ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Imaginative and Contrastive Based Self Learning Agent

Thesis submitted in partial fulfillment of the requirements for the Masters of Science in Computer Engineering

By Kalkidan Behailu Kebede

Advisor: Dr. Menore Tekeba

Date: June 2024

Addis Ababa ,Ethiopia

The undersigned have examined the thesis entitled '**Imaginative and Contrastive based Self Learning Agent** presented by **Kalkidan Behailu,** a candidate for the degree of **Master of Science** and hereby certify that it is worthy of acceptance.

Advisor Dr Menore Tekeba	Signature	Date
Internal Examiner Dr Surafel Lemma	Signature	Date
External Examiner Dr Biniam Tadese	Signature	Date
Chair person Dr Fitsum Asmamaw	Signature	Date

UNDERTAKING

I certify that research work titled “imaginative and contrastive based self-learning agent” is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Kalkidan Behailu Kebede

ABSTRACT

Developing an agent in reinforcement learning (RL) that is capable of performing complex control tasks directly from high-dimensional observation such as raw pixels is a challenge as efforts still need to be made towards improving sample efficiency. This paper explores an unsupervised learning framework that leverages imaginative and contrastive-based representations to enhance sample efficiency in reinforcement learning, working directly with raw pixels. It incorporates an imaginative module and performs contrastive learning to train its deep convolutional neural network-based encoder to extract temporal and instance information representation to achieve a more sample efficiency for RL. It extracts high-level features from raw pixels using the hybrid of contrastive and imaginative based unsupervised representation learning. It performs off-policy control using the extracted features, enabling the agent to imagine its future states and capture temporal dependencies. The agent's dynamic behavior can be understood by generating learnable patterns. Our method outperforms prior both imaginative and contrastive pixel-based learning methods on complex tasks in of the DeepMind Control Suite at the 100K environment and interaction time-steps benchmarks.

Keywords:

Reinforcement Learning, Imaginative Learning, Contrastive Learning, Representation Learning

ACKNOWLEDGMENTS

First and foremost, I would like to thank God for giving me his strength and ability to undertake this research study. Without his countless blessing, everything would not have been possible.

I would like to thank my advisor Dr. Menor Tekeba for his guidance throughout this research.

Finally, I must express my very profound gratitude to my parents, my husband and friends for providing me with unfailing support and continuous encouragement through- out my years of study, through the process of researching and writing this thesis. This accomplishment would not have been possible without them Thank you.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF Abbreviations

BYOL	Bootstrap Your Own Latent
CNN	Convolutional Neural Network
CURL	Contrastive Unsupervised Representation Learning
DMC	Deep mind control
DREAMER	Dream to control
EMA	Exponential moving average
LSTM	Long Short-Term Memory
ML	Machine Learning
MOCO	Momentum Contrast
PlaNet	Planning with Latent Dynamics
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent neural network
SAC	Soft Actor-Critic
SIMCLR	Simple Framework for Contrastive Learning of Visual Representations
SPR	Self predictive representation

CHAPTER 1 INTRODUCTION

1.1 Overview of Reinforcement Learning

Artificial intelligence has seen the rise of Reinforcement Learning (RL) as an effective method. It deals with learning how to make judgments by trial and error in a setting where decisions are made and then actions are rewarded or penalized accordingly. Agents can gradually optimize their behavior and get optimal outcomes through this interactive learning process. It will be clear that this formalization applies to a wide range of tasks and encompasses key aspects of artificial intelligence, such as an understanding of uncertainty and unpredictability, as well as a grasp of cause and effect. [5]

Self-learning RL agent is an agent that can learn and improve its behavior through interactions with its environment, without the need of supervisions. This means that it modifies or acquires new behaviors and skills incrementally. It uses trial-and-error experience (as opposed to e.g., dynamic programming that assumes full knowledge of the environment a priori). Thus, the RL agent does not require complete knowledge or control of the environment. It only needs to be able to interact with the environment and collect information.

Representation learning refers to the process of learning a parametric mapping from the raw input data domain to a feature vector or tensor, in the hope of capturing and extracting more abstract and useful concepts that can improve performance on a range of downstream tasks. [5]

1.2 Reinforcement Learning Algorithm

RL algorithm can categorized with value-based, policy-based, and model-based or model-free sub types. [1]

Value-based: a method to quantify the desirability of being in a particular state and taking specific action. It estimates the value of state action pairs and learns an optimal value function.

Policy-based: method that learns a parameterized neural network that maps state to action. It has two approaches:

- Policy gradient: update the policy in the direction of increasing the expected reward.

- Actor-critic: an algorithm in which the actor learns the policy and the critic learns the value function. The critic provides feedback to the actor to learn better policy.

It can be classed into two based on the updated mechanism off-policy and on-policy,
On-policy: the policy update based on the experience generated by recent policy or by itself not from other's data.

Off-policy: the policy can learn from experience generated by any policy not fixed with its policy. It uses a replay buffer to store experience and it uses for updates.

Model-based:-It tries to learn a model of the environment, including the transition dynamics and the reward function. With this learned model, the agent can plan ahead and optimize its behavior. Model-based algorithms often use techniques such as dynamic programming or Monte Carlo simulation.

Model-free: Model-free algorithms, on the other hand, directly learn the policy or value function without explicitly modeling the environment. These algorithms learn from interaction with the environment by updating their value function estimates based on the observed rewards and state transitions

1.3 Challenges in Reinforcement Learning

There are three main challenges in RL, state representation, sample efficiency and stability and generalization.

1.3.1 State Representation

States in RL agents can be either discrete or continuous. When the state is an image, the challenge is how to select relevant features or encode the image for action selection. So, state representation is a vital aspect of reinforcement learning (RL) as it determines how an agent perceives and interacts with the environment. The quality of the state representation greatly impacts the effectiveness and efficiency of RL algorithms.

1.3.2 Sample Efficiency

Sample efficiency refers to the ability to achieve good performance on a difficult task using a relatively small amount of data. When agent has high-dimensional and continuous states making it difficult to represent and process the states effectively. The agent needs more samples in order to explore its environment and make generalizations. When we train with

limited sample, agent sees only a portion of the environment rather than all state information.

1.3.3 Stability and Generalization

Stability refers to the consistency and convergence of the learning process in RL. RL algorithms often rely on value function estimation or policy optimization techniques, which can be sensitive to changes in the learning dynamics. Instabilities can arise due to factors such as high variance in the estimates, non-stationarity of the environment, or improper value function approximation. These instabilities can lead to suboptimal or divergent policies and hinder the learning process.

Generalization refers to the ability of an RL algorithm to apply the learned policy to unseen or novel states and environments. Effective generalization allows the agent to transfer its learned knowledge and policies to new situations, accelerating learning and adapting to different scenarios. Challenges in generalization arise when the agent encounters states or situations that differ significantly from the training data. This can occur due to changes in the environment dynamics, variations in the initial state distribution, or differences in task settings.

1.4 Statement of Problem

Sample efficiency in reinforcement learning (RL), agents typically require a considerable amount of interaction with the environment to learn optimal policies. However, collecting real-world data can be challenging in many domains. Therefore, there is a pressing need to develop RL algorithms that can learn effectively and achieve high performance with limited samples or interactions with the environment.

In reinforcement learning, to make good decisions, an agent must explore more and a wider range of states and actions to understand the dynamics of the environment and discover optimal strategies.

The amount of experience depends on several factors as complex problems require collection of high number of samples (data) and training iterations. The agent may have high dimensional state space so the observations become images or any other multimedia type. It is essential to extract features from the high-dimensional continuous state that will aid in selecting the action that maximizes the reward. Because the high in dimension need big data for simulation of the environment.

To make RL more sample efficient using RL state representation approach and world model that predict its future state. the most recent research done is by CURL [4]. But the approach doesn't capture temporal dependencies as its learning only depends on instance augmented samples rather than patterns.

On the other hand, the imagination-based sample-efficient mechanism (DREMER) proposed by [12] captures temporal dependencies between consecutive time steps of the agent by predicting the latent feature representation of the future state. However, the world model is trained with offline data collected from another experience and the learning lacks the real-time response and can't also collect most relevant samples (data) for its learning. The imaginations stack on that lack of relevant and diversified experience.

This thesis work aims to resolve these two issues by focusing working on the hybrid of the two approaches with a major modification on the imaginative.

The research questions of this thesis work are:

- Does the proposed approach perform better than DREAMER in sample efficiency and performance of the RL learning?
- Does the proposed approach perform better than CURL in sample efficiency and performance of the RL learning?
- What is the impact of contrastive learning on the imaginative-based self-supervised representation learning in reinforcement learning and vice versa?

1.5 Objectives

1.5.1 General Objectives

- To develop self-supervised learning based on contrastive and imaginative representation learning methods for RL.

1.5.2 Specific Objectives

- To select the agent(s), environment, and reinforcement algorithm for policy update
- To develop an imaginative(predictive) based module
- To incorporate contrastive learning with the imaginative module
- To train the agent with the reinforcement learning algorithm and the predictive (imaginative) network
- Test and evaluate the model,
- Make comparisons(analyze) and discuss the results.

1.6 Scope

The scope of this thesis work is limited to modeling imaginative and contrastive-based modules. The experiments are done using agents in simulated environments.

1.7 Contributions

The main contribution of this thesis work is to integrate the imaginative model, and data augmentation with contrastive learning to improve the image encoder to capture more meaningful features.

1.8 Thesis Organization

The remaining of this document is structured as follows. Chapter 2, presents the related research works. Chapter 3, discusses each component of our approach. Chapter 4 discusses the experimental methods used and the result that we got. Finally, the thesis work concludes in Chapter 5, which also offers suggestions for more research.

CHAPTER 2 RELATED WORKS

Several literatures conduct to handle sample insufficient. They are organized based on the method they follow in two approach: feature(state) representation on the agent sensory observation and world model that predict the future.

2.1 State Representation using Contrastive Learning

Contrastive-based representation learning has gained significant attention in recent years as an effective approach for unsupervised learning of meaningful representations. Here are a few research papers that study contrastive-based representation learning in detail:

SimCLR (Simple Framework for Contrastive Learning of Visual Representations) [6], is developed by Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. SimCLR presents a straightforward yet powerful framework that enhances the agreement between different augmented views of the same image and reduces the agreement between views of different images. This approach involves large batch sizes and a memory bank to store and compare a wide variety of augmented images, leading to substantial improvements in unsupervised visual representation learning. SimCLR's ability to scale with large batch sizes and its simplicity make it a foundation in the field of contrastive learning.

Another influential method is **BYOL (Bootstrap Your Own Latent) [7]**, introduced by Jean-Bastien Grill and his team. BYOL is a self-supervised learning approach that eliminates the need for negative samples by employing a target network to create a consistent prediction target from the online network's output. This innovation addresses the instability often caused by negative samples in contrastive learning, leading to more robust and stable representations. BYOL's architecture includes an online network and a target network, with the target network providing stable targets for the online network, facilitating effective learning without negative samples.

MoCo (Momentum Contrast) [8], proposed by Kaiming He and colleagues, is another significant contribution. MoCo leverages a momentum-based queue of negative examples to facilitate contrastive learning. It maintains a dynamic dictionary with a queue to store a large number of negative samples, ensuring a steady supply of negative examples for

comparison. This momentum encoder allows the model to achieve state-of-the-art performance on unsupervised learning benchmarks by effectively managing negative samples and maintaining a consistent learning process.

CURL (Contrastive Unsupervised Representation Learning) [4], developed by Aravind Srinivas, Michael Laskin, and Pieter Abbeel, combines contrastive learning with reinforcement learning. CURL utilizes contrastive learning to generate useful representations from observations, while the RL component selects actions to maximize rewards. The method employs an encoder to reduce the dimensionality of pixel images, followed by data augmentation to generate contrastive representations. These representations are then used by the RL algorithm to determine the best actions. CURL significantly enhances RL performance by providing useful semantic representations of the agent's environment.

In representation learning they used encoder that encode lower dimension from the pixel image. Before encoding, augmentation is needed to obtain the contrastive representation. The data is generated to compute the loss, which involves creating positive and negative samples relative to the anchor (the original observation). They select a random crop, which enhances the network's generalization and prevents it from being fixed on specific metrics.

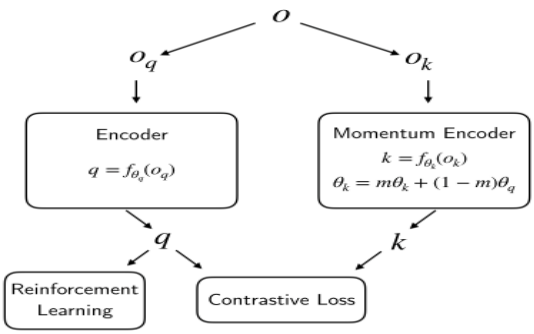

Figure 2.1: **Block diagram CURL [4]**

As shown in Figure 2.1, the encoder takes the anchor (Oq), which is the query, as input, while the momentum encoder processes the sample generated from random augmentation, which serves as the key (Ok). To generate the query and key, both pass through the same network, but the momentum encoder's parameters are updated using a moving average of the query encoder.

The output of the query encoder is used in reinforcement learning to determine the possible action that maximizes the reward. The query encoder is updated using the InfoNCE loss between the query and the key.

They aim to enhance the performance of reinforcement learning by providing a useful semantic representation of the agent's environment (observation). The effectiveness of the representation learning is evaluated after the reinforcement learning process is complete, making it difficult to test whether the features are good during the learning process.

2.2 Learning World Model that Predict the Future

Here is a paper that explores the details of world models and their ability to predict future states:

PlaNet (Planning with Latent Dynamics) [11], developed by Danijar Hafner and his team, is a pioneering model-based RL agent that uses a latent dynamics model to predict future states. The model is trained on a compressed latent representation of the environment's state, enabling it to plan actions effectively. By predicting future states, PlaNet allows the agent to make informed decisions, improving its performance and sample efficiency.

DREAMER (Dream to control: learning behaviors by latent imagination) [12], The paper build upon the world models framework and introduces Dreamer, an agent that learns to perform complex tasks by training a world model and a policy network jointly. The paper focuses on the use of latent imagination, where the agent generates multiple hypothetical trajectories to improve policy learning and decision-making.

The Authors tried to learn behaviors by latent imagination. They generate world model which is trained with the past experience of the agent with state action pairs and with gained reward. Driven by the behavior from the world model’s imagination, the agent takes an instance of observation, encodes it using a convolutional encoder, and predicts actions and rewards through the actor and value networks (proximal policy optimization). The imagination process is carried out three steps ahead in the latent space. Additionally, they

attempt to reconstruct the encoded observation to facilitate the representation network..

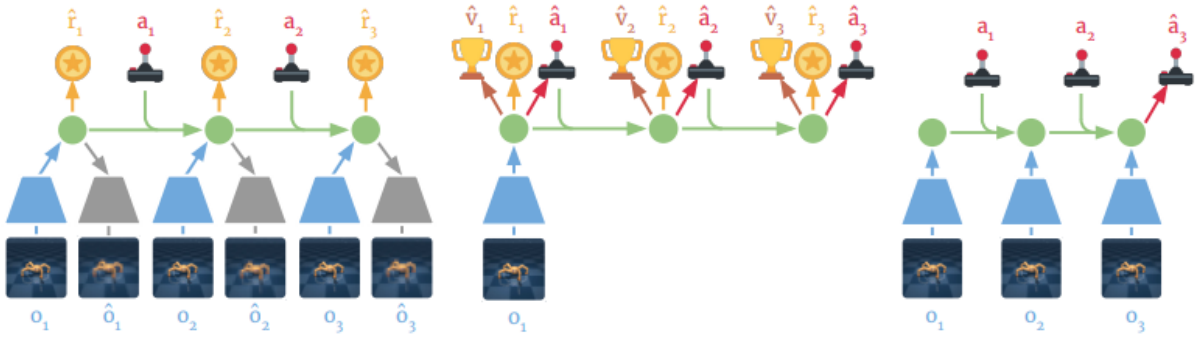

Figure 2.2: Block diagram of DREAMER[12]

As show figure 2.2 below, The original observation (o1) is encoded, and the imagination process generates v1, r1, and a1s. The imagination in latent space is representation of the observation, transition which is the action taken and the expected reward.

Mostly action is generated from the reinforcement learning but in dreamer the action is selected without interacting with the environment. The imagination or behaviors depend on the world model we trained. The data we provide is specific, which limits the imagination and affects its generalization capability.

Generalization is the capacity to achieve good performance in an environment where limited data has been gathered. World models are often trained on limited data, which can lead to biased predictions. If the training data is not representative of the true dynamics of the environment, the learned model may produce inaccurate or unreliable predictions. Model bias can delay the performance of the world model and limit its ability to accurately simulate future states.

SPR (Self predictive representation) [14], The paper trains an agent to predict its own latent state representations several steps into the future. They compute target representations for future states using an encoder which is an exponential moving average of the agent's parameters and they make predictions using a learned transition model.

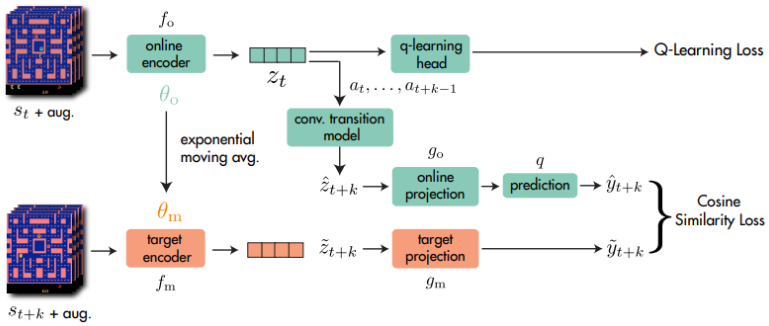

Figure 2.3: Block diagram of SPR[14]

Representations from the online encoder are used in the reinforcement learning task and for prediction of future representations from the target encoder via the transition model. The target encoder and projection head are defined as an exponential moving average of their online counterparts and are not updated via gradient descent

CHAPTER 3 PROPOSED APPROCH

The general architecture of the proposed approach is presented in Figure 3.1. It consists of three major components: Soft Actor-Critic (SAC), Contrastive Representation, and an Imaginative Module.

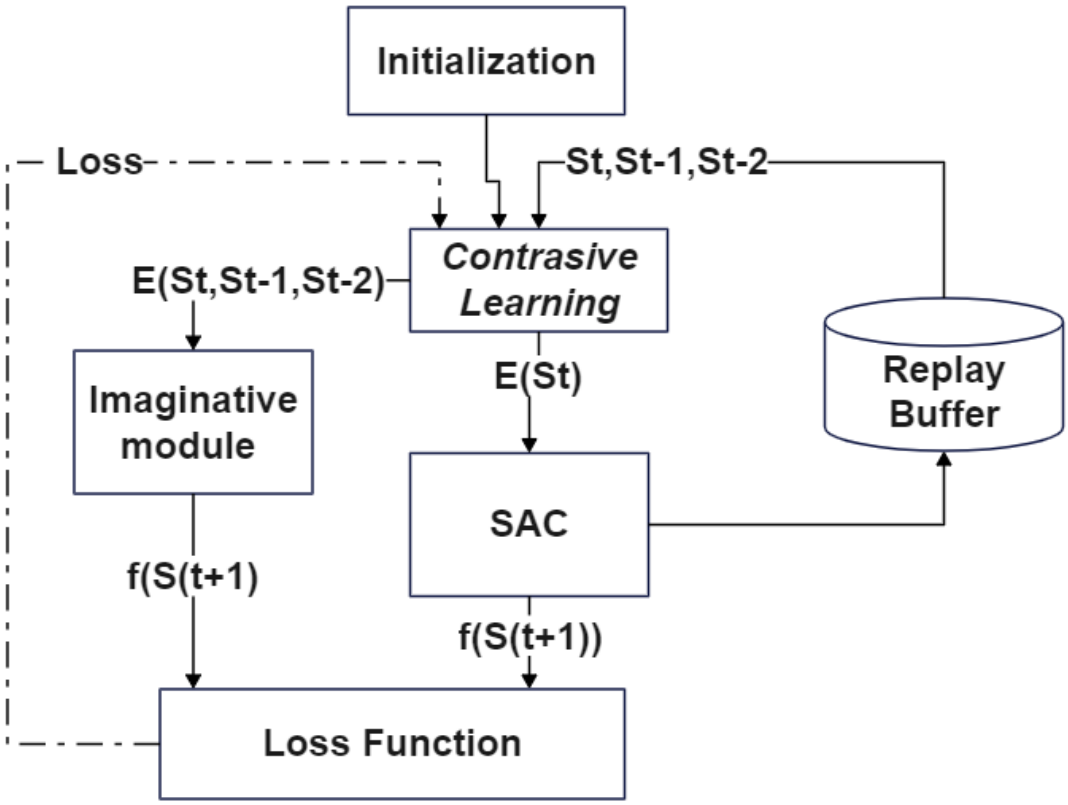

Figure 3: Block diagram of Proposed Approach

We described each building blocks in detail bellow.

3.1 SAC: -Soft Actor-Critic

First introduced by Haarnoja, Tuomas, SAC is off-policy reinforcement algorithm that combines the actor critic architecture with separate policy and value function network and maximization of the entropy. The "soft" in the name refers to this entropy-regularized objective, which makes the policy less greedy and more stochastic. Entropy is a measure of policy uncertainty on the given state. Off policy algorithm enables the reuse of previously collected data for efficiency. [18]

It encourages policy to have high uncertainty in action selection. This encourages exploration by discouraging the policy from becoming deterministic and promoting the exploration of different actions. It seeks to balance the exploration and exploitation of the policy to make diverse policy. It combines ideas from deep Q-learning and actor-critic methods to learn policies for continuous action spaces. The paper demonstrates the effectiveness of the soft actor-critic algorithm for continuous control tasks in reinforcement learning. It is designed to optimize both the policy and the value function simultaneously, enabling stable and efficient learning in continuous control tasks. [18]

3.1.1 Soft Actor-Critic algorithm

Actor-Critic Architecture: SAC is based on the actor-critic framework, which consists of two main components: an actor and a critic. The actor is responsible for learning and generating actions, while the critic evaluates the value or quality of those actions.
The action is selected using the Actor Network through the following steps:

- ✓ Take the state of the agent from the environment
- ✓ Feed the state vector into the actor neural network to obtain the mean (mu) and logarithmic of standard deviation of the distribution policy(log_std)
- ✓ Sample action from the distribution by adding random noise multiplied by exponential of log_std to the mean (mu).
- ✓ Change to valid range with tanh function

The Critic tries to evaluable the selection of the action by the actor network.

Entropy Regularization: To encourage exploration and stochasticity in the learned policy, SAC introduces an entropy regularization term. This term encourages the policy to have a high entropy, meaning it produces diverse and exploratory actions. Maximizing entropy helps to prevent the policy from getting stuck in suboptimal solutions.

Value Function Optimization: It optimizes the value function by minimizing the mean squared Bellman error between the estimated value and the target value. The target value is computed using the soft Q-function and incorporates the entropy regularization term. This approach encourages the critic to estimate the soft Q-value accurately.

Policy Optimization: The policy is optimized by maximizing the expected return, which is the sum of the Q-values multiplied by the policy probabilities. This is done using stochastic gradient ascent, where the gradients of the policy parameters are updated to maximize the expected return. The entropy regularization term is also incorporated into the policy optimization to encourage exploration.

Target Networks: To improve stability during training, SAC employs target networks for both the Q-function and the value function. These target networks are periodically updated with the parameters from the main networks using a target network update rate. Using target networks helps to reduce the variance and stabilize the learning process.

Hyper parameters: SAC has several hyper parameters, such as the learning rate, discount factor, entropy regularization coefficient, and target network update rate that need to be tuned.
It has two Q networks which have the same structure but different initial parameters. They serve to improve the robustness and stability of the learning progress. Each network estimates the action state pair value function and provide Q-values.

Value function provides an estimate of the future rewards that can be expected from a given state. Taking into account the policy action selection and the dynamics of the environment, it helps decision-making by indicating the desirability of different states.

Q-values measure the quality of taking a particular action in a given state. They enable decision making by allowing the selection of action to maximizing the reward.

The two Q network represented as target_Q and Q_. They calculate the Q-values of the policy output at state at time t+1 ($\prod(s_{(t+1)})$ and at time t ($\prod(s_{(t)})$ respectively. The minimum value between the two Q networks used for value estimation and action selection process. The actor network is updated the parameter based on the minimum of the Q-value from the Q network.

The target network is the copy of the Q network. the only difference is the target network update using soft update mechanism to solve the instability and divergence of the policy learning.

soft update the mechanism is update parameter of the target Q network. It uses exponential moving average (EMA)approach rather than update parameter direct copy. It improves stabilize the learning progress and improve the convergence of the Q function and Value function. Instead of direct copying of parameter from the online network to the target network. the target network adjusted slowly towards the Q network. This achieved by the target network parameters is updated using some fraction (EMA) from the Q network parameters.

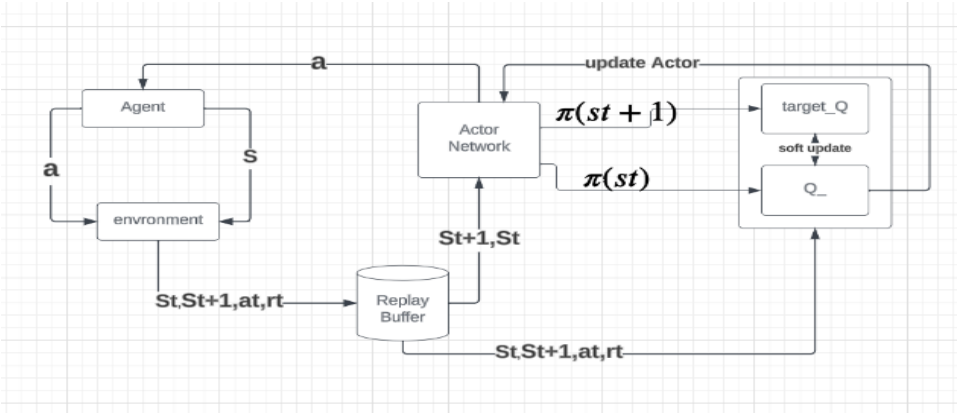

Figure 3.1.1: Block diagram of the SAC

As shown in the figure the actor network takes two state from the environment and give the action probability to agent to interact in the environment and the actor network evaluate the accuracy of probability by taking the output of the environment.

3.2 Contrastive Learning

Contrastive Learning is one way of state representation on the top of RL. It tries to learn representations that capture the underlying structure and semantics of the data, by pushing similar data samples closer together and dissimilar data samples farther apart in the representation space. This is typically achieved by defining a contrastive loss function that encourages the model to learn representations that maximize the similarity between positive (similar) pairs and minimize the similarity between negative (dissimilar) pairs.

The Contrastive learning has been successfully applied to various domains, including computer vision, natural language processing, and state representation learning. Some notable examples include CURL [4], SimCLR [6], MoCo [8], and BYOL [7] for image

representation learning. Specially in CURL it uses state representation learning on the top of RL agent training.

Contrastive learning ([23] [22] [24]) can be understood as learning a differentiable dictionary look-up task.

To specify a contrastive learning objective, we need to define

- ✓ the discrimination objective
- ✓ the transformation for generating query-key observations
- ✓ the embedding procedure for transforming observations into queries and keys
- ✓ the inner product used as a similarity measure between the query-key pairs in the contrastive loss.
- ✓ The specific details of these aspects primarily influence the quality of the learned representations.

The specific details of these aspects primarily influence the quality of the learned representations. The discrimination objective is typically implemented by applying data augmentation techniques to create multiple views or transformations of the same underlying data instance. In our case random cropping is used. Then training the model to maximize the similarity between the representations of these positive pairs (the transformed views of the same instance) and minimize the similarity between the representations of negative pairs (different instances).

The choice of positives and negative samples relative key aspect in contrastive learning. SimCLR [6] and MoCo [8] provide options for a simpler design of instance discrimination by maximizing the mutual information between an image and its augmented version. By learning to distinguish individual instances from each other, the model is encouraged to capture unique and distinguishing features of the data, which can lead to more informative and generalizable representations. It focuses on separating individual instances, the contrastive loss function also encourages the model to learn representations that capture the underlying semantic similarities between related instances (e.g., images of the same state from different viewpoints).

In generation of query and key pairs [6][7], the anchor and positive observations are two different augmentations of the same image while negatives come from other images. To

make different view, we use random crop data augmentation, where a random square patch is cropped from the original rendering. The encoders that map from high dimensional pixels to more meaningful latent [6][8]. InfoNCE is an unsupervised loss that learns encoders f_q and f_k mapping the raw anchors (query) x_q and targets (keys) x_k into latent $q = f_q(x_q)$ and $k = f_k(x_k)$, on which we apply the similarity dot products. It is common to share the same encoder between the anchor and target mappings, that is, to have $f_q = f_k$

Where query (q) is typically the representation of the current input data sample that we want to learn to distinguish from other samples. And Key(k) are the representations of the positive and negative samples that are compared to the query.

From the perspective of viewing contrastive learning as building differentiable dictionary lookups over high dimensional entities, increasing the size of the dictionary and enriching the set of negatives is helpful in learning rich representations. MoCo which uses the exponentially moving average (momentum average) version of the query encoder fq for encoding the keys in K.

3.2.1 CNN

A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for processing and analyzing structured grid-like data, such as images or 2D signals. They are designed to automatically learn hierarchical representations of patterns and features directly from the input data. [27]

3.2.1.1 Components and operations in a CNN:

Convolutional Layers: The core building blocks of a CNN are convolutional layers. These layers use learnable filters (also called kernels) to scan the input data using a sliding window operation known as convolution. Each filter captures specific local patterns or features from the input. During training, the CNN learns to adjust the filter weights to detect relevant features. Multiple filters are typically used in each convolutional layer to learn different features simultaneously.

Pooling Layers: After one or more convolutional layers, pooling layers are often added. Pooling reduces the spatial dimensions of the features maps produced by the convolutional layers. The most common pooling operation is max pooling, which selects the maximum

value within each pooling window. Pooling helps to down sample the feature maps, reducing the computational complexity and providing a form of translational invariance to small local variations.

Activation Function: Non-linear activation functions are applied element-wise to the outputs of the convolutional and pooling layers. Activation functions used in CNNs include Rectified Linear Unit (ReLU), which sets negative values to zero and keeps positive values unchanged, thereby introducing non-linearity to the network. Activation functions help introduce non-linearity, enabling the CNN to learn complex representations.

Fully Connected Layers: Following the convolutional and pooling layers, one or more fully connected layers are often added at the end of the CNN. These layers connect every neuron in one layer to every neuron in the next layer, similar to a traditional neural network. They help to capture high-level abstractions by combining features learned from earlier layers.

Convolutional Filters and Feature Maps: The convolutional filters (kernels) are small matrices that slide over the input data, extracting local features. Each filter produces a feature map, which represents the responses of the filter at different spatial locations of the input. Multiple filters are used to learn different features, and their feature maps are stacked along the depth dimension to form the output of a convolutional layer.

Before the data training, we normalized the data. The convolutional layer extracts feature from the original input. The CNN is only used for feature extraction. The backpropagation depends on the contrastive imaginative learning. [8] MoCo is a contrastive learning framework that utilizes an encoder network to generate the query and key representations. It uses query and key encoder separately. The query encoder is the main encoder network that generates the query representation from one of the augmented views of the input data instance. This encoder network is updated using standard backpropagation during the training process.

The key encoder is a separate encoder network that generates the key representation from the other augmented view of the same input data instance. Unlike the query encoder, the key encoder is not updated directly using backpropagation. Instead, its weights are updated using a momentum-based approach, where the key encoder's weights are a moving average

of the query encoder's weights. This momentum update helps to maintain the consistency of the key representations,

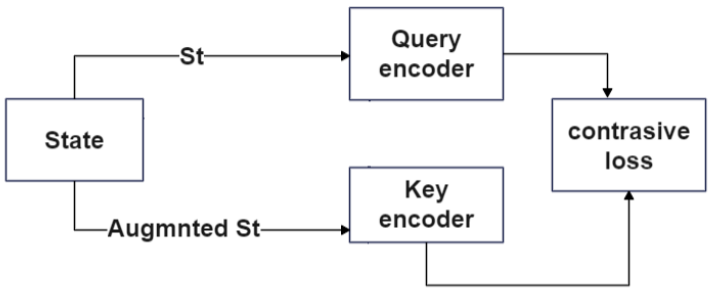

Figure 3.2: Encoder

As shown in Figure 3.2, Both query and key encoder are the same encoders but the key encoder update moving average of the query encoder. The contrastive loss we used, as described in [6][7], is InfoNCE. The query encoder is influenced by the imaginative module, which allows the predicted loss to be back propagated.

The InfoNCE loss is to compare the similarity of positive pairs (instances that are derived from the same underlying data) against a set of negative pairs (instances that are derived from different augmentation data points). The loss function aims to maximize the agreement between positive pairs while minimizing the agreement between negative pairs.

3.2.2 The InfoNCE loss Steps

For each instance, multiple views or augmentations are generated. These views can be different transformations or perturbations applied to the original instance. The model encodes each view of the instance into a latent representation, typically using an encoder neural network.

For each view [4], a similarity score is computed between the encoded representations of positive pairs (views from the same instance) and negative pairs (views from different instances). This can be accomplished using a bilinear product, which is applied to the query and key points.

The InfoNCE loss is calculated by applying a softmax function to the similarity scores and maximizing the log likelihood of positive pairs while minimizing the log likelihood of negative pairs. The loss encourages the model to assign high probabilities to positive pairs and low probabilities to negative pairs. By optimizing the loss, the model learns to capture

meaningful and discriminative representations that capture important information about the data.

To be specific, given a query q and keys K = {k0, k1, ...} where K includes the positive key k+ and the negative keys K\{k+}. The goal of contrastive learning is to ensure that q matches with k+ and is far apart from K\{k+}. Loss function is calculated using Equation 3.1

$$Lq = \frac{logexp(qTWk+)}{\exp(qTWk+) + \sum_{i=0}^{K-1} \exp(qTWki)} \quad \text{----------(3.1)}$$

3.3 Imaginative Module

The imaginative module performs prediction based on the Long Short-Term Memory (LSTM) neural network, utilizing the feature vector generated by the query encoder. The data gathered from the online training from SAC and taking the last three steps.

Figure 3.3: Imaginative module

As show in Figure 3.3, imaginative module tries to choose a batch of three states in replay buffer by using an index that is randomly generated. The latent representation of the states is then formed using the contrastive representation. The next state is predicted using the LSTM based on temporal dependency and historical patter generated by the feature extracted. The imaginative module working on the latent representation of states and to train prediction error will be our loss function. In our world model, the ability to construct unseen states depends on the state representation learned through contrastive learning.

3.3.1 LSTM

LSTM has the capability of learning long term dependencies. It contains its information in a memory that can be seen as a gated cell in which the cell decides whether to store or delete information based on the importance of LSTM network assigned to the information. The LSTM network can store, write, read, delete information on its memory. The memory

of LSTM is similar to computers memory. LSTM networks can remove or add information to the cell state using three gates such as input gate (to let or not to let the new input), forget gate (to delete the information if it is not important) and output gate (to see the impact of the information on the current time step) [13]

LSTM Networks have a chain-like structure as RNN.

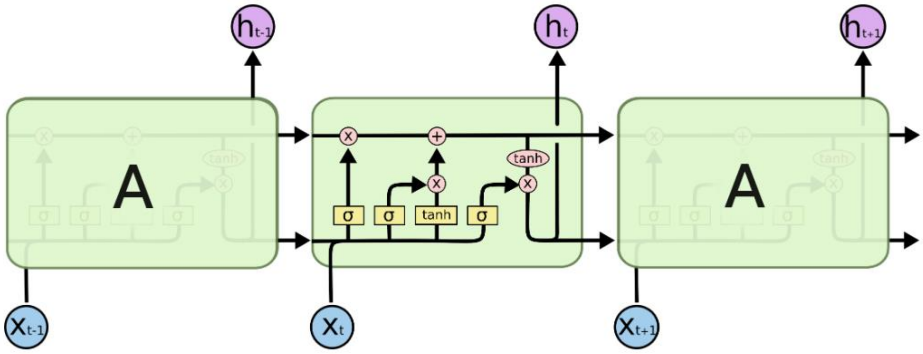

Figure 3.3.1 LSTM Layers [13]

In Figure 3.3.1, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed of a sigmoid neural net layer and a pointwise multiplication operation.

Sigmoid layer(σ): A Sigmoid layer decides whether the new information should be updated or ignored. The sigmoid layer gives a number between zero (let nothing through) and one (let everything through).

Hyperbolic tangent (tanh layer): To overcome the vanishing gradient problem, we need a function whose second derivative can sustain for a long range before going to zero.

Their internal structure involves three core gates (input, forget, output) that regulate information flow and memory cell updates, enabling them to learn and retain relevant information over extended sequences.

The imaginative module comprises an LSTM network that processes a sequence of input tokens. At each time step, the LSTM generates a hidden state (ht) that encapsulates the network's understanding of the sequence up to that point.

The inclusion of three consecutive states as input enables the module to consider the immediate past, present, and near future context, enriching its understanding and influencing its creative direction. It takes advantage of the information contained in batch of three consecutive LSTM hidden states (ht-2, ht-1, ht) as input. This allows the module to not only consider the current context but also leverage the immediate and slightly more distant past, providing a richer understanding of the sequence's flow and potential directions. This enhances the model's capacity to understand context and generate more logical and imaginative outputs.
where the current state depends not only on the immediate observations but also on past states and actions. The encoder is trained to enhance the performance of the imaginative or predictive network, utilizing features that provide insight into dynamics and historical patterns. The predictor predicts with minimal loss so if the encoder generates good feature the agent can know about dynamics and can select appropriate future state.

3.4 Integration of Contrastive and Imaginative Learning:

The proposed approach is hybrid of the contrastive and imaginative learning in reinforcement learning and the sample efficiency can be enhanced by the sampling mechanism used to generate the predicted future states by using the three batch states.

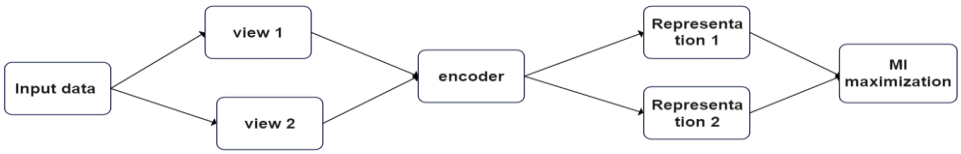

Figure 3.4.1: contrastive learning

As shown in the Figure 3.4.1, in self-learning contrastive learning, which generates positive pairs through augmentations and attempts to maximize mutual information to

build unique features based on the encoded feature. [6][8], the model learns to extract useful features from the input data in order to effectively perform the contrastive task of distinguishing between positive and negative pairs. It focuses on learning representations that capture the differences and similarities between data points, rather than predicting their next generation.

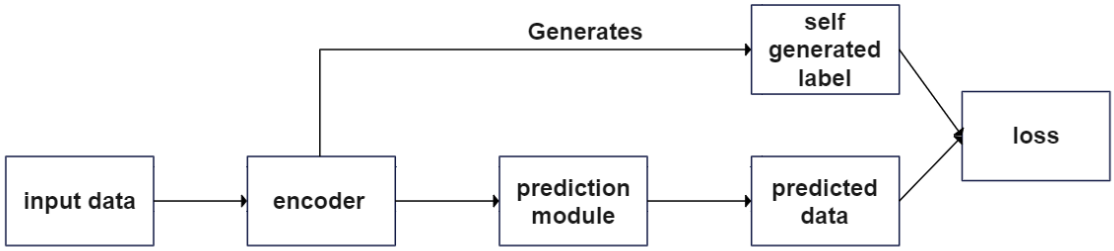

Figure 3.4.2: imaginative(predictive) module

As shown in Figure 3.4.2, imaginative (predictive) modeling makes advantage of data-generated labels. Because the network is affected by predictability or unpredictability in encoder or feature extractor selection, the encoder has an impact on the predictor's capacity.

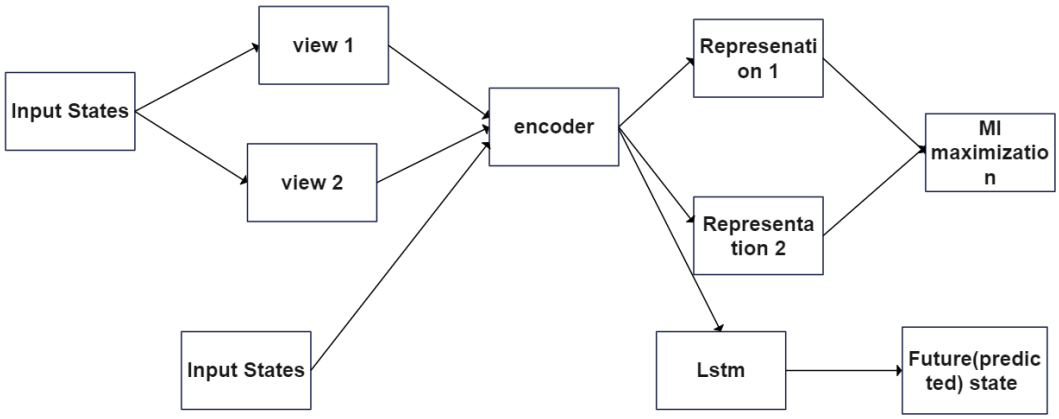

Figure 3.4.3: Integration of contrastive and Imaginative module

This integration enhances the ability to generate meaningful semantic features and improves sample efficiency. By generating better features that enhance the RL performance. In order to capture the contextual and relational information in the data, contrastive learning helps the agent to learn representations that can successfully

distinguish between related and unrelated data samples. Through imaginative learning, an agent can gain a more thorough and consistent grasp of the problem by understanding the underlying dynamics and causal relationships in the environment. Self-generated labels can be improved through the implicit generation of unique feature vectors. This allows world models to learn the behavior of the environment and adjust contrastive learning to select better features.

Generally, the proposed approach combines auxiliary tasks with world models for data-efficient reinforcement learning. Contrastive learning requires data pairs for training, utilizing augmentations of the original data, while the imaginative or predictive component generates labels by creating its own. This combination makes the labeling process explicit, allowing for semantic features to be extracted from the input, with the extraction process enhanced by the predictive capabilities of the features.

To verify if the encoder is selecting meaningful features, we can examine the quality and accuracy of the generated predictions. If the predictions closely match the actual future states or exhibit a high degree of correlation, it indicates that the encoder has successfully captured relevant features from the sensory input.

If the predictions are consistently accurate, it suggests that the encoder has learned to extract and encode the important information for predicting future states. This means that the selected features in the encoding process are indeed meaningful and relevant to the task at hand.

On the other hand, if the predictions are consistently inaccurate or exhibit low correlation with the actual future states, it may indicate that the encoder is not effectively selecting meaningful features. This could imply that the encoder is either capturing irrelevant information or failing to capture important aspects of the sensory input.

CHAPTER 4 EXPERIMENT

4.1 Reinforcement Learning Setup

To implement reinforcement learning [1], we need the following components: -

Agent: - is an entity that learns or makes decisions to behave in the environment. It observes its current state and tries to select an action to maximize the expected reward The agent is the learner who engages with the world around them. The agent's policy must be established; this is a mapping between states and responses. Its policy controls the agent's actions in the world.

Environment: - is the world model that the agent interacts with. It is a transition from one state to another based on the action selected by the agent, it gives a reward or penalty. The agent's environment is the external world in which it must function. It may be either a simulated or actual setting. You must provide the environment's reward function, state space, and action space. The reward function describes the agent's reward or punishment depending on its activities in the action space, while the state space reflects the different states of the environment.

Observation: - is the information that the agent has about the environment at a given time.

Action: - is the move or decision that the agent can make to interact with the environment.

Reward: - is a signal that the agent receives from the environment for taking an action. The agent seeks to incrementally maximize the cumulative reward it earns over time.

Policy:- is a function that maps state to action and determines the action to be taken in a given state. It defines the behavior of the agent. A policy can be stochastic or deterministic.

The RL agent learns by taking actions in the environment, receiving rewards, and updating its policy based on the observed outcomes. The learning process involves an interchange between exploration and exploitation. Exploration involves trying out new actions to discover potentially better strategies, while exploitation focuses on leveraging known effective actions to maximize rewards.

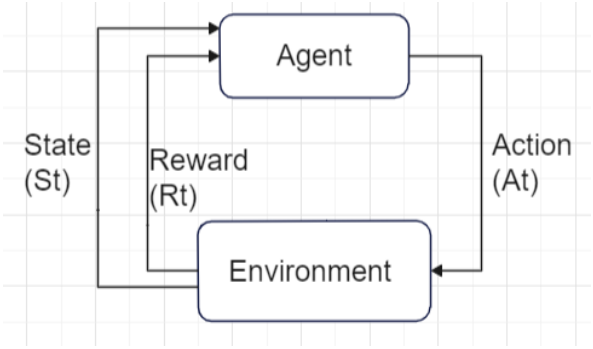

Figure 4.1: Agent-environment interacting in RL

As shown in Figure 4.1, the agent learns by interacting with the environment, receiving rewards for actions that lead to desired outcomes. Over time, the agent learns to maximize rewards.

Dataset preparation

To train the agent, data is collected while it is running and stored in a replay buffer with a capacity of 100,000. There is no need for a pre-existing dataset, as the training is conducted online. The buffer is used solely to store the agent's experiences. Experiment Setup

We have used contrastive representation learning and imaginative jointly trained along the reinforcement learning task. We have five DMControl tasks. These reinforcement learning tasks are trained with 100k time-steps. Those DMControl tasks are listed below and train using pytorch library.

4.1.1 DeepMind Control Suites

The DeepMind Control Suite is a set of continuous control tasks with a standardized structure and interpretable rewards, intended to serve as performance benchmarks for reinforcement learning agents. We will describe those RL agents with their observation and action spaces. [25]

Cart-Pole

Swing up and balance an unactuated pole by applying forces to a cart at its base. The physical model conforms to four benchmarking tasks: in swing up and swingup_sparse the pole starts pointing down while in balance and balance sparse the pole starts near the upright.

The agent's observations typically consist of the positions and velocities of the cart and pole. This includes the cart position, cart velocity, pole angle, and pole angular velocity. The goal is to apply forces to the cart to swing the pole up and balance it in the inverted position, starting from a near-vertical down position. We train the agent with the task of Swing-up.

Ball in cup

A planar ball-in-cup task. An actuated planar receptacle can translate in the vertical plane in order to swing and catch a ball attached to its bottom. The catch task has a sparse reward: 1 when the ball is in the cup, 0 otherwise.

In this task, the agent controls a robotic arm with a cup at the end, and the goal is to swing a ball hanging on a string into the cup. The observations typically include the positions, orientations, and velocities of the arm joints, as well as the position and velocity of the ball. This gives the agent information about the current state of the arm and the ball, which it can use to predict how its actions will affect the ball's trajectory and the cup's position.

Reacher

The simple two-link planar reacher with a randomized target location. The reward is one when the end effector penetrates the target sphere. In the easy task the target sphere is bigger than on the hard task. The agent's observations include the positions and velocities of the robotic arm's joints. Typically, this consists of the angles and angular velocities for each of the arm's segments/links. The goal is to move the end-effector of the robotic arm to a target position in space. The agent needs to learn a policy that can coordinate the joint movements to reach the desired target position. The observations give the agent information about the current state of the robotic arm that it can use to plan and execute the reaching motion.

Cheetah

A running planar biped based on the reward r is linearly proportional to the forward velocity v up to a maximum of 10m/s. The agent's observations typically include the positions, velocities, and orientations of the cheetah's body parts, such as the torso, legs, and joints. This might include things like the x/y/z coordinates of the torso, the angles and

angular velocities of the leg joints, and the linear and angular velocities of the various body parts. The goal is to make the simulated cheetah run as fast as possible, so the agent needs to learn a policy that can coordinate the complex movements of the quadrupedal robot. The detailed observations of the cheetah's state provide the necessary information for the agent to reason about how to generate effective locomotion.

Walker

An improved planar walker based on the one introduced in [16]. In the standing task reward is a combination of terms encouraging an upright torso and some minimal torso height. The walk and run tasks include a component encouraging forward velocity

This involves a simulated bipedal robot that needs to learn how to walk effectively. The agent's observations usually include the positions, velocities, and orientations of the robot's joints, torso, and feet. This provides the agent with detailed information about the current state of the walker's body, which it can use to reason about how to generate stable and efficient walking gaits. The challenge is that walking is a highly dynamic and unstable behavior, so the agent needs to learn a policy that can maintain balance and propel the walker forward.

In general, the observation consists of key values that are passed to the AI agent as an input during a decision point. These can be static values used by the model, the results of some calculations raw or transformed model outputs, and so on. And the action contains the definition of what the AI agent should output for the model to act on at each step. The code field is where you apply this received action to the model’s variables (or functions that perform the action), after which the model will proceed with the next step in the simulation.

4.1.2 Software and Hardware Specifications

The proposed approach is implemented with spider 3 using Pytorch library.

Pytorch

Pytorch is an open source machine learning framework based on the python programming language and the Torch library. Torch is an open source ML library used for creating deep neural networks It's one of the preferred platforms for deep learning research. The framework is built to speed up the process between research prototyping and deployment. [19]

RL agents need to interact with an environment to learn and improve their policies. Pytorch can be used in conjunction with various RL environments, such as those provided by OpenAI Gym and DMC Agents. These environments allow the agent to observe the current state, take actions, and receive rewards, which are then used to update the agent's model.

At the core of PyTorch is the tensor, which is similar to a NumPy array but can be efficiently computed on GPUs. RL algorithms often involve matrix and vector operations, which can be easily expressed and optimized using Pytorch tensor operations. PyTorch's automatic differentiation feature also plays a crucial role in RL, as it allows you to efficiently compute the gradients of the loss function with respect to the model parameters, which is required for training the agent using gradient-based optimization methods. Pytorch provides various utilities for saving and loading models, including the ability to save the model architecture, the trained parameters, and even the optimizer state.

We have trained the proposed approach along with SAC based reinforcement learning and compared the performance with the state of the art baselines. These baselines are CURL [4] and Dreamer [12]. We have made experimentation using 100k time-step.

We use the online data collecting using buffer. This interaction happens in real-time, with the agent making decisions and updating its knowledge based on the current state of the environment. The agent learns and updates its parameters or policies continuously, as it interacts with the environment. The learning process is an ongoing, iterative process, where the agent incorporates new experiences and updates its knowledge incrementally. It can potentially lead to more efficient use of samples or experiences, as the agent can learn from each interaction with the environment.

For handling the dynamics, the replay buffer can help mitigate this issue by maintaining a diverse set of experiences, allowing the agent to learn from a mixture of recent and past data. It can help reduce this variance by providing a more diverse and representative set of experiences for each learning update.

The proposed approach uses the same network model architectures for five agents to evaluate the robustness of the framework even though carefully choosing network model architectures potentially generates better results.

In particular, the query encoder (QE) architecture consists of four convolutional layers with ReLU activation followed by a fully connected projection layer which is similar to SAC-AE [20]. The imaginative module is modeled as LSTM with one hidden layer linear. This linear layer takes the final hidden state hidden [-1, :] from the LSTM layer and maps it to an output of the same size as the hidden state.

The key encoder (KE) architecture is identical to QE. The KE weights are the moving average of the KE weights which are similar to MoCo [8]. The EMA coefficient $\tau = 0.01$. Soft Actor Critic (SAC) [18] is used as a base RL algorithm. The actor and critic use the QE for feature extraction. the batch size is set to 32, the target critic and the target actor are updated every two updates of the main critic. We use random cropping for data augmentation throughout the experiments. The imaginative module is optimized using Adam optimizer [21] with default parameters and initial learning rate $1e{-}3$. All other settings are the same as mentioned in Curl [8].

4.2 Result

The evaluation is to see how our contrastive and imaginative based self-supervised representation learning method, brings enhancement of the reinforcement learning task in both sample efficiency and learning performance. Then we make comparison of the results with state of the art baselines. The learning performance is evaluated using two evaluation metrics of reinforcement learning: the performance of the reinforcement learning (the average returns of the episodes) and the sample efficiency of the learning process.

During training, we simultaneously evaluated the RL agent every 100K environment steps. The result demonstrates our approach significantly improved performance over the baselines: CURL [4], PlaNet [11], SAC-Pixel [18] in all tasks.

100k Steps Score	PROPOSED	CURL	DREMER	PlaNet	PIXEL-SAC
CARTPOLE,SWINGUP	414±84	**582±146**	326±27	303±71	419±40
REACHER,EASY	**556±111**	538±233	314±155	140±256	145±30
CHEETAH,RUN	258±68	**299 ±48**	235± 137	165±123	197±15
WALKER,WALK	**514±155**	403±24	277±12	125±57	42±12
BALL IN CUP,CATCH	**916±194**	769 ± 43	246 ± 174	198±442	312± 63

Table 1: Scores achieved by PROPOSED (max and standard deviation) and baselines on DMC evaluated at 100k environment step.

PROPOSED approach achieves state-of-the-art performance on the 3 out of 5 environments and just below CURL on Cart pole- swing up and Cheetah-Run.

4.2.1 Comparing PROPOSED Method with the DREAMER (RQ 1)

As show in the table 1, the score we have got in all five DMC agent is higher than the DREAMER. We used an auxiliary task (contrastive) and a world model (imaginative) for the representation method.

Contrastive representation learning is a technique used to learn meaningful representations from unlabeled data. It aims to capture the underlying structure and relationships within the data by contrasting positive pairs (similar samples) against negative pairs (dissimilar samples). It encourages the encoder to map similar inputs close together in the learned feature space while pushing dissimilar inputs apart.

The encoder learns to capture high-level features that capture the semantics and relevant information of the states. These features can encode important aspects such as object positions, scene geometry, or other relevant factors depending on the RL task (agent type).

These latent representations contain compressed and meaningful information about the states. the encoder is capable of capturing and encoding the key features necessary for decision-making, even in unseen or novel states. Predicting from raw sensory data is

challenging due to the increasing number of labels generated. The contrastive approach provides advantages in generalization and transfer learning.

By learning meaningful features, the model can better generalize to new, unseen data and transfer its knowledge to related tasks. This is in contrast to models that rely purely on raw data, which may struggle to extrapolate beyond the specific patterns they were trained on.

Focusing on the most relevant and informative features can result in more compact and efficient data representations, reducing the overall complexity of the model and potentially enhancing its performance and sample efficiency.

By combining contrastive representation learning with RL, the RL agent can benefit from the semantically rich and relevant features captured by the trained encoder. This allows the RL agent to make informed decisions based on the latent representations, even without explicit knowledge of the dynamics and patterns of the state space.

When discussing sample efficiency, training the agent using online raw data offers advantages in data quality and representation learning, enhancing its ability to understand environmental behavior.

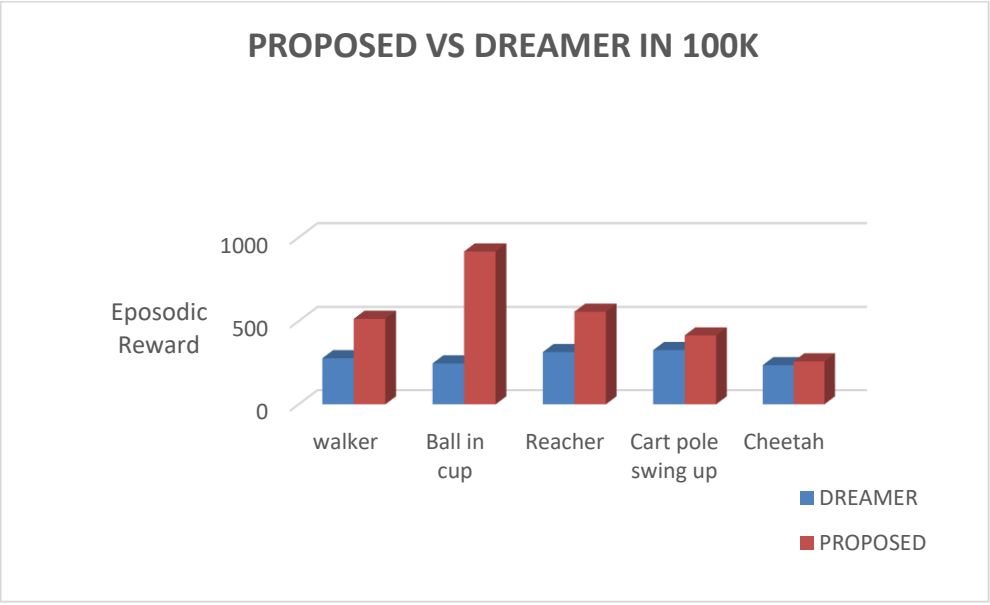

Figure 4.3.1: Average Score over five DMC agents of Dreamer and PROPOSED

As shown in the Figure 4.3.1, for five DMC agents training with 100K iteration the new approach performs better episodic reward than the dreamer. We extend the ability to

generate relevant features that can predict the future state of the agent or an unseen environment. According to the result, at the 100K environment step, OUR (proposed approach) gains 3.7× higher score reward than learning from Dreamer in ball in cup, around 1.8× higher in walker. It positively impacts the performance of future predictions.

4.2.2 Comparing the Our Method with CURL (RQ 2)

The hybrid of contrastive and imaginative based representation performs better than the CURL. It is able to predict near future state is ability of the network using the temporal dependency based on the selected or generated latent representations that have good impact on the contrastive representation learning. According to the result, at the 100K environment step, PROPOSED gains 1.1× higher score reward performance than learning from CURL in ball in cup, around 1.2× higher in walker.

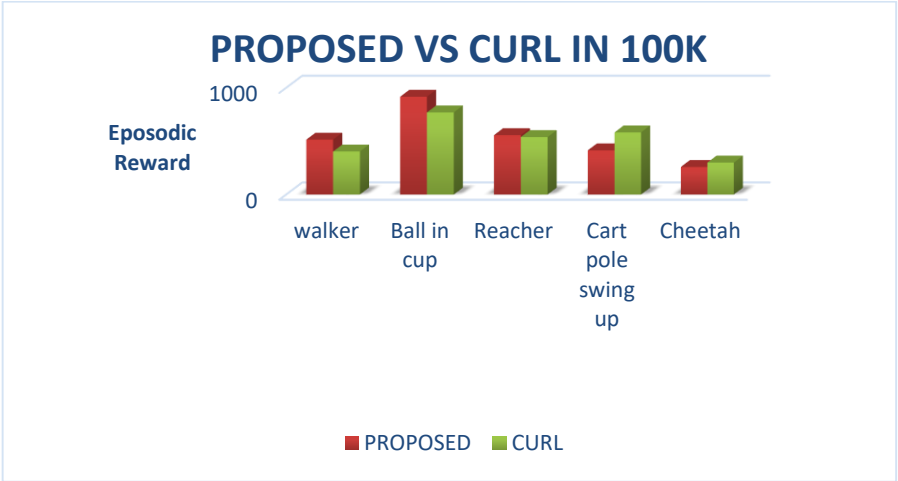

Figure 4.3.2 Average Score over five DMC agent of CURL and PROPOSED

4.2.3 The Impact of the Contrastive Representation Learning on Imaginative Module or vice versa (RQ 3)

During RL training, the RL agent interacts with the environment, receives states as input, and takes actions based on a learned policy. By using the encoder, the RL agent can transform the raw states into latent representations that capture important information. These representations serve as inputs to the RL algorithm, enabling the agent to make informed decisions based on the compressed and semantically rich features.

The benefits of combining contrastive representation learning with RL are several-fold. First, the process allows the encoder to capture relevant features from a replay buffer, enabling the RL agent to generalize well to unseen states. This generalization is particularly valuable when the state space is vast or continuous, as it allows the RL agent to make meaningful decisions without relying on explicit knowledge of the state dynamics.

Second, the semantically rich latent representations facilitate more efficient exploration and learning. The RL agent can leverage the compressed features to identify similarities and dissimilarities between states, enabling it to discover relevant patterns and generalize its knowledge to new situations.

Lastly, combining contrastive representation learning with RL can lead to improved sample efficiency. By training the encoder on history of current policy live, the RL agent starts with a more informative representation of the states. This can reduce the number of interactions with the environment needed for learning, as the agent can make effective use of the compressed information provided by the encoder. The label generated by the imaginative module become depend o the feature generated by the contrastive learning

In summary, combining contrastive representation learning with RL enhances the capabilities of the RL agent by leveraging semantically rich and relevant features captured by encoder. This approach enables the agent to generalize well to unseen states, make informed decisions based on latent representations, and improve sample efficiency in complex and dynamic environments. The sample efficiency enhanced by the encoder always have what will be unseen environments based on predicting error.

The data is collected online when the agent running, this improves the projection of the states by training the Imaginative module with appropriate data. Non-stationary data is beneficial for understanding the dynamic behavior of the environment. The proposed method has good score on 3 out of 5 from our state-of-the-art paper(CURL).

This is because the predictive loss of the imagined future state and the actual one used to train the feature extractor(encoder) enhance the representation learning. It has big factor on the agent performance. Using the consecutive state for prediction in latent space representation is useful because the network can adapt the dynamic behavior of the environment. It became capable of generating valuable information representations that enhance performance.

Lastly the output feature from the encoder can affect the imaginative prediction capacity. If the encoder gives useful information the imaginative model can predict near to the actual data. The predicted network also affects the contrastive learning.

CHAPTER 5 CONCLUSIONS AND RECCOMENDATIONS

5.1 Conclusions

The self-learning technique, which is both imaginative and contrastive, involves predicting future latent states based on historical data collected online. The latent state representation is enhanced with the prediction error of the imaginative network.

The imaginative module has improved the encoder to select more useful representations by using nonstationary data for robust and reliable reinforcement learning. It adapted and transferred knowledge across to select the best and most meaningful information. It uses the latent representation of the state for RL. The imaginative module has improved the encoder to select more useful representation.

Using latent representations of states learned using contrastive learning has shown to be advantageous for reinforcement learning. These representations, refined through contrastive learning, allow the agent to generalize better and make informed decisions on each step of the agents' learning.

The imaginative module's ability to improve the encoder's selection of useful representations has been validated. Since the imaginative module helps the agents' learning to capture temporal relationships between time consecutive states, it improves the selection of the features based on predicting error in contrastive learning. On the other hand, the imaginative module's limitation to learn discriminative and high level invariant and compact latent features is addressed using the contrastive learning module. This enhancement ensures that the agent has access to high-quality, relevant information, further boosting its performance.

5.2 Recommendations

Future research should explore different architectures and configurations of imaginative networks to further enhance the predictive capabilities and robustness of the reinforcement learning agents.

REFERENCES

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

[2] Lindenberg, S. (2001). Intrinsic motivation in a new light. *Kyklos*, *54*(2-3), 317-342.

[3] Pathak, D., P Agrawal, AA Efros, T Darrell (2017). Curiosity-driven exploration by self-supervised prediction. *International Conference on Machine Learning*. PMLR.

[4] Laskin, M., Srinivas, A., & Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. *International Conference on Machine Learning*. PMLR.

[5] Le-Khac, P. H., Healy, G., & Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, *8*, 193907-193934.

[6] Chen, T., S Kornblith, M Norouzi, G Hinton (2020). A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*. PMLR.

[7] Grill, J.-B., F Strub, F Altché, C Tallec, P Richemond, E Buchatskaya, C Doersch, B Avila Pires (2020). Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, *33*, 21271-21284.

[8] K He, H Fan, Y Wu, S Xie, R Girshick (2020). Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[9] PP Liang, Z Deng, MQ Ma, JY Zou, LP Morency, R Salakhutdinov (2024). Factorized contrastive learning: Going beyond multi-view redundancy. *Advances in Neural Information Processing Systems*, *36*.

[10] Ha, D., & Schmidhuber, J. (2018). World models. *arXiv* preprint arXiv:1803.10113.

[11] D Hafner, T Lillicrap, I Fischer, R Villegas, D Ha, H Lee, J Davidson (2019). Learning latent dynamics for planning from pixels. *International Conference on Machine Learning*. PMLR.

[12] D Hafner, T Lillicrap, J Ba, M Norouzi (2019). Dream to control: Learning behaviors by latent imagination. *arXiv* preprint arXiv:1912.01603.

[13] Olah, C. (2015). Understanding LSTM networks. *colah's blog*. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

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

[15] Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *5*, 834-846.

[16] TP Lillicrap, JJ Hunt, A Pritzel, N Heess, T Erez, Y Tassa, D Silver, D Wierstra (2015). Continuous control with deep reinforcement learning. *arXiv* preprint arXiv:1509.02971.

[17] AlMahamid, F., & Grolinger, K. (2021). Reinforcement learning algorithms: An overview and classification. *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE.

[18] T Haarnoja, A Zhou, P Abbeel, S Levine (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*. PMLR.

[19] N Ketkar, J Moolayil, N Ketkar, J Moolayil (2021). Introduction to PyTorch. *Deep learning with python: learn best practices of deep learning models with Pytorch*, 27-91.

[20] D Yarats, A Zhang, I Kostrikov, B Amos, J Pineau, R Fergus (2021). Improving sample efficiency in model-free reinforcement learning from images. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12).

[21] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv* preprint arXiv:1312.6114.

[22] Oord, A. van den, Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv* preprint arXiv:1807.03748.

[23] Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2*. IEEE.

[24] Z Wu, Y Xiong, SX Yu, D Lin. (2018). Unsupervised feature learning via non-parametric instance discrimination. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[25] Y Tassa, Y Doron, A Muldal, T Erez, Y Li, DL Casas, D Budden, A Abdolmaleki, J Merel (2018). DeepMind control suite. *arXiv* preprint arXiv:1801.00690.

[26] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv* preprint arXiv:1506.00019.

[27] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv* preprint arXiv:1511.08458.