Contrastive Decision Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract: Decision Transformers (DT) have drawn upon the success of Transformers by abstracting Reinforcement Learning as a target-return-conditioned, sequence modeling problem. In our work, we claim that the distribution of DT's target-returns represents a series of different tasks that agents must learn to handle. Work in multi-task learning has shown that separating the representations of input data belonging to different tasks can improve performance. We draw from this approach to construct ConDT (Contrastive Decision Transformer). ConDT leverages an enhanced contrastive loss to train a return-dependent transformation of the input embeddings, which we empirically show clusters these embeddings by their return. We find that ConDT significantly outperforms DT in Open-AI Gym domains by 10% and 39% in visually challenging Atari domains.

Keywords: Reinforcement Learning, Decision Transformers, Contrastive Loss

1 Introduction

The decision-making is a reasoning process resulting in the selection of a belief or a course of actions that leads to a desired outcome [1]. Decision-making has been conventionally approached by Reinforcement Learning (RL) methods [2] in robotics [3, 4], game-playing [5], and multi-agent collaboration [6, 7]. In RL, decision-making is framed as learning a mapping from a state observation to an action that maximizes the cumulative discounted environment rewards (i.e., return), and has seen success with traditional methods that fit value functions [8] or compute policy gradients [9]. Recently, Chen et al. [10] introduced the Decision Transformer (DT) to the realm of RL to draw from the simplicity and scalability of transformer architectures [11]. DT abstracts the decision-making process in RL as a sequence modeling problem and attempts to learn a *return-conditioned* state-action mapping [10]. The return-conditionality means that given a history of return-state-action tokens, such that the last token represents the desired return at the current-state, the DT predicts the action required to achieve this desired return. DT achieved promising performance in many decision-making problems Chen et al. [10], Lee et al. [12], but there still exists a sizable margin for transformers to improve in the realm of RL as shown in the results of Chen et al. [10] where DT matched or performed worse than existing offline RL baselines in a bevy of tasks.

In this work, we hypothesize that integrating Contrastive Representation Learning (CRL) into the DT architecture will help DT's learn a higher quality state-to-action mapping (i.e., policy) and achieve better convergence results. The objective of CRL is to cluster data representations with respect to their respective classes in order to strengthen the discriminability of the representations generated by Neural Networks (NN) [13, 14]. The power of CRL can be seen from its success in a large range of learning problems [13, 12]. Therefore, we investigate decision transformers trained with a contrastive loss, since DT is an inherently *return-conditioned* system (i.e., the predicted optimal action depends on the magnitude of user-defined returns at the current time-step).

The distribution of the target returns with which the DT is conditioned upon represents a distribution of sub-tasks with which the DT must learn to handle. To simultaneously handle the conflicting gradients of differing tasks, we posit that modifying the DT architecture to include a contrastive loss

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

and an input transformation layer to transform the input data into mutually separated sub-spaces of the input-dimension. We posit that this will implicitly help with the performance of DTs. We accomplish this process by introducing Contrastive Decision Transformers (ConDT), which enables learning discriminable input-space embeddings for DT by categorizing the embeddings and the sub-task to which they belong. Our new modified architecture allows DT to more easily choose the optimal action corresponding to the current state, desired return, and history of information. Our results in Atari and Open-AI-Gym environments show consistent improvements over the conventional DT by achieving significantly higher return-performance. Our contributions are as follows:

1. We propose ConDT, a modified, contrastive DT architecture and training procedure that utilizes CRL and return-dependent embedding-space transformations to effectively discriminate state and action embeddings belonging to different return classes.

2. We propose an enhanced CRL objective, the $\mathcal{L}_{\text{SimRCRL}}$ loss, for optimizing representational distance of state and action embeddings, rather than using the discriminator approach initially proposed by Liu et al. [15]. We provide qualitative and quantitative analysis of the embeddings and show that our $\mathcal{L}_{\text{SimRCRL}}$ results in better *separation* in representation space.

3. We provide empirical results demonstrating that our modified RCRL objective, embedding-space transformation, and the combination of the two (i.e., full ConDT) improves return performance by an average of **10%** in Open-AI Gym and **39%** in Atari. These notable gains indicate the strength of applying CRL to the input embeddings of transformers and has the potential to be applied to other domains where the inputs can be categorized.

2 Related Work

RL has seen great success in decision-making problems in robotics [3, 4], game-playing [5], and multi-agent collaboration [7]. For improved sample-efficiency, researchers have studied offline RL [4], in which the agent learns from a pre-recorded dataset rather than environment interactions. Q-learning is one of the most successful offline methods [8, 16], and with other methods, has benefited from observing a history of states and actions, rather than just the last state [17]. We call this history temporal context. Recurrent-based architectures [18] were generally used with RL to solve domains (i.e., games) with temporal context, and with the introduction of transformers [11, 19], researchers have investigated replacing RNNs with transformers in offline RL [20, 21, 10]. In the following, we discuss the latest applications of transformers to RL with respect to ConDT.

Transformers in RL – Chen et al. [10] introduced Decision Transformer (DT) as a replacement to recurrent-based RL architectures for cases when temporal context is known. Additionally, Janner et al. [22] introduced Trajectory Transformer as a long-term planning framework for RL. In this work, we propose a modular input-embedding space transformation and contrastive procedure (ConDT), which was tested on DT, but applicable to other transformers. ConDT achieves success because its methods effectively categorize the distribution of returns as distinguishable sub-tasks. This is notably distinctive from Multi Game DT (MGDT), which augment the input sequence of DT with observations from multiple-games showing that a single DT can generalize to multiple different games. MGDT trains a DT to solve multiple games, while ConDT trains a DT to better solve the sub-tasks within a single game.

Contrastive Learning for Transformers – Contrastive Learning has been applied to RL, contrasting representations in the state space [14, 23, 24] and the action space [25]. Liu et al. [15] recently introduced Return-Based Contrastive Learning (RCRL) which applies a contrastive loss across the return-space in RL. Contrastive methods for transformers generally take advantage of the InfoNCE loss [14], which has seen success in auto-regressive tasks. The InfoNCE loss has been successfully applied to multi-modal video Transformers [26] and bi-directional language Transformers [13]. These methods are relevant to non-RL domains, but our method, ConDT, attempts to apply RCRL to the input embedding layers of DT, because DT's architecture is inherently return-conditioned.

Pre-Training Objectives for Transformers – Transformers, in both vision and NLP applications, have benefited from pre-training [27, 28]. These works benefit from either introducing pre-training tasks related to the original objective [28], whereas other works perform pre-training on a different training set and then fine-tune on the main set [29, 30]. ConDT builds off the intuition of the former works, but utilizes a return-based contrastive objective to train the input-embedding space, unlike the pretraining techniques introduced by Devlin et al. [28].

3 Preliminaries

Problem Formulation – We consider the problem of learning in a Markov Decision Process (MDP) defined by the four-tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ represents the state-space and $\mathcal{A}$ represents the action space. A particular state and action at a time-step, t is represented by s_t, a_t, respectively. $\mathcal{P}$ represents the transition function $0 \leq P(s_{t+1}|s_t, a_t) \leq 1$, and $\mathcal{R}$ represents the reward function and the reward is defined as: $r_t = R(s_t, a_t)$. In this paper, we follow convention of the original DT and define return, (g_t), as the non-discounted rewards-to-go: $g_t = \sum_t^{\infty} r_t$. As stated earlier, DT is an offline RL method where the learner only from some fixed limited dataset (i.e., environment trajectories) instead of obtaining data via environment interactions. Offline RL is known to be more challenging as it removes an agent's ability to explore [10].

Decision Transformer (DT) Objective – DT [10] takes as input a sequence of three-tokens: $(\langle g_{t-K}, s_{t-K}, a_{t-K} \rangle, \cdots, \langle g_t, s_t, a_t \rangle)$, where K is the game-specific context length. DT encodes each token into an embedding and adds a positional encoding to each embedding. The embeddings are then fed into the GPT-2 Causal Transformer [31] where an attention mechanism is applied to predict a left-shifted version of the input: $(\langle \hat{s}_{t-K}, \hat{a}_{t-K}, \hat{g}_{t-K+1} \rangle, \cdots, \langle \hat{s}_t, \hat{a}_t, \hat{g}_{t+1} \rangle)$. During inference, a_t is unknown and therefore, a_t is a null-token. For training, and for inference, the only relevant output token is $\hat{a}_t$, which represents the model's predicted action that will maintain the return in the current-state. Considering the true-optimal action as a_t and predicted action as $\hat{a}_t$, the training objective of DT in discrete and continuous environments can be shown as in Eq. 1.

$$\mathcal{L}_{\text{DT}} = \begin{cases} \|\hat{a}_t - a_t\|^2, & \text{if continuous,} \\ -a_t \log(\hat{a}_t[a_t]), & \text{if discrete, } a_t \text{ is an integer and } \hat{a}_t \in \mathcal{A} \end{cases} \tag{1}$$

Return-Based Contrastive Learning (RCRL) Objective – RCRL performs contrastive representation learning on state-action embeddings [15]. In any offline dataset, we are given a series of return-state-action tuples, (g_t, s_t, a_t). Returns will range from $[g_{min}, g_{max}]$, which we will split into L distinct return buckets, $\forall_{k \in L} b_k : \big[b_k(\text{lower}), b_k(\text{upper})\big]$, where b_k refers to the k-th return bucket that ranges from $b_k(\text{lower})$ to $b_k(\text{upper})$. RCRL adds a discriminator, $\mathcal{D}(x, y)$, to the network architecture, which returns a scalar indicating the degree of correlation between embeddings x and y. The RCRL loss (Eq. 2) samples a batch of $\mathcal{B}$ anchor state-action embeddings, z_{ah}, whom each have some bin designation, b_{ah}, as well as $\mathcal{B}$ positive embeddings, z_p, whom have bin designation b_p s.t. $b_p = b_{ah}$, and also $\mathcal{B}$ embeddings, z_n, whom each have bin designation b_n s.t. $b_n \neq b_{ah}$.

$$\mathcal{L}_{\text{RCRL}} = \frac{1}{2\mathcal{B}} \sum_{z_{ah}, z_p, z_n} \Big(\big(\mathcal{D}(z_{ah}, z_p) - 1\big)^2 + \big(\mathcal{D}(z_{ah}, z_n)\big)^2 \Big) \tag{2}$$

4 Methodology

Overview – The full architecture of the proposed Contrastive Decision Transformer, ConDT, is shown in Fig. 1. Contrary to the original DT [10], we create a sub-space transformation layer that represents a return-dependent transformation of the state and action embedding spaces. The transformation layer is trained by our modified contrastive objective, the SimRCRL loss introduced in Eq. 4, to discriminate state-action embeddings depending on target return.

Next, we present a theoretical motivation for our following sections on return-based sub-space transformations and our contrastive learning objective, SimRCRL.

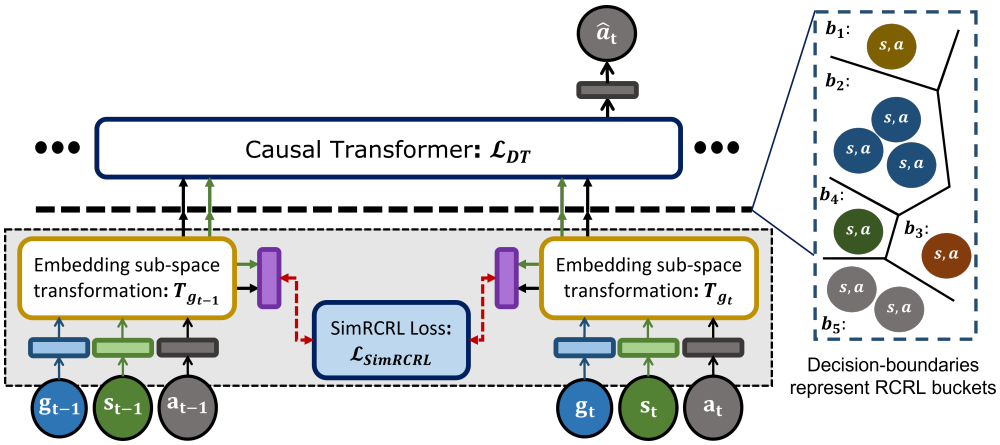


Figure 1: The architecture of the proposed Contrastive Decision Transformer. We create a sub-space transformation layer that represents a return-dependent transformation of the state and action embedding spaces. The transformation layer is trained by our modified contrastive objective, the SimRCRL loss in Eq. 4, to discriminate state-action embeddings depending on target return, as shown in the box on right. The purple box is a state and action embedding compression layer.

Motivation and Insight – In this section we seek to motivate the investigation of return-based transformations of the input embeddings by analyzing a simplistic reduction of the problem formulation. Consider we are given a memory bank $\mathbb{B}$ containing a set of three-tuples (g'_t, s'_t, a'_t) and that the distribution of returns is approximately uniformly distributed. Now, given some g_t and s_t, what is the optimal a_t that could achieve this g_t? A simple search-based solution could use k-nearest neighbors (k-NN), whose complexity is $O(|\mathbb{B}|(|\mathcal{G}| + |\mathcal{S}|)$, where $|\mathcal{G}|$ is the size of the return-space and $|\mathcal{S}|$ is the size of state-space. Upon finding a subset of tuples that match the return, g_t, and state, s_t, their associated a'_t can be chosen as the optimal action. However, imagine if the memory bank was organized by g'_t before-hand, i.e $\{g'_t : \left[(s'_t, a'_t), \cdots\right], \cdots\}$. We can index the memory bank by g_t in $O(1)$, resulting in $|\mathbb{B}|/|\mathcal{G}|$ state-action 2-tuples, and apply the k-NN across the state-space of the 2-tuples to find the optimal a'_t. Therefore, restructuring the memory bank reduces complexity to $O(|\mathbb{B}||\mathcal{S}|/|\mathcal{G}|)$. This is a simplistic reduction of the problem formulation, because transformers effectively search through many combinatorial permutations of (g'_t, s'_t, a'_t), dependent on the context-length; however, this analysis motivates the benefit of pre-indexing the data by return. Since our data is high-dimensional, we turn to the transformations and contrastive methods in the following sections to effectively "index" or "separate" the state and action embeddings by return.

Return-Based Embedding Sub-Space Transformation – As stated in Section 1, the distribution of potential, conditionable returns represents a distribution of sub-tasks which DT must solve. Cheung et al. [32] proposed several transformations that could be applied to the input data, such that data corresponding to different tasks existed in orthogonal sub-spaces of $\mathbb{R}^N$, where N is the dimension of the input, x. They considered a task-dependent context transformation, C_k, where $k \in L$ and L is the distribution of possible tasks. For a single-layer, linear network ($\mathcal{W}$), the output corresponding to the k-th task can be written as $y_k = \mathcal{W}(C_k x)$.

In our formulation, we denote the transformation C_k, as T_{g_t} which indicates a return-dependent, g_t, transformation applied to each embedding space. We note that, here we remove g_t from DT's input set of tokens, since the T_{g_t} should effectively *encode* g_t into the representations of s_t and a_t, the z^s_t and z^a_t, respectively. Therefore, we modify the input to the GPT [31] portion of DT as follows:

$$\text{GPT}\left(\left[T_{g_{t-k}} z^s_{t-K}, T_{g_{t-k}} z^a_{t-K}, \cdots T_{g_t} z^s_t, T_{g_t} z^a_t\right]\right) \rightarrow \left[s_{t-K}, a_{t-K}, g_{t-K+1}, \cdots s_t, a_t, g_{t+1}\right] \quad (3)$$

Now, we need to choose an optimal transformation function, T_{g_t}. To this end, we follow the static rotation-based approach proposed by Cheung et al. [32]. Following the return-bin notation of the RCRL objective in Section 3, we can generate the relevant rotation matrix, R_{b_k}, associated with some given return bucket-label, b_k, by sampling some orthogonal matrix from the Haar Distribu-

tion [32] for each possible bucket. Thus each sequence of (g_t, s_t, a_t) tokens can appropriately be transformed by defining $T_{g_t} = R_{b_k}$, s.t. k is the return bucket index that g_t belongs to. Additionally, we design T_{g_t} to be a learnable transformation, by considering T_{g_t} as a diagonal matrix whose entries are populated by the embedding generated by g_t. In this case, T_{g_t} is simply equivalent to the vector-product of the embedding generated by g_t and the embeddings generated by s_t and a_t. However, with a learnable T_{g_t}, there is no guarantee that the transformed embeddings be distant in representation space. Accordingly, we develop a contrastive objective for T_{g_t}.

Contrastive Learning of the Embedding Transformation – We explore contrastive learning as a direct optimization objective to train T_{g_t} to push state and action embeddings belonging to different b_k to be more distant in representation space. The was also the motivation behind the RCRL loss [15], introduced in Section 3, which claims that training a discriminator on pairs of state-action embeddings will cause the embeddings to grow distant/together in representation space. This claim was supported through an analysis of the cosine-similarity change over time [15], although, in our experiments, we found that optimization of the discriminator required by RCRL **did not always** strictly achieve this result.

Here, we improve upon the RCRL objective, in Eq. 2, by first sampling a batch, $\mathcal{B}$, of bins from L. Then, for each bin we sample two sets of (g_t, s_t, a_t) tuples, where the first is considered the the *anchor* and the second the *positive* sample associated with the anchor. Therefore the loss considers a total of $2\mathcal{B}$ samples. For a given anchor and its associated positive, the rest of the other anchors/positives serve as negative samples (i.e., $2N - 2$), with which we may contrast the embeddings upon. We define the strict-RCRL objective as the normalized-temperature-cross entropy loss of the state-action embeddings, whose construction is drawn from SimCLR [33], which seeks to maximize the dot product similarity between the anchor and its associated positive, while minimizing the dot product similarity with the rest of the samples. Accordingly, our improved contrastive objective, the SimRCRL[1] loss, is defined as in Eq. 4, where the compressed representation of the i-th anchor, which is a state-action embedding, is indicated by z^i_{ah} and its associated positive pair embedding as z^i_p, $\mathbb{1}(i,j)$ is the indicator function that is 0 when $i = j$ and 1 if not, and τ as the temperature hyper-parameter. $\mathcal{B}_C$ is the batch-size used in $\mathcal{L}_{\text{SimRCRL}}$.

$$\mathcal{L}_{\text{SimRCRL}} = \sum_{i=0}^{\mathcal{B}_C} -\log\left(\frac{\exp((z^i_{ah} \cdot z^i_p)/\tau)}{\sum_{j=0}^{\mathcal{B}_C} \mathbb{1}(i,j)[\exp((z^i_{ah} \cdot z^j_{ah})/\tau) + \exp((z^i_{ah} \cdot z^j_p)/\tau)]}\right) \tag{4}$$

Now, putting everything together, our total contrastive loss for training ConDT, $\mathcal{L}_{\text{ConDT}}$, is the sum of DT's objective and our improved contrastive objective, weighted by a β parameter that weighs how much the contrastive objective should affect the DT during training. $\mathcal{L}_{\text{ConDT}}$ is shown in Eq. 5. During experimentation, we consider pre-training the embeddings layers with $\mathcal{L}_{\text{SimRCRL}}$, followed by training the entire DT with a slowed learning rate on the embedding layers and $\mathcal{L}_{\text{DT}}$ loss.

$$\mathcal{L}_{\text{ConDT}} = \mathcal{L}_{\text{DT}} + \beta * \mathcal{L}_{\text{SimRCRL}} \tag{5}$$

5 Empirical Evaluation

Baselines - We tested our improvements upon the original DT architecture Chen et al. [10] by testing the below five methods. For each experiment, we examine the following methods where (1) is the original baseline DT [10] architecture, (2-3) are ablations that study the impact of learnable vs fixed return-transformations of the state and action embeddings, and (4-5) study the combination of (1) and (3) with the $\mathcal{L}_{\text{SimRCRL}}$ contrastive objective:

1. DT – Baseline DT proposed by Chen et al. [10]
2. DT+Rot – Baseline DT with *fixed*-rotation embedding sub-space-transformation
3. DT+Prod – Baseline DT with *learnable* input-embedding transformation

[1]The name SimRCRL comes from combining the SimCLR and the RCRL abbreviations.

4. ConDT w/o Prod (Ours) – Baseline DT trained with our $\mathcal{L}_{\text{ConDT}}$ loss
5. ConDT (Ours) – DT+Prod trained with our $\mathcal{L}_{\text{ConDT}}$ loss

Evaluation Environments – We empirically validate the enhanced performance of ConDT against the baselines in several decision-making domains, including Atari 2600 and Open-AI Gym. For environment descriptions and details, please refer to the provided supplementary material. Next, we present our evaluation results and investigative studies of the learned representations.

5.1 Results, Ablation Studies, and Discussion

Open-AI Gym – We first measure return performance across three standard Open-AI Gym locomotion domains: hopper, halfcheetah, and walker2d. We did not test in the Reacher domain, as done in DT [10], since the dataset for this domain was not publicly available. In each of the mentioned domains, the agent is provided with a continuous observation vector and the agent must output a continuous action vector. Agents are tasked with achieving human-level return performance (i.e., 3600 in hopper, 12000 in halfcheetah, and 5000 in walker2d for results in Table 1). In Gym and Atari, we report the mean, μ, and standard error, SE, of the return achieved in three random seeds.

We train the agents using the *Medium* (M.), *Medium-Replay* (M.R.), and *Medium-Expert* (M.E.) datasets provided by Fu et al. [34]. These datasets differ in size and the policies used to generate them. Thus, we tested on all three variations to study ConDT's robustness. Details of the distinction between these datasets can be found in the supplementary and in the original DT paper [10].

As shown in Table 1, DT+Prod and ConDT w/o Prod exceed the performance of DT, while ConDT exceeds or matches DT+Prod and ConDT w/o Prod in 8/9 tasks. This shows that learnable, return-based transformations benefit return performance, which can be further enhanced with the contrastive training objective. DT+Rot either matches or performs slightly worse than DT, although this trend dramatically falters across the datasets in *Medium-Replay*. We hypothesize the reason for the significant performance dip is because the return distribution in medium-replay is much more left-skewed than medium or medium-expert. This would cause the inputs to DT+Rot to only be transformed with R_{b_k} where b_k corresponded to bins in the lower range of the return-distribution, and thus harm inference when larger returns were provided to DT+Rot.

Dataset	Env.	DT [10]	DT+Rot	DT+Prod	**ConDT w/o Prod** (Ours)	**ConDT** (Ours)
M.	hopper	2097±38.9	1397±25.8	2330±33.2	2293±39.3	**2403**±30.6
	halfcheetah	4981±28.1	5035±36.8	5038±14.9	5020±11.7	**5060±11.7**
	walker2d	3466±56.3	3453±55.9	3521±30.5	**3609±44.8**	3545±49.4
M.R.	hopper	2393±43.4	13.6±0.07	2959±16.3	2869±29.2	**3076±27.7**
	halfcheetah	4070±52.4	703.4±39.4	4791±30.6	4491±35.8	**4803±36.2**
	walker2d	2988±114	968.3±96.1	**3717±65.7**	3322±77.4	3332±76.6
M.E.	hopper	3133±61.0	1528±65.5	3354±56.9	3505±26.3	**3571±8.49**
	halfcheetah	10779±29.2	8956±189	11021±62.1	10821±49.5	**11218±14.5**
	walker2d	4941±12.9	1195±73.4	4951±2.61	4850±35.6	**5011±1.44**

Table 1: Baseline Gym test results for medium (M), medium-replay (M.R.), and medium-expert (M.E.) datasets (as described in [10]) of hopper, half-cheetah, and walker2d, shown as $\mu \pm$ SE

.

Atari 2600 – We also tested on Atari 2600 environments [35], which contains of suite of classic Atari Games that are generally considered challenging due to the high dimensionality of the observation space and time-delayed credit assignment of rewards. We train each baseline on 1% of all samples in the DQN-replay dataset provided by Agarwal et al. [16], representing 500K out of the 50 million transitions observed by an online DQN. In this environment, each agent is provided a $(210, 160, 3)$-sized RGB image observation and must select a discrete action. We tested all the baselines across four domains, Breakout, Qbert, Seaquest, and Pong, and results are shown in Table 2.

Causal attention allows GPT [31] to understand the relation between each token and its preceding sub-sequence. Therefore, the removal of the return tokens from the input to GPT, as is the case

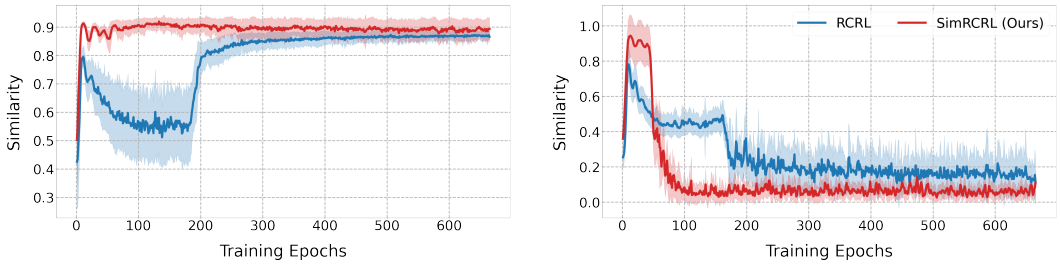
Env.	Target g_t	DT [10]	DT+Rot	DT+Prod	**ConDT w/o Prod** (Ours)	**ConDT** (Ours)
Breakout	90	48.8±6.69	42.7±2.08	70.4±4.73	67.8±4.73	**71.1±2.46**
Qbert	14000	3763±348	6877±302	3677±303	**10735±346**	6432±314
Pong	20	16.8±0.57	11±0.59	16.7±0.29	13.2±0.59	**17.9±0.48**
Seaquest	1150	948±34.7	880±32.1	1018±32.7	**1364±29.0**	1250±27.3

Table 2: Baseline Atari test results for Breakout, Qbert, Pong and Seaquest, shown as $\mu \pm$ SE.

in DT+Rot, DT+Prod, and ConDT, gives rise to the concern of whether the removal will have a negative effect on the performance of the overall DT. As shown in Table 2, even without learnable return-transformation, DT+Rot performs only slightly worse than DT across 3/4 of the Atari domains and doubles the performance of DT in Qbert, indicating return has successfully been encoded into the state and action embeddings. With learned transformations, DT+Prod and ConDT w/o Prod outperform DT in 3/4 of the experiments, except in Pong, where DT+Prod matches DT's performance, but most notably, in Qbert, ConDT w/o Prod nearly triples the performance of DT. Although ConDT does not always outperform ConDT w/o Prod, we find that ConDT **always** results in a universal performance improvement from DT and DT+Prod, and along with ConDT w/o Prod, outperforms human-performance in Seaquest. We hypothesize that ConDT w/o Prod outperforms ConDT in 2/4 tasks, because its causal attention allows it to relate return embeddings from any preceding time-step to a specific state, action embedding, whereas in ConDT, return embeddings are only related to the state, action embeddings at the same specific timestep.

5.2 Investigating the Learned Representations

We hypothesize that our return-based transformation, T_{g_t}, of the input embedding-space with the $\mathcal{L}_{\text{ConDT}}$ objective helps distance the state and action embeddings based on the return-class they belong to. To support this hypothesis, we study how the distance in the state and action embeddings change over time. Additionally, we visually investigate clustering behavior in the embedding-space.



(a) Positive Similarity during Pre-Training

(b) Negative Similarity during Pre-Training

Figure 2: Comparing the loss convergence between $\mathcal{L}_{\text{RCRL}}$ and $\mathcal{L}_{\text{SimRCRL}}$

Cosine Similarity Analysis – For this analysis, we seek to understand how the representation distance of the state-action embeddings from ConDT change over time with respect to the two loss functions, $\mathcal{L}_{\text{RCRL}}$ and $\mathcal{L}_{\text{SimRCRL}}$. The overall goal is to show that $\mathcal{L}_{\text{SimRCRL}}$ results in more stable training and more optimal convergence to the target distance. For this purpose, we utilize the cosine similarity as the distance metric [33]. Here a value of 1 represents parallel embeddings, and 0 represents orthogonal embeddings. During pre-training of ConDT with our modified $\mathcal{L}_{\text{SimRCRL}}$ objective, we sampled a fixed number of data samples across different return bins. Then we measure (1) the average cosine similarity to samples within the same return bin and (2) the average cosine similarity to samples in different return bins, which we call positive and negative similarity, respectively. In Fig. 2a and Fig. 2b, we show the positive and negative similarities from the contrastive training of ConDT for the Atari-Breakout experiment, comparing our $\mathcal{L}_{\text{SimRCRL}}$ objective with the traditional $\mathcal{L}_{\text{RCRL}}$ objective. As seen in the Fig. 2a-2b, $\mathcal{L}_{\text{SimRCRL}}$ results in more stable convergence of the positive-similarity to a value of 0.9 and the negative-cosine similarity to values near 0. This is while

the $\mathcal{L}_{\text{RCRL}}$ loss does not result in a monotonic increase of the positive-similarity, in fact the positive similarity actually decreases over the first 200 epochs, before increasing to a value of near 0.85. Moreover, for the negative cosine-similarity, the $\mathcal{L}_{\text{RCRL}}$ loss converges to a value of 0.2. Therefore, our enhanced $\mathcal{L}_{\text{SimRCRL}}$ objective is more effective in achieving the transformation-based objective we formulated for application to DTs.

Embedding-Space Visualization – Here, we intend to visualize the state-action embedding-space for a comparison between DT and ConDT. The goal is to observe a clustering behavior as result of our embedding-space transformation with contrastive learning, such that the between-class distances would increase and the within-class distances would decrease between the embeddings. To visualize the joint state-action embedding space, we sampled a batch of ten return bins, with 30 samples per bin. Then, we generated the state-action embeddings of this input data for DT and ConDT, and created a 2D TSNE visualization of the embeddings, shown in Fig. 3. As shown, DT demonstrates no clustering behavior of the state-action embeddings based on return; points of the same color are scattered amongst points of different colors (i.e., colors indicate the return bins). However, for ConDT, points of the same color are clearly clustered together and have distance to points of different color, indicating that the contrastive embedding effectively pushes state-action embeddings of different returns apart, and similar returns together.

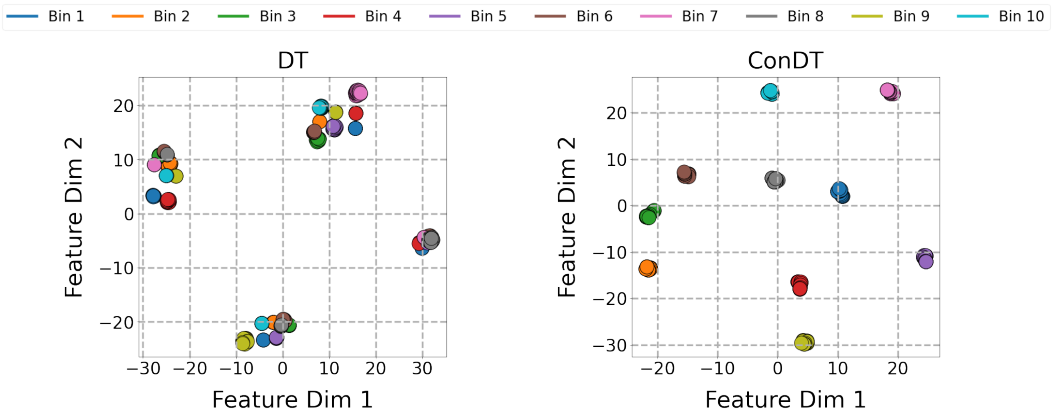


Figure 3: TSNE 2D visualization of state-action embedding-spaces for DT and ConDT.

5.3 Limitations

A limitation DT [10], which is shared in ConDT, is the size and distribution of returns in the offline data. As mentioned in the Open-AI Gym analysis, the left-skewed nature in the return distribution of medium-replay dataset harmed the performance of DT+Rot. The original DT [10] addresses this concern by sampling trajectories according to the magnitude of their cumulative return; thus, causing higher return trajectories to be sampled more often. Therefore, sampling techniques and the return distribution of the dataset can become a bottleneck in decision transformer methods.

6 Conclusion

Decision Transformer (DT) was one of the first introductions of transformers to RL, and while experimentation showed promise, there still lies a performance gap between DT and existing offline RL methods [10]. In this work, we hypothesized that performing a return-dependent transformation of the input embeddings to the DT can help enhance return performance. We proposed Contrastive Decision Transformer (ConDT). ConDT performs an embedding-space transformation of the state and action embeddings, where the transformation is trained using our enhanced $\mathcal{L}_{\text{SimRCRL}}$ objective, which we empirically show can better maximize distance in state-action embedding space. We verify the strength of ConDT's return-based transformation and contrastive learning objective by testing across several Atari and Open-AI Gym tasks, in which we show significant performance gains.

References

[1] W. Edwards. The theory of decision making. *Psychological bulletin*, 51(4):380, 1954.

[2] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[3] M. M. Botvinick. Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, 22(6):956–962, 2012.

[4] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[5] A. Rajeswaran, I. Mordatch, and V. Kumar. A game theoretic framework for model based reinforcement learning. In *ICML*, pages 7953–7963. PMLR, 2020.

[6] S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. *NeurIPS*, 29, 2016.

[7] S. Konan, E. Seraj, and M. Gombolay. Iterated reasoning with mutual information in cooperative and byzantine decentralized teaming. *arXiv preprint arXiv:2201.08484*, 2022.

[8] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 34, 2021.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[12] K.-H. Lee, O. Nachum, M. Yang, L. Lee, D. Freeman, W. Xu, S. Guadarrama, I. Fischer, E. Jang, H. Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

[13] C. Sun, F. Baradel, K. Murphy, and C. Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXive preprint arXiv:1906.05743v2*, 1906.

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

[15] G. Liu, C. Zhang, L. Zhao, T. Qin, J. Zhu, J. Li, N. Yu, and T.-Y. Liu. Return-based contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2102.10960*, 2021.

[16] R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. In *ICML*, pages 104–114. PMLR, 2020.

[17] B. Bakker. Reinforcement learning with long short-term memory. *NeurIPS*, 14, 2001.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] J. Q. Davis, A. Gu, K. Choromanski, T. Dao, C. Re, C. Finn, and P. Liang. Catformer: Designing stable transformers via sensitivity analysis. In *ICML*, pages 2489–2499. PMLR, 2021.

[21] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *ICML*, pages 7487–7498. PMLR, 2020.

[22] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *NeurIPS*, 34, 2021.

[23] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[24] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[25] M. Kim, K. Rho, Y.-d. Kim, and K. Jung. Action-driven contrastive representation for reinforcement learning. *Plos one*, 17(3):e0265456, 2022.

[26] H. Akbari, L. Yuan, R. Qian, W.-H. Chuang, S.-F. Chang, Y. Cui, and B. Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *NeurIPS*, 2021.

[27] L. Wang, F. Liang, Y. Li, W. Ouyang, H. Zhang, and J. Shao. Repre: Improving self-supervised vision transformer with reconstructive pre-training. *arXiv preprint arXiv:2201.06857*, 2022.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[30] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

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

[32] B. Cheung, A. Terekhov, Y. Chen, P. Agrawal, and B. Olshausen. Superposition of many models into one. *NeurIPS*, 32, 2019.

[33] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. *NeurIPS*, 2016.

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

[35] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.