PASTA: PRETRAINED ACTION-STATE TRANSFORMER AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised learning has brought about a revolutionary paradigm shift in various computing domains, including NLP, vision, and biology. Recent approaches involve pre-training transformer models on vast amounts of unlabeled data, serving as a starting point for efficiently solving downstream tasks. In the realm of reinforcement learning, researchers have recently adapted these approaches by developing models pre-trained on expert trajectories, enabling them to address a wide range of tasks, from robotics to recommendation systems. However, existing methods mostly rely on intricate pre-training objectives tailored to specific downstream applications. This paper presents a comprehensive investigation of models we refer to as pre-trained action-state transformer agents (PASTA). Our study uses a unified methodology and covers an extensive set of general downstream tasks including behavioral cloning, offline RL, sensor failure robustness, and dynamics change adaptation. Our goal is to systematically compare various design choices and provide valuable insights to practitioners for building robust models. Key highlights of our study include tokenization at the action and state component level, using fundamental pre-training objectives like next token prediction, training models across diverse domains simultaneously, and using parameter efficient fine-tuning (PEFT). The developed models in our study contain fewer than 10 million parameters and the application of PEFT enables fine-tuning of fewer than 10,000 parameters during downstream adaptation, allowing a broad community to use these models and reproduce our experiments. We hope that this study will encourage further research into the use of transformers with first-principles design choices to represent RL trajectories and contribute to robust policy learning.

1 INTRODUCTION

Reinforcement Learning (RL) has emerged as a robust framework for training highly efficient agents to interact with complex environments and learn optimal decision-making policies. RL algorithms seek effective strategies by maximizing the cumulative rewards obtained from interactions with the environment, leading to remarkable achievements in diverse applications, ranging from game-playing to robotics (Silver et al., 2014; Schulman et al., 2016; Lillicrap et al., 2016; Mnih et al., 2016). These algorithms often comprise multiple components that are essential for training and adapting neural policies. For example, model-based RL involves learning a model of the world (Racanière et al., 2017; Hafner et al., 2019; Janner et al., 2019; Schrittwieser et al., 2020) while most model-free policy gradient methods train a value or Q-network to control the variance of the gradient update (Mnih et al., 2013; Schulman et al., 2017; Haarnoja et al., 2018; Hessel et al., 2018). Training these multiple networks is challenging due to their nested nature (Boyan & Moore, 1994; Anschel et al., 2017) and the need to extract meaningful state-action space features along with relevant credit assignment in complex decision-making problems. Consequently, these factors contribute to fragile learning procedures, high sensitivity to hyperparameters, and limitations on the network's parameter capacity (Islam et al., 2017; Henderson et al., 2018; Engstrom et al., 2020).

To address these challenges, various auxiliary tasks have been proposed, including pre-training different networks to solve various tasks, such as forward or backward dynamics learning (Ha & Schmidhuber, 2018; Schwarzer et al., 2021) as well as using online contrastive learning to disentangle feature extraction from task-solving (Laskin et al., 2020; Nachum & Yang, 2021; Eysenbach et al., 2022). Alternatively, pre-training agents from a static dataset via offline RL without requiring

interaction with the environment also enables robust policies to be deployed for real applications. Most of these approaches rely either on conservative policy optimization (Fujimoto & Gu, 2021; Kumar et al., 2020) or supervised training on state-action-rewards trajectory inputs where the transformer architecture has proven to be particularly powerful (Chen et al., 2021; Janner et al., 2021).

Recently, self-supervised learning has emerged as a powerful paradigm for pre-training neural networks in various domains including NLP (Chowdhery et al., 2022; Brown et al., 2020; Touvron et al., 2023), computer vision (Dosovitskiy et al., 2020; Bao et al., 2021; He et al., 2022) or biology (Lin et al., 2023; Dalla-Torre et al., 2023), especially when combined with the transformer architecture. Inspired by impressive NLP results with the transformer architecture applied to sequential discrete data, most self-supervised techniques use tokenization, representing input data as a sequence of discrete elements called tokens. Once the data is transformed, first-principles objectives such as mask modeling (Devlin et al., 2018) or next token prediction (Brown et al., 2020) can be used for self-supervised training of the model. In RL, recent works have explored the use of self-supervised learning to pre-train transformer networks with expert data. While these investigations have yielded exciting outcomes, such as zero-shot capabilities and transfer learning between environments, methods such as MTM (Wu et al., 2023) and SMART (Sun et al., 2023) often rely on highly specific masking techniques and masking schedules (Liu et al., 2022a), and explore transfer learning across a limited number of tasks. Hence, further exploration of this class of methods is warranted. In this paper, we provide a general study of the different self-supervised objectives and of the different tokenization techniques. In addition, we outline a standardized set of downstream tasks for evaluating the transfer learning performance of pre-trained models, ranging from behavioral cloning to offline RL, robustness to sensor failure, and adaptation to changing dynamics.

Our contributions. With this objective in mind, we introduce the PASTA study, which stands for Pretrained Action-State Transformer Agents. This study provides comprehensive comparisons involving four pre-training objectives and two types of tokenization techniques, with multiple pre-training datasets and a collection of 23 downstream tasks, categorized into three groups and across three continuous control environments. The PASTA downstream tasks encompass imitation learning and standard RL to demonstrate the versatility of the pre-trained models. Moreover, we explore scenarios involving physical regime changes and observations alteration to assess the zero-shot performance and stability of the pre-trained models. We summarize the key findings of our study below:

1. **Tokenize trajectories at the component level.** Tokenization at the component level significantly outperforms tokenization at the modality level. In other words, it is more effective to tokenize trajectories based on the individual components of the state and action vectors, rather than directly tokenizing states and actions as is commonly done in existing works.
2. **Prefer first-principles objectives over convoluted ones.** First principles training objectives, such as random masking or next word prediction with standard hyperparameters match or can even outperform more intricate and task-specific objectives carefully designed for RL, such as those considered in MTM or SMART.
3. **Pre-train the same model on datasets from multiple domains.** Simultaneously pre-training the model on datasets from the three environments leads to enhanced performance across all three environments compared to training separate models for each environment.
4. **Generalize with a small parameter count.** All of the examined models have fewer than 10 million parameters. Hence, while these approaches are both affordable and practical even on limited hardware resources, the above findings are corroborated by experimentation with three transfer learning scenarios: a) probing (the pre-trained models generate embeddings and only the policy heads are trained to address downstream tasks), b) parameter-efficient fine-tuning (PEFT) (introducing a limited number of weights to the pre-trained model and fine-tuning them for solving downstream tasks), and c) zero-shot transfer.

2 Related Work

Self-supervised Learning for RL. Self-supervised learning, which trains models using unlabeled data, has achieved notable success in various control domains (Liu & Abbeel, 2021; Yuan et al., 2022; Laskin et al., 2022). One effective approach is contrastive self-prediction (Chopra et al., 2005; Le-Khac et al., 2020; Yang & Nachum, 2021; Banino et al., 2021) which have proven effective

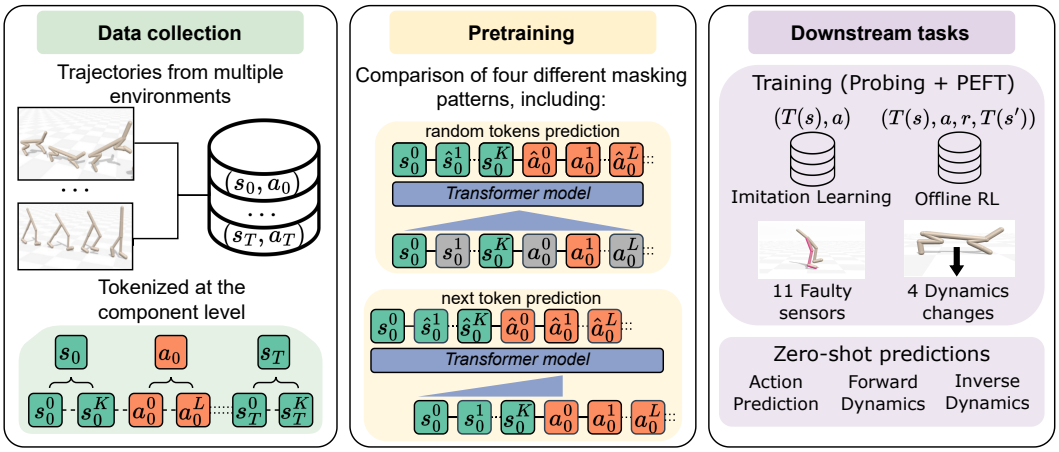

Figure 1: Illustration of the PASTA study. **Left:** State-action trajectories are collected from multiple environments and are tokenized at the component level. **Middle:** A transformer model is pre-trained by processing fixed-size chunks of these sequences. It learns latent representations $T(s)$ of the environments' states. In this study, we compare different tokenization schemes, masking patterns, and pre-training objectives, *e.g.*, random tokens prediction (BERT) or next token prediction (GPT). **Right:** The representations of the pre-trained transformer models are evaluated on multiple downstream tasks in which the learned representation $T(s)$ serves as a surrogate state for the policy.

in efficient data augmentation strategies, enabling downstream task solving through fine-tuning, particularly in RL tasks (Laskin et al., 2020; Nachum & Yang, 2021). Our study aligns with this trend, focusing on domain-agnostic self-supervised mechanisms that leverage masked predictions to pre-train general-purpose RL networks.

Offline RL and Imitation Learning. Offline learning for control involves leveraging historical data from a fixed behavior policy π_b to learn a reward-maximizing policy in an unknown environment. Offline RL methods are typically designed to restrict the learned policy from producing out-of-distribution actions or constrain the learning process within the support of the dataset. Most of these methods usually leverage importance sampling (Sutton et al., 2016; Nair et al., 2020; Liu et al., 2022c) or incorporate explicit policy constraints (Kumar et al., 2019; Fujimoto & Gu, 2021; Fakoor et al., 2021; Dong et al., 2023). In contrast, Imitation learning (IL) focuses on learning policies by imitating expert demonstrations. Behavior cloning (BC) involves training a policy to mimic expert actions directly while Inverse RL (Ng et al., 2000) aims to infer the underlying reward function to train policies that generalize well to new situations. In contrast, the models investigated in PASTA focus on learning general reward-free representations that can accelerate and facilitate the training of any off-the-shelf offline RL or imitation learning algorithm.

Masked Predictions and Transformers in RL. Recently, self-supervised learning techniques based on next token prediction (Brown et al., 2020) and random masked predictions (Devlin et al., 2018) have gained popularity. These methods involve predicting missing content by masking portions of the input sequence. These first-principles pre-training methods have achieved remarkable success in various domains, including NLP (Radford et al., 2018; 2019), computer vision (Dosovitskiy et al., 2020; Bao et al., 2021; Van Den Oord et al., 2017), and robotics (Driess et al., 2023). We explore the effectiveness of different variants of these approaches, with various masking patterns and pre-training objectives, in modeling RL trajectories and learning representations of state-action vector components. Transformer networks have been particularly valuable for these purposes. The decision transformer (Chen et al., 2021) and trajectory transformer (Janner et al., 2021) have emerged as offline RL approaches using a causal transformer architecture to fit a reward-conditioned policy, paving the way for subsequent work (Zheng et al., 2022; Yamagata et al., 2022; Liu et al., 2022a; Lee et al., 2023). Notably, GATO (Reed et al., 2022) is a multi-modal behavioral cloning method that directly learns policies, while PASTA focuses on pre-training self-supervised representations. Additionally, MTM (Wu et al., 2023) and SMART (Sun et al., 2023) propose original masking objectives for pre-training transformers in RL. MTM randomly masks tokens while ensuring some

tokens are predicted without future context. It uses modality-level masking and is limited to single-domain pre-training. Conversely, SMART uses a three-fold objective for pre-training a decision transformer with forward dynamics prediction, inverse dynamics prediction, and "random masked hindsight control" with a curriculum masking schedule. It focuses on processing real-valued visual observation sequences and investigates generalization across different domains. In PASTA, we compare several first-principles pre-training objectives without a masking schedule to these state-of-the-art approaches across multiple environments and diverse downstream tasks.

3 The PASTA Study

3.1 Preliminaries

Self-supervised Learning framework. In this paper, we study self-supervised learning (Balestriero et al., 2023) techniques to pre-train models on a large corpus of static (offline) datasets from interactions with simulated environments, as done in Shah & Kumar (2021); Schwarzer et al. (2023). By solving pre-training objectives, such as predicting future states or filling in missing information, the models learn to extract meaningful features that capture the underlying structure of the data. We focus our study on the use of the transformer architecture due to its ability to model long-range dependencies and capture complex patterns in sequential data. In addition, the attention mechanism is designed to consider the temporal and intra-modality (position in the state or action vectors) dependencies. After pre-training the models, we evaluate their capabilities to solve downstream tasks. This analysis is done through the lenses of three mechanisms: (i) probing, (ii) parameter-efficient fine-tuning (PEFT) (Liu et al., 2022b; 2021; Hu et al., 2021), and (iii) zero-shot transfer. The goal of the study is to investigate which pre-training process makes the model learn the most generalizable representations to provide a strong foundation for adaptation and learning in specified environments. An illustration of the approach adopted in PASTA is given in Figure 1.

Reinforcement Learning framework. In this paper, we place ourselves in the Markov Decision Processes (Puterman, 1994) framework. A Markov Decision Process (MDP) is a tuple $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition kernel, $\mathcal{R}$ is the bounded reward function and $\gamma \in [0,1)$ is the discount factor. Let π denote a stochastic policy mapping states to distributions over actions. We place ourselves in the infinite-horizon setting, *i.e.*, we seek a policy that optimizes $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. The value of a state is the quantity $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s]$ and the value of a state-action pair $Q^\pi(s,a)$ of performing action a in state s and then following policy π is defined as: $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$.

3.2 Tokenization

Building Motivation. Tokenization is a fundamental technique in self-supervised learning as it is an effective way to apply first-principles objectives for neural network pre-training. By representing states and actions as vector components and learning from diverse environments within the same physics engine, tokenization at this level enables representation models to reason effectively about interactions among different parts of a robot. This approach is expected to enhance performance across various morphologies and downstream tasks. For example, in a bipedal locomotion task, breaking down leg movements into thigh and shin components can facilitate the learning of more stable and efficient walking or running. We explore whether this granularity of tokenization offers a promising approach to address continuous control problems solely from a sequential perspective.

Component-level Tokenization. A key focus of the PASTA study is the representation of trajectories at the level of vector components for states and actions. Instead of considering each trajectory as a sequence of state, action (and often return) tuples, as done in most previous work, including SMART (Sun et al., 2023) and MTM (Wu et al., 2023), we break the sequences down into individual state and action *components*. Additionally, we exclude the return to develop a general method applicable to reward-free settings and learn representations that are not tied to task-specific rewards (Stooke et al., 2021; Yarats et al., 2021). This level of tokenization allows capturing dynamics and dependencies at different space scales, as well as the interplay between the agent's morphological actions and resulting states across different robotic structures. As we observe in Section 4, this results in more detailed representations that improve the performance of downstream tasks.

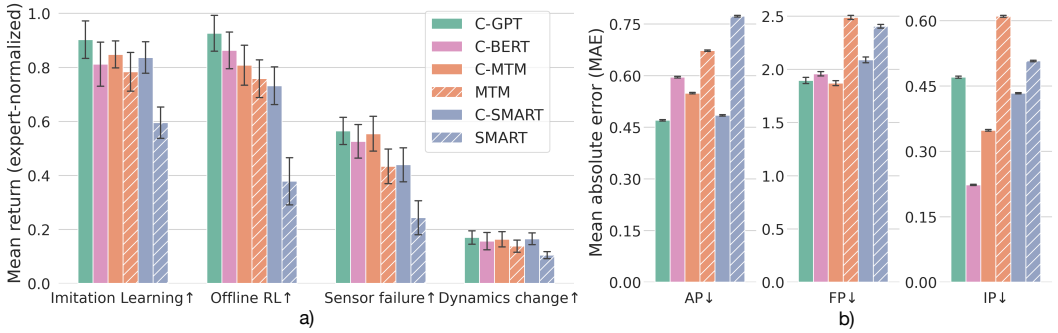

Figure 2: Performance aggregation of the PASTA pre-trained models (C-*) and modality-level models (MTM and SMART) with different masking and training objectives, evaluated on **a)** the representation learning tasks with fine-tuning and **b)** the zero-shot transfer tasks: Action Prediction (AP), Forward Prediction (FP), and Inverse Prediction (IP). Results are aggregated over all environments. We developed our own implementation of MTM and SMART using the same masking patterns and training objectives. ↑ (resp. ↓) indicates that higher (resp. lower) is better.

3.3 Pre-training

Trajectory modeling. The PASTA study includes different types of self-supervised learning strategies, each using different combinations of random token masking and/or next token prediction. Next token prediction uses autoregressive masking, while random masked prediction aims to learn from a sequence of trajectory tokens denoted as $\tau = (s_0^0, ..., s_0^K, a_0^0, ..., a_0^L, ..., s_T^0, ..., s_T^K)$. The model's task is to reconstruct this sequence when presented with a masked version $\hat{\tau} = T_\theta(\texttt{Masked}(\tau))$, where K is the observation space size, L is the action space size and T is an arbitrary trajectory size. Here, T_θ refers to a bi-directional transformer, and $\texttt{Masked}(\tau)$ represents a modified view of τ where certain elements in the sequence are masked. For instance, a masked view could be $(s_0^0, ..., s_0^K, a_0^0, ..., a_0^L, ..., \_, ..., \_)$, where the underscore "_" symbol denotes a masked element. In this scenario, representation models must predict and fill in the missing state or action components. We directly mask tokens at the input and output levels and do not use masking in attention weights.

Pre-training objectives. Next, we introduce the masking patterns investigated in the experimental study. First, the C-GPT masking pattern mimics GPT's masking mechanism and uses causal (backward-looking) attention to predict the next unseen token in RL trajectories. Second, we have the C-BERT masking pattern, derived from BERT's masking mechanism which uses random masks to facilitate diverse learning signals from each trajectory by enabling different combinations. Figure 1 provides a visual representation of the C-BERT and C-GPT masking mechanisms. Third, the MTM masking scheme (Wu et al., 2023) combines random masking (similar to BERT) and causal prediction of the last elements of the trajectory. This latter aims to prevent the model from overly relying on future token information. While MTM operates at the modality level, we adapt it to operate directly on components by masking random tokens within the trajectory and additionally masking a certain proportion of the last tokens. We refer to this method as C-MTM, which stands for component-level MTM. Finally, SMART's training objective encompasses three different masking patterns (Sun et al., 2023): forward-dynamics, inverse-dynamics and masked hindsight control. The training involves adding up the three losses corresponding to the three masking patterns. Similarly, we derive C-SMART, where instead of masking an entire modality at each stage, we mask a random fraction of the tokens within that modality. See Appendix C for additional details.

3.4 Downstream evaluation

In this study, we evaluate the effectiveness of PASTA models in transfer learning from two perspectives. Firstly, we examine the ability of pre-trained models to generate high-quality representations. This evaluation is carried out through probing and parameter-efficient fine-tuning. Secondly, we investigate the capability of pre-trained models to solve new tasks in a zero-shot transfer setting. To accomplish this, we introduce two sets of tasks: **Representation learning** tasks (17) and **Zero-shot transfer** tasks (6), comprising a total of 23 evaluation downstream tasks. These task sets are further

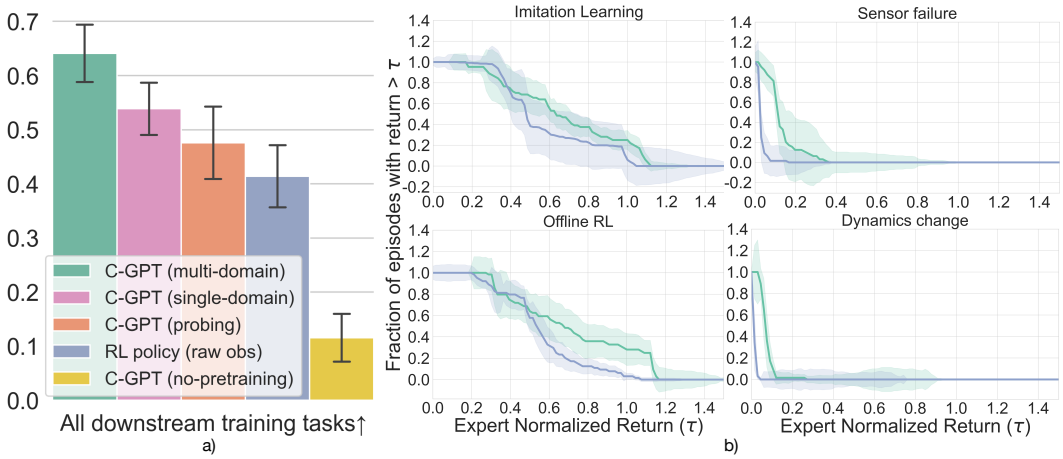

Figure 3: **a)** Evaluation in all downstream training tasks with multi- and single-domain pre-training, no-pretraining, probing, and training from raw observations. Remarkably, C-GPT outperforms all other methods and significantly surpasses policies trained from raw observations, despite having an equivalent number of parameters. ↑ indicates that higher is better. **b)** Performance profile of C-GPT against RL policies trained from raw observations. Shaded areas show the interquartile range over 3 seeds. C-GPT demonstrates higher performance versus policies trained from raw observations.

divided into sub-categories. These categories are designed to provide a general-purpose assessment for pre-trained agents, irrespective of the specific environment or domain. All tasks considered in this study involve either classification or regression.

Representation learning. The representation learning tasks encompass four sub-categories: Imitation Learning, Offline RL, Sensor Failure, and Dynamics Change. We evaluate the quality of raw representations learned by pre-trained agents using probing on these tasks. In this setting, the weights of the pre-trained models are kept fixed, and the embeddings produced by the final attention layer are fed into a single dense layer network. As the expressive power of such networks is limited, achieving good performance is contingent upon the embeddings containing sufficient information. Furthermore, we assess the quality of the produced representations through fine-tuning where the weights of the pre-trained agents are further updated to solve the downstream tasks. For this purpose, we use parameter-efficient fine-tuning, which focuses on updating only a small subset of (or newly introduced) weights representing a small fraction of the total weight volume. Recent studies have shown that these techniques can match or surpass the performance of standard fine-tuning methods while being faster and more memory-efficient. In this study, we use $(IA)^3$ (Liu et al., 2022b) for fine-tuning, which only updates an average of 0.1% of the total pre-trained weight count.

Zero-shot transfer. The zero-shot tasks are organized into three categories: Action Prediction (AP), Forward dynamics Prediction (FP), and Inverse dynamics Prediction (IP). These categories evaluate the pre-trained models' ability to directly predict states or actions based on trajectory information. Specifically, the prediction problems can be expressed as follows; AP: $(\tau_{t-1}, s_t \rightarrow a_t)$, FP: $(\tau_{t-1}, s_t, a_t \rightarrow s_{t+1})$ and IP: $(\tau_{t-1}, s_t, s_{t+1} \rightarrow a_t)$, where the input to the model is shown on the left side of the parentheses, and the prediction target is shown on the right side. For each category, we examine both component prediction and modality (state or action) prediction.

4 Experimental Analysis

In this section, we present the experimental study conducted to examine the impact of pre-training objectives, tokenization, and dataset preparation choices on the generalization capabilities of pre-trained PASTA models.

Table 1: Comparison of models with different tokenization (modality-level, component-level), pre-training datasets (single-domain, multi-domain), and MLP network (RL policy) in the 17 fine-tuned downstream tasks. (max) indicates maximum performance between respectively SMART & MTM and C-SMART & C-MTM. (↑) indicates higher is better and [11] indicates 11 tasks per category.

Domain	Task	RL policy (raw obs)	Modality-level (max)	Component-level (max)	C-GPT (single-domain)	C-GPT (multi-domain)
HalfCheetah	IL (↑) [1]	0.719 ± 0.10	1.006 ± 0.05	1.033 ± 0.02	0.974 ± 0.07	1.031 ± 0.05
	Off-RL (↑) [1]	0.903 ± 0.04	0.982 ± 0.08	1.100 ± 0.03	1.031 ± 0.07	1.092 ± 0.03
	Sensor failure (↑) [11]	0.560 ± 0.09	0.831 ± 0.08	0.871 ± 0.10	0.868 ± 0.09	0.926 ± 0.07
	Dynamics change (↑) [4]	0.124 ± 0.05	0.175 ± 0.03	0.223 ± 0.02	0.212 ± 0.02	0.248 ± 0.02
Hopper	IL (↑) [1]	0.571 ± 0.06	0.588 ± 0.09	1.093 ± 0.10	1.132 ± 0.04	1.119 ± 0.10
	Off-RL (↑) [1]	0.412 ± 0.08	0.873 ± 0.10	1.074 ± 0.01	1.169 ± 0.08	1.209 ± 0.09
	Sensor failure (↑) [11]	0.147 ± 0.01	0.173 ± 0.04	0.355 ± 0.06	0.100 ± 0.03	0.447 ± 0.03
	Dynamics change (↑) [4]	0.076 ± 0.00	0.139 ± 0.00	0.246 ± 0.03	0.172 ± 0.03	0.258 ± 0.03
Walker2d	IL (↑) [1]	0.520 ± 0.04	0.520 ± 0.05	0.519 ± 0.05	0.354 ± 0.04	0.558 ± 0.03
	Off-RL (↑) [1]	0.652 ± 0.11	0.337 ± 0.05	0.376 ± 0.02	0.399 ± 0.03	0.477 ± 0.05
	Sensor failure (↑) [11]	0.281 ± 0.03	0.206 ± 0.03	0.103 ± 0.05	0.058 ± 0.02	0.322 ± 0.05
	Dynamics change (↑) [4]	0.000 ± 0.00	0.004 ± 0.00	0.000 ± 0.00	0.000 ± 0.00	0.004 ± 0.00

4.1 Experimental Setup

Domains. To assess the effectiveness of our approach, we select tasks from the Brax library (Freeman et al., 2021a), which provides environments designed to closely match (Freeman et al., 2021b) the original versions found in MuJoCo's environment suite (Todorov et al., 2012). Brax provides significant advantages over MuJoCo, as it offers a highly flexible and scalable framework for simulating robotic systems based on realistic physics. More information about the environments is given in Appendix D.2. The pre-training datasets consist of trajectories collected from three Brax environments: HalfCheetah, Hopper, and Walker2d. Following the protocols used in previous work (Fu et al., 2020; Sun et al., 2023) we trained 10 Soft Actor-Critic (SAC) (Haarnoja et al., 2018) agents initialized with different seeds and collected single- and multi-domain datasets composed of 510 million tokens in total. For details about the pre-training datasets, we refer the reader to Appendix D.3.

Consequently, the 23 downstream tasks presented in Section 3 are set up for each environment resulting in a total of 69 tasks across environments. We introduce multiple environments for pre-training and evaluation (i) to evaluate the reproducibility of our findings across domains and (ii) to study the performance of models pre-trained on the three datasets simultaneously (multi-domains model) compared to single domain models. The implementations of tasks related to sensor failure and dynamics changes need to be adjusted to each domain to account for their specific dynamics. For further details about the implementation of downstream tasks, please refer to Appendix D.4.

Implementation details. In this study, we focus on reasonably sized and efficient models, typically consisting of around 10 million parameters. To capture positional information effectively, we incorporate a learned positional embedding layer at the component level. Additionally, we include a rotary position encoding layer following the approach in Su et al. (2021) to account for relative positional information. More implementation details are provided in Appendix B. To convert the collected data (state or action components) into tokens, we adopt a tokenization scheme similar to Reed et al. (2022). Continuous values are mu-law encoded to the range [-1, 1] and discretized into 1024 uniform bins. The sequence ordering follows observation tokens followed by action tokens, with transitions arranged in timestep order.

4.2 Results

Tokenization granularity. We first examine the impact of tokenization granularity on the generalization performance of the models. We train models using the SMART and MTM training procedures with two granularities: modality-level (predicting at the level of observations and actions) for SMART and MTM and component-level (predicting at the level of observation and action components) for C-SMART and C-MTM. All four models share the same architecture and are trained under identical conditions using the multi-domain dataset. After pre-training, the models undergo fine-tuning for downstream tasks under identical conditions. Figure 2 (a) reports the performance expressed as the mean return normalized by the expert return for each domain. We aggregate the

Table 2: Breakdown of Expert-normalized returns in the sensor failure and dynamics change tasks. (↑) indicates that higher is better.

Model	Sensor Failure	Dynamics Change
C-GPT (multi-domain) (↑)	0.56 ± 0.04	0.17 ± 0.02
C-GPT (single-domain) (↑)	0.34 ± 0.05	0.13 ± 0.02
RL policy (raw obs) (↑)	0.33 ± 0.04	0.07 ± 0.02
C-GPT (no-pretraining) (↑)	0.17 ± 0.04	0.03 ± 0.02

results across all domains and task categories: one Imitation Learning task, one Offline RL task, 11 Sensor Failure tasks, and 4 Dynamics Change tasks. Furthermore, Table 1 provides a breakdown of performance for both tokenization techniques across different domains. Overall, we observe that transitioning from modality-level to component-level improves performance for both methods. This improvement is particularly significant for SMART, where the shift nearly doubles the performance for Sensor Failure and Offline RL tasks.

Masking objectives. Then, we compare first principles tokenization techniques *i.e.*, masked language modeling (BERT) and next word prediction (GPT) with state-of-the-art transformer RL methods MTM and SMART which incorporate more customized design choices. To adapt BERT and GPT to our problem, we introduce C-BERT and C-GPT. These models are trained using component-level tokenization on the multi-domain dataset under similar conditions as C-SMART and C-MTM, which are the component-level counterparts of MTM and SMART. We systematically fine-tune all models for all downstream tasks and domains. Our findings reveal that C-BERT performs competitively compared to C-SMART and C-MTM across all task categories, as depicted in Figure 2 (a). Additionally, C-GPT exhibits slightly superior average performance for all downstream tasks compared to other masking techniques, as shown in Table 1. This demonstrates that simple and first-principles objectives are sufficient to achieve robust generalization performance.

Multi-domain representation learning. Then, we investigate the benefits of pre-training multi-domain representation models using component-level tokens. We synthesize our results using C-GPT as it showed the best performance among the other models in the study. Figure 3 (a) provides a summary of the results aggregated across all domains and downstream tasks, while Figure 3 (b) presents a breakdown of the performance profiles for each downstream task group.

First, we confirm that policies trained using C-GPT outperform policies trained from raw observations with neural networks comprised of an equivalent number of parameters (cf. Appendix D.1). This validates the capability of the model to produce useful representations. We also compare the performance of C-GPT against a randomly initialized model (no-pre-training) with the same architecture and confirm the positive effect of pre-training on the observed performance.

Second, we compare the performance of C-GPT against specialized models trained independently in each environment (C-GPT (single-domain)). To ensure a fair comparison, all models are trained for an equal number of epochs and have the same representation capability (architecture and number of learned parameters). Importantly, the results demonstrate that C-GPT outperforms specialized models in terms of final performance, indicating that C-GPT (multi-domain) learns a more generalizable representation than C-GPT (single-domain). For a detailed breakdown of the results for each downstream task, please refer to panels provided in Appendix A.1.

Robust representations. In this section, we focus on resilience to sensor failure and adaptability to dynamics change. These factors play a crucial role in real-world robotics scenarios, where sensor malfunctions and environmental variations can pose risks and impact decision-making processes. We used BC as the training algorithm and during evaluation, we systematically disabled each of the 11 sensors individually by assigning a value of 0 to the corresponding coordinate in the state vector. In Table 2, C-GPT (multi-domain) exhibits higher performance compared to the baselines, demonstrating its enhanced robustness in handling sensor failures. Furthermore, we introduced four gravity changes during the inference phase, and the results reaffirm the resilience of C-GPT (multi-domain) in adapting to dynamics change, thus validating our previous findings.

Zero-shot predictions. In this section, we investigate the zero-shot performance of pre-trained models, complementing our previous findings with fine-tuning. We consider an additional set of tasks outlined in Section 3.4, originally introduced in MTM. Notably, Figure 2 (b) shows that C-GPT's errors are comparable to those of specialized models like C-SMART. This suggests a significant alignment between C-GPT's pre-training approach and the inference tasks, even though C-GPT was not explicitly trained to minimize these specific tasks like C-SMART. These results further support the validity of the PASTA methodology, which incorporates first-principles masking patterns, a straightforward objective function, and component-level tokenization. Additionally, we note that C-GPT performs similarly to both C-BERT and C-SMART, except for the inverse dynamics prediction task. This difference is expected since C-GPT is an autoregressive model. Nevertheless, overall performance in Figure 2 (a) suggests that the inclusion of the inverse-dynamics task is not necessary for achieving improved performance across the analyzed downstream tasks.

Raw representations. Finally, we examine the representational power of raw pre-trained model embeddings through probing. Probing involves freezing all parameters of the pre-trained neural network during downstream task training. From Figure 3 (a), we observe that, on average, fine-tuning C-GPT outperforms probing by a significant margin. Importantly, the employed parameter-efficient fine-tuning method only introduces a minor average increase of 3.7% in learned parameters, resulting in negligible additional training time.

5 Discussion

This paper presents the PASTA study, which focuses on Pretrained Action-State Transformer Agents. The study aims to comprehensively explore self-supervised learning models for RL with downstream training and zero-shot performance evaluation. This study contributes pre-training datasets, a diverse set of 23 downstream tasks, and a comparison of four pre-training objectives and two tokenization techniques. The study was conducted across three continuous control environments to demonstrate the versatility and effectiveness of pre-trained models in various transfer learning scenarios, including probing, parameter-efficient fine-tuning, and zero-shot transfer.

One key finding of this study is the superiority of first-principles objectives over convoluted ones. The results indicate that standard self-supervised objectives, such as random masking or next token prediction, with standard hyperparameters, can match or even outperform more intricate and task-specific objectives designed specifically for RL. This suggests that simpler objectives can serve as a strong foundation for representation learning in RL tasks, simplifying the pre-training process. Additionally, it was observed that tokenizing trajectories based on individual components of the state and action vectors (component-level) rather than directly tokenizing states and actions (modality-level) leads to improved performance in representation models. This observation emphasizes the significance of carefully selecting tokenization strategies to enhance the expressiveness of the learned representations. Furthermore, the study revealed the benefits of pre-training a single model on datasets from multiple domains. Simultaneous pre-training on datasets from different environments resulted in improved performance across all three environments compared to training separate models for each domain. This finding indicates the potential for knowledge transfer and generalization when using diverse pre-training data. For example, the learned representations significantly enhanced the sample efficiency and performance of traditional offline RL algorithms, with an average increase of 23% in final returns compared to the same algorithms using raw inputs. Additionally, the investigation in Section 4.2 highlighted the importance of developing algorithms like C-GPT that can effectively adapt and make decisions in the presence of sensor failures or dynamic changes, ensuring safety and mitigating risks in robotics applications.

Overall, the findings from this study provide valuable guidance to researchers interested in leveraging self-supervised learning to improve RL in complex decision-making tasks. The models presented in this study are lightweight, and the fine-tuning approach involves training fewer than 10,000 parameters, facilitated by PEFT. This feature enables the replication of both pre-training and fine-tuning experiments on readily available hardware, making them accessible to any practitioner. In future work, it is anticipated that further exploration of other self-supervised objectives and tokenization strategies will be conducted. Additionally, expanding the range of downstream tasks will allow for a more comprehensive evaluation of model adaptation and robustness under varying conditions, further enhancing the practical applicability of pre-trained agents in real-world scenarios.

REFERENCES

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pp. 176–185. PMLR, 2017.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

Andrea Banino, Adria Puigdomenech Badia, Jacob C Walker, Tim Scholtes, Jovana Mitrovic, and Charles Blundell. Coberl: Contrastive bert for reinforcement learning. In *International Conference on Learning Representations*, 2021.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, 7, 1994.

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

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.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 539–546. IEEE, 2005.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, pp. 2023–01, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Kefan Dong, Yannis Flet-Berliac, Allen Nie, and Emma Brunskill. Model-based offline reinforcement learning with local misspecification. *arXiv preprint arXiv:2301.11426*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1etN1rtPB.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Infinite horizon model predictive control for nonlinear periodic tasks. *Manuscript under review*, 4, 2011.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Rasool Fakoor, Jonas W Mueller, Kavosh Asadi, Pratik Chaudhari, and Alexander J Smola. Continuous doubly constrained batch reinforcement learning. *Advances in Neural Information Processing Systems*, 34:11260–11273, 2021.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021a. URL http://github.com/google/brax.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021b.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

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

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

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

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934, 2020.

Jonathan N. Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning, 2023.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Fangchen Liu, Hao Liu, Aditya Grover, and Pieter Abbeel. Masked autoencoding for scalable and generalizable decision making. *arXiv preprint arXiv:2211.12740*, 2022a.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022b.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

Yao Liu, Yannis Flet-Berliac, and Emma Brunskill. Offline policy optimization with eligible actions. In *Uncertainty in Artificial Intelligence*, pp. 1253–1263. PMLR, 2022c.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Ofir Nachum and Mengjiao Yang. Provable representation learning for imitation with contrastive fourier features. *Advances in Neural Information Processing Systems*, 34:30100–30112, 2021.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Martin Puterman. *Markov Decision Processes*. Wiley, 1994. ISBN 978-0471727828.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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

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

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Max Schwarzer, Johan Obando-Ceron, Aaron Courville, Marc Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. *arXiv preprint arXiv:2305.19452*, 2023.

Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Yanchao Sun, Shuang Ma, Ratnesh Madaan, Rogerio Bonatti, Furong Huang, and Ashish Kapoor. Smart: Self-supervised multi-task pretraining with control transformers. *arXiv preprint arXiv:2301.09816*, 2023.

Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1): 2603–2631, 2016.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Paweł Wawrzyński. A cat-like robot real-time learning to run. In *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*, pp. 380–390. Springer, 2009.

Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*, 2023.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. *arXiv preprint arXiv:2209.03993*, 2022.

Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*, pp. 11784–11794. PMLR, 2021.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021.

Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-trained image encoder for generalizable visual reinforcement learning. *arXiv preprint arXiv:2212.08860*, 2022.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *International Conference on Machine Learning*, pp. 27042–27059. PMLR, 2022.

A Additional Experiments

A.1 Detailed Breakdown of Downstream Tasks Results

Figure 4: [1/2] Detailed breakdown of the results obtained in the 17 fine-tuning downstream tasks for the three environments HalfCheetah, Hopper and Walker2d.

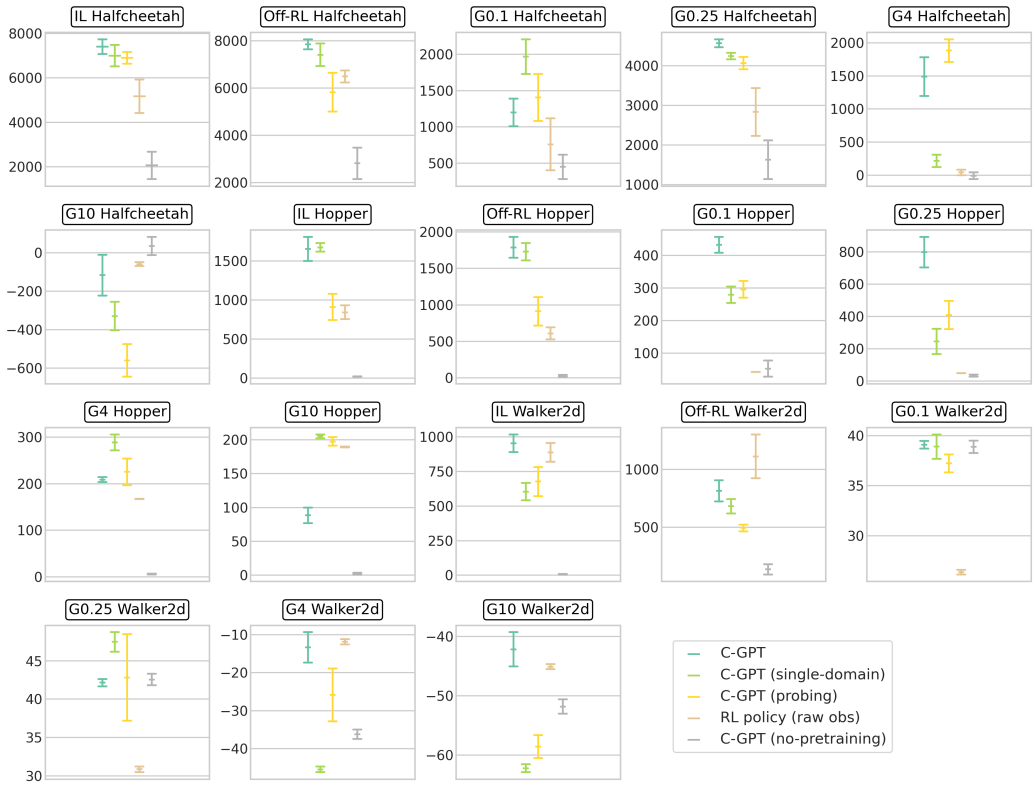

Figure 5: [2/2] Detailed breakdown of the results obtained in the 17 fine-tuning downstream tasks for the three environments HalfCheetah, Hopper and Walker2d.

B Implementation Details

In the sequence tokenization phase, we do not use return conditioning but since the representation models are pre-trained on multiple environments and tasks, we use environment conditioning, *i.e.*, during training, an environment token is appended at the beginning of the sequences in each batch, providing the model with additional contextual information. In practice, the length of the last two modalities (state and action concatenated) varies across different environments. Therefore, the maximum portion of masked tokens at the end of the sequence differs depending on the environment. For instance, in the Hopper environment with 3 actions and 11 observation tokens, the maximum portion of masked tokens is 14, while in HalfCheetah with 6 actions and 18 observation tokens, it is 24. Additionally, as we maintain a fixed-size context window of 128, the sequences' starting points will have varying truncations for different environments, ensuring a non-truncated state at the end of the window. Another design choice is the embedding aggregation, *i.e.*, how to come from a context_window x embedding_dimension tensor to a 1 x embedding_dimension tensor. We decided to use a Conv1d with a kernel size of 1.

Computational Cost. A significant advantage of the component-level sequencing approach is its reduced input dimension, allowing cheaper computational costs. By capturing the components of states and actions at different time steps, the input space expands linearly rather than quadratically mitigating the challenges associated with the curse of dimensionality. To illustrate this, consider a simple example of a 2-dimensional state space with a discretization size of 9. With a component-level granularity, the input size becomes $2 \times 9 = 18$. In contrast, a state-level granularity results in an input size of $9 \times 9 = 81$. The former exhibits linear growth within the observation space, while the latter demonstrates quadratic growth. Moreover, while it effectively multiplies the length of the input sequence by the average number of components in a state, this drawback is absorbed by the

increased context window of transformer models. Lastly, for an equal number of trajectories, the number of tokens is also trivially larger than that with a state- and action-level granularity.

C Additional Details on Masking Patterns

In this section, we provide further details on the masking patterns and schedule used in the SMART (Sun et al., 2023) and MTM (Wu et al., 2023) baselines. In C-GPT or C-BERT, we focused on reducing the technicalities to their minimum: a simple masking pattern, *i.e.*, GPT-like or BERT-like, and no masking schedule.

In SMART, the objective involves three components: Forward Dynamics Prediction, Inverse Dynamics Prediction, and Random Masked Hindsight Control. The masking schedule involves two masking sizes, k and k', which determine the number of masked actions and observations during pre-training. The masking schedule for actions (k) is designed to gradually increase the difficulty of the random masked hindsight control task. It starts with $k = 1$, ensuring the model initially predicts masked actions based on a single observed action. As training progresses, the value of k is increased in a curriculum fashion. The masking schedule for observations (k') ensures that the model learns to predict masked actions based on a revealed subsequence of observations and actions, rather than relying solely on local dynamics. Similar to the action masking schedule, k' starts at 1 and gradually increases during training. SMART's paper suggests that the masking schedule is essential for effective pre-training in control environments. By gradually increasing the masking difficulty, the model is exposed to a range of training scenarios, starting with simple local dynamics and gradually transitioning to complex long-term dependencies.

In MTM, the masking pattern is implemented by requiring at least one token in the masked sequence to be autoregressive, which means it must be predicted based solely on previous tokens, and all future tokens are masked. In addition, MTM uses a modality-specific encoder to elevate the raw trajectory inputs to a common representation space for the tokens. Finally, MTM is trained with a range (between 0.0 and 0.6) of randomly sampled masking ratios.

D Experimental Details and Hyperparameters

In this section, we provide more details about the experiments, including hyperparameter configuration and details of each environment (*e.g.*, version). For all experiments with C-GPT and the baselines, we run 64 rollouts with different random seeds and report the mean and 95% confidence interval (CI) across them.

D.1 Fair Comparison

To ensure a fair comparison between the representation models using an MLP or a transformer architecture, we made sure to have a comparable number of parameters. Both models consist of a minimum of three layers with a size of 256 for the baseline, while C-GPT uses a single layer with a size of 512 for the policy. Consequently, the maximum number of parameters comprised in the parameter-efficient fine-tuned versions (using $(IA)^3$) of PASTA methods is around 140k, which is comparable to that of the RL policies trained from raw observations.

In addition, we choose to fine-tune the MLP baselines to achieve the best performance in each environment. In contrast, we use the same set of hyperparameters for all domains involving PASTA models. This approach ensures a fair comparison between the two methods, with PASTA at a slight disadvantage. However, it also holds the promise of potentially achieving even better performance with the PASTA methods.

Finally, when comparing different pre-training methods, we save 10 checkpoints during the course of training and evaluate them all on the Imitation Learning task. For each method we select the one with highest performance and use it as initialization for all the other downstream tasks.

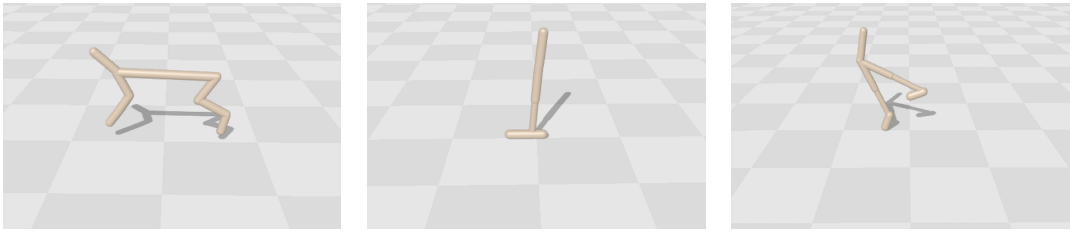

Figure 6: Continuous Control Downstream Tasks.

D.2 Environment Details

For all experiments, we use the 0.0.15 version of Brax (Freeman et al., 2021a). Each environment in Brax, illustrated in Figure 6, provides a realistic physics simulation, enabling agents to interact with objects and the environment in a physically plausible manner. The tasks studied in this paper feature (i) a HalfCheetah robot (Wawrzyński, 2009) with 9 links and 8 joints. The objective is to apply torques on the joints to make the cheetah run forward as fast as possible. The action space for the agents consists of a 6-element vector representing torques applied between the different links; (ii) a Hopper robot (Erez et al., 2011) which is a two-dimensional one-legged figure consisting of four main body parts: the torso, thigh, leg, and foot. The objective is to make hops in the forward direction by applying torques on the hinges connecting the body parts. The action space for the agent is a 3-element vector representing the torques applied to the thigh, leg, and foot joints; (iii) a Walker robot (Erez et al., 2011) which is a two-dimensional two-legged figure comprising a single torso at the top, two thighs below the torso, two legs below the thighs, and two feet attached to the legs. The objective is to coordinate the movements of both sets of feet, legs, and thighs to achieve forward motion in the right direction. The action space for the agent is a 6-element vector representing the torques applied to the thigh, leg, foot, left thigh, left leg, and left foot joints.

D.3 Dataset Details

In this section, we provide further detail on the collection of the datasets. We trained 10 SAC (Haarnoja et al., 2018) agents for a total of 5 million timesteps in each environment. From each, we select the 20% latest trajectories of size 1000, resulting in a combined total of 30 million transitions. With each environment comprising different observation and action sizes, the overall multi-domain dataset is composed of 510 million tokens. We also have one dataset for each domain.

Next, we give the hyperparameters of the SAC agents used to collect the pre-training trajectories. These are given in Table 3.

Table 3: Hyperparameters used in SAC.

Hyperparameter	**Value**
Adam stepsize	$3 \cdot 10^{-4}$
Discount (γ)	0.99
Replay buffer size	10^6
Batch size	256
Nb. hidden layers	2
Nb. hidden units per layer	256
Nonlinearity	ReLU
Target smoothing coefficient (τ)	0.005
Target update interval	1
Gradient steps per timestep	1
Training steps	20,000

We also provide a concrete example of the state and action components with their corresponding properties for the simplest robot structure, Hopper. The number of components for each property

is given in parentheses. In this case, the action space consists of torques applied to the rotors (3), while the observation space includes the following components: z-coordinate of the top (1), angle (4), velocity (2), and angular velocity (4).

D.4 Downstream Tasks Details

In this section, we provide the hyperparameters used in the training of the imitation learning algorithm Behavioural Cloning (BC) (Table 4) and the offline RL algorithm TD3-BC (Table 5).

Table 4: Hyperparameters used in the BC downstream task.

Parameter	Value
Horizon T	1000
Batch Size	1024
Non-Linearity	GELU (Hendrycks & Gimpel, 2016)
Nb. hidden layers	1
Nb. hidden units per layer	512
Adam stepsize	$3 \cdot 10^{-4}$
Training steps	8,000

Table 5: Hyperparameters used in the TD3-BC downstream task.

Parameter	Value
Horizon T	1000
Batch Size	1024
Discount γ	0.99
Non-Linearity	GELU (Hendrycks & Gimpel, 2016)
Nb. hidden layers	1
Nb. hidden units per layer	512
Adam stepsize (actor)	$1 \cdot 10^{-4}$
Adam stepsize (critic)	$3 \cdot 10^{-4}$
Target update rate	$5 \cdot 10^{-3}$
Policy noise	0.2
Policy noise clipping	(-0.5, 0.5)
Policy update frequency	2
Conservatism coefficient α	2.5
Training steps	90,000

Then, we give additional details about the sensor failures downstream tasks. In Table 6, 7 and 8 we include the correspondence between each sensor number and its associated name in all environments. In the 11 downstream tasks, we switch off each one of these sensors.

Finally, in the dynamics change downstream tasks, we vary the gravity coefficient (after training with a coefficient of 1) by multiplying it with the following constant values: 0.1, 0.25, 4, and 10.

Table 6: Sensor name / Sensor number in Halfcheetah.

Sensor name	Sensor number
z-coordinate of the center of mass	1
w-orientation of the front tip	2
y-orientation of the front tip	3
angle of the back thigh rotor	4
angle of the back shin rotor	5
angle of the back foot rotor	6
velocity of the tip along the y-axis	7
angular velocity of front tip	8
angular velocity of second rotor	9
x-coordinate of the front tip	10
y-coordinate of the front tip	11

Table 7: Sensor name / Sensor number in Hopper.

Sensor name	Sensor number
z-coordinate of the top (height of hopper)	1
angle of the top	2
angle of the thigh joint	3
angle of the leg joint	4
angle of the foot joint	5
velocity of the x-coordinate of the top	6
velocity of the z-coordinate (height) of the top	7
angular velocity of the angle of the top	8
angular velocity of the thigh hinge	9
angular velocity of the leg hinge	10
angular velocity of the foot hinge	11

Table 8: Sensor name / Sensor number in Walker2d.

Sensor name	Sensor number
z-coordinate of the top (height of hopper)	1
angle of the top	2
angle of the thigh joint	3
angle of the leg joint	4
angle of the foot joint	5
angle of the left thigh joint	6
angle of the left leg joint	7
angle of the left foot joint	8
velocity of the x-coordinate of the top	9
velocity of the z-coordinate (height) of the top	10
angular velocity of the angle of the top	11

D.5 HYPERPARAMETERS

In Table 9, we show the hyperparameter configuration for C-GPT across all experiments.

Table 9: Hyperparameters and configuration details for C-GPT across all experiments.

Hyperparameter	**Value**
Transformer Layers	10
Transformer Heads	8
Noising Ratio	0.15
Masking Probability	0.8
Random Token Probability	0.1
Non-Linearity	GELU
Learning Rate	$3e-4$
Num Epochs	3
Batch Size	4096
Num Quantization Tokens	1024
Embedding Dimension	256