PASTA: Pretrained Action-State Transformer Agents

Raphaël Boige* **Yannis Flet-Berliac***
Lars C.P.M. Quaedvlieg **Guillaume Richard** **Arthur Flajolet** **Thomas Pierrot†**
InstaDeep

Abstract

Self-supervised learning has brought about a revolutionary paradigm shift in various computing domains, including NLP, vision, and biology. Recent approaches involve pretraining transformer models on vast amounts of unlabeled data, serving as a starting point for efficiently solving downstream tasks. In reinforcement learning, researchers have recently adapted these approaches, developing models pretrained on expert trajectories. However, existing methods mostly rely on intricate pretraining objectives tailored to specific downstream applications. This paper conducts a comprehensive investigation of models, referred to as pre-trained action-state transformer agents (PASTA). Our study covers a unified framework and covers an extensive set of general downstream tasks including behavioral cloning, offline Reinforcement Learning (RL), sensor failure robustness, and dynamics change adaptation. We systematically compare various design choices and offer valuable insights that will aid practitioners in developing robust models. Key findings highlight improved performance of component-level tokenization, the use of fundamental pretraining objectives such as next token prediction or masked language modeling, and simultaneous training of models across multiple domains. In this study, the developed models contain fewer than 7M parameters 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 principle 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 efficient agents to learn optimal decision-making policies. This approach has led to remarkable achievements in diverse fields, including gaming and robotics (Silver et al., 2014; Schulman et al., 2016; Lillicrap 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 multifaceted networks poses challenges due to their nested nature (Boyan & Moore, 1994; Anschel et al., 2017) and the necessity to extract meaningful features from state-action spaces, coupled with assigning appropriate credit in complex decision-making scenarios. 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 pretraining 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

*Equal Contribution
†Corresponding author: t.pierrot@instadeep.com

extraction from task-solving (Laskin et al., 2020; Nachum & Yang, 2021; Eysenbach et al., 2022). Alternatively, pretraining 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 pretraining 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 using transformer neural networks, most self-supervised techniques use tokenization, representing input data as a sequence of discrete elements called tokens. Once the data is transformed, simple 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 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 tokenization schemes. In addition, we outline a standardized set of downstream tasks for evaluating the transfer learning performance of pretrained models, ranging from behavioral cloning to offline RL, robustness to sensor failure, and adaptation to changing dynamics.

Our contributions. The PASTA study, which stands for pretrained action-state transformer agents, provides comprehensive comparisons including 4 pretraining objectives, two tokenization techniques, 5 pretraining datasets (from Brax and Atari), and 7 downstream tasks. In addition to imitation learning and standard RL, we explore scenarios involving 4 physical regime changes and 11 observation alterations to assess the robustness of the learned representations. Finally, we assess the zero-shot performance of the models for predictions related to decision-making. 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 principle objectives over convoluted ones.** First principle training objectives, such as random masking or next-token prediction with standard hyperparameters match or outperform more intricate and task-specific objectives carefully designed for RL, such as those considered in MTM or SMART.

3. **Pretrain the same model on datasets from multiple domains.** Simultaneously pretraining the model on datasets from all environments leads to enhanced performance across all tasks compared to training separate models for each individually.

4. **Generalize with a small parameter count.** All of the examined models have fewer than 7M parameters. Hence, while these approaches are both affordable and practical even on limited hardware resources, the above results are corroborated by experimentation with 4 transfer learning scenarios: a) probing (the pretrained models generate embeddings and only the policy head is trained to address downstream tasks), b) last layer fine-tuning (only the pretrained model's last layer is fine-tuned), c) full fine-tuning and d) zero-shot transfer.

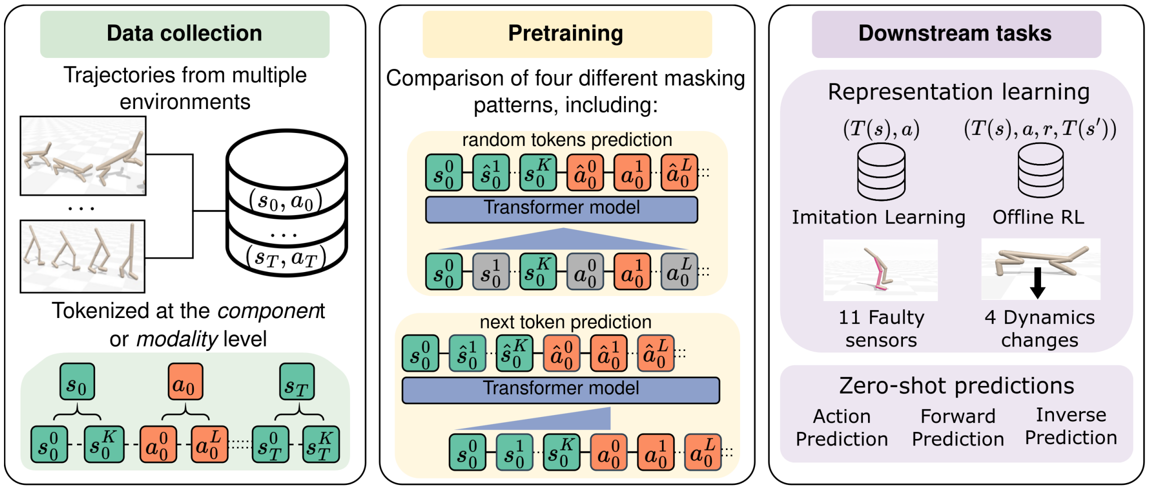

Figure 1: Illustration of the PASTA study. **Left:** State-action trajectories are collected from multiple environments and tokenized. **Middle:** A transformer model learns latent representations $T(s)$ of the environments' states. In this study, we compare different masking patterns, *e.g.*, random tokens prediction (BERT) or next token prediction (GPT). **Right:** The learned representations $T(s)$ serve as surrogate states for the policy and are evaluated on multiple downstream tasks.

2 Related Work

Self-supervised Learning for RL. Self-supervised learning trains models using unlabeled data and has been successful 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 has proven valuable in 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 pretrain RL policy 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 (Kumar et al., 2019; Fujimoto & Gu, 2021; Fakoor et al., 2021; Dong et al., 2023) or constrain the learning process within the support of the dataset via importance sampling (Sutton et al., 2016; Nair et al., 2020; Liu et al., 2022b). 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 IL 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. Transformer-based models, notably the decision transformer (Chen et al., 2021) and trajectory transformer (Janner et al., 2021), have proven effective in offline RL by implementing a causal transformer structure for direct reward-conditioned policy fitting, inspiring further developments (Zheng et al., 2022; Yamagata et al., 2022; Liu et al., 2022a; Lee et al., 2023; Badrinath et al., 2023). Notably, GATO (Reed et al., 2022) is a multi-modal behavioral cloning method that directly learns policies. These work contrast with PASTA which studies pretrained self-supervised representations learned from different masking patterns and objectives. MTM (Wu

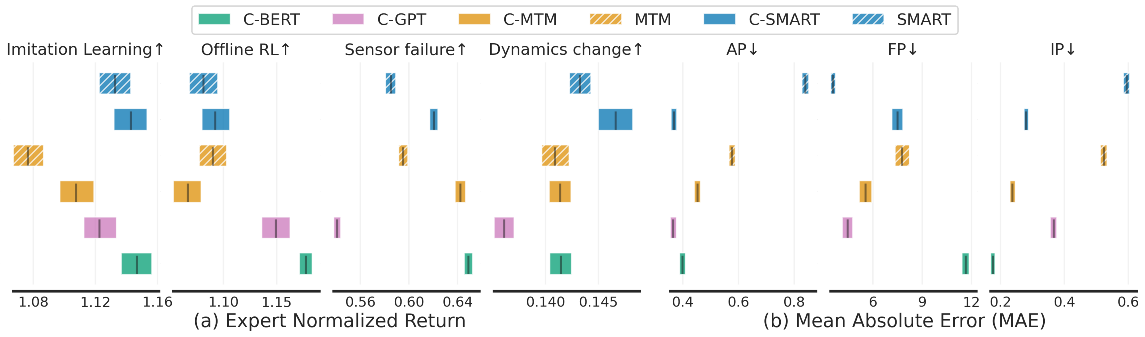

Figure 2: Interquartile Mean (IQM) of the expert-normalized scores, aggregated over all 4 environments, computed with stratified bootstrap confidence intervals (95% CI) over 5 seeds and 256 rollouts. ↑ (resp. ↓) indicates that higher (resp. lower) is better. **(a)** Representation learning and **(b)** Zero-shot transfer tasks.

et al., 2023) and SMART (Sun et al., 2023) are relevant to this study: MTM uses modality-level masking for single-domain pretraining whereas SMART uses a comprehensive objective involving forward and inverse predictions in addition to "random masked hindsight control" for cross-domain generalization with real-valued visual observations.

3 The PASTA Study

3.1 Preliminaries

Self-supervised Learning. In this paper, we study self-supervised learning (Balestriero et al., 2023) techniques to pretrain 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 pretraining 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 pretraining the models, we evaluate their capabilities to solve downstream tasks. This analysis is done through the lenses of 3 mechanisms: (i) probing, (ii) fine-tuning, and (iii) zero-shot transfer. The goal of the study is to investigate which pretraining 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. 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. In the infinite-horizon setting, we seek a policy that optimizes $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right)]$.

3.2 Component-level Tokenization

A key focus of this study is the *component-level* representation of the states and actions, *i.e.*, their vector components are dissected into individual tokens, as depicted in the middle panel of Figure 1, rather than at the *modality-level* where one state corresponds to one token. Most previous work, including SMART (Sun et al., 2023) and MTM (Wu et al., 2023) use the *modality-level* and consider a trajectory as a sequence of state-action (often -return) tuples. Instead, in this study, we break the sequences down to individual state and action *components* and exclude the return to allow applicability to reward-free settings and the learning of representations not tied to task-specific

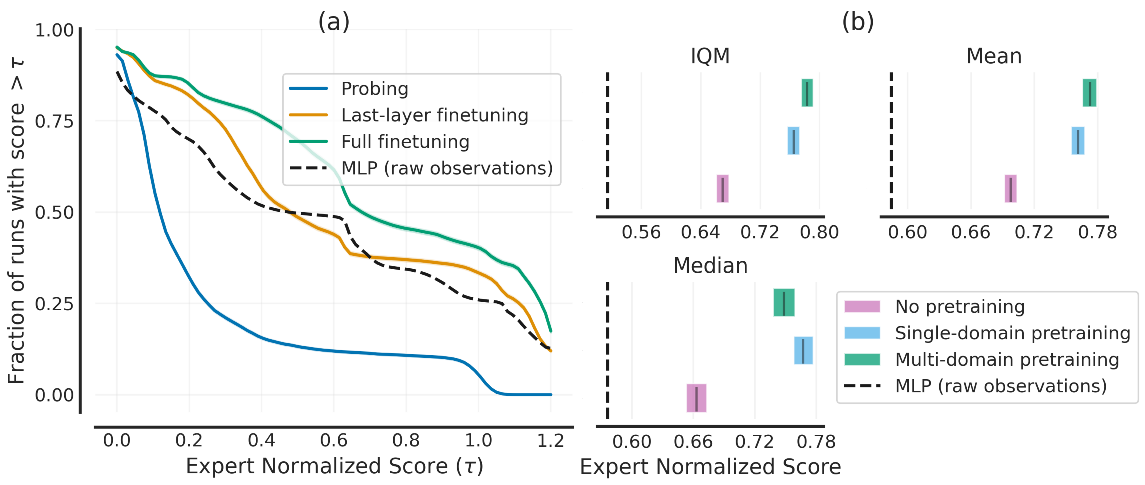

Figure 3: **(a)** Performance profile of models after full, last layer and no fine-tuning (probing), and MLP policies trained from raw observations. Shaded areas show bootstrapped CI over 5 seeds and 256 rollouts. **(b)** Evaluation in all downstream tasks with multi- and single-domain pretraining, no-pretraining and training from raw observations. Remarkably, multi-domain pretraining performs better or on par with single-domain pretraining, despite being trained on the same amount of data.

rewards (Stooke et al., 2021; Yarats et al., 2021). Based on our experimental results (Section 4), we argue that *component-level* level tokenization allows capturing better dynamics and dependencies at different space scales. As a result, more generalizable representations are learned that improve the performance of downstream tasks across different robotic structures.

3.3 Pretraining

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.

Pretraining objectives. Our study explores various masking strategies for pretraining. 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, the C-BERT masking pattern is derived from BERT and uses random masks to facilitate diverse learning signals from each trajectory by enabling different combinations. Figure 1 (middle) illustrates C-BERT's and C-GPT's 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 a certain proportion of the last tokens. We refer to this method as C-MTM, *i.e.*, component-level MTM. Finally, SMART uses 3 different masking patterns (Sun et al., 2023): forward prediction, inverse prediction and "random masked hindsight control". 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.

Table 1: Expert-normalized returns obtained with representations learned from modality-level tokenization, component-level tokenization, and from an MLP policy network in the 4 representation learning downstream tasks. We include the maximum performance obtained using modality- or component-level tokenization. (↑) indicates higher is better and [11] means 11 variations per task. We trained all methods with 5 different random seeds and evaluated them using 256 rollouts.

Domain	Task	MLP (raw observations)	Modality-level tokenization	Component-level tokenization
HalfCheetah	IL (↑) [1]	1.132 ± 0.003	**1.151 ± 0.003**	**1.154 ± 0.003**
	Offline-RL (↑) [1]	0.571 ± 0.030	**1.152 ± 0.004**	**1.154 ± 0.003**
	Sensor failure (↑) [11]	0.896 ± 0.003	1.006 ± 0.002	**1.048 ± 0.002**
	Dynamics change (↑) [4]	0.251 ± 0.003	0.339 ± 0.003	**0.369 ± 0.004**
Hopper	IL (↑) [1]	0.898 ± 0.022	0.847 ± 0.019	**1.078 ± 0.021**
	Offline-RL (↑) [1]	0.890 ± 0.022	0.812 ± 0.020	**0.971 ± 0.022**
	Sensor failure (↑) [11]	0.307 ± 0.005	0.554 ± 0.006	**0.584 ± 0.007**
	Dynamics change (↑) [4]	0.169 ± 0.035	**0.290 ± 0.035**	**0.290 ± 0.038**
Walker2d	IL (↑) [1]	0.736 ± 0.010	1.128 ± 0.029	**1.178 ± 0.031**
	Offline-RL (↑) [1]	0.911 ± 0.025	0.923 ± 0.025	**1.046 ± 0.023**
	Sensor failure (↑) [11]	0.339 ± 0.003	0.419 ± 0.003	**0.511 ± 0.003**
	Dynamics change (↑) [4]	0.000 ± 0.000	**0.004 ± 0.001**	**0.005 ± 0.001**
Ant	IL (↑) [1]	0.876 ± 0.032	**1.203 ± 0.008**	**1.209 ± 0.005**
	Offline-RL (↑) [1]	0.846 ± 0.030	0.907 ± 0.035	**1.213 ± 0.021**
	Sensor failure (↑) [11]	0.082 ± 0.004	0.615 ± 0.007	**0.717 ± 0.007**
	Dynamics change (↑) [4]	0.015 ± 0.001	0.065 ± 0.001	**0.068 ± 0.001**

3.4 Downstream evaluation

In this study, we evaluate the learned representations from two perspectives: (i) their ability to generate high-quality representations through probing, full fine-tuning, and last layer fine-tuning (4 Representation learning tasks), and (ii) their capability to solve new tasks in a zero-shot transfer setting (3 Zero-shot transfer tasks). **Representation learning tasks:** We use Imitation Learning, Offline RL, Sensor Failure, and Dynamics Change. First, we evaluate the quality of raw representations learned by pretrained agents using probing, *i.e.*, the pretrained models weights are frozen and the final attention layer's embeddings are fed into a single dense layer network. Second, we assess the quality of the representations through full fine-tuning and last layer fine-tuning, *i.e.*, the weights of the pretrained agents are further updated to solve the downstream tasks. Fine-tuning just the last layer updates only a small fraction of the total weight volume (<1M parameters), enhancing efficiency and computational cost. In all settings, a held-out dataset is used for training on the downstream tasks, and in the Sensor Failure and Dynamics Change tasks, the alterations are only introduced when evaluating the learned policies in the environments. **Zero-shot transfer tasks:** These tasks entail Action Prediction (AP), Forward Prediction (FP), and Inverse Prediction (IP). They evaluate the pretrained 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 of the impact of pretraining objectives, tokenization, and dataset preparation on the generalization capabilities of pretrained PASTA models.

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 MuJoCo versions (Todorov et al., 2012) while offering a highly flexible and scalable framework for simulating robotic systems. More information about the environments is given in Appendix D.2. The pretraining datasets consist of trajectories collected from 4 environments: HalfCheetah, Hopper, Walker2d and Ant. 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 680M tokens in total. For details about the pretraining datasets, we refer the reader to Appendix D.3.

To assess the reproducibility of our findings and compare the performance of multi-domain versus single-domain pretrained models, we provide 7 downstream tasks across 4 environments, totaling 28 tasks, with further details in Appendix D.4.

Furthermore, we validate the generalization of our findings on a different domain with experiments on Atari 2600 (Bellemare et al., 2013). We refer the reader to Appendix E for details and results.

Implementation details. In this study, we focus on reasonably sized and efficient models, typically consisting of around 7M 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 details are provided in Appendix B. To convert state or action components into tokens, we adopt a tokenization scheme similar to Reed et al. (2022). Continuous values are mu-law encoded to [-1, 1] and discretized into 1024 uniform bins. The sequence order follows observation tokens then action tokens with transitions arranged in timestep order. Finally, we put in perspective the performance of the different pretrained models by comparing them to an agent learning directly from raw observations without pretraining. For fairness, its hyperparameters have been tuned by taking the best performance for each domain and downstream task.

4.2 Results

Component-level Tokenization. Our initial analysis probes the influence of tokenization, how finely we dissect the data (component- or modility-level), on the models' performance. We train models using both the SMART and MTM protocols at two levels of granularity: modality-level (predicting entire observations and actions) for SMART and MTM, and component-level (predicting individual observation and action elements) for C-SMART and C-MTM. Despite sharing identical architectures and training conditions, and being trained on the same multi-domain dataset, the models' fine-tuning performance vary. As depicted in Figure 2 (a), component-level tokenization markedly enhances performance across a spectrum of tasks, including Imitation Learning, Offline RL, variations of Sensor Failure, and Dynamics Change tasks. Furthermore, Table 1 provides a breakdown of performance for both tokenization techniques across different domains. Our experiments on the Atari domain in Appendix E reveal the same conclusion, *i.e.*, transitioning from modality-level to component-level tokenization improves performance.

Masking objectives. In the light of the previous section demonstrating the advantages of using component-level tokenization, we design C-BERT for masked language modeling and C-GPT for next token prediction. In this section, we compare these two fundamental masking approaches against the state-of-the-art methods C-MTM and C-SMART which incorporate more tailored design choices. These models are trained on the multi-domain dataset and we systematically finetune all models for all downstream tasks and domains. Figure 2 (a) reveals that C-BERT exhibits on average higher performance on the considered downstream tasks compared to other masking schemes and training objectives. Based on C-BERT showing the best performance among other models, it is selected for further analysis within this study. Our experiments on Atari shown in

Figure 6 in Appendix E reveal that C-GPT and C-BERT outperform all other models, confirming that simpler masking objectives are sufficient to achieve robust generalization performance.

Multi-domain representation. Our exploration of learning multi-domain representations via component-level tokenization reveals that models pretrained in this manner outperform those that are specialized and trained on single-domain data. Multi-domain representations also surpass models randomly initialized, confirming the positive impact of pretraining on model performance, and vanilla MLP policy networks. These findings, illustrated in Figure 3(b), underscore the benefits of leveraging diverse domain knowledge, thereby enhancing the model's ability to generate useful representations across various tasks and domains. This suggests that multi-domain models effectively consolidate the representation knowledge from various domains into a unified model. To ensure a fair comparison, all models were trained on an equal amount of tokens and possess equivalent representation capabilities in terms of architecture and learned parameters. Detailed results for each specific task can be found in the appendices, specifically in Appendix A.1.

Fine-tuning and Zero-shot. Figure 3 (a) presents the performance profiles for various fine-tuning strategies: probing (the transformer's parameters are frozen), last layer fine-tuning (only the last layer's parameters are trained) and full fine-tuning (all the parameters are trained). Full fine-tuning results in a higher fraction of runs achieving near-expert scores, followed by last-layer fine-tuning, MLP, and probing. This shows that fine-tuning appears to bridge the gap between the generic representations and the specialized requirements of the downstream tasks. We further study the zero-shot capabilities of the pretrained models which we evaluate on an additional suite of tasks, outlined in Section 3.4, originally introduced in MTM (He et al., 2022). Figure 2 (b) reveals that the errors in Action Prediction (AP), Forward Prediction (FP), and Inverse Prediction (IP) for C-GPT and C-BERT are on par with those of more sophisticated models like C-MTM or C-SMART. This suggests that even simple pretraining objectives are well-aligned with the inference tasks, despite the models not being explicitly trained for these tasks. Such findings reinforce our conclusion that simple objective functions and masking patterns combined with component-level tokenization are sufficient to produce good performance. Importantly, we note that the masking strategy of C-BERT and C-GPT allows the emergence of competitive Action Prediction (AP) performance, which, according to the results in Figure 2 (a) is sufficient and indicative of strong downstream performance.

Robust representations. Here, 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 in Appendix A.2, multi-domain models exhibit higher performance compared to the baselines, demonstrating their enhanced robustness in handling sensor failures. Furthermore, we introduced 4 gravity changes during the inference phase, and the results reaffirm the resilience of multi-domain learning in adapting to dynamics change, corroborating our previous findings.

5 Discussion

This paper introduces the PASTA study which investigates self-supervised pretrained transformers for RL applications. The study contributes analyses across 4 training objectives, 2 tokenization methods, 5 training datasets and 7 downstream tasks. PASTA evaluates the efficacy of different design choices in probing, fine-tuning, and zero-shot evaluation in the Brax and the Atari domains.

Key findings include the effectiveness of *simple self-supervised objectives* such as random masking or next token prediction over more complex ones. *Component-level tokenization* proved superior to modality-level, underscoring the importance of finer tokenization for richer representations. Additionally, *multi-domain pretraining* led to better performance than domain-specific training, demonstrating the value of cross-domain knowledge transfer. Finally, results highlighted these models' adaptability to sensor failure or dynamic change, mitigating risks in robotics applications.

We hope PASTA will provide valuable guidance to researchers interested in leveraging self-supervised learning for RL in complex decision-making tasks. The models studied are relatively lightweight, enabling the replication of both pretraining and fine-tuning experiments on readily available hardware. In future work, we anticipate further exploration of self-supervised objectives, tokenization methods, and a broader spectrum of tasks to evaluate adaptability to online learning.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

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.

Anirudhan Badrinath, Yannis Flet-Berliac, Allen Nie, and Emma Brunskill. Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets, 2023.

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.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

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.

Felix Chalumeau, Raphael Boige, Bryan Lim, Valentin Macé, Maxime Allard, Arthur Flajolet, Antoine Cully, and Thomas PIERROT. Neuroevolution is a competitive alternative to reinforcement learning for skill discovery. In *The Eleventh International Conference on Learning Representations*, 2022.

Felix Chalumeau, Bryan Lim, Raphael Boige, Maxime Allard, Luca Grillotti, Manon Flageat, Valentin Macé, Arthur Flajolet, Thomas Pierrot, and Antoine Cully. Qdax: A library for quality-diversity and population-based algorithms with hardware acceleration. *arXiv preprint arXiv:2308.03665*, 2023.

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.

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.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

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.

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.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 694–711. Springer, 2016.

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.

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

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

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.

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.

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 Brax additional results

A.1 Detailed Breakdown of Downstream Tasks Results

We present in Figure 4 the detailed results per-downstream task on the Brax domain.

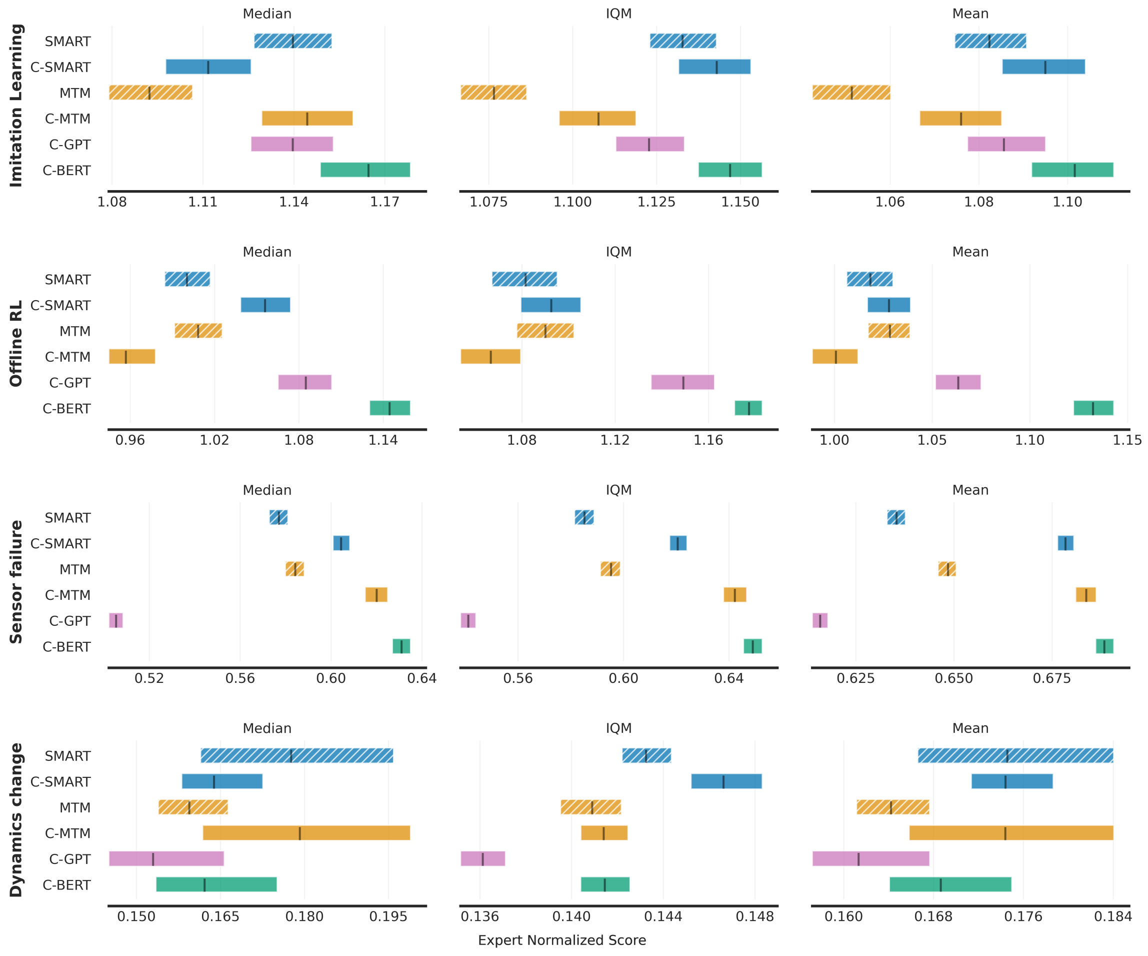

Figure 4: Detailed breakdown of the Mean, Interquartile Mean (IQM) and Median expert normalized scores, computed with stratified bootstrap confidence intervals, obtained in the 4 fine-tuning downstream tasks for the 4 environments HalfCheetah, Hopper, Walker2d and Ant. We repeatedly trained all methods with 5 different random seeds and evaluated them using 256 rollouts.

A.2 Robust Representations

We present in Table 2 the results for different pretraining settings on the task of Sensor Failure and Dynamics Change.

B Sequence Modeling Details

Positional encoding We use two positional-embedding methods to account for the inherent causal nature of the RL trajectories. The first positional embedding is at the component-level, similarly to Reed et al. (2022) we assing an arbitrary order to the components within one state vector or one action vector, and we learn associated representations with a learned embedding layer. For example, in Hopper, one observation consists in 11 components, these components will be indexed

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** (↑)
Multi-domain pretraining	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.17 \pm 0.01}$
Single-domain pretraining	0.66 ± 0.01	$\mathbf{0.18 \pm 0.01}$
MLP (raw observations)	0.41 ± 0.02	0.11 ± 0.01
No pretraining	0.55 ± 0.01	0.16 ± 0.01

from 0 to 10 and passed to the positionnal embedding layer, then these embeddings are added to the tokens embeddings. The second embedding layer is a Rotary Embedding Layer (Su et al., 2021) that accounts for relative positions insisde the sequence, capturing both within-timestep and between-timestep dependencies.

Handling action-spaces in the Brax domain In the sequence tokenization phase, we do not use return conditioning but since the representation models are pretrained on multiple environments and tasks, we use environment conditioning, *i.e.*, du ring 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 take the embedding from the last observed token.

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 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 3 components: forward prediction, inverse 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 pretraining. 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 pretraining 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.

Note that in order to accurately compare different design choice and training objectives, we developed our own implementation of the methods presented in this study.

D Brax 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, we run 256 rollouts with 5 different random seeds and report the mean and stratified bootstrap confidence intervals.

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 3 layers with a size of 256 for the baseline, while transformer models use a single layer with a hidden size of 512 for the policy. We tested bigger architecture for the MLP without performance gain.

Moreover, 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 puts PASTA models at a slight disadvantage while holding the promise of potentially achieving even better performance with the PASTA methods with further hyperparameter tuning.

Finally, when a pretrained model is involved, we always select the final checkpoint after the fixed 3 epochs done over the pretraining dataset.

D.2 Environment Details

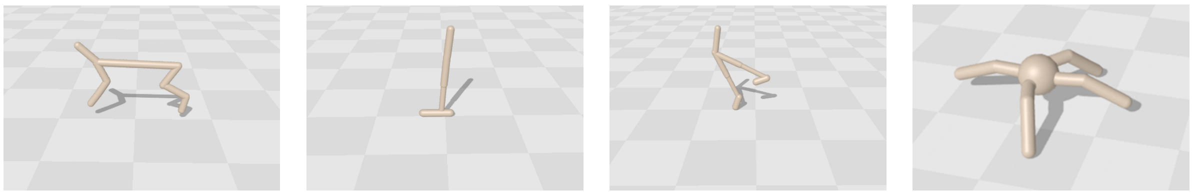
Figure 5: Continuous Control Downstream Tasks.

For all experiments, we use the 0.0.15 version of Brax (Freeman et al., 2021a). Each environment in Brax, illustrated in Figure 5, 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 4 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; (iv) an Ant robot (Schulman et al., 2016) which is a one torso body with 4 legs attached to it with each leg having two body parts. The objective is to coordinate the movements of the 4 legs to achieve forward motion in the right direction. The action space for the agent is an 8-element vector representing the torques applied at the hinge 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 5M timesteps in each of the 4 environments. From each, we select the 20% latest trajectories of size 1000. This choice aims to explore the potential of self-supervised learning in leveraging “expert” knowledge. Future explorations will investigate the impact of diverse data qualities, including suboptimal or exploratory behaviors. This results in a combined total of 40M transitions. With each environment comprising different observation and action sizes, the overall multi-domain dataset is composed of 680M tokens. We also have one dataset for each domain.

Next, we give the hyperparameters of the SAC agents used to collect the pretraining 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.

Hyperparameter	**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	80,000

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

Hyperparameter	**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	140,000

Then, we give additional details about the Sensor Failures downstream task. In Table 6, 7, 8 and 9 we include the correspondence between each sensor number and its associated name in all environments. In the 11 variations of the Sensor Failure downstream task, we switch off each one of these sensors.

Finally, to implement the Dynamics Change downstream task we use the GravityWrapper for Brax environments of the QDax library (Chalumeau et al., 2023) and similarly to Chalumeau et al. (2022) we train the policies with a gravity multiplier of 1 and we vary this coefficient at inference by 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

Table 9: Sensor name / Sensor number in Ant.

Sensor name	**Sensor number**
z-coordinate of the torso (centre)	1
x-orientation of the torso (centre)	2
y-orientation of the torso (centre)	3
z-orientation of the torso (centre)	4
w-orientation of the torso (centre)	5
angle between torso and first link on front left	6
angle between the two links on the front left	7
angle between torso and first link on front right	8
angle between the two links on the front right	9
angle between torso and first link on back left	10
angle between the two links on the back left	11

D.5 Hyperparameters

In Table 10, we show the shared hyperparameters for all transformer backbones used during the pretraining phase as well as C-BERT specific parameters.

Table 10: Hyperparameters and configuration details for transformer backbones.

Hyperparameter	**Value**
Shared	
Transformer Layers	10
Transformer Heads	8
Non-Linearity	GELU (Hendrycks & Gimpel, 2016)
Learning Rate	$3e-4$
Num Epochs	3
Batch Size	4096
Num Quantization Tokens	1024
Embedding Dimension	256
C-BERT specific	
Noising Ratio	0.15
Masking Probability	0.8
Random Token Probability	0.1

E Atari experiments

In this section we present the details of the additional experiments we conducted on the Atari domain.

E.1 Implementation details

The Atari benchmark (Bellemare et al., 2013) offers a playground for agents across 60 Atari 2600 games. Within this domain, the DQN Replay Dataset, as proposed by Agarwal et al. (2020), comprises a collection of five logged training trajectories of models trained using deep Q-learning for each of the 60 Atari 2600 environments. Each of these training trajectories consists of $50,000,000$ transitions. This large dataset allows for reproducible benchmarks of proposed offline RL algorithms (Gulcehre et al., 2020).

We use the DQN Replay Dataset in our experiments on the Atari domain. Since the dataset's observations are images, we choose to encode them using a VQ-VAE (Van Den Oord et al., 2017). This approach follows the methodology outlined by Micheli et al. (2022), and we train the autoencoder by minimizing a combination of the reconstruction loss and a perceptual loss (Johnson et al., 2016). Additional hyperparameters for the VQ-VAE model are described in Table 13. The discrete encodings of these images are then used as tokens for the transformer model.

For the pretraining phase, we closely follow the settings used for the Brax domain, and use the 20% last trajectories to construct the pretraining dataset. See Table 11 for detailed hyperparameters.

For the downstream tasks, we limit ourselves to the Imitation Learning setting. To construct the BC dataset, we follow Gulcehre et al. (2020) and we use all the 50M transitions for each of the 5 seeds in the dataset, resulting in a dataset with 250M transitions equating to about 4.25B tokens.

E.2 Results

Figure 6 compares the performance of different pretraining objectives for the task of Imitation Learning in the Ms-Pacman environment. Consistently with the observations made for the Brax domain, the methods with component-level masking show higher performance than the modality-level ones. Additionally, C-GPT and C-BERT exhibit the best overall performance across Mean, Interquartile Mean (IQM), and Median scores, which strengthen the claim that simple but foundational pretraining objectives can foster the learning of strong representations for RL downstream tasks.

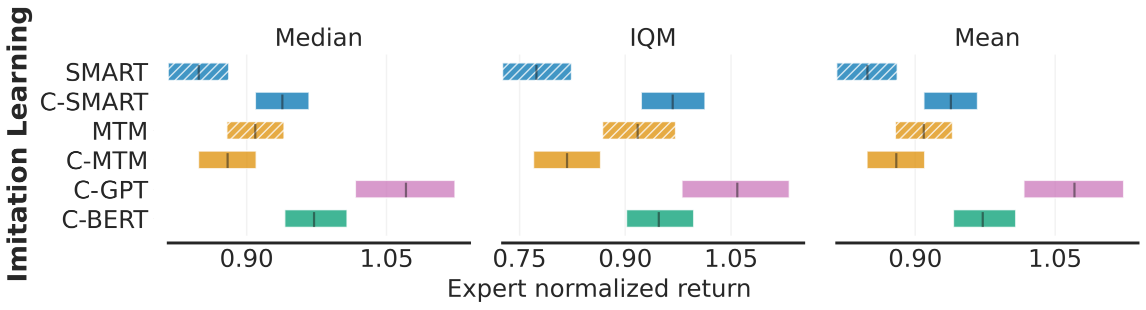

Figure 6: Detailed breakdown of the Median, Interquartile Mean (IQM) and Mean expert-normalized scores, computed with stratified bootstrap 95% confidence intervals, obtained in the fine-tuning downstream task for the Ms-Pacman environment. We repeatedly trained all methods with 3 different random seeds and evaluated them using 128 rollouts.

E.3 Hyperparameters

Table 11 shows the hyperparameters for the pretraining, Table 12 for the Imitation Learning downstream task, Table 13 for the training on the VQ-VAE model used for the Atari domain.

Table 11: Hyperparameters and configuration details shared across all methods for the pretraining in the Atari domain.

Hyperparameter	**Value**
Transformer Layers	4
Transformer Heads	8
Non-Linearity	GELU (Hendrycks & Gimpel, 2016)
Learning Rate	$3e-4$
Num Epochs	4
Batch Size	1024
Num Quantization Tokens	1024
Embedding Dimension	256

Table 12: Hyperparameters used in the BC downstream task.

Hyperparameter	**Value**
Horizon T (in frames)	108K
Frame skip	4
Batch Size	256
Non-Linearity	GELU (Hendrycks & Gimpel, 2016)
Nb. hidden layers	2
Nb. hidden units per layer	256
Adam stepsize	$3 \cdot 10^{-4}$
Training steps	1M

Table 13: Hyperparameters for the VQVAE model.

Hyperparameter	**Value**
Frame dimensions	80×80
Layers	4
Channels in convolutions	64
Codebook size	1024
Embedding Dimension	512
Tokens per frame	16
Self-attention layers at resolution	$[10, 20, 40]$