Contrastive Representations for Combinatorial Reasoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Contrastive learning (CL) has emerged as a powerful framework for learning structured representations that enable a wide range of downstream tasks. Its applications span sample-efficient reinforcement learning (RL), retrieval-augmented generation, and improved selection of model-generated samples, among others. Despite these successes, its potential for combinatorial reasoning problems remains largely untapped. In this paper, we take a step in this direction by using temporal contrastive learning to learn representations conducive to solving combinatorial problems, which will reduce our reliance on planning. Our analysis reveals that standard CL approaches struggle to capture temporal dependencies over complex trajectories. To address this, we introduce a novel method that leverages negatives from the same trajectories. Across three complex reasoning tasks, our approach outperforms traditional supervised learning.

1 Introduction

Deep contrastive learning (CL) has achieved remarkable progress in vision [40], control [18], and language modeling [54]. However, it continues to fall short on tasks that require structured, combinatorial reasoning. Even relatively simple problems, such as planning in puzzles or verifying symbolic constraints, remain challenging for end-to-end learning systems [50, 31]. Addressing these problems currently still requires resorting to often computationally expensive search algorithms, such as A* or Best First Search (BFS).

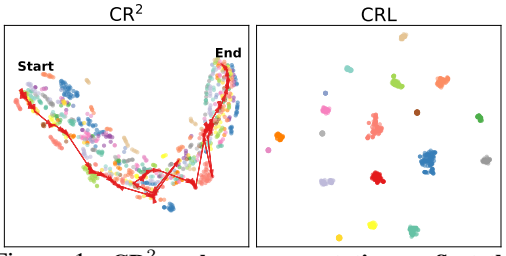

Figure 1: **CR^2 makes representations reflect the structure of the combinatorial task.** t-SNE visualization of representations learned by CR^2 (left) and CRL (right) for Sokoban, with one trajectory highlighted using arrows connecting consecutive points. Colors correspond to trajectories. CRL representations (right) cluster within trajectories, making them useless for planning.

This work centers on the question: *Can we learn structured representations that reduce or eliminate the need for search in combinatorial reasoning tasks?*[1] We approach this question by leveraging temporal contrastive learning [45, 18, 15, 34]. These self-supervised techniques are designed to acquire compact, structured representations that capture the problem's temporal dynamics, enabling efficient planning directly within the latent space.

While CL has shown promise in control tasks, we observe that its effectiveness in combinatorial domains is significantly limited. Specifically, we identify a critical failure mode where contrastive representations overfit to superficial, instance-specific context, hindering the learning of underlying

[1]For instance, finding Euler paths in a graph would be trivial if a deep-learned representation could directly provide information about the degrees of its vertices.

Submitted to the 18th European Workshop on Reinforcement Learning (EWRL 2025). Do not distribute.

environment dynamics. Consequently, models fail to adequately capture the temporal and causal structure that is vital for effective decision-making. This failure mode—such as when the model overfits to wall layouts in Sokoban (Section 4.1)—manifests as a collapse of trajectory representations into small, disconnected clusters, as illustrated in Figure 1 (right).

To solve this, we introduce Contrastive Representations for Combinatorial Reasoning (CR^2), a simple, theoretically grounded CL method that uses in-trajectory negatives. By design, CR^2 forces the model to distinguish between temporally distant states within the *same* episode. This mechanism prevents it from exploiting irrelevant context—such as visual or layout cues—and instead focuses on learning temporally meaningful embeddings that reflect the problem's relevant dynamics. This inductive bias echoes recent findings in neuroscience, where overlapping event trajectories are represented increasingly distinctly in the hippocampus to reduce interference. [7]

We evaluate CR^2 across challenging combinatorial domains: Sokoban, Rubik's Cube, N-Puzzle, Lights Out, and Digit Jumper. Due to their large, discrete state spaces, sparse rewards, and high instance variability, they serve as rigorous testbeds for evaluating whether learned representations can support efficient, long-horizon reasoning. In each case, CR^2 significantly improves planning efficiency over standard contrastive learning, and approaches or surpasses the performance of strong supervised baselines. We also validate our approach on a robotic control task.

Our main contributions are the following:

- We identify and analyze a critical failure mode in standard contrastive learning (CL), showing its inability to capture relevant temporal or causal structure in combinatorial problems.
- We propose Contrastive Representations for Combinatorial Reasoning (CR^2), a novel and theoretically grounded CL algorithm that utilizes in-trajectory negative sampling to learn high-quality representations for combinatorial reasoning.
- Through extensive empirical evaluation, we show that CR^2 improves search efficiency compared to other approaches.

2 Related Work

We build upon recent advances in self-supervised RL and contrastive representation learning, showing that they can be applied successfully to complex combinatorial problems.

Contrastive Learning Contrastive learning has emerged as a widely adopted approach for model pretraining [27, 42]. It facilitates the discovery of rich representations [9, 8] from unlabeled data that improve learning downstream tasks [53], thereby reducing dependence on human annotations. Importantly, contrastive learning enabled effective learning of large-scale models in fields such as computer vision [57, 6], VLMs [40, 30], NLP [48] and real-world applications including RAG [21]. The foundational idea of contrastive learning is to learn representations by pulling similar data points, i.e. ones that belong to the same underlying concept, closer together and pushing dissimilar ones further apart in the representation space [52]. It has been shown that representations learned in this way demonstrate discriminative power for downstream tasks and exhibit properties such as generalization, robustness [49], and transferability [26].

Contrastive Representations for Sequential Problems Recently, self-supervised contrastive learning has been also applied to sequential (or temporal) problems, including goal-conditioned RL [18, 51, 34], skill-learning algorithms [37, 60, 14], or exploration methods [22]. Most temporal-based contrastive algorithms are based on optimizing InfoNCE objective [47] to distinguish real future states in the trajectory from random states. Interestingly, Eysenbach et al. [15] demonstrate that inferring intermediate state representations can be performed by linear interpolation between the initial and final representations. Based on these findings, we hypothesize that such representations might facilitate planning in complex combinatorial problems.

Combinatorial Problems Combinatorial environments are characterized by discrete, compact observations that represent exponentially large configuration spaces, often associated with NP-complete problems [28]. Recent RL advancements address these challenges using neural networks to learn efficient strategies, including policy-based heuristics [33, 2], graph neural networks for structural exploitation [5, 29], and imitation learning with expert demonstrations [46].

Planning in latent space. Planning in complex environments can be made more efficient by leveraging learned state representations. Techniques such as autoencoders have been employed to reduce the dimensionality of the state space and learn world models [23, 24]. Some approaches focus on learning representations that preserve only the features relevant for planning [44, 19]. For robotic applications, latent representations are trained to guide movement and decision-making [25, 19]. Furthermore, [17] frames goal-conditioned planning as a representation learning problem.

3 Preliminaries

Combinatorial problems and dataset properties We focus on combinatorial problems, which can be formulated as deterministic goal-conditioned controlled Markov processes $(\mathcal{S}, \mathcal{A}, p, p_0, r_g, \gamma)$. At each timestep t, the agent observes both state $s_t \in \mathcal{S}$ and goal $g \in \mathcal{S}$, and performs action $a_t \in \mathcal{A}$. We assume that the transition function $p : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$ is known and deterministic, while the initial states might differ as they are sampled from the distribution $p_0(s_0)$. We define reward function $r_g = 1$ for $s_t = g$ and $r_g = 0$ otherwise. The objective is to learn goal-conditioned policy $\pi(a \mid s, g)$ that maximizes the expected reward: $\max_\pi \mathbb{E}_{p_0(s_0), p_g(g)} \left[\sum_{t=0}^{\infty} \gamma^t r_g\left(s_t, a_t\right)\right]$. We study an offline learning setup with a dataset of successful yet suboptimal trajectories $\tau_i = ((s_1, a_1), (s_2, a_2), \ldots (g, -))$. We define the distance function $d : \mathcal{S}^2 \to \mathbb{R}$ as follows: for $s_1, s_2 \in \mathcal{S}$ $d(s_1, s_2) = n$ if s_n is reachable from s_1 in n actions, and there does not exist shorter path between s_1 and s_n. Formally, s_n is reachable from s_1 if there exist a path $a_1, a_2, \ldots, a_n$, such that $s_n = p(a_n, p(a_{n-1}, p(\ldots, p(a_1, s_1))))$.

Contrastive Reinforcement Learning We employ a contrastive reinforcement learning (CRL) method [18] to train a critic, $f(s, a, g)$, which estimates the correlation between the current state-action pair and future states. The critic consists of two embedding networks: one for state-action pairs, ϕ, and another for goals, ψ. These networks generate representations $\phi(s, a)$ and $\psi(g)$, respectively. The energy function, $f_{\phi,\psi}(s, a, g)$, then measures a form of similarity between these representations that reflects the structure of the task. To train the critic, we use the InfoNCE objective [47] as in previous CRL works [18, 16, 59, 58, 34, 3]. Specifically, we construct every batch $\mathcal{B}$, by sampling n random trajectories from the dataset. For each trajectory, we select a state-action pair (s_i, a_i) uniformly and draw goal g_i, using a $\mathrm{Geom}(1-\gamma)$ distribution over future states. Negative pairs consist of state-action pairs (s_i, a_i) and goals g_j from different trajectories. The critic's objective is:

$$\min_{\phi,\psi} \mathbb{E}_{\mathcal{B}} \left[-\sum_{i=1}^{|\mathcal{B}|} \log \left(\frac{e^{f_{\phi,\psi}(s_i, a_i, g_i)}}{\sum_{j=1}^{K} e^{f_{\phi,\psi}(s_i, a_i, g_j)}} \right) \right]. \tag{1}$$

Mutual-Information For two random variables X and Y with joint density $p(x, y)$ and marginals $p(x)$, $p(y)$, the mutual information (MI) can be understood as the KL-divergence between the joint distribution and the product of its marginals:

$$I(X;Y) \;=\; \mathbb{E}_{p(x,y)}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] \;=\; H(X) - H(X\,|\,Y), \tag{2}$$

Intuitively, MI quantifies the reduction in the uncertainty of one random variable from observing the value of the other. In **Conditional Mutual-Information**, $I(X;Y \mid C)$ measures the extra information Y provides about X once the context C is known; it captures the dependence that remains after "factoring out" C. It is zero precisely when X and Y are conditionally independent given C.

$$I(X;Y \mid C) \;=\; \mathbb{E}_{p(x,y,c)}\Big[\log \frac{p(x,y \mid c)}{p(x \mid c)\, p(y \mid c)}\Big] \;=\; H(X \mid C) - H(X \mid Y, C). \tag{3}$$

Search-based planning is a widely used approach for solving complex environments [46, 4, 55, 36]. In our study, we focus on the Best-First Search (BestFS) [38] planner. BestFS builds the search tree by greedily expanding nodes with the highest heuristic estimates, hence targeting paths that are most likely to lead to the goal. While not ensuring optimality, BestFS provides a simple yet effective strategy for navigating complex search spaces. The pseudocode for BestFS is outlined in Appendix B. In our work, we use distances in the latent space as the heuristic, as detailed in Section 3.

4 Method: Temporal Representations that Ignore Context

The main contribution of this paper is a method for learning representations that *facilitate search*. We use an off-the-shelf search algorithm (BestFS) and focus on how distance defined on learned representations can serve as an effective value function for guiding the search. We start by describing how naive temporal contrastive representations, obtained with CRL, fail in combinatorial problems. Using Sokoban as an example, we will highlight why this approach fails (Sec. 4.1), and use it to motivate (Sec. 4.2) a different contrastive objective that better facilitates search on many problems of interest. Section 4.3 summarizes our full method, CR^2, which integrates improved representation learning with BestFS search.

4.1 Failure of Naive CRL in Combinatorial Domains

In this work, we develop a representation learning method that enables the use of distances in representation space as a heuristic for search. We use a neural network $\phi : \mathcal{S} \mapsto \mathbb{R}^k$ to embed observations into k-dimensional representations. The critic output is defined as the norm between these embeddings: $f = ||\phi(s) - \phi(g)||$ (see details in Appendix C).

We assume access to a dataset of trajectories $(s_t)_{t=1..N}$, which could be collected from expert or random policies. Baseline algorithm, CRL (outlined in Sec. 3), fits the representations $\phi(s)$ using temporal contrastive learning and use representation distances as a heuristic function for BestFS.

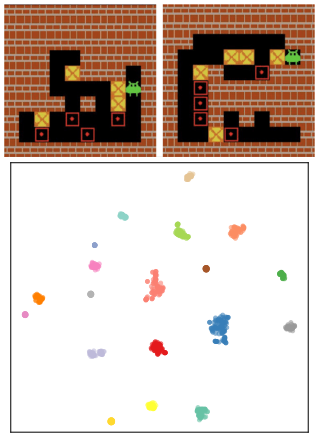

Figure 2: Applying temporal contrastive learning to the Sokoban environment *(Top)* results in representations *(Bottom)* that primarily indicate the layout of walls.

When we applied this approach to the game of Sokoban, we found that it struggled to find effective strategies. Sokoban is a puzzle game where an agent must push boxes to target locations in a maze. Each level (or problem instance) is generated with a random wall pattern, meaning that mazes vary significantly between episodes. Figure 2 shows two Sokoban boards from our dataset. Although they require similar high-level strategies (e.g., box-pushing, avoiding dead ends), their layouts are very different.

We attempted to apply the standard CRL method to this domain. In CRL, positive pairs are sampled from nearby states within the same trajectory, while negatives come from different trajectories. Due to the large variety of wall layouts, each batch element will usually correspond to a different maze. Therefore, looking only at the wall pattern to decide whether two states form a positive pair, results in a perfect accuracy. We demonstrate that using CRL results in a network that does exactly that. Figure 2 shows a 2D t-SNE projection of the learned representations. Embeddings from different mazes form tight, isolated clusters. This confirms that the model is primarily encoding the sokoban board layout, not the temporal structure of the task.

4.2 Learning Representations that Ignore Context

To mathematically understand the failure mode in the example from Sec. 4.1, we start by introducing a *context* random variable that is held constant across time. In Sokoban, we can decompose the state observation into two parts: the static part (positions of walls and box goals) and the dynamic part (positions of player and boxes). The context is this static part. In a general setting, assume that we have a state S, $S = (C, T)$, where C is constant through the trajectory and T varies with time.

Definition 4.1. Context C is a latent variable that parametrizes the distribution over trajectories. Specifically, we assume each trajectory $(X_1, \ldots, X_T) \sim \mathcal{P}(X_{1:T}|C)$, where X_t denotes the state at time t. The context C captures all static properties and initial conditions influencing trajectory evolution and remains constant throughout the trajectory.

For the sake of our analysis, we make the following assumption on the relationship between contexts and trajectories:

Assumption 4.2. For a context C and a trajectory $(X_1, \ldots, X_T) \sim \mathcal{P}(X_{1:T}|C)$, for $i > j$, X_i and C are conditionally independent given X_j. We write $X_i \perp C \mid X_j$.

Note that this holds in Sokoban if we take the context to be the static elements of the states, fixed throughout a trajectory.

Based on this mathematical understanding of the context, we now introduce an alternative method for learning the representations. Our method differs from CRL in a crucial way - it conditions the sampling of positives and negatives on the context. Specifically it works by first sampling the context $c \sim \mathcal{P}(C)$, then positive pairs (x, x_+) from the conditional joint distribution $\mathcal{P}(X, X_+|c)$ and negatives from the marginal conditional distribution $x_-^{(i)} \sim \mathcal{P}(X|c)$ for $i \in \{1, \ldots, N-1\}$.

The resulting contrastive learning objective is:

$$\mathcal{L} = \max_s \mathbb{E}_{c\sim\mathcal{P}(C)} \mathbb{E}_{\substack{(x_j, x_{j+})\sim\mathcal{P}(X,X_+|C), \\ x^i_{j-}\sim\mathcal{P}(X|c) \\ i,j\in\{1,\ldots,N\}}} \left[\frac{1}{N} \sum_{j=1}^{N} \frac{\exp(f(x_j, x_{j+}))}{\exp(f(x_j, x_{j+})) + \sum_{k=1}^{N-1} \exp(f(x_j, x^k_{j-}))} \right],$$

where $f(\cdot, \cdot)$ is the similarity score between state pairs. Our objective serves as a lower bound on the conditional mutual information $I(X; X_+|C)$, see Ma et al. [32].

We analyze the $I(X; X_+|C)$, showing that optimizing it results in learning representations that ignore the context variable, avoiding the failure mode from Sec. 4.1. First, applying the chain rule and symmetry of mutual information gives:

$$\begin{aligned} I(X; X_+|C) = I(X_+; X|C) &= I(X_+; X, C) - I(X_+; C) \\ &= I(X_+; X) - (I(X_+; C) - I(X_+; C|X)) \\ &= I(X; X_+) - (I(X_+; C) - I(X_+; C|X)) \end{aligned} \tag{4}$$

Using $X_+ \perp C \mid X$, we get $I(X_+; C|X) = 0$, yielding:

$$\mathcal{L} \leq I(X_+; X|C) = I(X; X_+) - I(X_+; C)$$

Our formulation reveals a core trade-off: optimal representations not only maximize $I(X; X_+)$ (as in CRL) but also minimize $I(X_+; C)$, encouraging representations that are as context-invariant as possible. Consequently, among all representations that are optimal under the standard objective (CRL), our method preferentially selects those that contain minimal information about the context. Therefore, the theoretical guarantees established for CRL are expected to remain valid.

This minimization of $I(X_+; C)$ closely parallels the goal of adversarial feature learning, where a discriminator is trained to predict the context from the representation and an encoder is trained to fool it, leading to context-invariant features. In contrast, our method achieves a similar effect through the objective itself, without the need for adversarial optimization. This connection underscores that our approach promotes invariance in a more stable and principled way, avoiding the challenges typically associated with adversarial training.

4.3 Method Used in Practice

Intuitively, incorporating in-trajectory negatives alongside standard in-batch negatives enables a trade-off between optimizing the conventional contrastive learning objective, which maximizes mutual information between inputs and their positives ($I(X; X_+)$), and our proposed objective, which additionally penalizes shared information between positives and their context ($I(X_+; C)$), i.e., optimizing $I(X; X_+) - I(X_+; C)$. Empirically, we find that a mixture of both negative types yields superior performance (see Section 5.6).

To implement this, we introduce a lightweight modification to the data loading procedure (detailed in Algorithm 4.3) that facilitates the sampling of in-trajectory negatives. Under this setup, standard in-batch contrastive learning naturally includes a subset of in-trajectory negatives. The repetition factor governs the proportion of such negatives, thereby providing a controllable mechanism to interpolate between the standard and proposed objectives. We provide further support for this design choice in Section 5.6.

4.4 What if the context is not constant?

The Sokoban example illustrates a common challenge in reasoning tasks: when the context remains constant throughout a trajectory, learned representations often fail to capture the underlying temporal

```
# dataset.shape == [num_traj, traj, obs_dim]
# CRL (prior work):
trajectories = np.random.choice(dataset.shape[0], batch_size)
batch = dataset[trajectories]

# CR^2 (our approach):
trajectories = np.random.choice(dataset.shape[0],
                                batch_size // repetition_factor)
trajectories = np.repeat(trajectories, repetition_factor, axis=0)
batch = dataset[trajectories]
# further batch processing, the same for CRL and CR^2
```

Figure 3: We propose a simple modification (CR^2) to how data is typically sampled for temporal contrastive learning (CRL) that results in representations discarding task-irrelevant context, boosting performance (See Fig. 4).

structure. This issue also manifests in domains like Sudoku, Minesweeper, and graph-based problems such as the Traveling Salesman, where the context provides limited or no temporal signal.

A related challenge arises when there is no fixed context. In the Rubik's Cube, for instance, all states are mutually reachable, making it difficult to define a stable reference context. Nevertheless, if we focus on the more shuffled portion of the trajectory, simple heuristics like Hamming distance can classify state pairs with 90% accuracy. This suggests that while each move introduces temporal change in some parts of the cube, others remain unchanged, implicitly forming a type of context. As a result, networks may latch onto features that correlate with this pseudo-context rather than true temporal proximity. In Section 5.2 we empirically demonstrate that our method can also improve performance in the case of such a context.

5 Experiments

Our experiments aim to answer the following research questions:

1. Does learning representations that ignore context improve performance on combinatorial reasoning problems? (Sec. 5.2)
2. Are representation learning methods competitive with successful deep learning methods for combinatorial reasoning once context is removed? (Sec. 5.3)
3. Does removing context also provide representations that enable stronger temporal reasoning and stitching in domains outside combinatorial reasoning? (Sec. 5.4)
4. Do learned representations alone suffice for reasoning or is search essential? (Sec. 5.5)
5. What is the relative importance of design decisions, such as how the negatives are sampled and the number of in-trajectory negatives? (Sec. 5.6)

5.1 Experimental Setup

Environments We evaluate all methods on five challenging combinatorial reasoning tasks: Sokoban, Rubik's Cube, N-Puzzle, Lights Out, and Digit Jumper. Most of these are NP-hard [12, 10, 41] and serve as standard RL benchmarks [1, 39, 56]. *Sokoban* is a grid-based puzzle where an agent pushes boxes to targets while avoiding irreversible states. *Rubik's Cube* requires aligning each face of a 3D cube to a single color. *N-Puzzle* involves sliding tiles within a 4×4 grid to reach a goal configuration. *Lights Out* is a toggle-based puzzle aiming to switch all cells to an *off* state. *Digit Jumper* is a grid game where each cell indicates the jump length from that position. See Appendix A for full environment details.

Baselines We compare against three baselines. The *contrastive baseline* follows standard contrastive RL [18], training representations without in-trajectory negatives. The *supervised baseline* predicts state distances using a value network trained via imitation on demonstrations. *DeepCubeA* [1], a strong combinatorial reasoning method, learns a value function via iterative one-step lookahead on in-

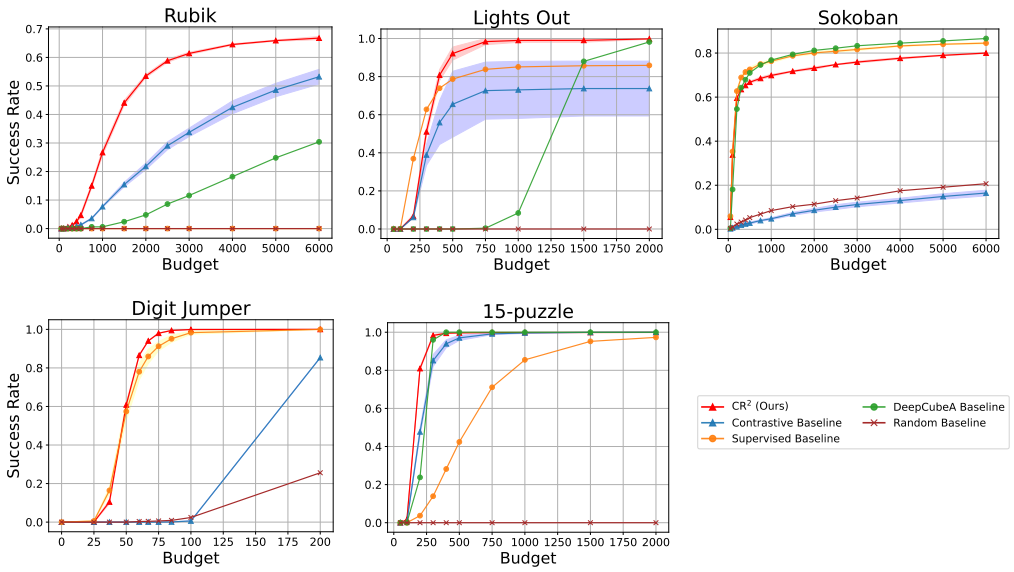

Figure 4: **CR^2 performs well in all the evaluated domains.** Performance of CR^2 compared to baselines.

creasingly challenging states. We also consider the performance resulting from taking representations from a randomly initialized network as a lower bound.

For fairness, the CRL and supervised baselines use the same architecture as CR^2. All methods, including DeepCubeA, use BestFS for planning. During tree search, all actions are considered for Rubik's Cube, N-Puzzle, Digit Jumper, and Sokoban; for Lights Out, expansion is limited to the top eight actions ranked by the value function. Further evaluation details are provided in Appendix D.

Metrics There are two metrics of main interest to us: correlation and solved rate. Correlation is Spearman's Rank correlation between the distance in the representation space and the actual distance in time. The solved rate at a given computational budget is the fraction of initial states from which a solution has been found to all states considered.

Our code is available online https://github.com/combinatorialreasoning/crcr. The training details are specified in Appendix C.

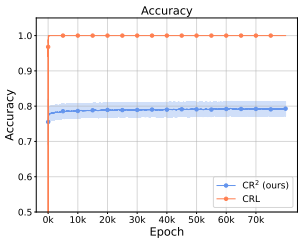

Figure 5: Accuracy of training objectives. CRL quickly acquires near-perfect accuracy, however this is due to relying only on superficial features, like walls.

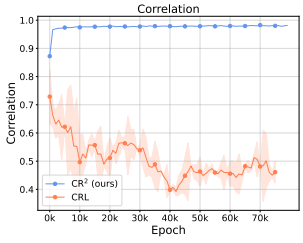

Figure 6: Correlation (Spearman's ρ) between the distance induced by learned embeddings and actual distance. CR^2 quickly acquired reasonable values, thus allowing for effective planning.

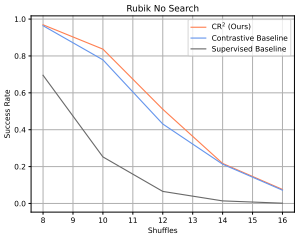

Figure 7: **Contrastive representations must be used alongside search.** Performance of baselines and CR^2 on the Rubik's Cube without search, evaluated across increasing levels of shuffling.

5.2 Impact of Context-Free Representations on Combinatorial Reasoning

Figure 4 compares CR^2 to standard Contrastive Reinforcement Learning (CRL) without in-trajectory negatives and random baseline. Across all domains, CR^2 consistently improves search efficiency, with the largest gains in Sokoban and Digit Jumper. We therefore conclude, that using CR^2 is correlated with improved performance on combinatorial reasoning problems.

Moreover, the improvement is caused by removing the context information: We consider the example of Sokoban. CRL immediately achieves 100% training accuracy (Figure 5) despite poor correlation (Figure 6), suggesting that it is, in fact, ignoring the temporal aspect and looking solely at the context.

In contrast, CR^2 is unable to pick up on the context, avoiding trivial solutions (Figure 5) and learning geometry aligned with state-space distances (Figure 6). A corresponding analysis for Digit Jumper is included in Appendix F.

5.3 Competitiveness of Context-Free Representations with Supervised Baselines

We test whether CR^2 is competitive with supervised approaches for solving combinatorial problems. We display the result in Figure 4. In all the environments considered, CR^2 demonstrates consistently strong performance. It ranks among the top-performing methods in each environment and is strictly the best one in two cases. In contrast, the supervised baselines are performing much worse in Rubik's Cube and Lights Out.

In our evaluations, each method uses the same planning algorithm – BestFS. This suggests that the advantage of CR^2 stems from the structure of its learned representations, which provide more effective guidance for planning compared to the direct value estimation approach used by the baselines. While in Sokoban, they achieve higher scores, the difference is small.

5.4 Generalization to Temporal Reasoning in Non-Combinatorial Domains

To investigate whether CR^2 also improves the temporal structure in non-combinatorial domains, we consider Adroit simulations from D4RL [20]. Those tasks require using a high-dimensional robotic hand, to perform various tasks. They are specifically designed to test fine motor control and long-horizon planning in challenging settings.

We look at the correlation through training for CR^2 and CRL (Fig. 8). CR^2 results in a higher correlation (more than 0.9 in comparison to 0.5-0.8 depending on the environment), as well as visibly better training stability – for standard CRL, the correlation is visibly unstable through training and in some cases even becomes smaller as the training progresses. We conclude that using CR^2 results in a better temporal structure in the representation space for non-combinatorial problems.

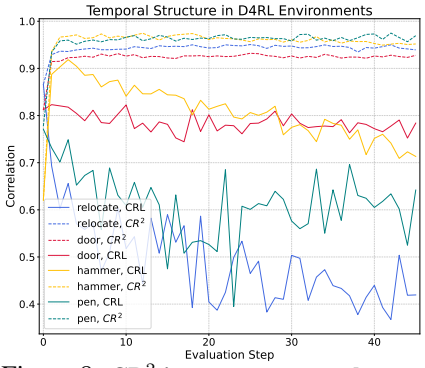

Figure 8: **CR^2 improves temporal structure in robotics environments.** Comparison of correlation metric for CRL (solid) and CR^2 (dotted) for D4RL offline datasets.

5.5 Is search still necessary?

One of our main questions was whether having good representations allows to use no search, or at least, decrease the amount of search needed. We test the approach, where we always only consider one action, predicted to be the best by our heuristic. We do this until we arrive at the same state for the second time, or exceed the budget of 6000 nodes. Table 1 demonstrates the results of not using search in CR^2, our contrastive baseline, and the supervised baseline. While our approach improves the performance without search, for Rubik's Cube and 15-puzzle, the solved rate is very close to 0. In Figure 7 we demonstrate the performance for the no-search approach, for the increasingly shuffled Rubik's cube. All the methods' performance decreases exponentially as the number of shuffles is increased. This is expected, as the number of states reachable within n shuffles follows an exponential trend, for $n \leq 18$. [43] We therefore conclude that while our method does perform better than the baselines, search is still necessary for achieving the optimal performance. For additional no-search results, see Appendix E.

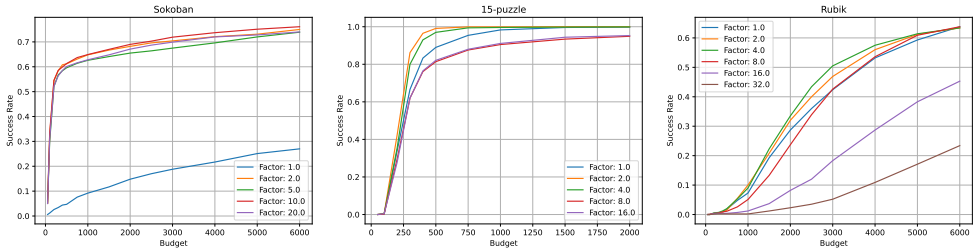

Figure 9: **Influence of the repetition factor depends on the environment type.** Increasing the repetition factor for Sokoban, N-Puzzle, and Rubik's Cube, respectively. Factor 2.0 corresponds to our CR^2, while factor 1.0 corresponds to the CRL baseline.

5.6 In-trajectory Negatives Design Choices

Limitations of the Standard Approach The standard way to introduce in-trajectory negatives is to follow conventional techniques for incorporating hard negatives. However, we found that such straightforward approaches consistently underperformed compared to CRL, resulting in lower solve rates and weaker correlations. Although we were able to achieve some improvement on the Rubik's Cube by applying an unconventional normalization scheme, this strategy did not yield similar gains on Sokoban. For full loss formulations and detailed results, see Appendix G.

Table 1: Performance of the baselines and CR^2 on Rubik's Cube, Lights Out, 15-puzzle and Sokoban without using search.

Problem	CR^2	Contrastive Baseline	Supervised Baseline
Rubik's Cube	**0.03**	0.02	0.0
Lights Out	**0.98**	0.47	0.91
15-puzzle	0.0	0.0	0.0
Sokoban	**0.30**	0.0	0.23

Balancing In-Trajectory and In-Batch Negatives Our method introduces a single additional hyperparameter: the repetition factor R. This parameter controls the proportion of in-trajectory negatives and is critical for achieving strong performance. As shown in Figure 9, the impact of increasing R varies by environment. For Sokoban, higher values of R lead to only a slight decline in performance. In contrast, in many other environments, excessive repetition can significantly degrade results. While $R = 2$ is not always optimal, it consistently improves performance across all environments we evaluated and serves as a strong default choice.

6 Conclusion

In our work, we introduced CR^2, an algorithm for learning high-quality representations in combinatorial reasoning tasks. Our analysis revealed a critical limitation of prior approaches: when training demonstrations are separable, their learned representations become trivial and ineffective for planning. CR^2 addresses this by balancing global negatives, which capture overall task structure, with local negatives, which enforce temporal consistency. Experimental results across four domains highlight its effectiveness.

Limitations This work focuses on understanding the role of representations in combinatorial reasoning. We are interested mostly (though, see Sec. 5.4) in domains with discrete states and actions and known dynamics. Moving forward, we are interested in applying these methods to domains of real-world interest, such as chemical retrosynthesis and robotic assembly. These domains also have rich combinatorial structure, but introduce additional complexity because the dynamics are unknown and observations can be noisy.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be highlighted here.

References

[1] Agostinelli, F., McAleer, S., Shmakov, A., and Baldi, P. (2019). Solving the rubik's cube with deep reinforcement learning and search. *Nat. Mach. Intell.*, 1(8):356–363.

[2] Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940.

[3] Bortkiewicz, M., Pałucki, W., Myers, V., Dziarmaga, T., Arczewski, T., Kuciński, Ł., and Eysenbach, B. (2024). Accelerating goal-conditioned rl algorithms and research. *arXiv preprint arXiv:2408.11052*.

[4] Brown, N., Bakhtin, A., Lerer, A., and Gong, Q. (2020). Combining deep reinforcement learning and search for imperfect-information games. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[5] Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Velickovic, P. (2021). Combinatorial optimization and reasoning with graph neural networks. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4348–4355. ijcai.org.

[6] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv: 2104.14294*.

[7] Chanales, A. J. H., Oza, A., Favila, S. E., and Kuhl, B. A. (2017). Overlap among spatial memories triggers repulsion of hippocampal representations. *Current Biology*, 27(15):2307–2317.e5.

[8] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

[9] Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775.

[10] Culberson, J. C. (1997). Sokoban is pspace-complete.

[11] Czechowski, K., Odrzygózdz, T., Zbysinski, M., Zawalski, M., Olejnik, K., Wu, Y., Kucinski, L., and Milos, P. (2021). Subgoal search for complex reasoning tasks. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 624–638.

[12] Demaine, E. D., Eisenstat, S., and Rudoy, M. (2018). Solving the rubik's cube optimally is np-complete. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[13] Dor, D. and Zwick, U. (1999). SOKOBAN and other motion planning problems. *Comput. Geom.*, 13(4):215–228.

[14] Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. *International Conference on Learning Representations*.

[15] Eysenbach, B., Myers, V., Salakhutdinov, R., and Levine, S. (2024). Inference via interpolation: Contrastive representations provably enable planning and inference.

[16] Eysenbach, B., Salakhutdinov, R., and Levine, S. (2021). C-Learning: Learning to Achieve Goals via Recursive Classification. In *International Conference on Learning Representations*. arXiv.

[17] Eysenbach, B., Zhang, T., Levine, S., and Salakhutdinov, R. (2022a). Contrastive learning as goal-conditioned reinforcement learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[18] Eysenbach, B., Zhang, T., Levine, S., and Salakhutdinov, R. R. (2022b). Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620.

[19] Fang, K., Yin, P., Nair, A., Walke, H., Yan, G., and Levine, S. (2022). Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks. In Liu, K., Kulic, D., and Ichnowski, J., editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 106–117. PMLR.

[20] Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2021). D4rl: Datasets for deep data-driven reinforcement learning.

[21] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv: 2312.10997*.

[22] Guo, Z., Thakoor, S., Pislar, M., Pires, B. Á., Altché, F., Tallec, C., Saade, A., Calandriello, D., Grill, J., Tang, Y., Valko, M., Munos, R., Azar, M. G., and Piot, B. (2022). Byol-explore: Exploration by bootstrapped prediction. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[23] Ha, D. and Schmidhuber, J. (2018). World models. *CoRR*, abs/1803.10122.

[24] Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. P. (2023). Mastering diverse domains through world models. *CoRR*, abs/2301.04104.

[25] Ichter, B. and Pavone, M. (2019). Robot motion planning in learned latent spaces. *IEEE Robotics Autom. Lett.*, 4(3):2407–2414.

[26] Islam, A., Chen, C.-F. R., Panda, R., Karlinsky, L., Radke, R., and Feris, R. (2021). A broad study on the transferability of visual representations with contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8845–8855.

[27] Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2020). A survey on contrastive self-supervised learning. *TECHNOLOGIES*.

[28] Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York.

[29] Kool, W., van Hoof, H., and Welling, M. (2019). Attention, learn to solve routing problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[30] Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2024). Visual instruction tuning. *Advances in neural information processing systems*, 36.

[31] Ma, K., Du, X., Wang, Y., Zhang, H., Wen, Z., Qu, X., Yang, J., Liu, J., Liu, M., Yue, X., et al. (2024). Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. *arXiv preprint arXiv:2410.06526*.

[32] Ma, M. Q., Tsai, Y.-H. H., Liang, P. P., Zhao, H., Zhang, K., Salakhutdinov, R., and Morency, L.-P. (2022). Conditional contrastive learning for improving fairness in self-supervised learning.

[33] Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Comput. Oper. Res.*, 134:105400.

[34] Myers, V., Zheng, C., Dragan, A., Levine, S., and Eysenbach, B. (2024). Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. *International Conference on Machine Learning*.

[35] Nauman, M., Ostaszewski, M., Jankowski, K., Miłoś, P., and Cygan, M. (2024). Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control.

[36] Orseau, L., Lelis, L., Lattimore, T., and Weber, T. (2018). Single-agent policy tree search with guarantees. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3205–3215.

[37] Park, S., Rybkin, O., and Levine, S. (2023). Metra: Scalable unsupervised rl with metric-aware abstraction. *International Conference on Learning Representations*.

[38] Pearl, J. (1984). *Heuristics - intelligent search strategies for computer problem solving*. Addison-Wesley series in artificial intelligence. Addison-Wesley.

[39] Racanière, S., Weber, T., Reichert, D. P., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P. W., Hassabis, D., Silver, D., and Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5690–5701.

[40] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*.

[41] Ratner, D. and Warmuth, M. K. (1986). Finding a shortest solution for the N × N extension of the 15-puzzle is intractable. In Kehler, T., editor, *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, USA, August 11-15, 1986. Volume 1: Science*, pages 168–172. Morgan Kaufmann.

[42] Rethmeier, N. and Augenstein, I. (2023). A primer on contrastive pretraining in language processing: Methods, lessons learned, and perspectives. *ACM Computing Surveys*, 55(10):1–17.

[43] Rokicki, T., Kociemba, H., Davidson, M., and Dethridge, J. (2014). The diameter of the rubik's cube group is twenty. *SIAM Rev.*, 56(4):645–670.

[44] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609.

[45] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. (2018). Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE.

[46] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489.

[47] Sohn, K. (2016). Improved Deep Metric Learning With Multi-Class N-Pair Loss Objective. In *Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[48] Srivastava, A., Rastogi, A., and et al., A. R. (2023). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*

[49] Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020). What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839.

[50] Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. (2023). Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987.

[51] Venkattaramanujam, S., Crawford, E., Doan, T., and Precup, D. (2019). Self-supervised learning of distance functions for goal-conditioned reinforcement learning. *arXiv preprint arXiv: 1907.02998*.

[52] Wang, T. and Isola, P. (2020). Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.

[53] Xiao, T., Wang, X., Efros, A. A., and Darrell, T. (2021). What should not be contrastive in contrastive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[54] Yin, Y., Wang, Z., Gu, Y., Huang, H., Chen, W., and Zhou, M. (2024). Relative preference optimization: Enhancing llm alignment through contrasting responses across identical and diverse prompts. *arXiv preprint arXiv:2402.10958*.

[55] Yonetani, R., Taniai, T., Barekatain, M., Nishimura, M., and Kanezaki, A. (2021). Path planning using neural a* search. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12029–12039. PMLR.

[56] Zawalski, M., Tyrolski, M., Czechowski, K., Odrzygozdz, T., Stachura, D., Piekos, P., Wu, Y., Łukasz Kucinski, and Milos, P. (2024). Fast and precise: Adjusting planning horizon with adaptive subgoal search.

[57] Zhang, Y., Jiang, H., Miura, Y., Manning, C. D., and Langlotz, C. P. (2022). Contrastive learning of medical visual representations from paired images and text. In Lipton, Z., Ranganath, R., Sendak, M., Sjoding, M., and Yeung, S., editors, *Proceedings of the 7th Machine Learning for Healthcare Conference*, volume 182 of *Proceedings of Machine Learning Research*, pages 2–25. PMLR.

[58] Zheng, C., Eysenbach, B., Walke, H., Yin, P., Fang, K., Salakhutdinov, R., and Levine, S. (2024a). Stabilizing Contrastive RL: Techniques for Offline Goal Reaching. In *International Conference on Learning Representations*. arXiv.

[59] Zheng, C., Salakhutdinov, R., and Eysenbach, B. (2023). Contrastive Difference Predictive Coding. In *Twelfth International Conference on Learning Representations*. arXiv.

[60] Zheng, C., Tuyls, J., Peng, J., and Eysenbach, B. (2024b). Can a misl fly? analysis and ingredients for mutual information skill learning. *arXiv preprint arXiv: 2412.08021*.

A Environments

Sokoban. Sokoban is a well-known puzzle where the player must push boxes onto target locations within a confined grid. Its high combinatorial complexity and PSPACE-hard nature [13] make it a benchmark for both classical planning and deep learning methods. Sokoban challenges algorithms to balance search efficiency and long-term planning. In our experiments, we use 12×12 Sokoban boards with four boxes.

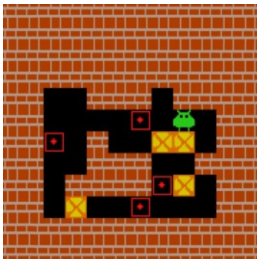

Figure 10: An example instance of Sokoban.

Figure 11: An example instance of Rubik's Cube.

Rubik's Cube. The Rubik's Cube is a 3D combinatorial puzzle with over 4.3×10^{19} possible configurations, making it an iconic testbed for algorithms tackling massive search spaces. Solving the Rubik's Cube requires sophisticated reasoning and planning, as well as the ability to navigate high-dimensional state spaces efficiently. Recent advances in using neural networks for solving this puzzle, such as [1], highlight the potential of deep learning in handling such computationally challenging tasks.

N-Puzzle. The N-Puzzle is a sliding-tile puzzle with variants like the 8-puzzle (3×3 grid), 15-puzzle (4×4 grid), and 24-puzzle (5×5 grid). The objective is to rearrange tiles into a predefined order by sliding them into an empty space. It serves as a classic benchmark for testing algorithms' planning and search efficiency. The problem's difficulty scales with puzzle size, requiring effective heuristics for solving larger instances.

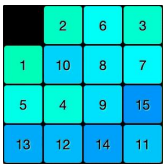

Figure 12: An example instance of N-Puzzle.

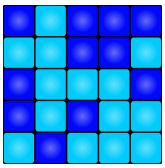

Figure 13: An example instance of Lights Out.

Figure 14: An example instance of Digit Jumper.

Lights Out The Lights Out is a single-player game invented in 1995. It is a grid-based game where each cell (or *light*) can either be on or off. Pressing a cell flips its state and those of its immediate neighbors (above, below, left, and right). Corner and edge lights have fewer neighbors and therefore affect fewer lights. The goal is to press the lights in a strategic order to turn off all the lights on the grid.

Digit Jumper Digit Jumper is a grid-based game, where the goal is to get from the top-left corner of the board to the bottom-right one. At each point, the player can move n steps to the left, right, up or down, where n is determined by the number written on the board. Digit Jumper is an example of an environment with a constant context as is Sokoban.

B Best-First Search

Best-First Search greedily prioritizes node expansions with the highest heuristic estimates, aiming for paths that likely lead to the goal. While not ensuring optimality, BestFS provides a simple yet efficient strategy for navigating complex search spaces. The high-level pseudocode for BestFS is outlined in Algorithm 1.

Algorithm 1 Pseudocode for Best-First Search

while has nodes to expand **do**
  Take node N with the highest value
  Select children n_i of N
  Compute values v_i for the children
  Add (n_i, v_i) to the search tree
end while

C Training Details

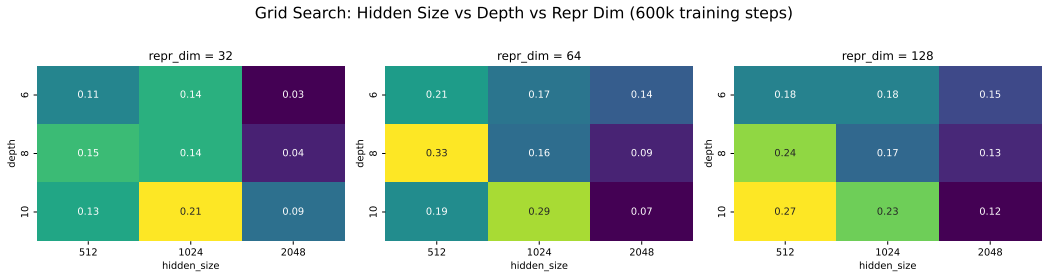

Figure 15: Grid of network's depth, representation dimension and hidden dimension.

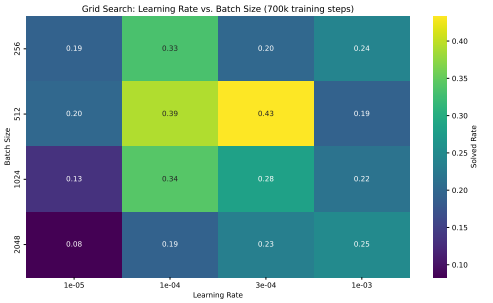

Figure 16: Learning rate and batch size grid for the Rubik's Cube. Solved rate is investigated on a cube that has been shuffled only 10 times.

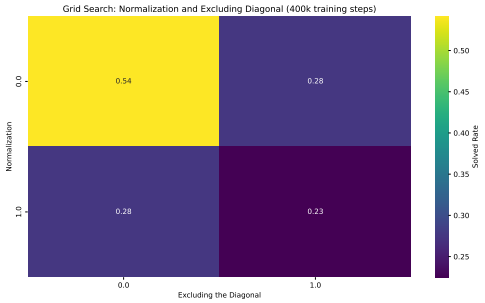

Figure 17: Use of normalization in contrastive learning and whether the distance from positives is divided by sum of all batch elements or only the in-batch negatives.

Code to reproduce all results is available in the anonymous repository referenced in the main text. Below, we document the training procedures for the supervised baseline, contrastive baseline, and CR^2.

Training Data For Sokoban, we use trajectories provided by Czechowski et al. [11], and train on a dataset of 10^5 trajectories. For 15-Puzzle, Rubik's Cube, and Lights Out, we generate training trajectories by applying a policy that performs n random actions, where n is set to 150, 21, and 49, respectively. In the case of 15-Puzzle, we additionally remove single-step cycles from the dataset to improve data efficiency. For Digit Jumper, we generate training data by sampling a random path

Figure 18: Trainings with different metrics for the Rubik's Cube. The solved rate is for a cube shuffled 10 times.

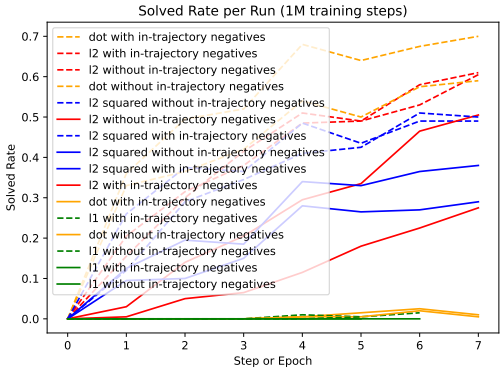

Figure 19: Symmetric contrastive loss vs. backward contrastive loss.

from the upper-left corner to the bottom-right corner on a standard 20×20 grid. All grid cells not required for this path are filled by sampling uniformly from the set $1, \ldots, 6$. The network for Digit Jumper typically saturates in performance after a few hours of training, so we train until convergence is observed. For Sokoban, Rubik's Cube, Lights Out, and 15-Puzzle, we adopt an unlimited data setup and train all models for two days. This results in the models seeing approximately 8×10^6 trajectories for Rubik's Cube, 7×10^6 for 15-Puzzle, and 9×10^6 for Lights Out.

Training Hyperparameters We use the Adam optimizer with a constant learning rate throughout training. A learning rate of 0.0003 was found to perform well across all environments, with the exception of Lights Out, where this setting led to unstable training. For this environment, we instead use a reduced learning rate of 0.0001. In all environments, we use a batch size of 512. The choice of learning rate and batch size was guided by the performance of the contrastive baseline on Rubik's Cube. Specifically, we evaluated solve rates on cubes shuffled 10 times, as shown in Figure 16.

Network Architecture We adopt the network architecture proposed by Nauman et al. [35], using 8 layers with a hidden size of 512 and a representation dimension of 64. This configuration was found to yield optimal performance for the contrastive baseline on Rubik's Cube, as illustrated in Figure 15. We observed that this architecture also performs well across all other environments, with two exceptions:

- In Sokoban, a convolutional architecture was required to achieve strong performance.
- In Lights Out, the convolutional network was necessary to ensure training stability.

Test Set For Sokoban, we construct a separate test set comprising 100 trajectories, which is used to compute evaluation metrics such as accuracy, correlation, and t-SNE visualizations. For all other environments, a separate test set is not required, as we operate in an infinite data regime and train for only a single epoch. In this setting, evaluation is performed directly on unseen data sampled during training.

Contrastive Loss We use the backward version of the contrastive loss, which we found to consistently outperform the symmetrized variant on Rubik's Cube, as shown in Figure 19.

For Rubik's Cube, we use the dot product as the similarity metric. Performance across different metrics is presented in Figure 18. While the contrastive baseline performs comparably under the ℓ_2 metric, CR^2 achieves significantly better results with the dot product. Based on similar empirical evaluations, we use the following metrics in other environments:

- Lights Out: ℓ_2 distance,
- Digit Jumper and 15-Puzzle: dot product,

Table 2: Average solution length of the baselines and CR^2 on Rubik's Cube and 15-puzzle without using search. Supervised baseline in the Rubik's Cube solved no of the boards.

Problem	CR^2	Contrastive Baseline	Supervised Baseline
Rubik's Cube	448.7	1830.3	NaN
15-puzzle	82.4	119.5	1054.3

- Sokoban: squared ℓ_2 distance.

We set the temperature parameter in the contrastive loss to the square root of the representation dimension.

Supervised Baseline The supervised baseline takes as input a pair of states and predicts the distance between them by performing classification into discrete bins, where the number of bins corresponds to the maximum trajectory length observed in the dataset.

In all environments—except Lights Out—the supervised baseline uses the same architecture as the contrastive baseline. For Lights Out, however, we employ a different architecture: a dense network, which achieves a performance of 0.7 solved rate, compared to a maximum of approximately 0.2 with a convolutional network. This significant difference in performance motivates our use of a dense architecture for the supervised baseline in this environment.

D Evaluation Details

We evaluate all networks on 1000 problem instances per environment. For Rubik's Cube, each instance is a cube shuffled 20 times. For 15-Puzzle, Lights Out, and Digit Jumper, evaluation boards are sampled randomly. For Sokoban, we follow the same instance generation procedure as described by Czechowski et al. [11].

E Additional Experiments

No-search results In the main part of the paper, we limit the maximum solution depth for the no-search results. In this section, however, we remove these constraints and allow arbitrarily long solutions. In such a setup, for both the Rubik's Cube and 15-puzzle, the contrastive methods achieve a solved rate of over 90%.

The no-search approach operates by selecting, at each step, the next state that appears most likely to move toward the solved state—based on the learned representations. If the representation were perfect, this would yield optimal solutions. In practice, however, suboptimal representations cause the agent to spiral quite randomly through the representations far away from the goal state before converging. Thus, the quality of the representation is reflected in the length of these trajectories: the better the representation captures directionality in latent space, the shorter the resulting solutions.

Table 2 reports the average solution length for the no-search approach on Rubik's Cube and 15-Puzzle. These results suggest that the representations learned by CR^2 are better suited for this approach than those learned by the contrastive baseline, and significantly outperform those derived from the supervised method. This supports the conclusion that CR^2 provides a more reliable notion of direction in latent space. Remarkably, both the average for CR^2 and CRL are smaller than the solutions lengths from the training data, indicating that we observe trajectory stitching.

F Digit-Jumper Analysis

In Digit Jumper, we observe a similar effect to that seen in Sokoban when comparing CR^2 to standard CRL. As shown in Figure 20, CRL rapidly achieves 100% training accuracy. However, despite this perfect accuracy, the resulting representations exhibit poor correlation with actual temporal structure (Figure 21). This is consistent with the t-SNE representations: the same as for Sokoban,

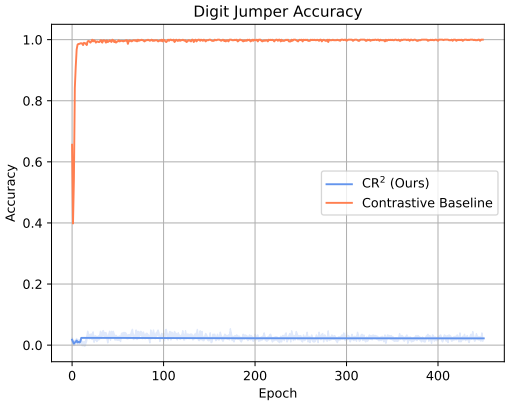

Figure 20: Accuracy of training objectives.

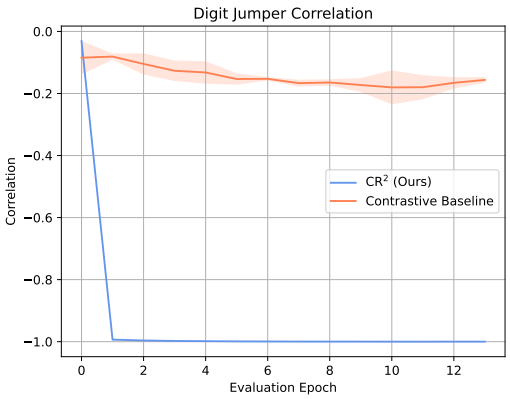

Figure 21: Correlation (Spearman's ρ) between the distance induced by learned embeddings and actual distance.

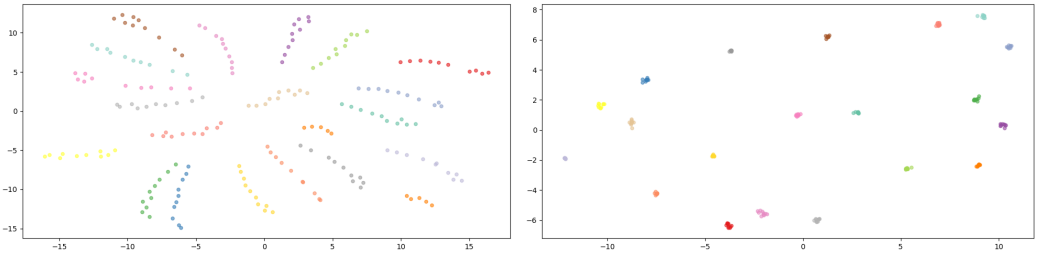

Figure 22: **CR^2 makes representations reflect the structure of the combinatorial task.** t-SNE visualization of representations learned by CR^2 (left) and CRL (right) for Digit Jumper. Colors correspond to trajectories. CRL representations (right) cluster within trajectories, making them useless for planning.

CRL collapses each trajectory into a single point in the representation space, discarding temporal information. In contrast, CR^2 preserves a clear temporal structure within the latent space (see Figure 22).

G Negatives

We explored alternative methods for incorporating in-trajectory negatives into the contrastive loss. The first approach mimics the standard addition of hard negatives: given a batch $\mathcal{B} = (x_i, x_{i+})_{i \in \{1..B\}}$, we sample additional negatives, $(x_{i-})_{i \in \{1..B\}}$, and compute the loss as

$$\mathcal{L} = \frac{1}{B} \sum_i \log \left(\frac{\exp\left(f(x_i, x_{i+})\right)}{\sum_{j \neq i} \exp(f(x_i, x_{j+})) + \exp(f(x_i, x_{i-}))} \right),$$

.

We considered three strategies for selecting in-trajectory negatives: sampling uniformly at random, selecting the first state, or selecting the last state of the trajectory. For Rubik's Cube, instead of choosing the last state—which is identical for all trajectories—we sample a random state further away from the solution to serve as a negative.

As shown in Figures 23 and 24, training with this approach failed to achieve strong performance. We hypothesized that the large error introduced by the in-trajectory negatives (x_{i-}) caused excessively large gradients, destabilizing training. To mitigate this, we applied a normalization scheme: so that

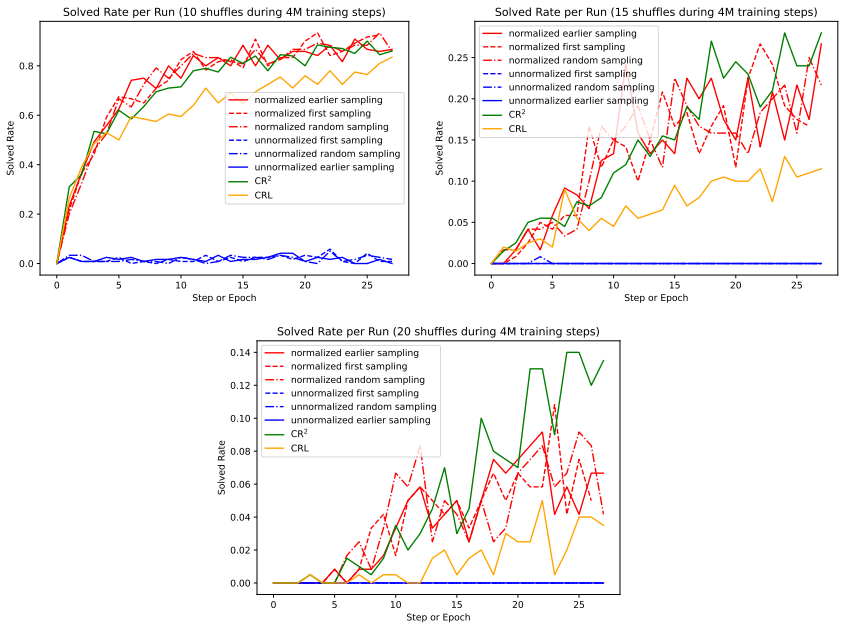

Figure 23: Comparison of different methods of introducing in-trajectory negatives in the Rubik's Cube environment, with increasing number of shuffles of the cube. While normalized negatives perform similarily to CR^2 for a small number of shuffles, their performance fails to be as good for more shuffles.

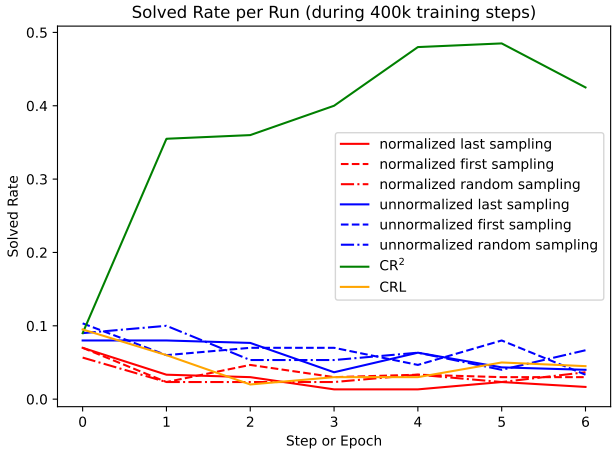

Figure 24: Comparison of different methods of introducing in-trajectory negatives in the Sokoban environment. The only negative sampling strategy that works is CR^2.

the vector

$$\begin{bmatrix} f(x_1, x_{1-}) \\ \vdots \\ f(x_B, x_{B-}) \end{bmatrix}$$

has the same norm as the norm (when viewing the matrix as a vector of size B^2) of the matrix

$$\begin{bmatrix} f(x_1, x_{1+}) & f(x_1, x_{2+}) & \dots & f(x_1, x_{B+}) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_B, x_{1+}) & f(x_B, x_{2+}) & \dots & f(x_B, x_{B+}) \end{bmatrix}$$

. This normalization enabled achieving comparable performance to CR^2 on Rubik's Cube shuffled 10 times (Figure 23). However, CR^2 still outperforms all negative sampling strategies on cubes shuffled 15 and 20 times.

For Sokoban, the only approach that consistently improved performance is CR^2, as demonstrated in Figure 24. We hypothesize that this is because removing contextual information is more challenging in Sokoban than in Rubik's Cube. In the latter, the context is more local and changes gradually over time, making it *softer*, while the context in Sokoban is constant through a trajectory. This is discussed in detail in Section 4.2.

H Computational Resources

All training experiments were conducted using NVIDIA A100 GPUs and took between 5 and 48 hours each. The solving runs ranged from 10 minutes to 10 hours. In total, the project required approximately 30,000 GPU hours to complete.