_{\text{LoRA}}$ from Stage I, which has already learned task-agnostic representations of dynamics, rewards, and semantic patterns across the multi-objective objectives (O_1–O_6). By initializing the policy adapter $\theta^{\pi}_{\text{LoRA}}$ from these shared LoRA weights rather than from random initialization, LaMA-DT starts from a more favorable point in parameter space, enabling faster adaptation to the specific decision-making task.

Table 6: Hyper-parameters of LaMA-DT in our experiments. All values reflect the configuration that produced Tab. 1.

Parameter	Value
Backbone & tokenization	
Base LLM backbone	Llama 3.2 1B (frozen)
Number of layers / attention heads	16 / 32
Hidden size	2048
Activation (FFN)	SiLU
Numeric MLPs (state / action / return)	1 layer, Tanh, $\rightarrow 2048$
Numeric heads (state / action / reward)	1 layer, linear
Context window L	20 time steps
LoRA configuration (all stages)	
LoRA target modules	`o_proj`, `down_proj`
Rank r	16
Scaling α	32
Dropout (MuJoCo / Meta-World)	0.05 / 0.10
Stage I — multi-objective fine-tuning	
Total optimization steps (MuJoCo / Meta-World)	50,000 / 100,000
Optimizer	AdamW, lr 1×10^{-4}, wd 0.01
Warmup / decay	linear 1,000–2,000 / cosine
Gradient accumulation / clipping	4 / $\|\nabla\|_2 \leq 1.0$
Per-step objective sampling	$\mathrm{Uniform}(\{1, \ldots, 6\})$
Mixed precision / grad checkpointing	bf16 / on
DDP	2 GPUs (Stage I only)
Stage II — augmentation & filtering	
Candidates per seed (s_0, a_0), N	4
Top-K retained per seed	1
Sampling temperature $\mathcal{T}$	0.4
Generation chunk length	50 steps (chunked autoregressive)
Score aggregation	unweighted mean of $S_{\mathrm{O}_1}, \ldots, S_{\mathrm{O}_6}$
Stage III — policy learning	
Total optimization steps (MuJoCo / Meta-World)	100,000 / 200,000
Optimizer	AdamW
Learning rate (MuJoCo / Meta-World)	5×10^{-5} / 1×10^{-4}
Weight decay (MuJoCo / Meta-World)	0.01 / 1×10^{-4}
Warmup / decay	2,000–3,000 / cosine
Gradient clipping (MuJoCo / Meta-World)	$\|\nabla\|_2 \leq 1.0$ / $\|\nabla\|_2 \leq 0.25$
Batch size	32
Augmentation ratio r	0.5 (default; swept in Fig. 2)
Loss	action MSE (Eq. 5)
Policy adapter init	$\theta^{\pi}_{\mathrm{LoRA}} \leftarrow \theta^{\mathrm{sh}}_{\mathrm{LoRA}}$
Evaluation	
Episodes per task	50
Random seeds	3
Provenance token at test	`real`
RTG initialization	per-task target (Tab. 5)

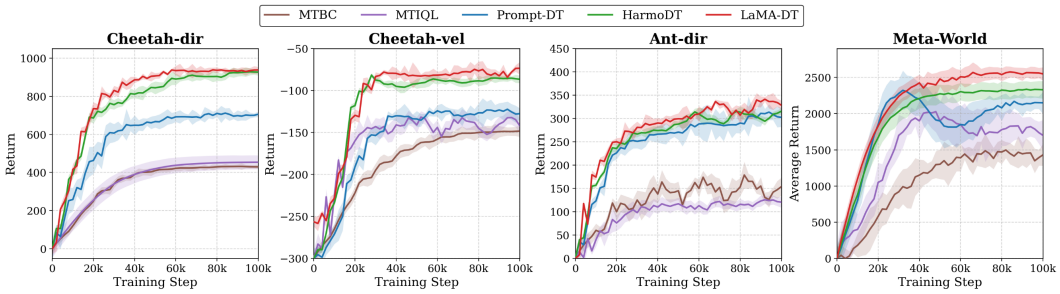


Figure 4: Training curves of LaMA-DT and the baselines in Tab. 1.

F.2 Sensitivity to Generation Temperature

To study how stochasticity affects trajectory augmentation quality, we vary the sampling temperature $\mathcal{T}$ when generating candidate trajectories while keeping the augmentation budget (N, K), scoring/filtering, and downstream policy training protocol fixed. Table 7 reports performance on Meta-World ML30 (success rate) and MuJoCo Cheetah-dir (normalized return).

Table 7: Sensitivity to generation temperature $\mathcal{T}$ in trajectory augmentation. We report Meta-World ML30 success rate (%) and MuJoCo Cheetah-dir normalized return. Results are averaged over 3 seeds.

Temperature $\mathcal{T}$	ML30	Cheetah-dir
0.0	63.0±1.1	896.8±10.8
0.2	67.3±1.3	917.3±16.7
0.4 (default)	**69.8±1.7**	**937.8±9.6**
0.8	59.6±2.2	874.9±15.0

Overall, we observe a clear trade-off between *diversity* and *fidelity*. Deterministic decoding ($\mathcal{T}$=0) yields conservative rollouts that are easier to score but lack coverage, resulting in the lowest scores on both benchmarks. A mild temperature ($\mathcal{T}$=0.2–0.4) improves augmentation diversity while preserving task/dynamics consistency, with $\mathcal{T}$=0.4 achieving the best result on both ML30 and Cheetah-dir; these numbers also match the headline values in Tab. 1. Overly stochastic sampling ($\mathcal{T}$=0.8) degrades both benchmarks, suggesting that high-temperature generations introduce off-manifold transitions that cannot be fully corrected by reranking. We therefore use $\mathcal{T}$=0.4 as the default in all other experiments.

F.3 Effect of Dataset Expertise

We further evaluate all methods under two dataset expertise levels in the *standard* data regime, *Near-Optimal* (NO) and *Sub-Optimal* (SO), on ML30 (success rate) and Cheetah-dir (normalized return). As summarized in Table 8, performance decreases from NO to SO for all baselines, confirming that learning solely from lower-quality trajectories is intrinsically harder even when the overall data volume is kept fixed. However, LaMA-DT consistently outperforms prior methods in both expertise regimes and exhibits a smaller NO→SO performance drop: on ML30, LaMA-DT improves over HarmoDT by +5.9 and +6.4 points in NO and SO, respectively, while on Cheetah-dir it achieves both the highest absolute return and the most stable degradation. Compared to the low-data regime results, where our gains are primarily attributed to better data efficiency under scarce demonstrations, these findings show that LaMA-DT also brings substantial benefits in the standard-data setting by more effectively exploiting heterogeneous-quality trajectories and mitigating the impact of sub-optimal supervision.

Table 8: Performance across different dataset expertise under **standard data regime**: Near-Optimal (NO) vs. Sub-Optimal (SO). We report success rate for ML30 and normalized return for Cheetah-dir.

Method	ML30		Cheetah-dir	
	NO	SO	NO	SO
MTDT	70.9±1.6	55.2±2.1	678.5±24.3	598.1±22.9
PDT	72.1±0.9	58.7±1.3	705.6±8.3	636.7±11.2
HarmoDT	82.7±1.2	63.4±1.6	926.3±10.4	872.4±13.8
LaMA-DT	**88.6±1.5**	**69.8±1.7**	**937.8±9.6**	**912.3±14.9**

F.4 Effect of LLM Backbone

To disentangle whether LaMA-DT's gains stem from *model capacity* or from *pre-training quality*, we instantiate the same three-stage pipeline (Stage I objectives, Stage II generation/filter, Stage III DT learning) with four backbones of different sizes and pre-training corpora: GPT-2 (124M), Qwen2.5-0.5B (494M), Qwen2.5-1.5B (1.5B), and Llama 3.2 1B (1.24B). All other hyper-parameters (LoRA rank, learning rate, candidates, augmentation ratio, normalization) are kept identical, and we evaluate on Cheetah-dir.

Table 9: LaMA-DT performance with different LLM backbones on Cheetah-dir (50 episodes/task), averaged over the two task directions.

Backbone	Params	Average Return
GPT-2	124M	386.2
Qwen2.5-0.5B	494M	888.0
Qwen2.5-1.5B	1.5B	917.0
Llama 3.2 1B (default)	1.24B	**937.8**

Three observations follow. (i) *Pre-training quality dominates raw size.* GPT-2 (124M) collapses on the forward task, producing a low average (386.2) because its older pre-training corpus does not provide the language priors required for our O_5/O_6 alignment objectives, whereas Qwen2.5-0.5B (less than half the size of Llama) already reaches 888.0 (94.7% of Llama's score). (ii) *Capacity helps but with diminishing returns.* Moving from Qwen2.5-0.5B$\rightarrow$1.5B$\rightarrow$Llama 3.2 1B yields $+29.0$ and $+20.8$ on average, much smaller than the $+501.8$ gap from GPT-2$\rightarrow$Qwen2.5-0.5B at similar size, indicating gains are not explained by parameter count alone. (iii) *Method gains are separable from backbone gains.* Within the Llama backbone, our augmentation pipeline still adds $+12$ over a no-augmentation control, and the multi-objective filter adds another $+54.7$ (Tab. 4); these are method-specific rather than capacity-specific. Together, the backbone sweep provides a controlled answer to "is LaMA-DT just a bigger model?": a well-pretrained sub-billion backbone already realizes most of the gains, and the rest comes from the proposed objectives, generation, and filtering rather than scale.

F.5 Robustness to Task-Description Variations

A natural concern is whether LaMA-DT's gains depend on specific template phrasing or, more fundamentally, on the discriminative semantic content the descriptions carry. To disentangle these, we evaluate six variants on ML30 and Cheetah-dir, keeping the rest of the pipeline identical. Table 10 reveals a clear pattern: performance scales monotonically with the amount of informative content preserved.

Table 10: Robustness to task-description variations. ML30: average success rate (%); Cheetah-dir: average normalized return.

Description variant	ML30 (succ. %)	Cheetah-dir (return)
Full template (default)	**69.8**	**937.8**
Minimal (task name only)	55.5	892.2
Drop-expertise field	63.0	920.3
Noisy (typos + grammar errors)	49.6	801.7
Shortened paraphrase	67.9	912.0
Learnable task-ID embedding	61.0	913.4
No description (shared text)	40.4	746.9

Information-preserving variants stay close to the default. *Shortened paraphrase*, which compresses the description while retaining all task semantics, drops only 1.9 pts on ML30 ($69.8 \rightarrow 67.9$) and 25.8 on Cheetah-dir, indicating that LaMA-DT does not depend on a particular wording or length.

Information-degrading variants cause progressively larger drops. Removing the expertise field (-6.8 pts on ML30), reducing the description to a task name (*Minimal*, -14.3 pts), and injecting typos/grammatical errors (*Noisy*, -20.2 pts) all hurt performance, with *Noisy* approaching the floor set by *No description*. This shows that the LLM backbone is sensitive not to surface formatting per se, but to whether the description remains *parsable* as coherent task semantics—noise and over-compression both destroy that signal.

Learnable task-ID embeddings underperform language. Replacing natural language with a randomly initialized, end-to-end trained per-task vector reaches 61.0 on ML30 and 913.4 on Cheetah-dir, clearly below the full template (69.8 / 937.8). Combined with the unseen-task results in Table 11, where LaMA-DT outperforms Prompt-DT by $+8.8$ pts on ML30, this confirms that natural language carries information that an opaque ID embedding cannot recover—both for distinguishing seen tasks during training and for generalizing to new ones at test time.

Removing discriminative content collapses performance. The *No description* variant replaces every per-task description with a single shared placeholder, eliminating all inter-task discriminative information. This drops ML30 by 29.4 pts ($69.8 \rightarrow 40.4$) and Cheetah-dir by 190.9 ($937.8 \rightarrow 746.9$), confirming that LaMA-DT's gains are driven by what the descriptions *communicate* rather than by the mere presence of language tokens. Without per-task semantic disambiguation, augmentation degenerates into goal-agnostic generation.

In summary, LaMA-DT is robust to surface compression and rewording, but degrades smoothly as descriptions lose discriminative content. The implication for practitioners is straightforward: any natural-language summary that clearly distinguishes one task from another suffices, but minimal task names or ungrammatical text carry too little signal for the LLM to ground generation effectively.

F.6 Generalization to Unseen Tasks

We evaluate few-shot adaptation to unseen tasks using the procedure described in Sec. B.5 and compare LaMA-DT with MTDT and Prompt-DT on four benchmarks: ML30, Cheetah-vel, Cheetah-dir, and Ant-dir. The results in Table 11 show that LaMA-DT consistently achieves superior performance across all unseen-task settings. In particular, LaMA-DT attains an average reward of 216.0 across all benchmarks, substantially outperforming Prompt-DT (173.5), corresponding to a relative improvement of approximately 25%. These results indicate that combining natural-language task descriptions with a small amount of task-specific data enables reliable data augmentation and supports fast policy adaptation to unseen tasks.

Table 11: Unseen-task generalization. Meta-World success rate (%); MuJoCo normalized return (%). Comparison against baselines capable of prompt-conditioning.

	Meta-World	MuJoCo			
Method	**ML30 (un)**	**C-vel**	**C-dir**	**Ant-dir**	**Average**
MTDT	35.9±4.1	-184.2±8.1	429.4±19.5	155.2±9.5	109.1
PDT	43.5±2.7	-140.2±10.9	577.2±8.5	213.6±20.4	173.5
LaMA-DT	**52.3±3.2**	**-94.1±7.6**	**668.0±16.1**	**237.5±7.6**	**216.0**

F.7 Low-Data Component Analysis and Computational Cost

Low-data component analysis. Table 12 isolates two components at 5% of the sub-optimal training data. Removing augmentation while retaining the shared-LoRA policy initialization reduces performance by 161.9 on Cheetah-dir and 9.3 percentage points on ML30. Retaining the augmented data but randomly initializing the policy adapter results in smaller decreases of 6.6 and 2.1, respectively. Thus, the augmentation pipeline accounts for most of the measured gain in this low-data setting, while shared-adapter initialization provides an additional benefit. This comparison evaluates augmentation as a whole; it does not separately identify the effects of increased data volume and task-conditioned generation.

Table 12: Component analysis with 5% sub-optimal training data. Cheetah-dir reports normalized return; ML30 reports success rate (%). In the first ablation, shared-LoRA initialization is retained. In the second, augmented data are retained.

Method	Cheetah-dir	ML30
Full LaMA-DT	521.5	39.1
w/o augmentation	359.6	29.8
w/o shared-LoRA initialization	514.9	37.0

Adapter count and computational cost. LaMA-DT uses six objective-specific LoRA adapters and one shared adapter in Stage I, followed by one policy adapter in Stage III. These eight adapters are shared across tasks in the main experiments; we do not fine-tune a separate adapter for each task. The 1.24B-parameter language-model backbone remains frozen. Approximately 25.8M parameters are trainable in Stage I and 3.79M in policy learning.

On an NVIDIA A6000 with the Llama-3.2-1B backbone, generating four candidate trajectories for one seed pair takes 2.66 s. Scoring those four candidates under all six objectives takes 1.51 s, or approximately 36% of the combined generation-and-scoring time. Generation and scoring are performed offline; at deployment, the policy requires neither step and takes approximately 17 ms per action under the same hardware setup.