Investigating the Effects of Representation Learning on Exploration in On-Policy Reinforcement Learning

by

Can Gözpınar

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Sciences and Engineering

KOÇ ÜNİVERSİTESİ

July 25, 2024

Investigating the Effects of Representation Learning on Exploration in On-Policy Reinforcement Learning

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Can Gözpınar

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Asst. Prof. Barış Akgün (Advisor)

_____

Assoc. Prof. Nazım Kemal Üre

_____

Dr. Fatma Güney

Date: _____

ABSTRACT

Investigating the Effects of Representation Learning on Exploration in On-Policy Reinforcement Learning
Can Gözpınar
Master of Science in Computer Sciences and Engineering
July 25, 2024

Reinforcement Learning (RL) in environments with high-dimensional state spaces is challenging. This is mainly due to the amount and quality of data required to adequately understand the environment, the consequences of actions, and to figure out high-value states/actions. Finding good actions and states, especially if they are sparse and/or there are long-term dependencies, is difficult. An RL agent must explore to find them all the while utilizing what it has learned. Additionally, the complexity of state and action spaces makes it challenging to generalize learned behaviors, requiring sophisticated function approximators and often leading to issues such as overfitting and sample inefficiency. Furthermore, the presence of noise in the data exacerbates these challenges. The effects of noise is more pronounced in high-dimensional spaces because the agent needs to discern meaningful patterns from noisy data, increasing the risk of overfitting to random fluctuations rather than true signals.

Proper exploration is crucial for Reinforcement Learning problems as it can increase the sample efficiency and shorten the training time. Unguided exploration is very sample inefficient in high-dimensional settings. This is especially the case for the hard-exploration problems (e.g. Montezuma's Revenge) in which the agents struggle to learn due to the sparsity of the rewards and the complexity of the state and action spaces. There are several approaches for guided exploration, some of which are proposed to deal with the issues of hard-exploration problems. One of these methods is based on using "prediction-errors" as intrinsic rewards. In prediction-error based methods, a prediction (e.g. next state, reward) is compared against the actual observations. If the discrepancy between those two is high, one concludes that further exploration of such states is required to decrease the error. Exploration

of these states is encouraged by providing extra rewards (intrinsic rewards) when the agent visits them. Such an approach adopts the optimism in the face of uncertainty principle by guiding the agent to the promising yet under-explored parts of the state space. However, in high-dimensional environments, unimportant observations and noise can lead the agent astray.

One promising direction to alleviate these aforementioned issues in high-dimensional and noisy/stochastic environments is learning smaller yet effective and robust state representations. Such an ideal latent representation would be robust to noise and focus on the important aspects of the environment while ignoring the unimportant ones. Utilizing deep neural networks is already a step in this direction. Another potential step is borrowing auxiliary representation learning objectives from self-supervised learning to augment RL.

In light of the observation that operating under small-dimensional state spaces is desirable for both the reinforcement learning agents and the exploration methods, we believe that for prediction-error based exploration methods, receiving support from representation learning methods appears as a viable solution. To this end we propose the *Modified RND* approach to investigate the effect of using an auxiliary self-supervised learning (SSL) loss for the model-predictive exploration methods. Additionally, we also propose the *ViT with Explorative Attention* method which aims to improve exploration performance by learning exploration and exploitation specific representations with just an architectural change without requiring any method from the self-supervised learning literature. Unfortunately, with our proposed methods we have failed to show justifiable performance gains. Only under certain circumstances we have managed to obtain better early training performance which later converged to the performance of our baseline models. Despite its short comings in empirical performance, we still believe that our work presents noteworthy ideas and serves to further one's understanding of the subject. We believe that our work may be a valuable tool to others who are also interested in the intersection of representation learning and prediction-error based on-policy exploration methods in reinforcement learning.

ÖZETÇE

Temsil Öğrenmesinin Politikalı Pekiştirmeli Öğrenmedeki Keşif Üzerindeki Etkilerinin İncelenmesi
Can Gözpınar
Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans
25 Temmuz 2024

Yüksek boyutlu durum uzaylarına sahip ortamlarda pekiştirmeli öğrenme zordur. Bu, esas olarak, ortamı, eylemlerin sonuçlarını ve yüksek değerli durumları/eylemleri anlamak için gereken veri miktarı ve kalitesinden kaynaklanmaktadır. İyi eylemler ve durumlar bulmak, özellikle bunlar seyreklerse ve/veya uzun vadeli bağımlılıklar varsa zordur. Bir pekiştirmeli öğrenme ajanı, bunların hepsini bulmak için keşif yapmalı ve aynı zamanda öğrendiklerini kullanmalıdır. Ayrıca, bu uzayların karmaşıklığı öğrenilen davranışları genelleştirmeyi zorlaştırır, sofistike fonksiyon yaklaşıklayıcıları gerektirir ve genellikle aşırı öğrenme ve örnek verimsizliği gibi sorunlara yol açar. Ayrıca, verilerdeki gürültünün varlığı bu zorlukları daha da artırır. Gürültünün etkileri yüksek boyutlu uzaylarda daha belirgin hale gelir. Bunun sebebi ajanın büyük miktarda gürültülü veriden anlamlı örüntüleri ayırt etmek zorunda kalmasıdır ki bu da rastgele dalgalanmaları aşırı öğrenme riskini artırır.

Pekiştirmeli öğrenme problemleri için doğru keşif çok önemlidir, çünkü örnek verimliliğini artırabilir ve eğitim süresini kısaltabilir. Yönlendirilmemiş keşif, özellikle yüksek boyutlu ortamlarda çok örnek verimsizdir. Bu, özellikle ödüllerin seyrekliği ve durum ve eylem uzaylarının karmaşıklığı nedeniyle ajanların öğrenmekte zorlandığı zor-keşif problemleri (örneğin Montezuma's Revenge) için geçerlidir. Yönlendirilmiş keşif için, zor-keşif problemlerinin sorunlarıyla başa çıkmak için önerilen birkaç yaklaşım vardır. Bu yöntemlerden biri, "tahmin-hataları"nı içsel ödüller olarak kullanmaya dayanmaktadır. Tahmin-hata tabanlı yöntemlerde, bir tahmin (örneğin, bir sonraki durum, ödül) gerçek gözlemlerle karşılaştırılır. Bu ikisi arasındaki farklılık yüksekse, bu tür durumların keşfinin hatayı azaltmak için gerektiği sonucuna varılır. Bu durumların keşfi, ajan bu durumları ziyaret ettiğinde ekstra ödüller (içsel ödüller) sağlanarak teşvik edilir. Böyle bir yaklaşım, ajanı durum uzayının vaatkar ama

az keşfedilmiş kısımlarına yönlendirerek belirsizlik karşısında iyimserlik ilkesini benimser. Ancak, yüksek boyutlu ortamlarda, önemsiz gözlemler ve gürültü ajanı yanlış yönlendirebilir.

Yukarıda bahsedilen yüksek boyutlu ve gürültülü/rastgele ortamlardaki sorunları hafifletmenin umut veren yöntemlerinden bir tanesi daha küçük ama etkili ve sağlam durum temsilleri öğrenmektir. Böyle ideal bir temsil, gürültüye karşı dayanıklı olacak ve ortamın önemsiz yanlarını göz ardı ederken ortamın önemli yönlerine odaklanacaktır. Derin sinir ağlarını kullanmak, zaten bu yönde bir adımdır. Diğer bir potansiyel adım ise, pekiştirmeli öğrenmeye öz gözetimli öğrenmedeki temsil öğrenme hedeflerini eklemektir.

Küçük boyutlu durum uzaylarında çalışmanın hem pekiştirmeli öğrenme ajanları hem de keşif yöntemleri için arzu edildiği gözlemi ışığında, tahmin-hata tabanlı keşif yöntemleri için temsil öğrenme yöntemlerinden destek almanın geçerli bir çözüm olarak göründüğüne inanıyoruz. Bu amaçla, model öngörücü keşif yöntemleri için yardımcı öz gözetimli öğrenme hedefi kullanmanın etkisini araştırmak için *Değiştirilmiş RND* yaklaşımını öneriyoruz. Ek olarak, öz gözetimli öğrenme literatüründen herhangi bir yönteme ihtiyaç duymadan sadece mimari bir değişiklikle keşife ve sömürüye özgü temsilleri öğrenerek keşif performansını arttırmayı amaçlayan *Keşifsel Dikkat ile ViT* yöntemini de öneriyoruz. Ne yazık ki, önerdiğimiz yöntemlerle haklı performans artışları gösteremedik. Sadece belirli koşullar altında erken eğitim performansında daha iyi sonuçlar elde etmeyi başardık, ancak bu performans daha sonra denek modellerimizin performansına yakınsadı. Deneysel performanstaki eksikliklere rağmen, araştırmamızın kayda değer fikirler sunduğuna ve ilgili konuların daha iyi anlaşılmasına hizmet ettiğine inanıyoruz. Çalışmamızın temsil öğrenme ve politikalı pekiştirmeli öğrenmedeki tahmin hatasına dayalı keşif yöntemlerinin kesişimi ile ilgilenen diğerleri için değerli bir araç olabileceğini düşünüyoruz.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Assistant Professor Barış Akgün, for his invaluable guidance and continuous support during my study. I would also like to extend my sincere thanks to my thesis committee members, Associate Professor Nazım Kemal Üre and Assistant Professor Fatma Güney, for their invaluable time, consideration, and feedback. I thank KUIS AI and Koc University for their support and resources. Additionally, many thanks to my family for their encouragement and support all through my studies. Their belief in me has kept my motivation high during this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AE	Autoencoder
BYOL	Bootstrap Your Own Latent
CL	Contrastive Learning
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
EMA	Exponential Moving Average
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GPT	Generative Pre-trained Transformer
ICM	Intrinsic Curiosity Module
MDP	Markov Decision Processes
NLP	Natural Language Processing
OFU	Optimisim in the Face of Uncertainty
PPG	Phasic Policy Gradient
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RND	Random Network Distillation
RNN	Recurrent Neural Network
SPR	Self-Predictive Representations
SSL	Self-Supervised Learning
UCB	Upper Confidence Bound
ViT	Vision Transformer

Chapter 1

INTRODUCTION

Programming agents that make decisions (i.e. take actions) to handle their tasks in complex environments is a challenge. Some issues include (i) lack of "correct action" labels for possible environment states, (ii) properly representing the environment states, (iii) difficulty of modelling and simulating how the environment, including other agents, behave and react, (iv) the dependence between sequential actions, and (v) inherent stochasticity. The field of reinforcement learning (RL) approaches this challenge by looking at the long-term consequences of actions and learning the "best" action given an environment state. The quality of actions are measured by the long-term effects in the form of cumulative sum of (usually discounted) immediate rewards. Thus, the aim of RL is to learn actions that maximize this sum by looking at "interaction data".

In the typical RL setting, an agent takes an action a in a state s, receives a reward r and transitions into the next state s'. Then this continues either indefinitely or until the agent transitions into a terminal state, depending on the task. Actions are inherent to the agent. The state is the representation of the environment by the agent, derived from its observations which is not necessarily the true/exact environment state. Rewards convey information about how the current transition aligns with the agent's objectives. RL algorithms work with this transition (aka interaction) data (mostly the (s, a, r, s') tuples but there are exceptions based on the problem setting or the type of the algorithm). Most of RL algorithms calculate the long-term value of a state, the value of a state-action pair, and/or directly the state-action mapping which is called a policy.

This data is usually gathered by the agent itself through interacting with the

environment. This aspect of RL resembles the way that most of the intelligent organisms, including the humans, acquire skills and make decisions. Even though this idea intuitively makes sense, it introduces some challenges. One such challenge, which we concentrate in this thesis, is the need for "exploration". It is difficult to find good actions, especialy if they are sparse and/or there are long-term dependencies. Even if the agent finds decent actions, there could always be better actions. As a result, the agent needs to visit novel states and try novel actions to find the valuable states and actions that lead to these states. This processes is referred to as exploration.

Pure exploration is costly and may not be feasible. Most of the time we are interested in staying in valuable parts of the state space and only look at promising states and/or actions. In other words, there is no need to explore bad states and bad actions. Also, finding "good enough" (or locally optimal) actions is usually sufficient. Achieving global optimality is extremely difficult, if not impossible for complex tasks and environments. Lastly, the agent shoud increase its confidence on its calculated values and/or policy. Due to these reasons, the agent should also take actions that it thinks are good, i.e., exploit its gathered knowledge. This leads to the typical exploration-exploitation trade-off in RL.

A simple way to handle this trade-off is taking random actions in addition to the best estimated action. The idea is to try novel actions so that the agent eventually visits novel states. This approach is surprisingly effective even for relatively complex tasks. However, when there are long-term dependencies and/or sparse rewards it becomes inefficient, especially if the actions and states are high-dimensional. For such problems, just looking at the rewards and/or taking random actions in all states is not feasible [Burda et al., 2018].

As a result, more complicated approaches have emerged that look at specific states or state-action pairs to guide exploration. The main idea is "optimism in the face of uncertainty"; novel states/state-action pairs potentially lead to better outcomes than what the agent already knows. As such, the novelty/uncertainty itself becomes an intrinsic reward. The source of this reward can be uncertainty of value estimates, uncertainty of actions, prediction errors of the learned model etc.

Using the novelty as reward also alleviates the "couch-potato" problem, ensuring the agents remain motivated to explore even when locally rewarding strategies are discovered.

One such method is called *Random Network Distillation (RND)* which we utilize in this thesis. In RND, a random but fixed neural network called the target network is used [Burda et al., 2018]. This network takes agent observations as input, and outputs a vector of certain size. Then, another network, usually with the same architecture, is trained to mimick the target network. The idea is that the output of these two networks would agree on previously visited states but differ for novel states. Thus, the distance between the network outputs is utilized as intrinsic rewards.

Despite significant developments and improvements of various exploration methods in RL, challenges remain. These include sample inefficiency, high sensitivity to hyperparameters, and scalability when applied to complex environments. For high-dimensional problems, the novelty/uncertainty is difficult to quantify accurately. For example, noise in agent observations or an irrevelant object in the background may lead agent to attribute novelty to an otherwise familiar state. Similarly, an important detail might be overlooked if it is relatively small.

In this thesis, we are motivated by the idea of utilizing a latent space instead of full observations. We posit that such a latent space has the potential to mitigate the aforementioned issues for RND type methods by being robust to noise and representing only the relevant information. The obvious question is then, how to learn such a space? One potential direction is deep learning. Problems with high-dimensional state and/or action spaces had been out of reach for RL until the advent of Deep RL (DRL). Values and/or policies are estimated with deep neural networks (DNN) in DRL. The reasons that make DRL successful are out of the scope of this discussion, but one relevant aspect is that DNNs learn good representations. Thus, the output of an appropriate intermediate layer has the potential to satisfy our latent space conditions.

We are further motivated by the successes of Self-supervised learning (SSL) for representation learning to improve the robustness and usefulness of the latent space. SSL methods use unlabelled data to learn robust and useful representations (feature

embeddings). Example SSL ideas include projecting similar data points together and different points away from each other, predicting original data points from transformed and/or noise added inputs, learning an uncorrelated latent space etc. These learned representations can then be used for downstream tasks ([Grill et al., 2020]) or the SSL ideas can be used as auxiliary objectives during training.

In addition to improving the performance of supervised learning, SSL methods are also applied to RL with some success. They are mainly used in relatively low-data settings to improve convergence which is among our aims. In RL, SSL ideas are mostly used in auxiliary objectives since pretraining almost never works due to non-stationarity. Using these methods during training and their effects on exploration is an interesting topic which is sparsely studied. In this thesis, we want to explore the interaction between exploration and representation learning. Furthermore, such ideas have been only tested with off-policy RL methods. In such methods, the transition data does not have to be generated by the current policy. In constrast, we are interested in on-policy methods, specifically *Proximal Policy Optimization (PPO)*, where the data is directly generated by the method itself. The main high-level distinction is that on-policy methods improve the current policy and learn the associated values whereas off-policy methods aim to learn the optimal policy and the value directly while following an exploratory policy.

The main reason we concentrate on on-policy methods is that exploration comes naturally. These methods concentrate on the "now" and search for the best policy while still exploring. Furthermore, the performance during training and online deployment is similar, and that they usually learn safer strategies. However, they are more susceptible to local minima, "forgetting", sample correlations, and have higher variance. Off-policy methods can re-use previous data to help break sample correlations, stabilize learning, and allow easier escape from local optima. However, previous data can be misleading which would slow down learning (or downright prevent convergence) and evaluating the learned policy while collecting data from another policy causes issues if they are not similar. There are also certain theoretical concerns which are out of the scope of this thesis. Re-using previous data is a tempting property to include representation learning in RL, both due to more stable

learning targets and the ability to perform updates without transitions. However, most of the impact of exploration is on the most recent steps. As such, previous data may not reflect the latest state of exploration, which hinders state/state-action based exploration algorithms. In contrast, on-policy methods use the most recent transitions to learn, and exploration affects the outcomes directly.

Motivated by (i) the potential of latent space exploration to alleviate the shortcomings of exploration methods in high-dimensional state spaces, (ii) SSL's potential to learn robust and useful representations from high-dimensional data, (iii) lack of representation learning and exploration studies in RL, and (iv) naturality of exploration in on-policy methods, we propose the *Modified RND* approach. In *Modified RND*, the backbone (feature extractor) of the PPO model is shared between policy learning, value (advantage) estimation, exploration (RND) and representation learning (an SSL method). In other words, the output of the backbone is our latent representation. The RND component uses the latent vectors as inputs to its models (i.e. target model and network model). The SSL component is designed to provide auxiliary information during learning. In short, *Modified RND*, combines PPO, RND, and SSL together which are all trained concurrently by optimizing a linear combination of their objectives.

We think that such a framework has the potential to learn a latent space that will lead to better exploration and improve training performance. Coupling an SSL objective with the PPO's backbone model could lead to better representations. In turn, this improvement in the learned representations could lead to better policies and value estimations since PPO's critic and actor heads would be operating on better representations. Similarly, RND could generalize better in environments with high-dimensional state spaces because ideally, the learned representations would allow RND's networks to differentiate between different states more easily and be more robust to noise. This result would follow from the observation that representations learned by self-supervised learning algorithms successfully capture relevant information and demonstrate significant generalization capabilities. In addition, states, which the original RND paper (referred to as the Original RND approach) uses as inputs, can be high-dimensional and constructing explorative models be-

comes challenging when dealing with high-dimensional continuous state spaces like images([Pathak et al., 2017], [Wagner and Harmeling, 2024]).

The proposed *Modified RND* approach uses the same general architecture as the Original RND approach. This architecture uses convolutional layers. Motivated by the achievements of transformers in tasks involving vision and sequential decision-making, we decided to take a first look at their potential for addressing exploration challenges. Transformer layers learn what to pay attention to directly based on the data and training objectives. We incorporate two separate Vision Transformer (ViT) modules, one for the values associated with extrinsic rewards (coming from the RL task) and the other one for the values associated with intrinsic rewards (coming from the RND module). The idea is to learn different attention mechanisms and representations for exploration and exploitation. Then, the policy module takes both of these representations. We call this the *ViT with Explorative Attention* approach. The immediate idea is to decouple the representations so that the policy makes more informed decisions. The long-term idea is to utilize the separate representations and the attention to devise novel exploration algorithms which we leave for future work. We hypothesized that by implementing this approach effectively, we could achieve better representations solely through architectural adjustments, without necessitating self-supervised learning methods. However, *ViT with Explorative Attention* can also be combined with our *Modified RND* approach as well.

We evaluate our methods in the Atari domain, focusing on the game "Montezuma's Revenge" since it is a hard exploration game. In contrast to our expectations, our experimental results failed to find justifiable performance gains with our proposed approaches. For the *Modified RND* approach, we suspect that the reasons for the lack of improvement include the simultaneous optimization of multiple objectives, tuning of hyperparameters, the adaptability of the RND module to changing feature extractors, and the architectural changes we made to the RND module compared to the Original RND approach. We tested the validity of our suspisions by proposing solutions and testing them. To this end we proposed the gradient projection method (Chapter 5), the pretraining method (Chapter 6), exponential moving average (EMA) updated target shared PPO backbone modification (Chapter 6), and

the architectural change to the *Modified RND* idea (Chapter 6). Despite our efforts, we only observed improvements in the early training performance. Moreover, for the *ViT with Explorative Attention* approach, we suspect that the data requirements for transformers to excel were not met during our experiments, and that the inherent biases in convolutional neural networks (CNNs) gave the Original RND approach an edge in our evaluation domain. Despite our lack of positive results, we see a potential in learning appropriate latent spaces to improve exploration. We believe that our work can serve as a stepping stone for the future studies and improve one's understanding of the subject, both intuitively and technically.

In the following chapters, we briefly introduce relevant parts of the RL exploration literature (Chapter 2) and the self-supervised learning literature (Chapter 2) with an emphasis on its computer vision (CV) applications. Then, in Chapter 3, we introduce RND (Chapter 3.3) as our choice of RL exploration method, and BYOL (Chapter 3.4) and Barlow Twins (Chapter 3.5) as our choices of SSL algorithms. This is followed by a description of our initially proposed *Modified RND* approach (Chapter 4), along with the modifications we proposed to it (Chapter 5 and 6), and the proposed *ViT with Explorative Attention* approach (Chapter 7). These are presented with evaluations of their experimental results, followed by their limitations (Chapter 8) and our suggestions for future directions (Chapter 9). Finally, we conclude by sharing our thoughts (Chapter 10).

Chapter 2

RELATED WORK

In this chapter, we go over the literature of RL exploration methods and self-supervised learning with an emphasis on the approaches which are relevant to our framework. It serves to better one's understanding of where our inspiration comes from and where our approach fits in the literature.

2.1 Exploration in Reinforcement Learning

Exploration is a fundamental part of the reinforcement learning (RL) training process. It involves taking actions in states which the agent is uncertain about in order to gather more information about the environment. The need for exploration in RL arises from the fact that RL agents gather the data which will be used for their training by interacting with their environment. This aspect of RL differs from the supervised and unsupervised learning paradigms. Unlike the aforementioned paradigms, in RL the training data is not static. RL agents gather the data by interacting with their environments. Usually, these interactions are in the form of actions taken by the agent based on its observations, and the next states and rewards are obtained as a result of the actions taken. It is these interactions that the agent will use to better itself. As a result, it is crucial for the gathered data to be informative and representative of the environment dynamics so that the agent can use it to arrive on a good generalizable policy. For example, if the gathered data only consists of interactions with a small part of the environment, then it might lead to a policy which makes sub-optimal decisions. The exploration of a variety of states is necessary for establishing a good understanding of environment dynamics and eventually being able to develop good value function estimators and policies. Furthermore unlike supervised learning, where data points are assumed to

be independent and identically distributed (i.i.d.), RL involves sequential decision making where the agent's actions influence future states and rewards. This dependency breaks the i.i.d. assumption, making it crucial for the agent to explore the environment to gather comprehensive information about the state space in order to learn effectively. To tackle these issues, exploration methods are developed. On the highest level, they are generally categorized into *undirected* and *directed* methods according to the type of information that they take into account ([Thrun, 1992]).

2.1.1 Undirected methods

These types of methods do not take into account any "exploration-specific" ([Thrun, 1992]) knowledge to decide on the actions. They explore the environment in a random manner. Even though they can be used to explore environments with small state spaces and dense extrinsic rewards, they fail to scale up to hard exploration problems (e.g. Montezuma's Revenge) with large environments and complex tasks. Random walk, ϵ-greedy, and Boltzman exploration methods belong to this category. Furthermore, the undirected methods suffer from safety concerns (notably in real life training scenarios) which stems from the fact that random exploration could result in the unsafe actions being repeatedly taken ([Amin et al., 2021]). These issues motivate the need for a more systematic and efficient exploration approach. This is where the *directed methods* come into play.

2.1.2 Directed methods

Unlike the *undirected methods*, the *directed methods* make use of "exploration-specific" knowledge to guide the agent to explore the environment. As a result, these methods tend to be more efficient than the *undirected methods*. Methods that fall under this category tend to differ with respect to the type of information they use to guide the exploration process. Inspired by [Amin et al., 2021], we further divide these methods into *Intrinsically Motivated Reward-Free Exploration* and *Reward-Based Exploration* methods depending on whether they use extrinsic rewards to guide their exploratory action selection process.

Intrinsically Motivated Reward-Free Exploration

These methods do not utilize extrinsic rewards to guide the action selection process ([Amin et al., 2021]). Instead of extrinsic rewards, they rely on different sources of information which they formulate as intrinsic rewards to select actions. Intrinsic Curiosity Module (ICM) ([Pathak et al., 2017]) in the absence of any extrinsic rewards is one example to this category of methods. Note that the RND method does not fall under this category as it not only uses intrinsic rewards but also uses extrinsic rewards to guide its action selection process.

Reward-Based Exploration

In this category of methods, extrinsic rewards have an impact on the action selection process. They can either choose to make use of other forms of information and formulate them as intrinsic rewards, or choose not to use any. These methods can further be categorized under the following categories ([Amin et al., 2021]).

- **Deliberate Exploration**

  These methods are based on "solving the exploration-exploitation trade-off optimally" ([Amin et al., 2021]).

- **Probability Matching**

  In this group of exploration methods, a heuristic is employed to determine the subsequent action by randomly selecting a single instance from the posterior belief about environments or value functions ([Amin et al., 2021]). The agent subsequently solves for the selected environment exactly and acts in accordance with that solution, typically for the duration of a single episode. This approach ensures that each action is taken with a probability aligned with the agent's belief that it represents the optimal choice. This heuristic effectively steers the exploration effort towards actions that are deemed promising.

- **Optimism/Bonus-Based**

  These methods introduce a bonus to favor actions which the agent is less certain about among all of the possible actions. They perform exploration by

preferring these uncertain actions via the bonus introduced. The motivation behind it is that novel states which are informative can be explored upon taking actions which yield the highest uncertainty. This design choice of preferring uncertain actions falls under the principle of "*optimisim in the face of uncertainty (OFU)*" ([Amin et al., 2021]). These methods are widely adopted for environments with sparse rewards. Upper Confidence Bound (UCB) methods are a key example of optimism-based exploration strategies in reinforcement learning. They tackle the exploration-exploitation dilemma by choosing actions based on both their expected reward and the uncertainty of the reward estimates, encouraging exploration of actions with high uncertainty that might reveal higher rewards. Addition of the bonus (intrinsic reward (r_i)) to the extrinsic reward (r_e) can be viewed as augmenting the extrinsic reward to obtain a new reward (r^+) which also guides the exploration of the environment. $r^+(s,a) := r_e(s,a) + r_i(s,a)$. It is important to note that, unlike the extrinsic reward which is provided by the environment, the intrinsic reward is calculated by the agent. The bonus can be calculated via different approaches, such as the *count-based* methods and the *prediction-error* based methods ([Amin et al., 2021]).

- **Prediction-Error Based**

  In this approach, agent contains a learned model which reflects the agent's understanding of its environment. The learned model is trained on the past experiences of the agent. This model's predictions will be compared against the real/observed values to identify states where the agents knowledge is lacking, and hence requires exploration. High discrepancy values between the predicted value and the target value indicates that the agent's learned model cannot generalize to those states. This would signal that those states are novel states. This follows from the assumption that the reason for the agent's model cannot predicting well in those states is that it has not observed similar states in its training data ([Amin et al., 2021]). Motivated by these findings, prediction-error based approaches introduce

an intrinsic reward which guides the agent to explore those states where the discrepancy between the agent's internal model and the real value is high. *Random Network Distillation (RND)* ([Burda et al., 2018]) is one such method which falls under this category. It introduces an intrinsic reward according to the discrepancy between the feature predictions of a randomly initialized fixed network and a learned model on the observed states. *RND* is further explained in Chapter 3.3.

2.2 Self-Supervised Learning (Representation Learning)

Deep supervised learning algorithms are one of the most popular and successful algorithms in deep learning. Their problem formulation requires the data to be labelled. Even though deep supervised learning algorithms tend to excel at tasks given that enough data is supplied, typically the required amount of data is quite large. Furthermore, the gathering and the labeling of the required data can be time-consuming and expensive. This process usually involves human-annotations which also introduces problems such as regulating the quality of the annotations produced by people. Moreover, this dependence of supervised learning on the vast amounts of data hinders its applicability to tasks with small scale datasets. To this end, transfer learning is introduced as a solution.

In transfer learning, one performs supervised pretraining to extract features from large datasets, and then these learned extractions are used (transferred) to excel at tasks with small data. For instance, ResNet-50 models that are pretrained on the ImageNet dataset can be fine-tuned on computer vision tasks with small scale datasets. The improvements in performance and convergence times while using transfer learning can be alluded to the used pretrained model acting as a good parameter initialization for the downstream task ([Gui et al., 2023]). Typically the features learned on the large scale datasets generalize to the downstream tasks. This generalization prevents issues such as overfitting which the training on small datasets are prone to ([Gui et al., 2023]). Despite its promises, transfer learning

has its own downsides. To begin with, the use of supervised pretraining results in models which are sensitive to changes in data ([Ozbulak et al., 2023]). Also, even though fine-tuning models which are pretrained on large scale datasets helps while working with downstream tasks with small scale datasets, supervised pretraining phase still requires the presence of a large scale labelled dataset. Contrary to this, unlabelled data is easier and cheaper (no need for human-annotation) to gather and is more abundant. Motivated by these observations and the shortcomings of transfer learning, self-supervised learning (SSL) is proposed as a viable solution.

SSL makes use of unlabelled data through self-supervision. During the SSL training, the model tackles a pretext task (surrogate task) with pseudo-labels automatically generated according to the chosen pretext task and the attributes of the data. This way of problem formulation in self-supervised learning allows the model to utilize the unlabelled data to learn useful and robust features. The SSL method can be applied as an auxiliary task or it can appear in a separate pretraining phase. A model pretrained in this way can then be fine-tuned in a supervised manner to accomplish downstream tasks. As an example, under the given definition, autoencoders can be viewed as an instance of self-supervised learning in which the pseudo-labels are the input data itself ([Gui et al., 2023]). Following from the success of SSL in the natural language processing (NLP) field, people have tried to match its success in the computer vision (CV) field. Due to the nature of our proposed approach we primarily focus on the applications of SSL in the CV field. The aforementioned pretext task is a separate task than the original one we are trying to solve. They can be formulated in various ways for different domains. These would result in different frameworks which can be categorized in multiple ways. We follow the categorization approach taken by [Ozbulak et al., 2023] due to its coverage of recent non-contrastive SSL methods, elegant categorization, and its focus on the CV domain. Following from their work, SSL methods are divided under the categories of *generative* frameworks and *discriminative* frameworks.

2.2.1 Generative Self-Supervised Learning

In generative self-supervised learning, the objective is to construct suitable data distributions within the input space such as pixel space in computer vision or token space in natural language processing. ([Ozbulak et al., 2023]). A frequent critique of generative self-supervision pertains to its high computational cost, limited compatibility with high-resolution images, and the suggestion that it might not be necessary for effective representation learning ([Ozbulak et al., 2023]). Autoencoders (AEs) and generative adversarial networks (GANs) are examples to models that are trained in this fashion. Generative pre-trained transformer (GPT) models from the natural language processing (NLP) domain are also examples of generative self-supervised learning, where they predict subsequent tokens to learn meaningful representations from large text corpora.

2.2.2 Discriminative Self-Supervised Learning

In discriminative self-supervised learning, the goal is to learn useful representations of the data [Ozbulak et al., 2023]. The loss functions used by these methods tend to be functions which assess the "discriminative power of learned representations" ([Ozbulak et al., 2023]). Methods that fall under this category can be further divided into the following subcategories.

- **Clustering**

  These methods rely on clustering to extract pseudo-labels for the self-supervised pretraining pretext. Unfortunately, they tend to suffer from issues such as "(1) offline training that prevents their usage for large-scale data, (2) large clusters dominating the majority of the labels or small clusters leading to extremely granular labels, (3) empty clusters, (4) requiring knowledge about the number of clusters beforehand, and (5) trivial solutions where all data are gathered in a single cluster which causes the network to collapse" ([Ozbulak et al., 2023]). There exists some works which make use of clustering to drive exploration in RL. For example, [Wagner and Harmeling, 2024] implements pseudo-count

based exploration in RL by clustering state representations using the cluster centers obtained by the Gaussian Mixture Models (GMMs). The inverse square root of the pseudo-counts are then used to assign intrinsic rewards to the corresponding states.

- **Contrastive Learning (CL)**

  Contrastive learning methods make use of the unlabelled data by using pretext tasks which compare ("contrast") the data against each other. These comparisons are based on the positive and negative samples. The positive samples correspond to data instances that are similar to an anchor data instance, whereas the negative samples correspond to different data instances. The interpretation of positive and negative instances can change based on factors like the modality under examination and the specific criteria, such as the spatial and temporal consistency in video comprehension or the interplay of modalities in multi-modal learning situations ([Gui et al., 2023]). For example, for the modality with 2D images, image augmentations are applied to create various perspectives from a single image to generate positive samples. *Instance discrimination* is one such pretext which views augmented images as positive samples which originated from the original (anchor) image, and by contrasting these samples, it directs the model to learn robust representations which are invariant to the augmentations ([Ozbulak et al., 2023]). The general approach for generating negative samples in contrastive learning relies on choosing them randomly from the dataset, operating under the assumption that most random pairs will not be similar. This method is straightforward and effective for many applications. However, in reinforcement learning picking negative samples for contrastive learning involves additional considerations due to the sequential and state-dependent nature of the environment (i.e. interactions within a rollout are non-i.i.d). For off-policy methods negative samples can be randomly sampled from the stored experiences in the replay buffer. This approach is not feasible for on-policy methods like PPO as they operate using trajectories gathered using the current policy and they do not have a replay

buffer. In asynchronous RL methods, where multiple agents interact with parallel environments, negative samples can be selected from different parallel environments. This assumes that the parallel environments are independent and thus should yield negative samples that are sufficiently diverse and non-similar. Overall, the fundamental idea is to encourage closeness among positive instances while maximizing the gap between negative instances within the latent space ([Gui et al., 2023]). In contrast to their strengths, CL methods tend to suffer from network collapse and dimensional collapse issues ([Ozbulak et al., 2023]). It appears that negative samples are important to prevent the network collapse problem ([Ozbulak et al., 2023]). There exists various works which have applied contrastive learning to RL problems. Usually, the goal appears to be increasing the sample efficiency of the RL method and encouraging better exploration. [Nguyen et al., 2021] is one such method which introduces a contrastive learning based forward dynamics model to increase the sample efficiency of an off-policy RL method and to provide intrinsic rewards to guide the exploration process. It increases the sample efficiency by learning better state encoders via the contrastive loss, and it yields intrinsic rewards by mapping the dissimilarity between the anchor and the positive pair to intrinsic reward values. It relies on the replay buffer to obtain the negative and positive pairs of the contrastive learning formulation where the consecutive transitions constitute the positive pairs, and the non-consecutive pairs constitute the negative pairs.

- **Distillation**

  Unlike the contrastive learning methods which rely on negative samples to prevent network collapse, distillation methods do not require negative samples through a process called distillation. The distillation process refers to the training of a student model which tries to predict the representations produced by another model called the teacher model ([Ozbulak et al., 2023]). There are some works which tried to ascribe distillation's independence from negative samples to other factors such as batch normalization ([Fetterman and Albrecth,

2020]) and model size ([Li et al., 2022]). *Bootstrap Your Own Latent (BYOL)* ([Grill et al., 2020]) is one such method which is of interest to us. It relies on Siamese networks where one acts as the teacher network and the other acts as the student network. It is further investigated in Chapter 3.4. The distillation based SSL methods have also been applied to the reinforcement learning problems. For instance, BYOL-Explore ([Guo et al., 2022]) integrated the distillation based BYOL method to the world model of the RL method. By doing so they managed to learn better representations and also guided the exploration process by using the self-supervised world model loss to generate intrinsic reward signals.

- **Information-maximization**

  Information-maximization methods work under the principle of maximizing the information conveyed by the decorrelated learned representations ([Ozbulak et al., 2023]). Similar to the distillation methods, information-maximization methods too do not rely on negative samples to prevent collapse issues. They mitigate these issues by utilizing carefully designed loss functions ([Ozbulak et al., 2023]). These loss functions tend to take the variance, invariance, covariance, and the cross-correlation of the learned features into account ([Gui et al., 2023]). *Barlow Twins* ([Zbontar et al., 2021]) is a method which belongs to this category and is of concern to us. Its loss function promotes resemblance of the learned features derived from altered versions of a data instance while reducing the redundancy among their elements ([Gui et al., 2023]). This idea is mathematically enforced by ensuring that the normalized cross-correlation matrix of the feature representations closely resembles the identity matrix. These methods are further explained in Chapter 3.5. Similarly, the idea of information-maximization has been applied to reinforcement learning to drive exploration. For example, [Chmura et al., 2023] proposed an intrinsic reward which incentivized the maximization of the information content of a trajectory in the state space using Shannon's Entropy. Their method uses trajectory statistics to estimate the entropy of the state visitation distribution, reward-

ing the agent for relative enhancements to this metric. Ultimately, it achieves exploration via the exploration of the trajectories with the highest information content.

2.3 Representation Learning in RL

In reinforcement learning, representation learning is generally used for learning good state representations. Given environments with large state spaces, these learned representations could help with approximating value functions with fewer parameters and improve generalization capabilities to encounters with previously unseen states ([Lan et al., 2022]). Deep reinforcement learning applications with image state spaces commonly employ deep neural networks with multiple layers. Intuitively, these layers take in visual inputs (large raw state representations) and learn to extract useful higher level representations of the image in their intermediate layers which the latter layers make use of to output the desired targets ([Lan et al., 2022]). This view of looking at the deep RL architectures is in accord with our work. Similarly, in our work we view the PPO's backbone layer (intermediate layer of the overall PPO network architecture) as a representation learner. This leads us to relate it with the encoder network (representation generating layer) of the self-supervised methods.

This idea of learning representations is very powerful and it is important to note that the learned representations are not only limited to the state representations. Even though they are not very widely adopted, there are other works which employ representation learning to learn useful action representations for various RL tasks. [Schneider et al., 2023] has investigated the effect of learning action representations by trying to draw relations between the empirical performance of the models and their corresponding loss surfaces. Their results show that the action representations make a difference on the loss surfaces and learning action representations can result in improved efficiency and performance. Similar to RND, [Sun et al., 2023] relied on prediction errors to incentivize exploration. Unlike RND, which only learns state representations and uses state information to guide its exploration process, their

work learns both state and action representations and conducts its exploration in the state-action space.

Additionally, in our proposed Modified RND approach, we are employing representation learning methods that also have the potential to mitigate the sample inefficiency problem by capturing the hidden structures and patterns of the data ([Fujimoto et al., 2023]). [Sun et al., 2023] argued that representation learning through forecasting the latent state representations of the agent several steps ahead could enhance the sample efficiency of the on-policy methods. Similarly, [Schwarzer et al., 2021a] built upon the idea that having state representations which can be used to forecast future states based on future actions should enhance the data efficiency of reinforcement learning algorithms. To this end, they proposed that an agent can enhance its learning efficiency by supplementing reward maximization with self-supervised objectives derived from the structure of its visual input and sequential interactions with the environment. They showed the viability of their claims by integrating a model-free RL algorithm with their proposed Self-Predictive Representations (SPR) as a supplementary loss to enhance sample efficiency. Furthermore, [Schwarzer et al., 2021b] improved the data efficiency of the RL methods by carrying a separate pretraining phase with task-agnostic self-supervised objectives. In our Modified RND code we introduced the option to pretrain the feature extractor network via SSL which could improve the sample efficiency of our approach.

Offline RL methods appear to be a more feasible option when interacting with the environment in an on-policy fashion is not viable. Under such circumstances, the performance of the offline method depends on the quality of the available static training data. Following from this observation, [Sun et al., 2023] aimed at the data gathering procedure and tried to better it. They devised a framework which incorporates unsupervised learning and explores the environment during the data gathering phase. We believe their work can be viewed as an attempt to perform exploration for offline RL using self-supervised learning, whereas our Modified RND approach

attempts to perform exploration for on-policy RL using self-supervised learning.

[Kwon et al., 2024] identified the main issues for end-to-end training as learning good representations, and leveraging the good representations to learn a good policy. In our proposed approaches, learning good representations is handled by the SSL objectives such as BYOL, and learning good policy is handled by the PPO. [Kwon et al., 2024] explained that end-to-end training with just an RL objective such as PPO could fail to learn good representations specifically during the early training phase. To this end, in our approaches when we introduced an auxiliary SSL loss we expected to see better performance during the earlier training phases.

[Schwarzer et al., 2021b] thought about which questions should the representations learned through SSL in pretraining and direct training be able to answer. When examining the questions, they realized that the questions which needed to be answered were task agnostic and hence could be learned separately from the RL objective in an unsupervised way. To answer all of the questions, they combined three tasks namely, forward dynamics modeling, inverse dynamics modeling, and self-supervised goal-conditioned RL. The task-agnostic nature of these questions lead them to learn representations in a separate pretraining phase. Similar to their work, in our Modified RND approach, we have introduced an option for a separate pretraining phase to learn representations in a task-agnostic self-supervised manner.

In their work [Liu and Abbeel, 2021] have pointed out that pretraining on a dataset which is different from their target environment will not be able to perform better than training from scratch in the target environment. They explain this by claiming that under such circumstances the pretraining data ends up being out-of-distribution for the target environment. Similar to this, in our pretraining approach we are pretraining on data which is gathered on the same environment as the agent will be trained. Parallel to [Cai et al., 2023] we are using a random policy to gather the pretraining data.

[Anderson et al., 2015] demonstrated that using a backbone pretrained on the state dynamics in an unsupervised manner results in faster learning agents. [Seo et al., 2022] showed that self-supervised pretraining can prove to be efficient in reinforcement learning tasks that rely on visual inputs. Their findings are in accord with our expectations for Just PPO with SSL to learn quicker than Just PPO.

Chapter 3

BACKGROUND

In this chapter we provide background and briefly describe the methods we use in our thesis.

3.1 Markov Decision Processes

Most of the time, RL settings are formulated as a Markov Decision Processes (MDP). An MDP is typically represented by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma >$. $\mathcal{S}$ is the state space and $\mathcal{A}$ is the actions space. The transition dynamics (aka model) is $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ (or $\mathcal{T}(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$), which represents the probability of transitioning from one state to another given an action and can be deterministic depending on the problem. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [\mathcal{R}_{\min}, \mathcal{R}_{\max}]$ is the deterministic reward function[1]. Lastly, the discount factor $\gamma \in [0, 1)$ is used to prioritize recent rewards and to weigh down the rewards to ensure their sum does not diverge. With this formulation the goal of the agent is to find a policy that maps the states to actions. A policy is formulated as $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ (stochastic policy $\pi(s, a) = P(A = a | S = s)$) or $\pi : \mathcal{S} \to \mathcal{A}$ (deterministic policy $\pi(s) = a$).

Recall that in an RL setting, an agent takes an action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$, receives a reward $r = \mathcal{R}$, and transitions to the next state s'. Then this continues either indefinitely or until the agent transitions into a terminal state, depending on the task. For on-policy RL methods, the action comes from a stochastic policy. RL methods use this interaction/transition data for learning.

The policy is learned with the objective of maximizing the expected cumulative discounted rewards (aka return). Note that the rewards represent the immediate de-

[1]Reward function can take the form $\mathcal{R}(s)$ or $\mathcal{R}(s, a, s')$, and can even be stochastic depending on the problem.

sirability of the state-action pair and not their long-term value. For most problems, it is easier to define rewards in comparison to long-term values. However, values are calculated based on the rewards. The value of a state is defined as the expected return starting from that state, and the value of a state-action pair is the expected return starting from that state and applying that action. These values depend on the policy, and the optimal values correspond to the optimal policy. On-policy methods directly learn these based on the current policy (π), with the corresponding value functions represented as V^{π} and Q^{π} for the states and state-action pairs, respectively. For example, the value function $V^{\pi}(s) = \mathbb{E}_{\pi,\mathcal{T}}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t | s_0 = s, a_t \sim \pi(\cdot|s_t)]$ is the expected sum of discounted rewards when the agents starts from the state $s \in \mathcal{S}$ and continues to take actions according to its policy π. Similarly, $Q^{\pi}(s,a) = \mathbb{E}_{\pi,\mathcal{T}}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t | s_0 = s, a_0 = a, a_t \sim \pi(\cdot|s_t) \text{ for } t \neq 0]$. The policy that maximizes the value function for every state is known as the optimal policy (π^*).

3.2 Proximal Policy Optimization (PPO)

Reinforcement learning methods can be classified as on-policy, off-policy, and offline RL according to the data they make use of during the training phase. In our work, we are using PPO which is an on-policy method. On-policy RL methods require the collection of new experiences using the latest version of the policy (target policy) they are learning. This requirement of continuous data gathering leads them to be sample inefficient. On the other hand, offline RL methods can reuse experiences which were collected by another policy (behavior policy π_β), which allows them to be more sample efficient ([Prudencio et al., 2024]). But this sample efficiency comes at the cost of offline methods being prone to distribution shift. This stems from the fact that offline methods cannot interact with the environment and, hence, are incapable of performing exploration, which is crucial for learning good policies. When such an offline method is deployed in the real environment, if it encounters a state which was not represented in its static dataset (i.e. state which is outside of the training distribution), it will likely make mistakes that can compound over time and

lead the policy to diverge ([Prudencio et al., 2024]). This phenomena is known as the distributional shift problem, and on-policy methods are less prone to this issue.

Offline RL can suffer from divergence due to the extrapolation error problem, which stems from the lack of feedback when using static datasets ([Fujimoto et al., 2023]). The extrapolation error occurs when deep value functions tend to project towards impractical values for state-action pairs that are infrequently observed in the dataset ([Fujimoto et al., 2023]).

Following from these findings, using PPO allows us to mitigate the distributional shift and extrapolation error problems, and due to its active use of the target policy (π) to interact with the environment, we can conveniently investigate the exploration problem in RL. Furthermore, coming up with an efficient way to explore the environment should ideally help with the inherent sample inefficiency problem of PPO.

PPO ([Schulman et al., 2017]) is one of the most commonly used on-policy reinforcement learning algorithm. It falls under the category of policy gradient methods. It is motivated by the observation that traditional policy gradient methods' performances are too dependent on the value of the learning rate. This stems from the fact that small learning rates result in minor changes to the policy, which extend the training time and decrease sample efficiency. In contrast to this high learning rates result in overshooting updates which yield unstable and divergent policies. To handle this dependence issue, PPO proposes the clipped surrogate function which constrains the policy updates to ensure that the policy updates would make significant improvements while avoiding making drastic updates. The clipped surrogate function which is given as $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t, clip(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t, 1-\epsilon, 1+\epsilon))]$ contains a probability ratio between the new policy and the old policy, which is clipped within a specified range to prevent excessively large updates. $\hat{A}_t$ is the advantage function which measures how a given action compares against the average action at a given state, and ϵ which controls the degree of constraint imposed on the policy up-

date. During the PPO training, the agent interacts with the environment to collect trajectories using its current policy. Then for every state in the collected trajectory, the corresponding advantage values and the empirical return values are computed. The data is divided into minibatches and the following updates are performed for multiple epochs. The policy is updated by optimizing the clipped surrogate function, and the value function is updated by minimizing the mean squared error between the predicted value and the computed empirical return. These steps are repeated until the policy converges. Figure 3.1 shows the architecture of PPO which is commonly used in Atari environments with stacked grayscale image states.

It is important to note that despite the improvements of PPO, such as better sample efficiency, stability, and simplicity, the empirical performance of the PPO agent is shown to be highly dependent on various implementation details, such as the policy initialization methods used ([Andrychowicz et al., 2020]). There exists work, such as [Andrychowicz et al., 2020] and [Engstrom et al., 2020], which investigates this implementation aspect of the PPO method.

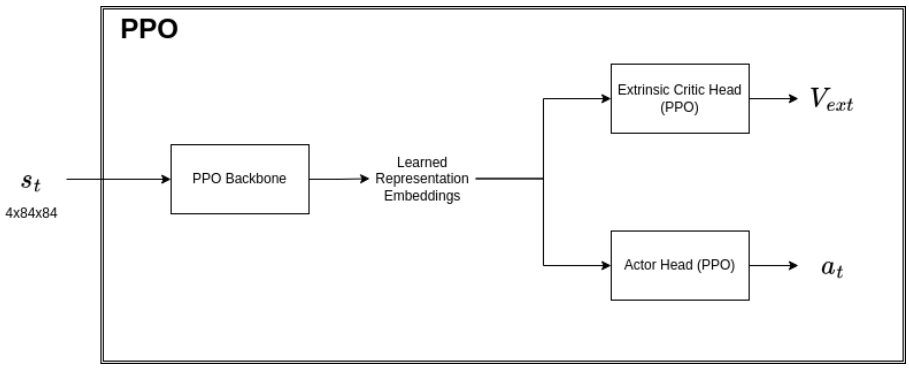


Figure 3.1: Overview of the PPO architecture.

3.3 Random Network Distillation (RND)

Random Network Distillation (RND) ([Burda et al., 2018]) is a *prediction-error based* exploration method (refer to Chapter 2.1.2). It proposes a scalable solution to the exploration problem of hard exploration tasks such as Montezuma's Revenge

in which the rewards (extrinsic reward, r_e) are sparse. It introduces an exploration bonus (intrinsic reward, r_i) which favors the exploration of states (novel) where the discrepancy between the features produced by a fixed randomly initialized *target* neural network (f) and another *predictor* neural network ($\hat{f}$, trained on the data collected by the agent) is high. The *target* network maps observations into embeddings, while the *predictor* network is trained by trying to minimize the expected mean squared error $||\hat{f}(x;\theta) - f(x)||^2$ with respect to its parameters $\theta_{\hat{f}}$ ([Burda et al., 2018]). Through this process, the randomly initialized neural network (*target*) is "distilled" into a trained one (*predictor*) ([Burda et al., 2018]). This bonus is motivated by the idea that neural networks typically exhibit notably reduced prediction errors when dealing with examples that closely resemble those used for their training ([Burda et al., 2018]). Following from this observation, if a model trained on the past experiences of the agent yields higher prediction errors, it should signal that those states possess higher degree of novelty. Additionally, the authors introduced reward and observation normalizations to increase the generalization capabilities of the method and stabilize the process. In the original paper, *Proximal Policy Optimization (PPO)* is modified to implement this method, and the environment's rewards (r_e) are replaced by $r_t = r_e + r_i$, which combines the extrinsic rewards (sparse) with the intrinsic rewards to guide the agent. In Figure 3.2 the rectangular area labelled PPO corresponds to the PPO part of the RND approach. A comparison of Figure 3.2 with Figure 3.1 reveals that the PPO sections are similar with the exception that the RND approach introduces an additional value head (V_{int}) for predicting the intrinsic rewards.

3.4 BYOL

Bootstrap Your Own Latent (BYOL) ([Grill et al., 2020]) is a self-supervised learning method which falls under the category of *distillation* methods (refer to Chapter 2.2.2). It is used for learning image representations which can be used for downstream tasks. It does not rely on negative samples, instead it applies image augmentations to generate augmented views of an image to use as inputs to its pretext

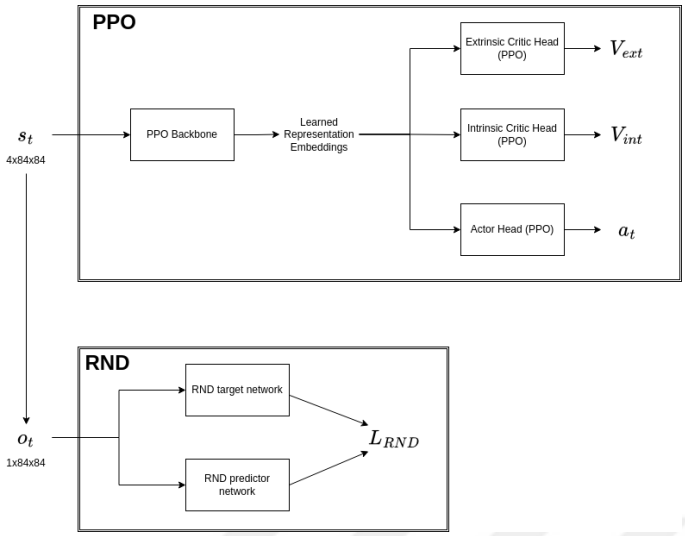


Figure 3.2: Overview of the approach which was proposed by the original RND paper ([Burda et al., 2018]). Given here as a reference to clarify the modifications we make on it.

task. During its self-supervised pretraining, given these augmented views, it directly bootstraps the representations by having its online network try to predict its target network's representations for another augmented view of the same input. The relation between the online (student) and the target (teacher) network is reminiscent of the student-teacher framework. The online network is comprised of an *encoder* (f_θ), a *projector* (g_θ), and a *predictor* (q_θ). The target network has the same parts except for the *predictor* and uses a different set of weights (ξ). The parameters of the target network (ξ) are updated as an exponential moving average of the parameters of the online network (θ).

$$\xi \leftarrow \tau\xi + (1-\tau)\theta, \text{ where } \tau \in [0,1] \text{ is the target decay rate.} \tag{3.1}$$

During the self-supervised training phase, two different views of the same image are generated by sampling two augmentations. Then, one of these views is passed through the target network to produce the target projections. Meanwhile, the other view is passed through the online network to produce predictions for the target projections (i.e. projections produced by the target network). To sum up, online network is trying to predict the projections produced by the target network for a different view of the same image. This yields a loss function which is the mean

squared error between the normalized predictions and the target projections ([Grill et al., 2020]). In the end, we only keep the *encoder* (f_θ) and use it to obtain image representations. Note that in our proposed approach, this encoder corresponds to the backbone (feature extractor) model of our PPO.

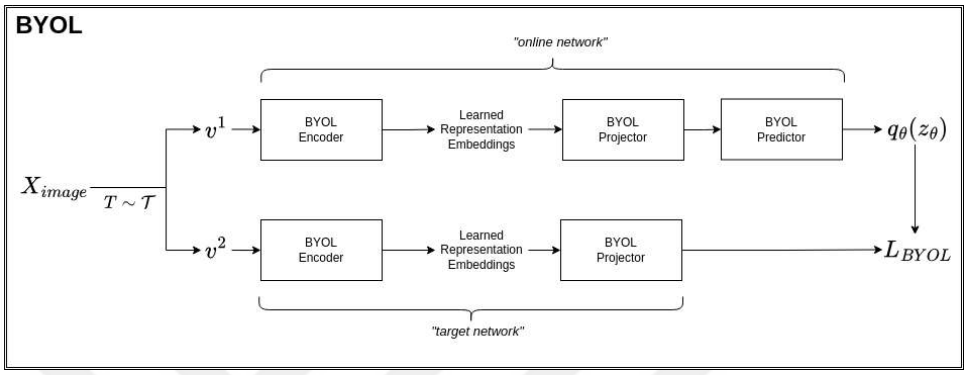


Figure 3.3: Overview of the approach which was proposed by the BYOL paper ([Grill et al., 2020]).

3.5 Barlow Twins

Barlow Twins ([Zbontar et al., 2021]) is a self-supervised learning method which can be categorized under the *information-maximization* methods (refer to Chapter 2.2.2). It is used for learning image representations which can be used for downstream tasks. It mainly relies on data augmentations and its carefully designed loss function to prevent issues such as network collapse and trivial solutions. Unlike *BYOL* (refer to Chapter 3.4), *Barlow Twins* does not rely on an asymmetric Siamese network architecture but two identical networks. During the self-supervised pretraining, two image augmentations are sampled and applied to the same image to generate two views of the original image. These views are passed through the identical networks (f_θ) to produce their corresponding image embeddings (Z^A and Z^B) for the two different views of the same image (Y^A and Y^B). Then the following

loss function is applied.

$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance component}} + \lambda \underbrace{\sum_i \sum_{j \neq i} \mathcal{C}_{ij}{}^2}_{\text{redundancy reduction component}} \quad \text{([Zbontar et al., 2021])} \tag{3.2}$$

where λ is a positive constant, and $\mathcal{C}$ is the cross-correlation matrix as given below.

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2} \sqrt{\sum_b (z^B_{b,j})^2}} \tag{3.3}$$

where b is indexing in the batch dimension, and i, j are indexing in the embedding vectors dimensions.

The design of the loss function can be explained as follows. The *invariance component* of the objective seeks to equate the diagonal elements of $\mathcal{C}$ to 1. This yields embeddings which are "invariant to the distortions applied" ([Zbontar et al., 2021]). Additionally, the *redundancy reduction component* aims to align the off-diagonal elements of $\mathcal{C}$ with 0. This decorrelation results in reduced redundancy between the elements of the embeddings, ensuring that each embedding element carries a unique information about the image it represents ([Zbontar et al., 2021]).

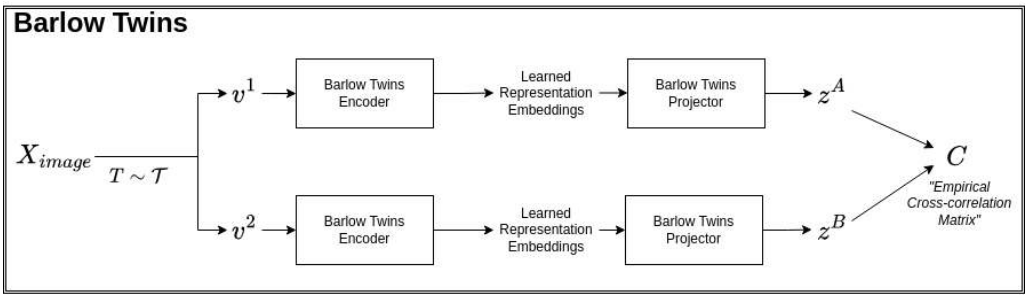


Figure 3.4: Overview of the approach which was proposed by the Barlow Twins paper ([Zbontar et al., 2021]).

3.6 Experimental Setup

In this chapter, we go through the environments in which we have tested our approach and the metrics we chose to evaluate the performance of our methods.

Our experiments made use of the Montezuma's Revenge, Pong, and Breakout environments from the OpenAI GYM's Atari environments. We chose these environments for their wide adoption by the RL research community which makes the hyperparameter tuning process easier. Also, due to the fact that many others have tested their approaches in these environments, it got easier to evaluate our findings by comparing our results to theirs.

In Montezuma's Revenge the agent tries to navigate through multiple rooms of an Aztec pyramid while avoiding dangers. Some of these rooms are shown in Figure 3.5. In the context of reinforcement learning, Montezuma's Revenge is regarded as a hard-exploration problem because of its sparse rewards and the necessity for long-term planning to succeed. This makes it a widely adopted benchmark for evaluating the effectiveness of exploration strategies in RL algorithms. Authors such as [Aytar et al., 2018], [Burda et al., 2018], and [Guo et al., 2022] have deemed it a convenient environment to test exploration algorithms and have shared their results. Following from these, in our experiments, we decided to use the Montezuma's Revenge environment to evaluate the exploration capabilities of our approaches. We facilitated the hyperparameter tuning process by adapting the hyperparameters shared in their work. It is important to note that other works, such as [Seo et al., 2021], have used the MiniGrid environment which detaches the exploration problem from the visual perception problem to investigate the exploration problem in RL. We decided not to use MiniGrid since we are investigating the impact of learning image representations on the exploration problem.

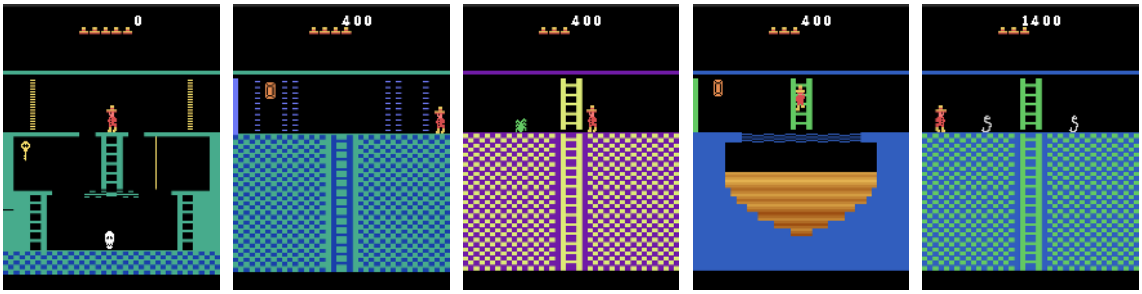


Figure 3.5: Some of the available rooms in Montezuma's Revenge.

We have also used the Pong and the Breakout Atari environments in our experiments to evaluate performance on non-hard exploration problems. We chose these environments as they were widely adopted by the RL community and they did not require the exploration of the environment. In the Pong environment one tries to hit the ball to get it pass the opponents paddle to score a point (Figure 3.6). In the Breakout environment, one tries to bounce the ball using a paddle they control to hit the bricks that are arranged at the top. The goal is to break all the existing bricks without getting the ball pass the controlled paddle (Figure 3.7). Unlike the Montezuma's Revenge environment, both the Pong and the Breakout environments provide denser immediate rewards and they do not require very complex long-term planning to achieve their task. As a result, these two environments are considered as non-hard exploration problems which are easier to solve.

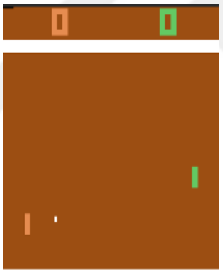


Figure 3.6: Screenshot of the Pong environment.

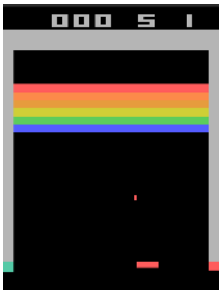


Figure 3.7: Screenshot of the Breakout environment.

Our implementation operates on stacked image observations and makes use of various gym wrappers to make predictions in discrete-action spaces. We are con-

verting images to downscaled grayscale images which are 84 x 84 byte arrays, and stacking these images to allow our CNN backbone to extract some temporal information (This would not be required for a recurrent neural network (RNN) structure but it seems to introduce instabilities in the training (see [Burda et al., 2018])).

Evaluation of an RL agent depends on the environment and the task at hand. We have decided on the following metrics to evaluate our agents based on the characteristics of our environments:

- **Mean episode lengths (over last 100 episodes)**: In environments where the agent's wrong decision can lead to early termination of an episode (e.g. dying in Montezuma's Revenge) or indicate competence (e.g. scoring against your opponent in Pong), this metric can signal the performance of the agent.

- **Mean of rollout rewards (extrinsic)**: Usually, higher values would imply better performing agents.

- **Mean of rollout rewards (intrinsic)**: Gives an insight about how much the agent explores in its environment. Encounter with novel states should result in higher values.

- **Mean undiscounted episodic return (over last 100 episodes) (extrinsic)**: An increasing trend would indicate a better performing agent. This follows from the observation that RL agents goal is to maximize its cumulative rewards.

- **Total number of visited rooms (for Montezuma's Revenge)**: This metric is specific to the Montezuma's Revenge environment. It specifies the number of distinct rooms that the agent has visited through its lifetime. Montezuma's Revenge has sparse rewards and many rooms. Due to the sparsity of the rewards, it is hard for RL agents to find these rooms with the lack of guiding reward signals ([Burda et al., 2018]). We would expect the value of this metric to be high if the utilized exploration method works well.

- **Training loss curves**: Unlike other paradigms in machine learning such as supervised learning, in RL the training curves do not directly signal the performance of the training. This follows from RL concepts such as delayed rewards and exploration-exploitation trade-off. Despite these, we still monitor these curves for further insight but do not rely on them completely like we would in a supervised learning setting.

Chapter 4

MODIFIED RND

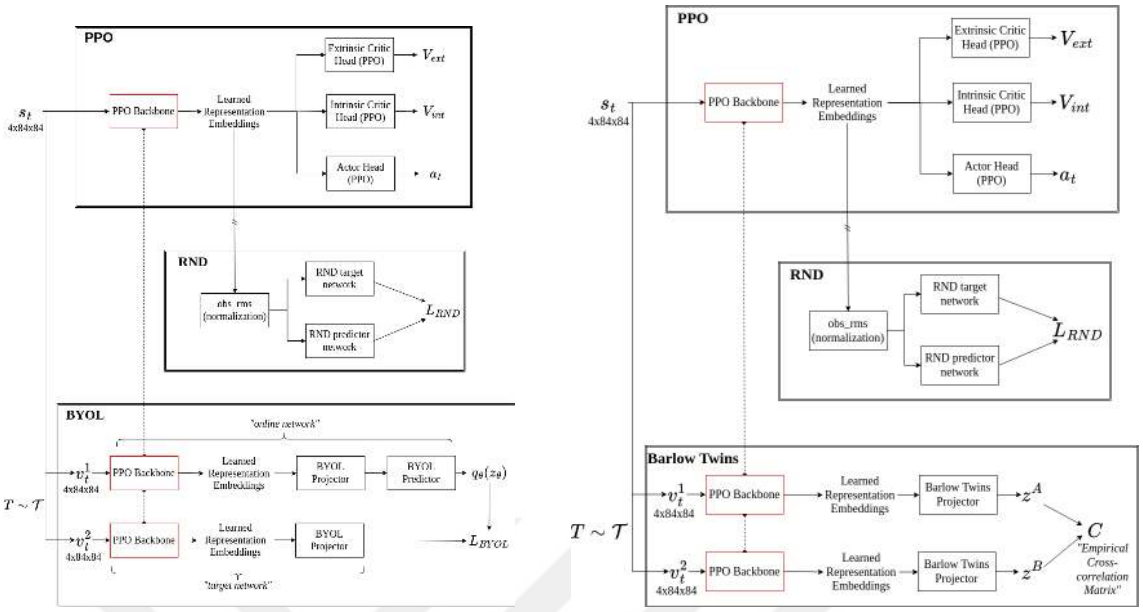
We propose a method which performs exploration in the latent space where the latent space is learned by a combination of an auxiliary SSL loss and an on-policy RL loss. This method which we called the Modified RND approach is used to investigate the impact of bringing *prediction-error based* (Chapter 2.1.2) exploration methods together with *discriminative* (Chapter 2.2.2) self-supervised learning methods.

4.1 Approach

The Modified RND approach builds up on the RND approach and modifies it to bring in the representation learning algorithms in order to reap their potential benefits. At a high level, it can be thought of as being comprised of the following parts:

- Reinforcement Learning Method:
  - Proximal Policy Optimization (PPO)
- Exploration Method:
  - Random Network Distillation (RND)
- Representation Learning Method:
  - Bootstrap Your Own Latent (BYOL)
  - Barlow Twins

Figure 4.1 visualizes the overall design of our proposed approach and highlights its main parts. Next we go over these parts and introduce our method in detail.



Modified RND using BYOL Modified RND using Barlow Twins

Figure 4.1: Modified RND approach with different SSL methods employed in the SSL module.

4.1.1 Reinforcement Learning Method

Recall that we are interested in on-policy methods. We utilize the PPO algorithm as our approach due to its performance, widespread use, and readily available implementations. However, other algorithms can also be used. In our implementation, PPO consists of a CNN backbone (feature extractor), one actor head, and two critic heads (one head for the intrinsic returns, the other for the extrinsic returns). Figure 4.1 groups the aforementioned components of PPO within a rectangle frame marked as PPO. The CNN backbone is used to extract features which are then separately passed to the heads. One thing to note here is that we are using two critic heads. This stems from the fact that since we are using RND, the returns can be decomposed as $R = R_e + R_i$ (R_e is the extrinsic return and R_i is the intrinsic return), and we can fit two separate value heads as V_e and V_i ([Burda et al., 2018]). This design choice is motivated by the observation that "the extrinsic reward function is stationary whereas the intrinsic reward function is non-stationary" ([Burda et al., 2018]).

4.1.2 Exploration Method

For exploration, we use the RND method due to its performance on hard exploration tasks with sparse rewards, relative simplicity, and readily available implementations. [Burda et al., 2018] fed the observations obtained from the environment into the RND networks. In our approach, we substituted the observations with the feature embeddings extracted by the PPO's backbone. We believe that performing RND on the extracted feature embeddings could result in better scalability and increased performance over the original implementation. As shown in Figure 4.2, the last layer of the shared PPO backbone is a linear layer, which means that the inputs to the RND networks become a vector. In contrast to this, the Original RND approach used a few convolutional layers as the first layers of the RND networks, as its inputs were one channel grayscale image observations. This difference leads us to change the RND networks' convolutional layers to linear layers. Additionally, in order to use RND, we modify the critic head of the PPO by introducing a separate critic head for the intrinsic rewards as was explained before. We follow the same reward and observation normalization scheme from [Burda et al., 2018]. Figure 4.1 groups the parts related to the exploration method under a rectangular frame labelled RND and the observation normalization scheme is indicated in the figure with a rectangle labelled *obs_rms (normalization)*. The name *obs_rms* stands for observation running mean statistics, which corresponds to a Python object in our code implementation that aggregates the overall running mean and variance statistics of the inputs to the RND module. During training, these statistics are updated with the new inputs (observations), and at the same time, they are used to normalize these inputs (observations).

4.1.3 Representation Learning Method

For the representation learning method, we relied on the *discriminative* self-supervised learning methods (Chapter 2.2.2). Among the different approaches in the self-supervised learning, we decided to use the methods that belong to the *distillation* and *information-maximization* categories, as these methods do not rely on negative

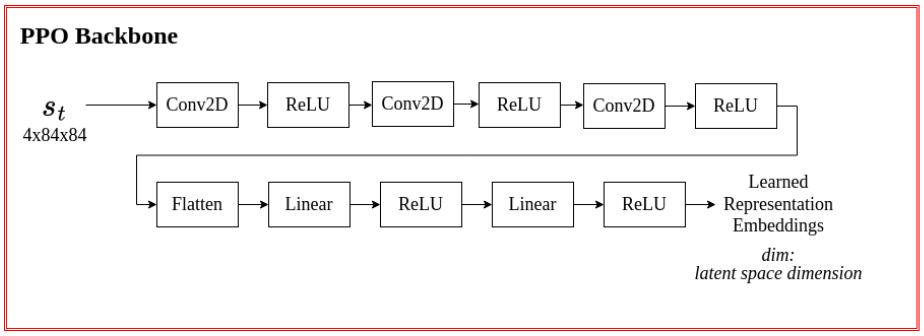


Figure 4.2: Architecture of the shared PPO backbone with a linear last layer.

pairs. We chose two algorithms from these methods, namely BYOL and Barlow Twins. The main modification to integrate these methods is that the encoder (feature extractor) network of these approaches are using the PPO's backbone model. In other words, the PPO backbone is shared with the SSL methods. Inspired by the work of [Guo et al., 2022] and [Schwarzer et al., 2021a] who have managed to successfully train the joint representations via both the RL objective and the self-supervised world dynamics objective, we believe that our shared PPO backbone can be successfully trained by the gradients coming from both the RL objective and the SSL objective. One important thing to note is that since our inputs to the representation learning networks are stacked frames, we could choose to either apply the same random augmentation to all of the stacked images within an input or sample different augmentations for each frame within an input and apply them independently. In our experiments we chose to apply the same augmentation to all of the stacked frames within an input and sample different augmentations for different inputs (stacked frames of states) within a batch. In Figure 4.1, $T \sim \mathcal{T}$ implies that different data augmentations ($\mathcal{T}$) are sampled from the set of all possible data augmentations (T) to produce their corresponding different views of the same input (v_t^1 and v_t^2). Our decision aligns with the work of [Srinivas et al., 2020], which is based on the idea that using identical random crop coordinates across all frames in the stack ensures consistent spatial jittering over time. Furthermore, [Cai et al., 2023] explained that data augmentation-based methods (e.g. BYOL and Barlow Twins) neglect the connection between one state and other states within a trajectory. This

means that the resulting representations lack information about the dynamics of the environment. To alleviate this issue, we are feeding the stacked image observations (i.e. stacked frame states) to our shared backbone which corresponds to the encoder of the SSL network. This way our method can exploit the relations between consecutive frames and encode it in the representations it produces. Following this observation, when we feed the extracted features to the RND module in our Modified RND approach, we convey temporal and environmental dynamics information to the RND module. In contrast, the Original RND approach lacks this information because it feeds a single image frame as input. [Park et al., 2023] pointed out that the use of stronger data augmentations result in better contrastive learning, but the application of intense augmentation is restricted because significant alterations to the input frame can disrupt the downstream task. It is important to note that in our proposed approach we are not suffering from this issue as the inputs to the RL objective and the SSL objective are separate even though the same backbone is being shared. Additionally, we believe that the shared PPO backbone being shared in our Modified RND approach makes our method more resilient to the representation collapse issue, as [Schwarzer et al., 2021a] have demonstrated that concurrently optimizing a reinforcement learning objective alongside an SSL objective (SPR) deters representation collapse.

4.1.4 Loss Formulation

In our approach we are optimizing all of its objectives simultaneously. This is done by optimizing on a loss function which is a linear combination of the individual losses as follows:

$$\mathcal{L} = \alpha_{RL} * \mathcal{L}_{\text{RL}} + \alpha_{SSL} * \mathcal{L}_{\text{SSL}} + \alpha_{RND} * \mathcal{L}_{\text{RND}} \tag{4.1}$$

- where RL loss is:

$$\mathcal{L}_{\text{RL}} = \alpha_{act} * \mathcal{L}_{\text{act}} + \alpha_{crt} * \mathcal{L}_{\text{crt}} - \alpha_{\mathcal{H}} * \mathcal{H} \text{ and} \tag{4.2}$$

$$\mathcal{L}_{\text{crt}} = \alpha_{crt^{(e)}} * \mathcal{L}_{\text{crt}^{(e)}} + \alpha_{crt^{(i)}} * \mathcal{L}_{\text{crt}^{(i)}} \tag{4.3}$$

The individual components are defined in Table 4.1. One important point is that the RND loss does not propagate to the PPO backbone.

$\mathcal{L}$	Total combined loss
$\mathcal{L}_{\text{RL}}$	RL loss (i.e. PPO)
$\mathcal{L}_{\text{SSL}}$	SSL loss
$\mathcal{L}_{\text{RND}}$	RND loss
$\mathcal{L}_{\text{act}}$	Actor loss
$\mathcal{L}_{\text{crt}}$	Total Critic loss
$\mathcal{L}_{\text{crt}^{(\text{e})}}$	Critic loss computed from the extrinsic rewards
$\mathcal{L}_{\text{crt}^{(\text{i})}}$	Critic loss computed from the intrinsic rewards
$\mathcal{H}$	Entropy Loss
α	Coefficients used for linearly weighting the terms

Table 4.1: Variables in equations 4.1-4.3

4.1.5 Motivation

The motivation behind the Modified RND approach is as follows. Exploration is an important concept in RL and it becomes even more important when working with environments with high-dimensional state spaces. Under such conditions, RND faces certain issues. Networks in RND gets larger. Potentially, the noise in the data makes it harder for RND to detect novel states. Also, because the target network in RND is frozen by design, it might struggle to extract useful features which could result in RND not being able to distinguish novel states well. Similarly, the PPO's backbone gets larger, and it might get harder for the PPO to learn useful image representations from just reward signals. To this end, our proposed approach promises the following improvements. In the proposed approach, the RND networks can be smaller since they operate on features extracted by another network (i.e. PPO's backbone) and not the original raw observations. RND can distinguish novel states better due to it taking useful learned representations as inputs which could result in better

exploration. RND's online network can learn quicker due to operating on already extracted useful image features. Learning useful embeddings whose size can be tuned as a hyperparameter is owed to the use of SSL. Furthermore, due to the sharing of the PPO backbone between RND and PPO, PPO should be able to learn better image representations in a shorter amount of time because of the introduced auxiliary SSL loss. This could result in better performing actor and critic heads and overall learn a better policy. Furthermore, in the original RND paper, RND networks only take the last frame (i.e. most recent frame of the stacked frames) as input. This could result in RND not being able to fully understand certain situations. For example, from a single frame where the agent appears is in mid air, it cannot infer whether the agent is going to go up in the next frame or if the agent is going to fall down. On the other hand, in our proposed approach, RND takes an embedding which was extracted from the stacked frames and hence it potentially encodes the required temporal information to know what will happen next. For example, assume that the last four consecutive frames were as given in Figure 4.3. Then the Original RND approach would take the last frame at $t + 3$ as input, in which the agent appears in mid air. It is not possible to understand the direction in which the agent has jumped to appear in such a state, and hence it cannot infer which direction the agent will move in the next frame. On the other hand, the Modified RND approach operates on the last four stacked frames. By looking at $t+2$ and $t+3$ in Figure 4.3, it can possibly determine that the agent has jumped to the right, which helps clarify the position the agent will appear in the next frame.

4.2 Experimental Analysis

In order to test the performance of the Modified RND and its SSL components, we compare them against the Original RND approach and a regular PPO approach (referred to as Just PPO). The comparisons against the Original RND was used to evaluate the explorative performance of our method, and the Just PPO approach was used to evaluate our method's performance against a common RL approach, which is not as explorative as the Original RND approach. We have also both introduced and

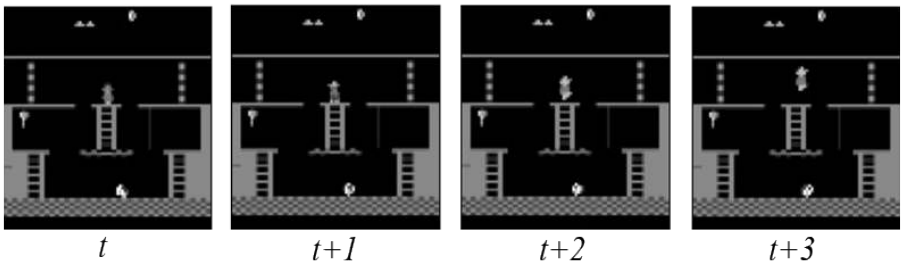


Figure 4.3: An example of four consecutive stacked frames where the agent appears in mid air.

removed the auxiliary SSL objectives from these methods in order to gain a better understanding of the effect that the auxiliary SSL loss has. Our choice of SSL for these experiments was BYOL because we were able to tune it easier during our initial tests. We ran these experiments on the Montezuma's Revenge environment because Modified RND is especially designed for hard exploration problems. Before examining whether our modifications deteriorated performance on less challenging exploration problems, we first tested whether it performed well on the types of problems where it was expected to excel.

The results of the experiments are given in Figure 4.4. From the experiments our findings are as follows. Approaches performance ranking are *originalRND* > *originalRND_448embed* > *justPPO* > *modifiedRDN_BYOL* > *modifiedRND_NoSSL*, where the numbers depict the embedding dimension (2048 if none specified) and the rest are self-explanatory. Original RND can explore more rooms then JustPPO which was expected as introducing intrinsic rewards should help Original RND under sparse reward conditions of the Montezuma's Revenge environment. The Original RND paper had used latent space dimension of 448. We scaled the embedding dimension up to 2048. We have done this thinking that higher embedding dimension can help our proposed Modified RND approach to perform better and more distinctly showcase its potential improvement over the Original RND approach. We expected Modified RND to perform better given that using higher embedding dimension could help the SSL method (BYOL) learn better features. Additionally,

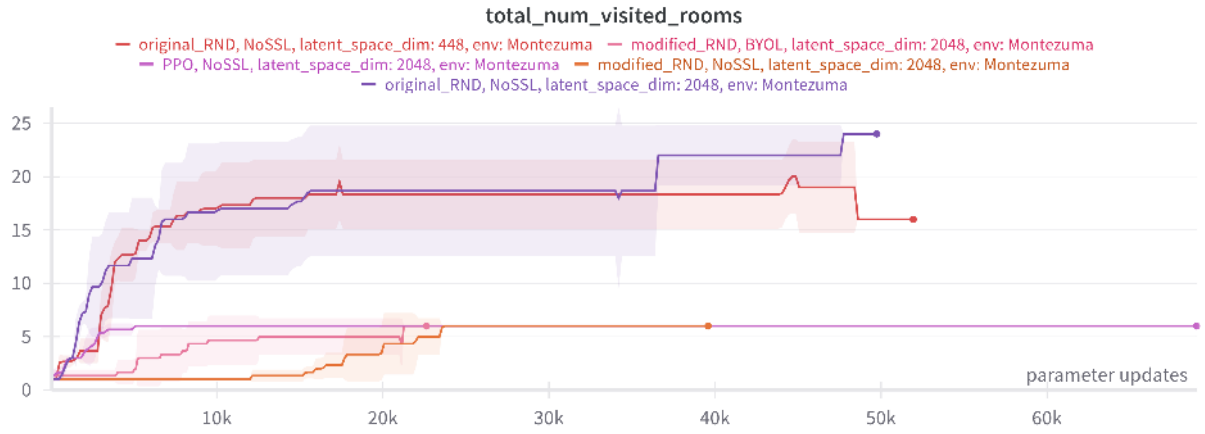


Figure 4.4: Total number of visited rooms vs parameter updates in Montezuma's Revenge. The dark purple and red lines (two at the top) correspond to Original RND with 2048 and 448 latent space dimensions respectively. The light purple line corresponds to Just PPO. The pink and orange lines correspond to Modified RND with and without the BYOL loss respectively.

by making the SSL loss that we have employed in our work use latent space dimension of 2048, we have more closely mathched the original SSL paper's design choice as those papers have used a ResNet-50 architecture which had output dimension of 2048. Unfortunately the results did not completely reflect our expectations and Modified RND ranked the worst by even performing worse than Just PPO. Using higher latent space dimension (2048) matched the performance of using the smaller latent space dimension (448). Following from this, we pursued our experiments with Modified RND using the latent space dimension of 2048. Comparing the Original RND with a latent space dimension of 448 to the Original RND with a latent space dimension of 2048 allowed us to generalize our results. While the Original RND approach keeps exploring newer rooms, the Just PPO approach seems to explore up to 6 rooms and get stuck. In order to justify any modification to the Just PPO framework, the proposed method should perform better than Just PPO approach and find more than 6 rooms. Our Modified RND approach cannot surpass this threshold and hence it does not meet this criteria. As a result, we cannot justify our proposed Modified RND approach which is computationally more expensive than both the Just PPO and Original RND approaches. Adding an auxiliary SSL (BYOL)

loss improved Modified RND's performance. This result is in accordance with our expectation that introducing an SSL loss should result in the model learning useful stacked image (state) representations. The better representations should help the RND module to distinguish unexplored states better by taking in more expressive features instead of raw images. Also, PPO's heads (the critic heads and the policy head) should perform better by leveraging these better extracted features. In the absence of the auxiliary SSL loss the model has to learn to extract useful features using the reward signals. In environments such as Montezuma's Revenge where these signals are sparse, learning feature representations gets harder and it takes longer which also negatively impacts the learned policy. We believe that introducing an auxiliary SSL loss helps model to learn extracting useful features quicker in a better way. This can be explained by the SSL loss not depending on the sparse reward signals and exploiting the inherent structure in the images to train itself in a self-supervised manner which happens aside of the RL objective.

To explain the poor performance of Modified RND in these runs we theorized a few ideas. We hypothesized that the underperformance of Modified RND was a result of conflicting gradients from the PPO and BYOL objectives. As Modified RND was backpropagating both the PPO and the SSL gradients concurrently, it might have appeared that the SSL gradients were predominantly opposing the PPO's gradients, thereby hindering the RL performance. We refer to this issue as the clashing gradients problem. To explore this issue and provide a potential solution, we further investigated the SSL module of the Modified RND approach, which is presented in Chapter 5.

We also hypothesized that the Modified RND performed poorly due to non-stationary inputs to the RND module. Modified RND uses the features extracted by the shared backbone as inputs to the RND module. These extracted features change over time as the shared backbone is trained via the SSL and PPO objectives. Unlike the Original RND framework, this would result in the inputs to the RND networks changing over time. This change in inputs might have caused the outputs of the RND network

to vary, making it difficult for the RND module to distinguish states. In Chapter 6, we investigated this issue and proposed modifications as possible solutions. These ideas are explored after addressing the clashing gradients problem.

Chapter 5

CLASHING GRADIENTS AND GRADIENT PROJECTION

In this chapter, we attempt to investigate and potentially alleviate the suspected clashing gradients problem observed in the Modified RND experiments (Chapter 4.2). Our starting hypothesis is that the gradients of the SSL loss and the gradients of the RL loss might be pulling the models in different directions. We propose the Gradient Projection method, where the gradient of the SSL loss is projected to the tangent plane of the RL gradient if they have components in the opposite direction.

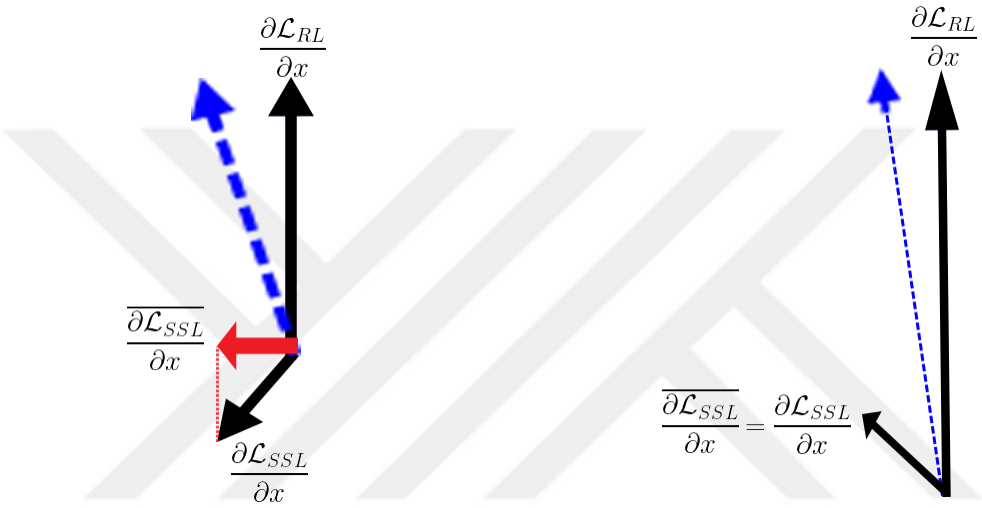
5.1 Gradient Projection Method

5.1.1 Motivation

It was motivated by the idea that optimizing multiple objectives at the same time can give rise to certain problems. The objectives might not be aligned leading to unstable and inferior training performance. The respective loss weights of the objectives would have to be tuned carefully. Given that one of the objectives is an auxiliary objective designed to improve and complement the main objective, one might want to ensure that the auxiliary objective does not hinder the learning of the main objective.

We believe that training updates which cause the auxiliary objective to hinder the main objective's learning can be detected by inspecting their gradients. The gradients can be viewed as two vectors of the same size. If the gradient vector of the auxiliary objective is pointing in the opposite direction to the gradient vector of the main objective, one can conclude that such an update would cause the auxiliary objective to clash with the main objective. Under such conditions, one can choose

to project the gradient of the auxiliary objective orthogonal to the main objective's gradient and use the projected orthogonal gradient in place of the original gradient of the auxiliary objective for parameter updates (see Figure 5.1). This is the idea behind our proposed gradient projection method, which is based on the notion that accumulating gradient vectors with components pointing in the opposite directions would hinder their learning.



auxiliary gradient is projected auxiliary gradient is not projected

Figure 5.1: The arrows correspond to the gradient vectors. The red arrow corresponds to the projected gradient of the auxiliary objective. The dashed blue arrow corresponds to the final accumulated gradient (i.e. combination of the main objective's gradient and the (un)projected auxiliary objective's gradient). *Image on the left:* the auxiliary SSL objective's gradient is pointing in the opposite direction so it is projected orthogonal to the main RL objective. *Image on the right:* the auxiliary SSL objective's gradient is pointing in the same direction to the main RL objective so its gradients are kept the same.

In our investigation, the main objective corresponds to the RL objective (PPO loss) and the auxiliary objective corresponds to the SSL objective (BYOL or Barlow Twins). We thought that the gradient projection method would help if the gradients of the PPO loss and the SSL loss happened to clash during training, allowing the PPO objective to improve even if the auxiliary SSL objective yielded gradients that would otherwise deteriorate the performance of the PPO objective. One can see how the gradient projection method might help with optimizing the main objective, given

the hypothetical scenario with the computed gradients and the loss surface presented in Figure 5.2. Also, the use of gradient projection could improve the combined loss function's tolerance to the weight of the auxiliary loss, as the gradients of the auxiliary loss that clash with the main objective would be projected orthogonally. Overall, we believe that using the proposed gradient projection method could lead to more stable training and better performance in training schemes where different losses are combined.

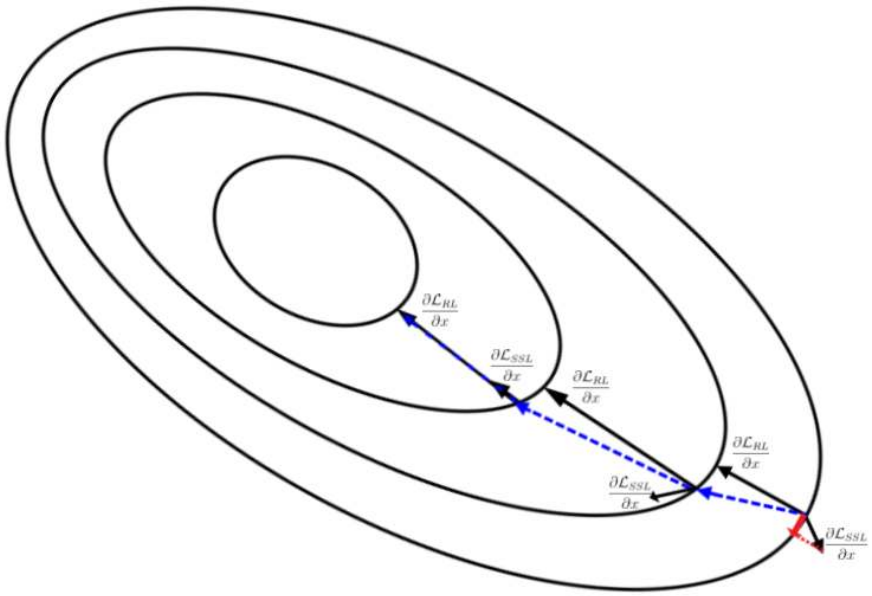


Figure 5.2: Visualization of the gradient projection method in 2D space. The ellipses depict the loss surface and the arrows correspond to the gradient vectors.

5.1.2 Implementation

During the implementation of the method, one can choose between performing gradient projection at the every layer or just at the last layer of the shared network. Applying gradient projection at every layer would be more robust to the clashing gradients problem. On the downside, it would be computationally more expensive as the number of angle calculations and required projections would increase. Conversely, applying gradient projection only at the last layer would be more compute efficient, though it would be less robust to the clashing gradients.

In our implementation, we decided to use the second approach and project the gradients only at the last shared layer, which corresponds to the last layer of the shared PPO backbone (feature extractor). We made this decision to improve computational efficiency. Additionally, we believe that the gradients at the last shared layer are the main source of the clashing gradients problem. By projecting the gradients at this layer and propagating them backwards in a non-clashing way, we aim to mitigate their potential hindrance. In our approach, the last shared layer corresponds to the last layer of the shared PPO backbone, as this is the only part of the PPO method which is shared with the SSL methods.

At the layer where the gradient projection is applied, the gradients are obtained, and the relative direction between the gradient vectors is checked by computing the angle between them. If the cosine of the angle between the gradients of the SSL loss and the PPO loss is negative (i.e. $90° \leq \theta \leq 270°$, or $\cos\theta < 0$), then the gradients of the SSL loss are projected orthogonally to the gradients of the PPO loss (See equations 5.1 through 5.3). These projected gradient are then backpropagated and used to compute the gradients of the model parameters before performing the gradient-based model update.

$$\text{Let } \mathbf{a} = \frac{\partial \mathcal{L}_{SSL}}{\partial x} \text{ and } \mathbf{b} = \frac{\partial \mathcal{L}_{RL}}{\partial x}, \tag{5.1}$$

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{5.2}$$

$$proj_{\mathbf{b}}\mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|^2}\mathbf{b} \text{ and } \frac{\partial \overline{\mathcal{L}_{SSL}}}{\partial x} = \mathbf{b} - proj_{\mathbf{b}}\mathbf{a} \tag{5.3}$$

where $\frac{\partial \overline{\mathcal{L}_{SSL}}}{\partial x}$corresponds to the orthogonally projected gradient vector.

If the gradients are tensors rather than vectors, we first flatten the tensors into vectors. We then perform the angle calculation and projection operations on the flattened vectors. Finally, the resulting vectors are converted back to their original tensor shapes and used in place of the original gradients.

5.1.3 Experimental Analysis

Effects of Gradient Projection on Modified RND

We test the gradient projection method within the Modified RND approach. We believe that observing an improvement over the previous results with the Modified RND approach, without the Gradient Projection method, would indicate that the suspected clashing gradients problem was indeed the underlying issue of the inferior performance of the Modified RND approach.

We first looked at the 448 and 2048 embedding dimensions with gradient projection to mirror the Modified RND experiments (Chapter 4.2). The number of visited rooms and the mean extrinsic returns can be seen in Figures 5.3 and 5.4. Similar to the Modified RND results, a higher dimensional latent space appears to perform better earlier but they all seem to converge to the same number of rooms found. Gradient Projection appears to have slightly improved the early performance of the agent compared to Modified RND. Unfortunately, similar to the Modified RND results without gradient projection, the Modified RND with gradient projection could not find more than 6 rooms and eventually got stuck. Despite the improvement in early performance, the introduction of the Gradient Projection method to the Modified RND approach is still not as performant as the Original RND approach. The additional code and computational complexity introduced by the gradient projection cannot be justified by these findings.

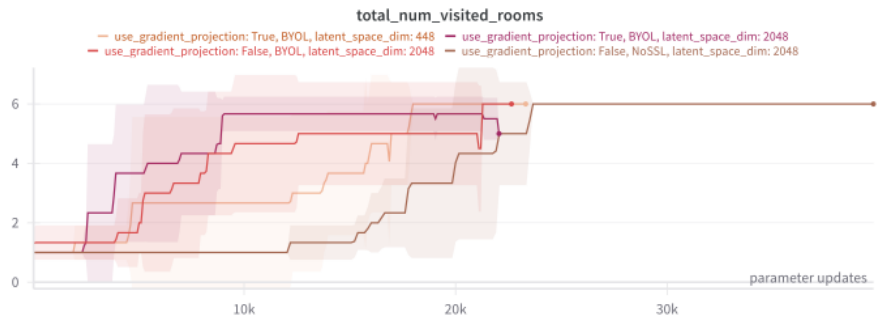


Figure 5.3: Total number of visited rooms vs parameter updates with and without gradient projection for 448 and 2048 latent space dimensions.

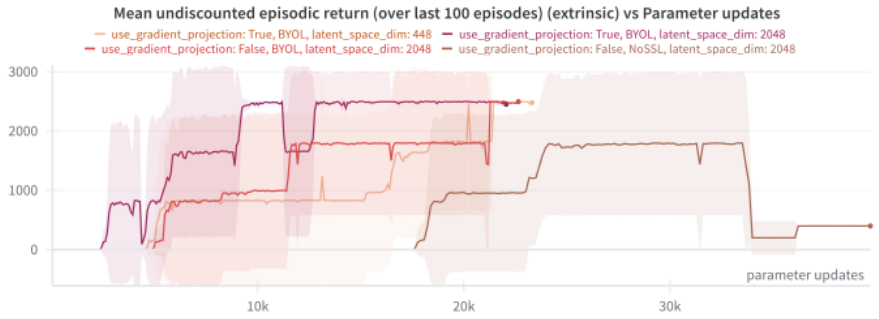


Figure 5.4: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates with and without gradient projection for 448 and 2048 latent space dimensions.

An assessment of these experimental results, under the assumption that our proposed gradient projection method did indeed solve the clashing gradients problem, would have lead us to believe that the clashing gradients between the PPO and BYOL objectives were not the cause of the poor performance of the Modified RND approach. However, since the effectiveness of the proposed Gradient Projection method was not certain, we refrained from jumping to such conclusions.

Effects of SSL on PPO

Instead we devised a further set of experiments with the aim of determining the effects of the SSL module by isolating it from the effects of the exploration module (RND). To this end, we removed the RND module and used the Pong and the Breakout environments which do not require heavy exploration as opposed to the Montezuma's revenge. We ran our experiments using PPO with and without an auxiliary SSL loss. This allowed us to forgo the exploration part and focus solely on the impact that SSL has on RL in an on-policy setting. Also, we repeated our experiments with 2048 and 16 embedding dimensions. 16 was used to examine whether the performance of PPO with a higher embedding dimension (2048) can be matched by using PPO with SSL which uses a smaller embedding dimension (16), as a side assessment.

We believed that observing a performance degradation with the introduction of SSL to PPO would indicate the clashing gradients problem. On the other hand, if there was no deterioration or if performance even improved with the introduction of the SSL module, it would suggest that the exploration module was responsible for the inferior performance of the Modified RND method. With this in mind, we expected BYOL to enhance the overall Just PPO performance and give better results early on in the training. Furthermore, we anticipated that the introduction of SSL would make a bigger difference with a smaller embedding dimension, as SSL could help convey more information with fewer parameters.

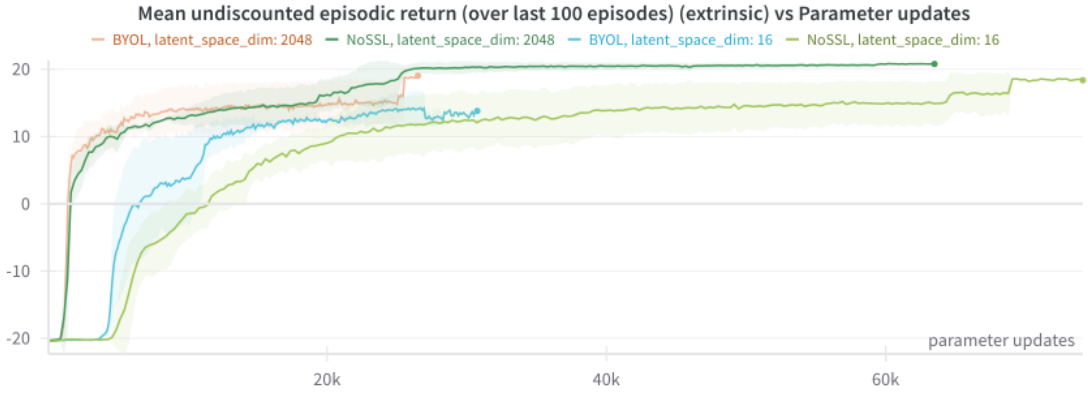


Figure 5.5: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates on the Pong environment for PPO with and without an auxiliary SSL loss.

From our experiments on the Pong environment (Figure 5.5) adding an auxiliary SSL loss seems to improve RL in an on-policy manner when the embedding dimension is small (16). This improvement diminished when the embedding dimension was increased (2048). We believe that when the embedding dimension is smaller, it is harder for the PPO backbone to learn useful representations which are expressive. Under such conditions adding SSL loss helps with learning discriminative image representations and helps the PPO's performance. On the other hand, when the embedding dimension is large even features extracted by a random CNN network can be used to discriminate between different states (this idea was the premise of the original RND paper) and hence, the importance of SSL loss diminishes.

Moreover, introducing auxiliary SSL loss appears to result in better initial results. In other words, during the earlier stages of the training, the agent performs better. This pattern is more apparent in Figure 5.5 for the latent space dimension of 16. The curve for PPO with BYOL rises quicker than the curve for PPO with no SSL. We believe that this improvement stemmed from the fact that SSL objective helps with learning image representations quicker which in turn allows PPO agent's heads to perform better by leveraging these learned representations.

The results on the Pong environment indicate that adding an additional SSL objective to the PPO objective does not deteriorate PPO's performance and promises a quicker learning experience during the early stages of training, though it eventually converges to the performance of Just PPO.

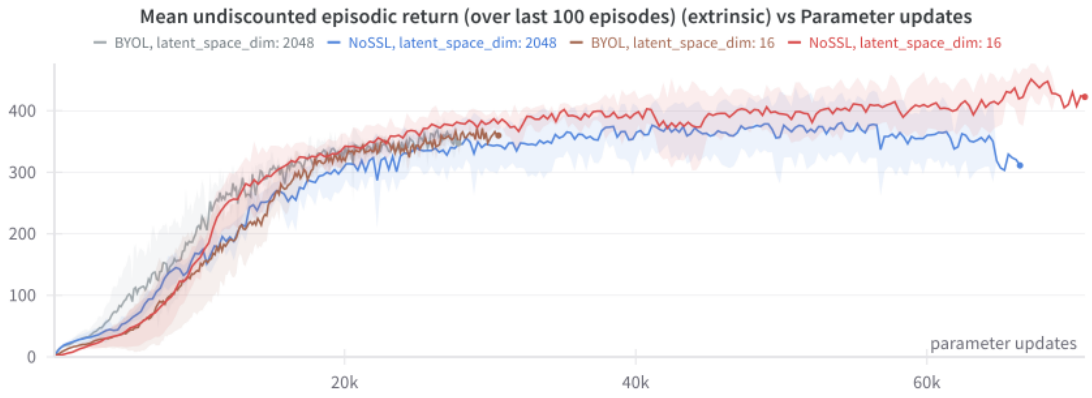


Figure 5.6: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates on the Breakout environment for PPO with and without an auxiliary SSL loss.

In the Breakout environment one gets a reward for destroying a brick and in the Pong environment one gets a reward for making the ball pass the opponent's paddle. This leads the Breakout environment to yield denser extrinsic reward signals during the early phases of the game in comparison to the Pong environment. We believe this diminishes the benefit of using an auxiliary SSL loss. Unlike Pong where rewards were sparser and the addition of auxiliary loss improved the results especially in

lower dimensional spaces, in Breakout we did not see a clear difference when SSL was introduced (Figure 5.6). This indicates that auxiliary SSL loss is beneficial for environments with sparse reward signals, since under such circumstances, learning state representations from reward signals becomes harder. This can be solved by exploiting the inherent structure in the image data by utilizing SSL loss. Alleviating the problem of learning state representations helps PPO to learn better performing PPO heads by exploiting these learned state representations.

Overall, introducing an auxiliary SSL loss to the regular PPO method does not appear to deteriorate performance and even yields minor improvements in the early training performance of agents operating in non-hard exploration environments with relatively sparse rewards. Additionally, the results conform to our initial expectation that the benefit of introducing an auxiliary SSL objective to an on-policy RL method becomes more apparent as the size of the embedding dimension decreases.

Effects of SSL on PPO with Gradient Projection

Now that we understood introducing SSL does not deteriorate the performance of PPO, we ran experiments using PPO with BYOL using the Gradient Projection method. Our intention was to both understand the impact of the Gradient Projection method better, and to further look for an evidence that the clashing gradients problem existed in the PPO with BYOL formulation. To this end we repeated the previous experiment's setting with only introducing the Gradient Projection method to the PPO with BYOL case. Note that this formulation allowed us to use the previous experimental results as baselines.

We believed that if the Gradient Projection improved the performance of PPO with SSL, it would indicate a clashing gradient problem between the PPO and the BYOL objectives. This would also suggest that there might be a clashing gradients problem in the Modified RND's case, which we had missed so far in our findings. On the other hand, lack of any substantial improvement would have lead us to think that the PPO with SSL method was not suffering from the clashing gradients problem and hence the introduction of the Gradient Projection method was ineffective. The

latter case would also be in accord with our findings so far that the modified RND approach was not suffering from the clashing gradients problem.

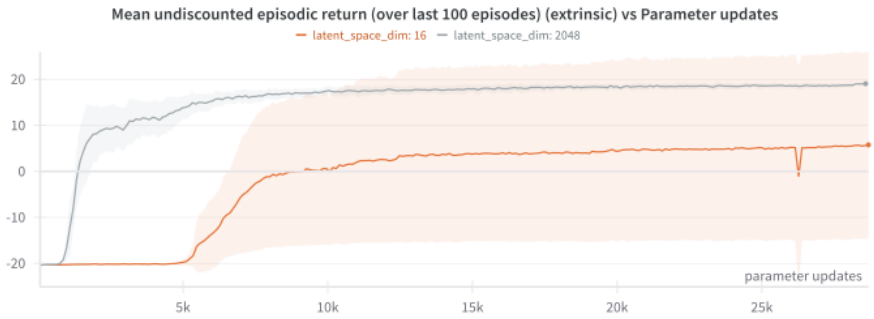


Figure 5.7: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates in the Pong environment with PPO, SSL and Gradient Projection.

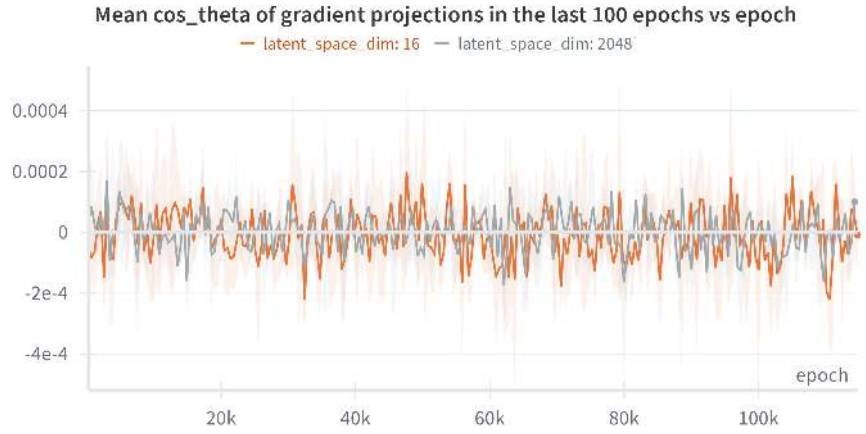


Figure 5.8: Mean of the cosine angle theta between the main objective (RL) and the auxiliary objective (SSL) observed over the last 100 epochs recorded during the gradient projection method experiments in the Pong environment with PPO, SSL and Gradient Projection.

The results were in accordance with our previous finding that higher latent space dimensions resulted in better performance. The graph in Figure 5.9 shows the number of gradient projections in the last 100 epochs, indicating that for approximately half of the epochs, the angle between the gradients exceeded 90°, requiring the

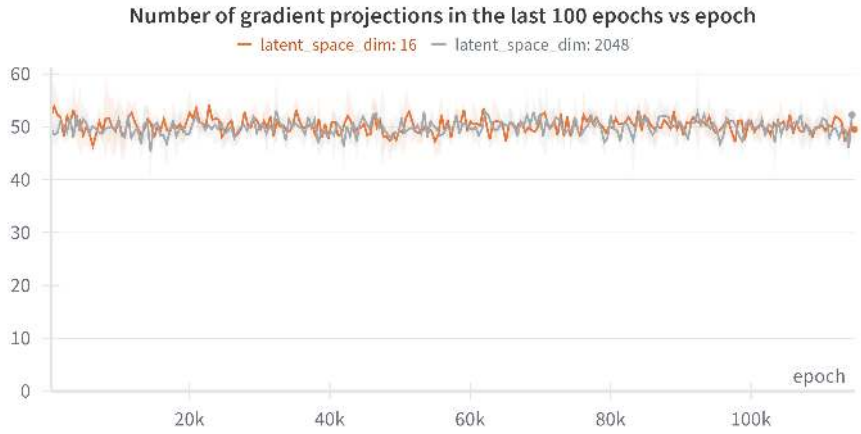


Figure 5.9: Number of gradient projections that took place in the last 100 epochs vs epoch for the Pong environment with PPO, SSL and Gradient Projection.

projection of the gradients. The amount of gradients clashing seems to be unrelated to the latent space dimension. From Figure 5.8, $cos^{-1}(0.0004) = 89.98°$ and $cos^{-1}(-2e^{-4}) = 90.01°$ which indicate that the angle between the gradients are close to 90°. As a result, we think that the gradients are almost orthogonal to each other and they do not hinder the learning. We allude the fact that introducing gradient projection did not yield significant improvements (see Figure 5.10) to the gradients being already almost orthogonal.

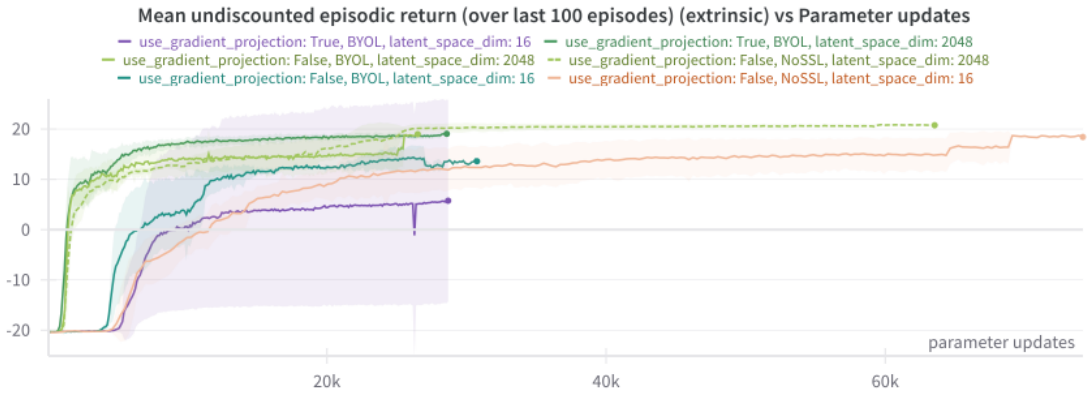


Figure 5.10: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates on the Pong environment for PPO with and without SSL loss, and with and without gradient projection.

A comparison of the results in the Pong environment (Figure 5.10) indicates that using gradient projection with an embedding size of 2048 performs better initially and eventually converges to the same value as without gradient projection. For the embedding size of 2048, gradient projection results in a smaller standard deviation (i.e. more stable). Unfortunately, for the embedding size of 16, the standard deviation is much higher than without gradient projection. These results indicate that when the embedding dimension is large enough (e.g. 2048), gradient projection can provide more stable training with quicker early learning. However, when the embedding dimension is small (e.g. 16), it can lead to worse results and a more fragile training experience.

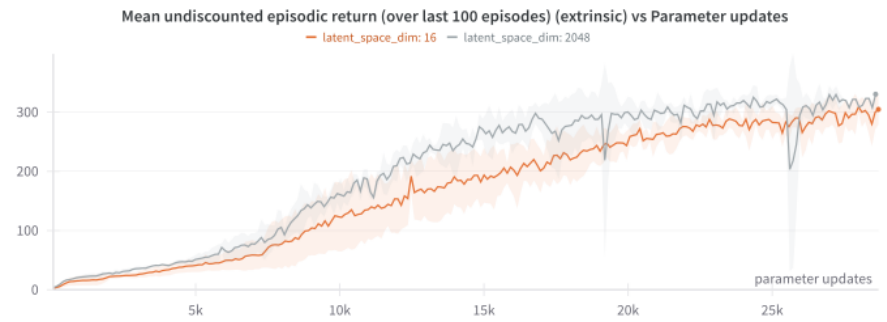


Figure 5.11: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates for the Breakout environment with PPO, SSL and Gradient Projection.

The results of the Breakout experiments are largely consistent with the results of the Pong experiments.

A comparison of results in the Breakout environment (Figure 5.14) indicates that using gradient projection with an embedding size of 2048 does not perform better earlier in training, as was the case in the Pong environment experiments. Moreover, for an embedding size of 16, performance deteriorated drastically when gradient projection was used. Furthermore, the degree of deterioration appears to lessen as the embedding size is increased. These findings from the Breakout environment were not in accord with the previous results from the Pong environment. We think this had to do with the Breakout environment having relatively denser earlier reward

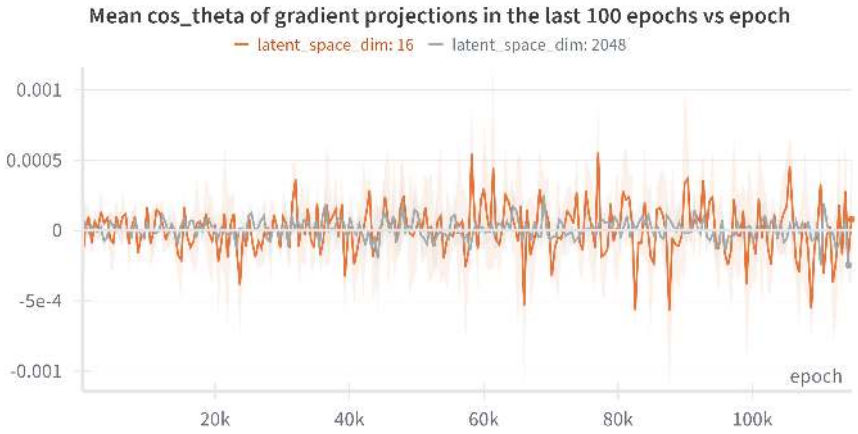


Figure 5.12: Mean of the cosine angle theta between the main objective (RL) and the auxiliary objective (SSL) observed over the last 100 epochs recorded during the gradient projection method experiments in the Breakout environment with PPO, SSL and Gradient Projection.

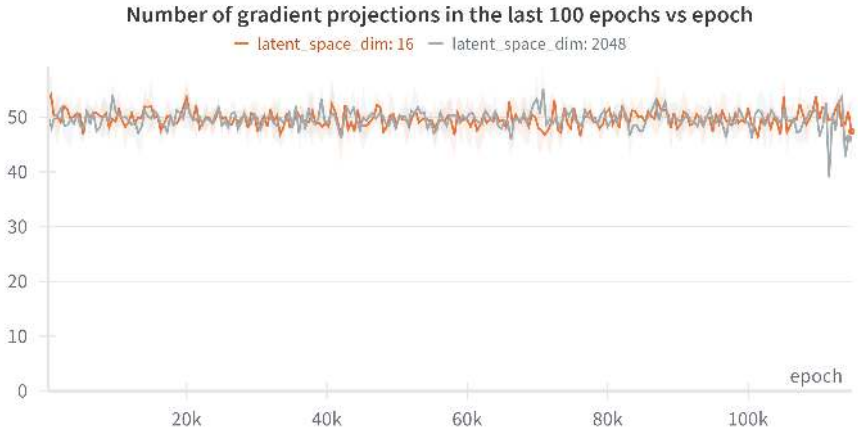


Figure 5.13: Number of gradient projections that took place in the last 100 epochs vs epoch for the Breakout environment with PPO, SSL and Gradient Projection.

signals than the Pong environment.

Having observed that the introduction of an auxiliary SSL objective did not degrade the performance of the PPO agent, and that the introduction of the Gradient Projection method to the Modified RND approach did not mitigate its performance issues, we could not find any evidence indicating that the clashing gradients problem

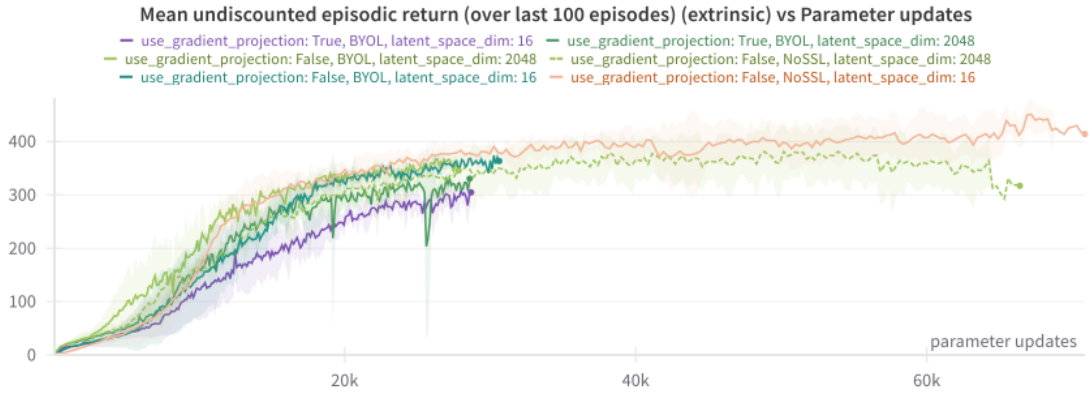


Figure 5.14: Mean undiscounted extrinsic episodic return over the last 100 episodes vs parameter updates on the Breakout environment for PPO with and without SSL loss, and with and without gradient projection.

was the underlying issue. The near orthogonality of the recorded gradients further supports this. As a result, we decided to focus our efforts on the possibility that the exploration module might be the source of the poor performance of our Modified RND approach.

Chapter 6

FURTHER INVESTIGATING THE EXPLORATION MODULE

We suspected that our modifications to the exploration module (RND) were behind the inferior performance of the Modified RND approach in comparison to the performance of the Original RND approach. Initially, we thought that in our modified RND approach, using the features extracted by the shared PPO backbone, which was updated in an online fashion, might have caused the problems. In the Original RND approach, the RND module kept taking image observations as inputs. Contrary to this, in the Modified RND approach the RND module took the features which were extracted by the shared PPO backbone as inputs. We suspected that since the shared PPO backbone was being trained in an online fashion, this might have lead the extracted features to considerably change over time and hence become non-stationary inputs to the RND module's networks. This change could have led the predictor RND network to be unable to predict the output of the target RND network (target feature) even for the previously visited states. This would have resulted in not useful intrinsic rewards being produced, which could hinder the agent's learning by incentivizing it to explore uninformative states. We refer to this idea as the non-stationary inputs to the RND module problem.

6.1 Pretraining Method

To investigate the non-stationary inputs to the RND module, we introduced the pretraining method with a separate pretraining phase in which only the state representations are being learned via the SSL objective. During the pretraining phase, the training data is gathered online using a random policy. Then in the second phase, the learned parameters are used as frozen initializations for our agent's shared PPO

backbone. The rest of the architecture is trained using objectives other than the SSL objective.

With the use of pretraining method, we could start to utilize the RND module once we have good state representation extractors that are not being modified and produce stationary outputs. As a result, similar to the Original RND approach, the distribution of the inputs to the RND module in the Modified RND approach should not change. We believed that by comparing the performance of the pretrained Modified RND runs to the performance of the previous Modified RND runs (Chapter 4.2) we could evaluate the correctness of our suspicions about the non-stationary inputs to the RND module. Observation of an improved performance with the pretraining could hint at the validity of the non-stationary inputs to the RND module issue. Conversely, not observing an improvement could indicate that our suspicions were incorrect.

Furthermore, typically in the applications of SSL methods, a self-supervised pretraining phase takes place at the beginning. Contrary to this, in our approaches we used SSL objectives as an auxiliary loss. This can introduce some issues, such as the tuning of the SSL loss coefficient. Since SSL objectives, such as BYOL's loss, operate on l_2 loss, their range can change drastically between different applications. Also, SSL applications work on a vast amount of data to learn useful representations. This requirement might not be met in our RL application, or it might increase the time required for the training. [Kwon et al., 2024] suggested that the phase of learning historical representations needs to be distinct for on-policy methods. Otherwise, a considerable number of trajectory samples may go to waste if we fail to effectively address the simpler supervised learning task using the provided representations.

6.2 Experimental Analsysis

Motivated by these ideas, we performed a set of experiments using the pretraining method to test the performance of the modified RND approach with the pretraining

method. We wanted to compare our results with our first experiment (Chapter 4.2), so we used the Montezuma's Revenge environment with the hyperparameter configurations from that experiment. The results showed that loading the pretrained model and then freezing the pretrained backbone drastically deteriorated the performance. The agent failed to explore any room other than the one it was spawned in.

In light of these results, we thought that perhaps the gradients from the RL objective are essential for the shared PPO backbone to perform well. To test this hypothesis, we conducted experiments using the same pretrained backbone as the initial weights for the shared PPO backbone during the second phase of the training. We tried three different approaches: completely freezing the backbone ("Completely frozen shared backbone"), training the shared backbone with just the RL objective ("RL trained shared backbone"), and training the shared backbone with both the RL and SSL objectives ("RL and SSL trained shared backbone"). The results in Figure 6.1 indicate that indeed the gradients from the RL objective are vital during the second phase of the pretraining method. Also, we observe that "RL and SSL trained shared backbone" exhibits better early training performance which then converges to the performance of the "RL trained shared backbone". This falls in line with our previous finding about introducing an auxiliary SSL objective improving the early training performance of the agent.

6.3 EMA Updated Target PPO Backbone

Our investigation of the pretraining method did not provide sufficient evidence to determine that the non-stationary inputs to the RND Module were not the underlying issue. Also having seen that the gradients from the RL objective flowing back to the shared PPO backbone were essential for the performance of the Modified RND approach, we wanted to try another approach that could tackle the non-stationary inputs to the RND Module problem while still allowing the RL objective to have an impact on the shared PPO backbone.

To this end, we explored the idea of alternative approaches, such as applying Expo-

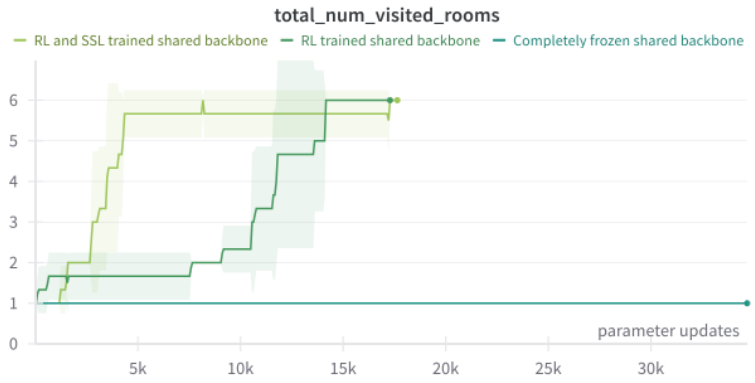


Figure 6.1: Number of visited rooms vs parameter updates in Montezuma's Revenge. All of the models use the same pretrained backbone but they are trained via different objectives during the second phase of the pretraining method.

nential Moving Average (EMA) to the shared PPO backbone. Motivated by works like [Fujimoto et al., 2023], which addressed potential instability caused by inconsistent inputs through target networks, we considered that having a slower updated target network for the shared PPO backbone might mitigate the issue of the non-stationary inputs to the RND Module while allowing the shared PPO backbone to continue being trained by the RL and the SSL objectives. These ideas are further reviewed in Chapter A.2.

Inspired by these, in order to handle the non-stationary inputs to the RND Module problem, we tried having two separate copies of the shared PPO backbone. One copy (online shared PPO backbone) is trained normally using both the RL and the SSL objectives. The other copy (target backbone) is updated every n steps (we tried updating after every training epoch) using EMA Updates from its counterpart (online backbone). Given that the online shared PPO backbone has parameters θ, the target shared PPO backbone has parameters ϕ, and the smoothing factor τ, the EMA update for the target backbone's parameters is performed as $\phi \leftarrow \tau\theta + (1-\tau)\phi$. This target backbone is then used to provide more stationary inputs to the RND Module. We believed that this approach should produce more stationary inputs for the RND Module as the parameters of the target backbone change more slowly compared to the initially proposed Modified RND approach. Additionally, the target

backbone benefits from the representations learned by the RL and the SSL objectives through the EMA updates. Furthermore, unlike our pretraining method, the PPO heads would not suffer from the shared PPO backbone being frozen, as the online backbone will be trained normally with both the PPO and SSL Modules.

6.4 Experimental Analysis

In our experiments, we tested this idea with $\tau = [0.01, 0.05]$ and compared the results against those of the initially proposed modified RND approach from the original Modified RND experiments. The results are shown in Figure 6.2. We observed that introducing EMA did not improve the performance of the Modified RND approach. Even with EMA updates, the Modified RND approach was unable to explore more rooms than in the original Modified RND experiments. Additionally, the initially proposed Modified RND approach can be seen as a special case of our EMA updated target PPO backbone modification with $\tau = 1.0$. Our empirical results with $\tau = [0.01, 0.05, 1.00]$ suggest that the EMA updated target PPO backbone does not offer promising improvements. Although further tuning of the τ parameter might yield better results, we are skeptical about this possibility.

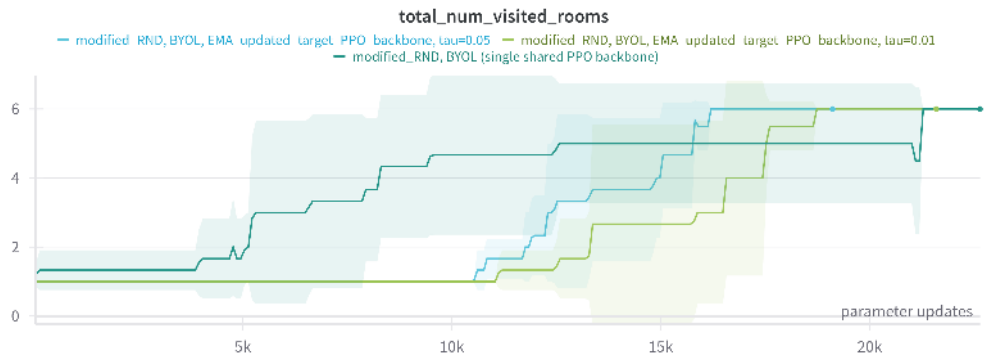


Figure 6.2: Number of visited rooms vs parameter updates in Montezuma's Revenge for the EMA updated target shared PPO backbone experiments. Runs labelled "EMA_updated_target_PPO_backbone" correspond to the new proposed modification, while "modified_RND, BYOL (single shared PPO backbone)" refers to the initially proposed modified RND approach from the earlier original Modified RND experiments.

6.5 Investigation of Training Statistics of the RND Module

Having failed to find any sign of the non-stationary inputs to the RND module issue, and yet still suspecting that our modifications to the RND Module might be responsible for the poor performance of the Modified RND approach, we next investigated various statistics related to the RND Module in both the Original RND and the Modified RND methods from the original Modified RND experiments (Chapter 4.2) to gain further insight.

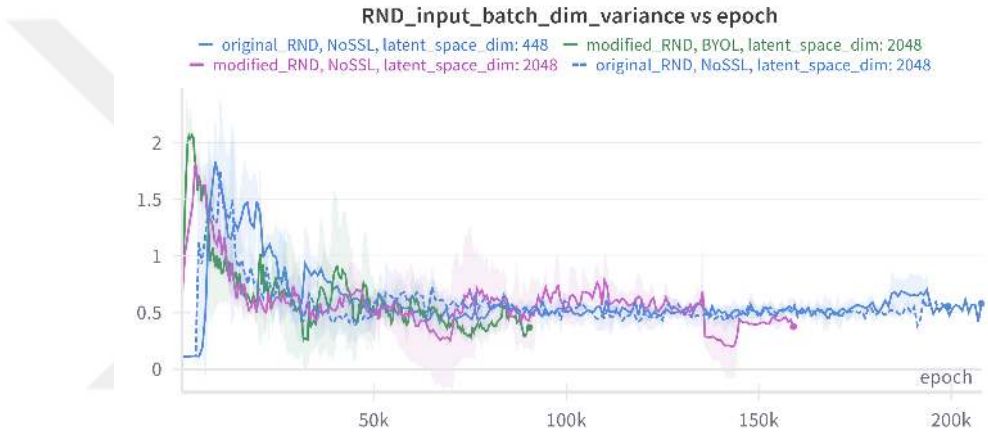


Figure 6.3: Batch dimension variance of the inputs to the RND module vs epoch in the original Modified RND experiments.

Note that the graphs with the *RND_input* prefix are related to the training statistics of the inputs to the RND Module. In other words, the tensors that the RND module's target network and the predictor network take in as inputs. Also note that the inputs to the RND Module have statistics which are close to zero-mean and unit-variance because these logged statistics are calculated over features which were normalized by their corresponding empirical running statistics. The *RND_input* graphs demonstrate that the Modified RND's inputs to the RND module closely resemble the Original RND's inputs. This suggests that the poor performance of the Modified RND approach is not due to differences in the statistical characteristics of the inputs to the RND module, but rather to some other factor. One hypothesis is that the poor performance might be due to the inputs to the RND module lacking

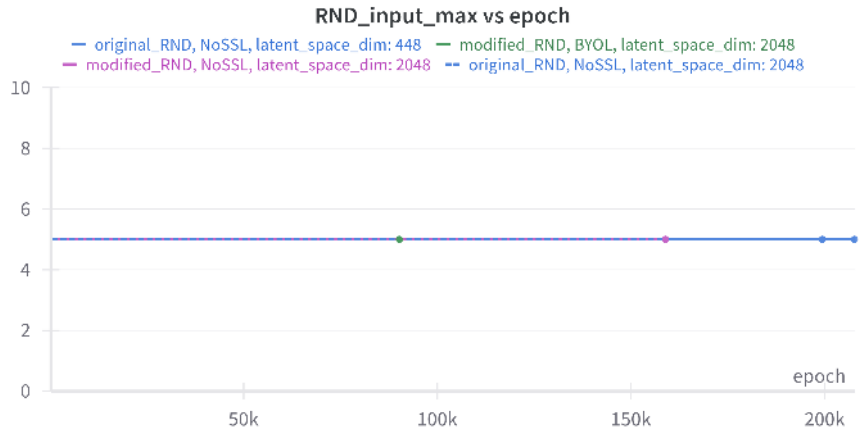


Figure 6.4: Maximum of the inputs to the RND module vs epoch in the original Modified RND experiments.

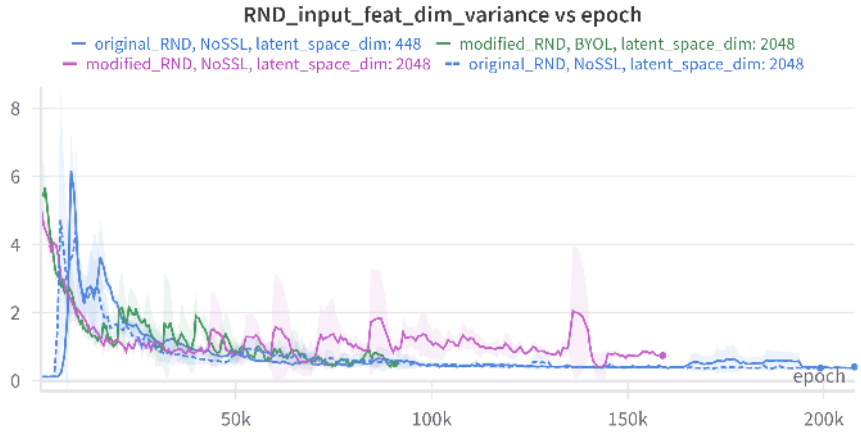


Figure 6.5: Feature dimension variance of the inputs to the RND module vs epoch in the original Modified RND experiments.

in the quality of the information they convey, which could hinder the RND module's ability to differentiate between different states. Another possibility is that the information content of the inputs generated by the Modified RND might be changing too drastically, making it difficult for the predictor RND network to adapt. This could result in the predictor RND network failing to predict the target features, even for states similar to those the agent has encountered before. However, if we assume that the input features are similar in quality to their statistical characteristics, this

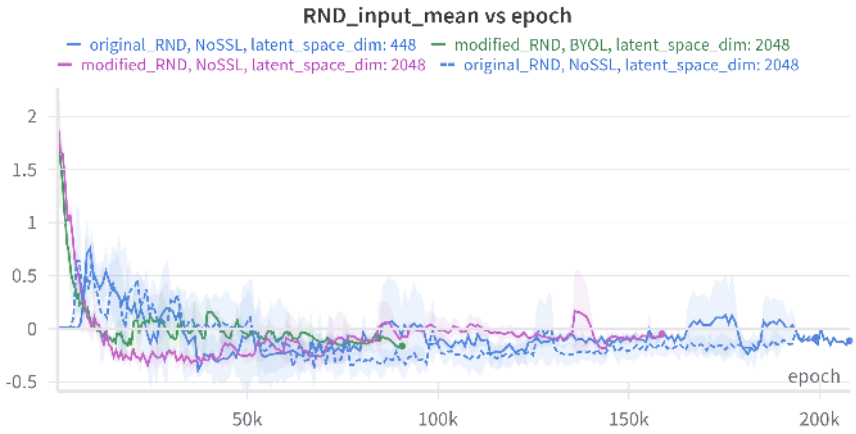


Figure 6.6: Mean of the inputs to the RND module vs epoch in the original Modified RND experiments.

suggests that differences in the formulation of the RND networks might be the underlying cause of the poor performance. Furthermore, the similarity in the input statistics between Modified RND with SSL and Modified RND without SSL might indicate that the auxiliary SSL objective does not result in statistical instabilities, which could hint at the viability of the SSL objective as an auxiliary objective.

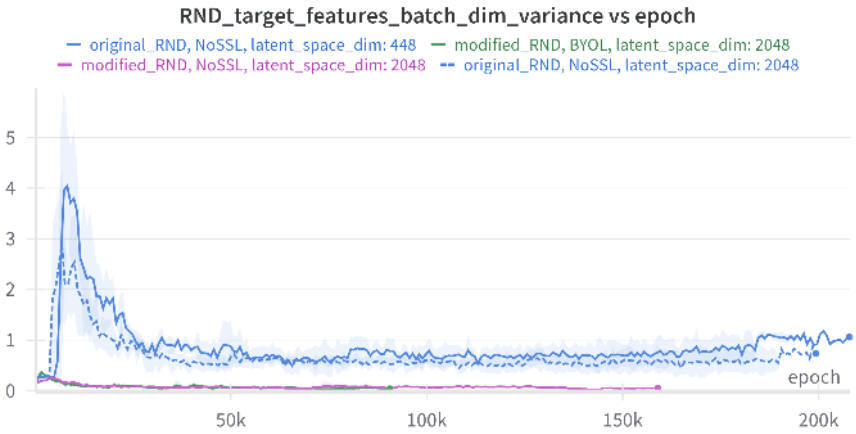


Figure 6.7: Batch dimension variance of the RND module targets vs epoch in the original Modified RND experiments.

The graphs with the *RND_target* prefix are related to the training statistics

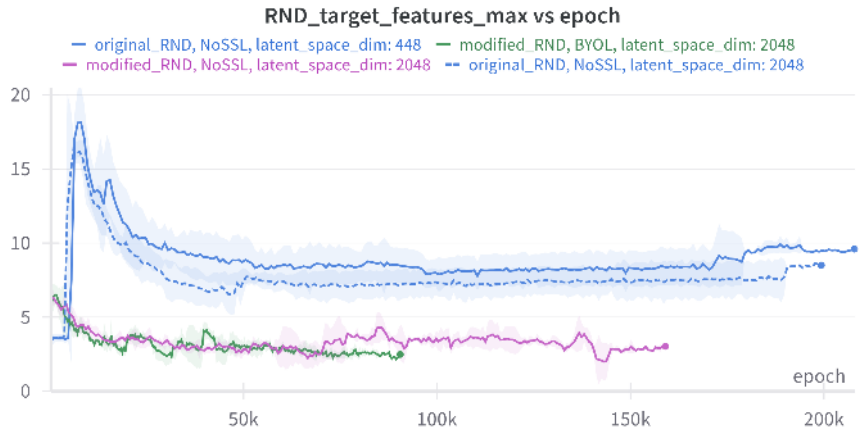


Figure 6.8: Maximum of the RND module targets vs epoch in the original Modified RND experiments.

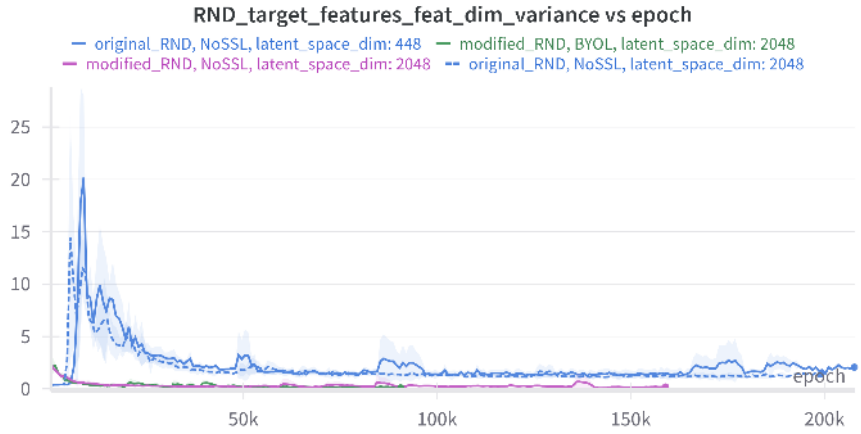


Figure 6.9: Feature dimension variance of the RND module targets vs epoch in the original Modified RND experiments.

of the vectors outputted by the RND module's target network. In other words, the vectors that the online predictor network of the RND Module has to learn to predict (match). The Modified RND approaches appear to have target feature statistics which are smaller than the Original RND approaches. We suspected that this change in the target features might be behind the inferior performance of the Modified RND approach. To investigate this idea, we performed experiments where we scaled the weights of the target RND network during its initialization, hoping to

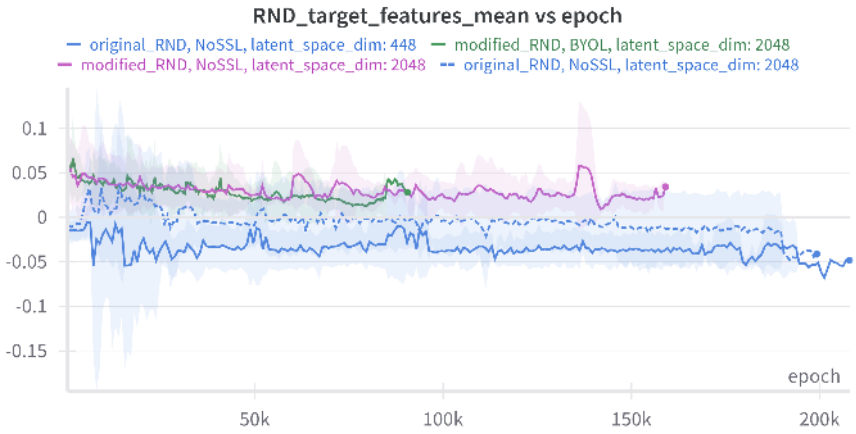


Figure 6.10: Mean of the RND module targets vs epoch in the original Modified RND experiments.

also scale the statistics of the produced target features and observe how this affected performance. Although we saw an increase in the target feature statistics, there was no corresponding change in the performance of the Modified RND approach. As a result, we concluded that this difference in target feature statistics was not the underlying problem with the Modified RND approach.

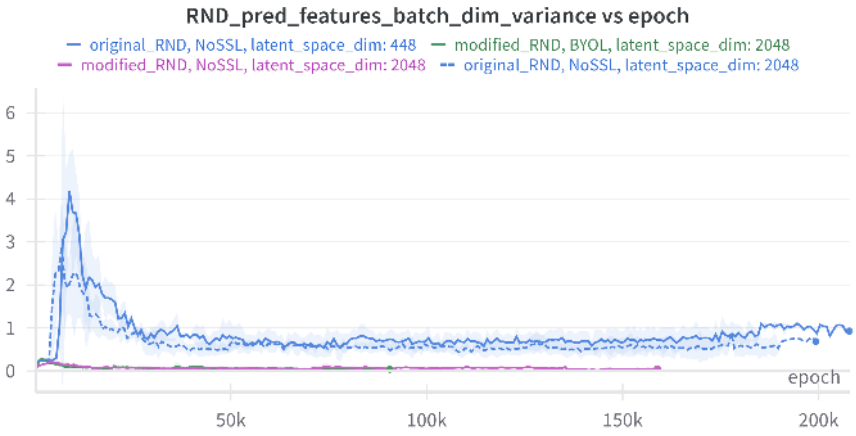


Figure 6.11: Batch dimension variance of the RND module predictions vs epoch in the original Modified RND experiments.

The graphs with the *RND_pred* prefix are related to the training statistics of

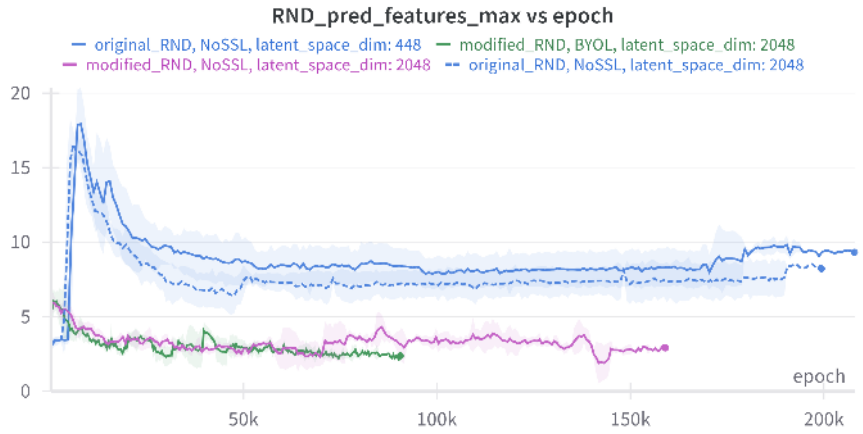


Figure 6.12: Maximum of the RND module predictions vs epoch in the original Modified RND experiments.

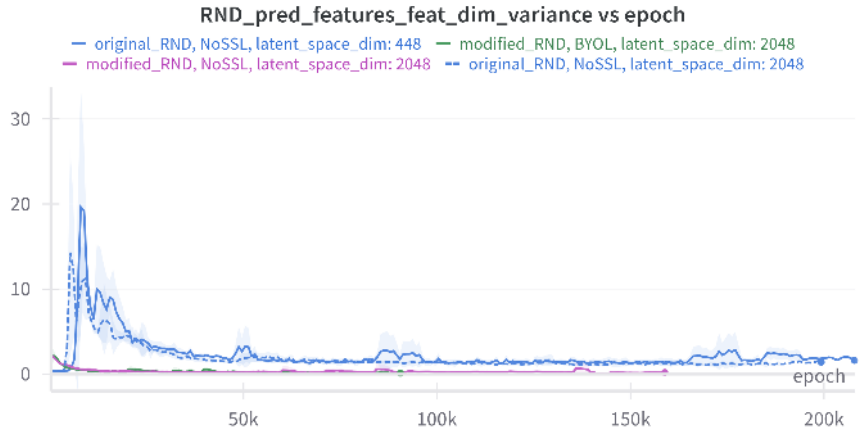


Figure 6.13: Feature dimension variance of the RND module predictions vs epoch in the original Modified RND experiments.

the vectors outputted by the RND module's predictor network. In other words, the vectors that the online predictor network predicts which should ideally learn to match the vectors outputted by the RND's target network. The statistics of the *RND_pred* appear to match the statistics of the target features produced by the target RND network more closely in the Modified RND approach than in the Original RND approach. Figure 6.17 and Figure 6.15 reflect this by showing smaller values for the Modified RND approach and higher values for the Original RND approach.

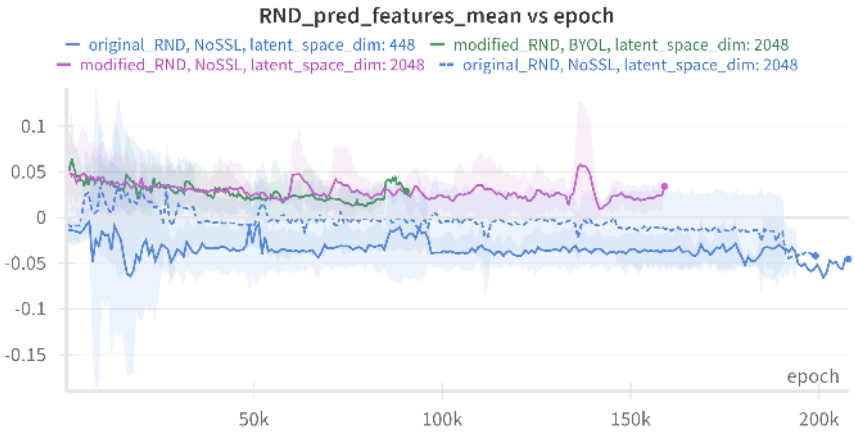


Figure 6.14: Mean of the RND module predictions vs epoch in the original Modified RND experiments.

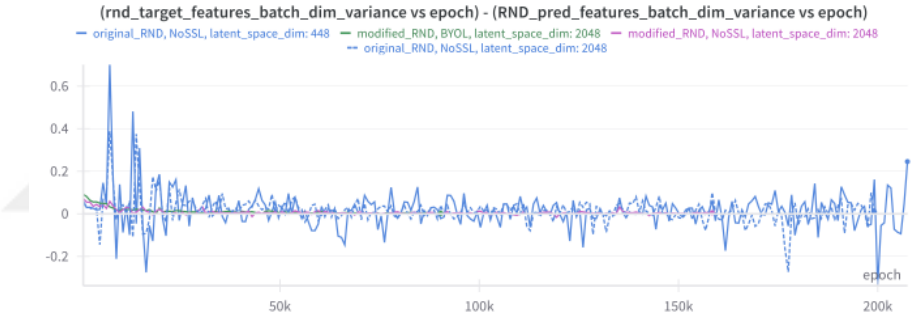


Figure 6.15: Difference between the batch dimension variance of the RND module targets and predictions vs epoch in the original Modified RND experiments.

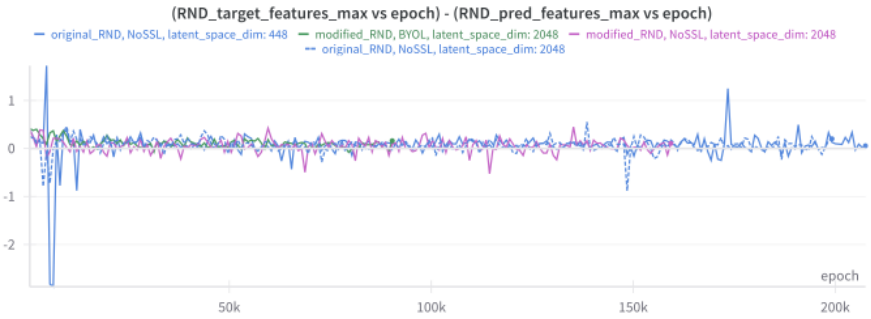


Figure 6.16: Difference between the maximum of the RND module targets and predictions vs epoch in the original Modified RND experiments.

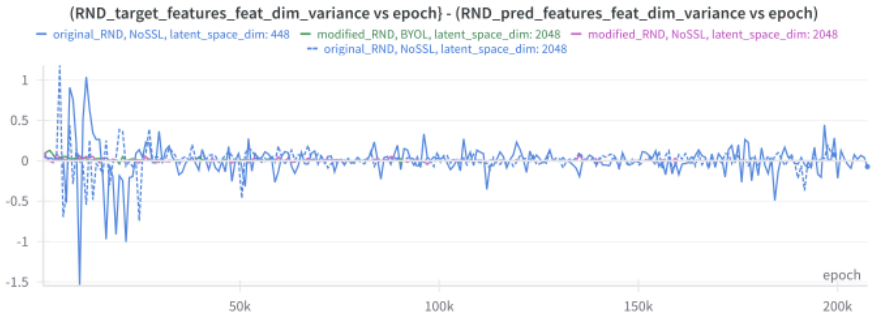


Figure 6.17: Difference between the feature dimension variance of the RND module targets and predictions vs epoch in the original Modified RND experiments.

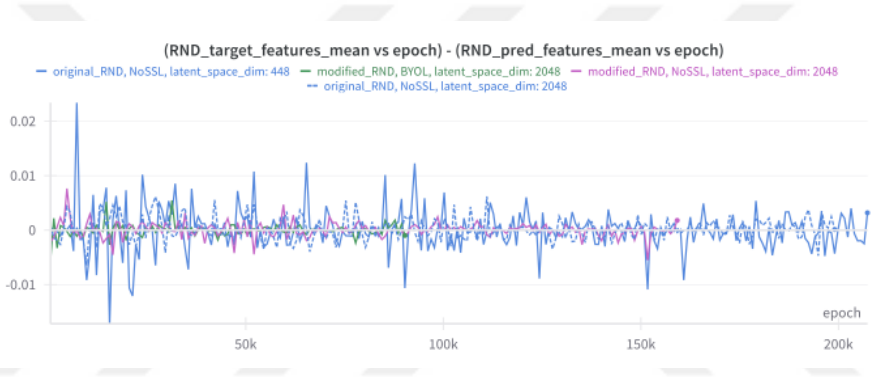


Figure 6.18: Difference between the mean of the RND module targets and predictions vs epoch in the original Modified RND experiments.

This indicates that the RND Module's predictor network was capable of predicting the features outputted by the RND's target network. Consequently, we do not believe that there is an issue with the formulation or hyperparameters of the predictor network in our approach, and the poor performance of the Modified RND does not stem from this.

We suspected that the problem with the Modified RND might have stemmed from the produced target features not reflecting the characteristics of the input states. This could have led to the predictor RND network's predictions being less useful for discerning novel states, resulting in meaningless intrinsic rewards. Consequently this lack of useful intrinsic rewards might have hindered the agent's exploration performance and deteriorated the overall performance of the Modified RND approach. We thought that the issues with the produced target features might have been caused

by the modifications made to the neural network architecture of the RND networks in the Modified RND approach.

6.6 Architectural Changes to the Shared PPO Backbone and the RND Networks

The Original RND was using convolutional layers in the RND Module but we had changed these to linear layers as the Modified RND's shared PPO backbone was outputting a linear vector. We thought that switching to linear layers was okay as we were passing features extracted by the shared PPO backbone as inputs to the RND Module. Hence we thought that it did not matter whether the frozen linear layers were able to extract useful image representations like the randomly initialized frozen convolutional layers did. But due to the poor performance of the Modified RND approach we started to think that maybe preserving local information in the images using convolutional layers was essential for the target RND network to produce useful target features.

To investigate and potentially solve this issue, we proposed modifications to our initially proposed Modified RND approach as follows. The shared PPO backbone is split into two parts: the initial convolutional layers become the new shared PPO backbone, while the remaining linear layers that followed these convolutional layers are retained as a continuation of the shared PPO backbone, applied only before the PPO heads. This configuration means that the inputs to the RND Module are produced solely by the convolutional layers, acting as downscaled images that can be further processed by convolutional layers within the RND Module. This change in the shape of the inputs to the RND Module allowed us to switch from linear layers to convolutional layers in the RND networks of our Modified RND approach. The overall architecture of the PPO Module remains unchanged. The modifications to the shared PPO backbone architecture can be understood by comparing Figure 4.2 and Figure 6.19. The overall architecture of our newest modification (Modified RND with a shared PPO backbone featuring a convolutional last layer) is illustrated

in Figure 6.20.

Figure 6.21 shows RND network architectures from different approaches. Our initially proposed Modified RND uses linear layers. Both the Original RND and our newest modification uses convolutional layers, but the RND Module in our newest modification has a relatively smaller input size compared to the Original RND approach, which leads to the use of larger paddings and smaller kernel sizes.

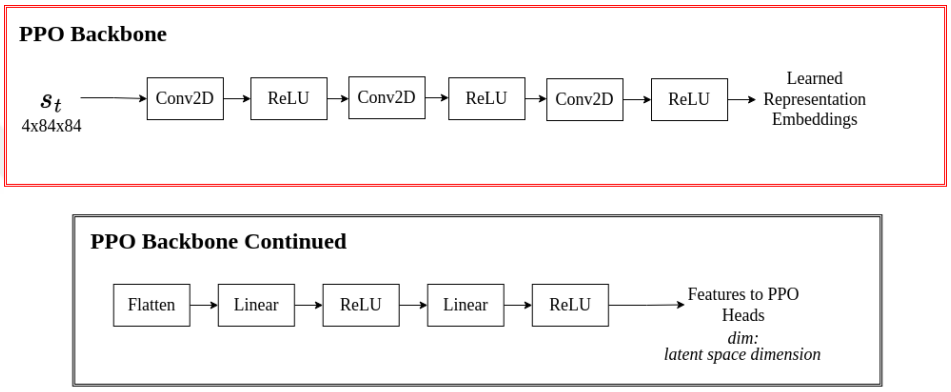


Figure 6.19: Architecture of the shared PPO backbone with a convolutional last layer.

6.7 Experimental Analysis

We devised a set of experiments to test the effectiveness of our proposed Modified RND with a convolutional last layer. The results are shown in Figure 6.22. Note that on the graphs, *shared_ppo_backbone_last_layer=Conv* corresponds to our newest modification idea, where the Modified RND has a convolutional last layer, while *shared_ppo_backbone_last_layer=Linear* corresponds to the Modified RND runs (i.e. initially proposed Modified RND with a linear last layer) from the original Modified RND experiments (Chapter 4.2). Unfortunately, the Modified RND with a convolutional last layer performed considerably worse than the initially proposed Modified RND with a linear last layer (see Figure 6.22).

By comparing the RND module statistics of the Modified RND with the convolutional last layer to those from the original Modified RND experiments (Chapter 4.2),

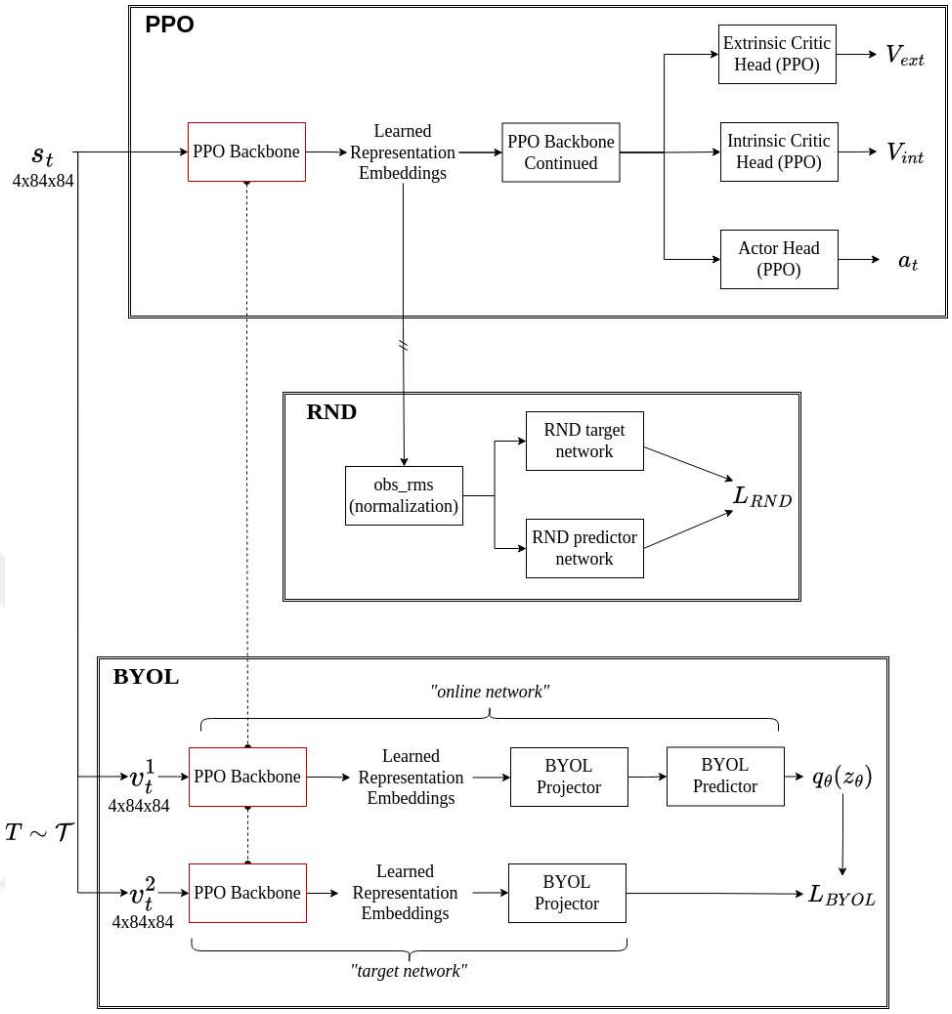


Figure 6.20: Architecture of the Modified RND approach with the shared PPO backbone and the convolutional last layer applied.

we aimed to identify what went wrong and caused the performance to deteriorated further (Figures 6.23-6.30). The inputs and targets of the Modified RND with the convolutional last layer's RND module (i.e., output of the shared PPO backbone's final convolutional layer) did not appear to be non-stationary. Also, the statistics of the Modified RND with the convolutional last layer were similar to those of the Original RND. We suspected that the relatively larger paddings and smaller kernel sizes used in our newest modification's RND networks might have led to a loss of information, which could explain the poor performance.

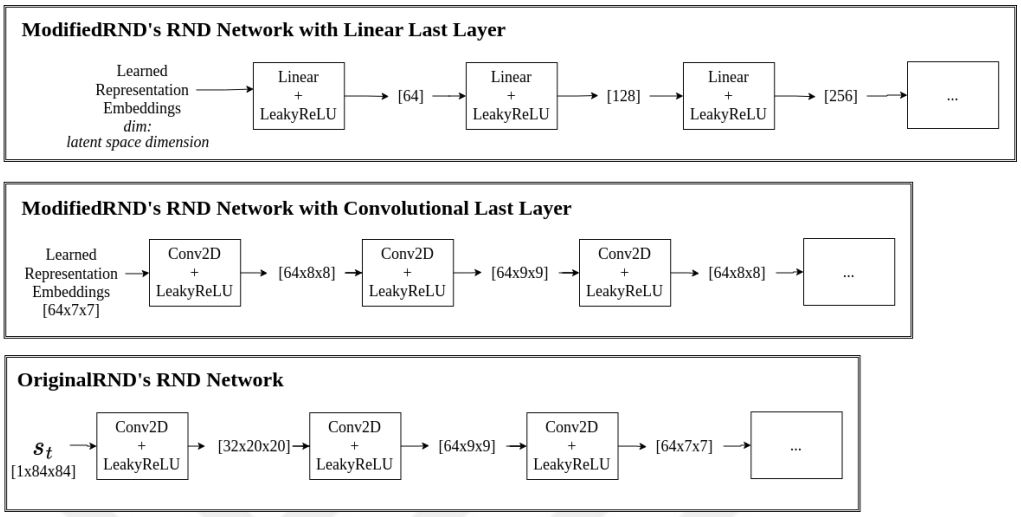


Figure 6.21: Overview of the different RND network architectures from different approaches.

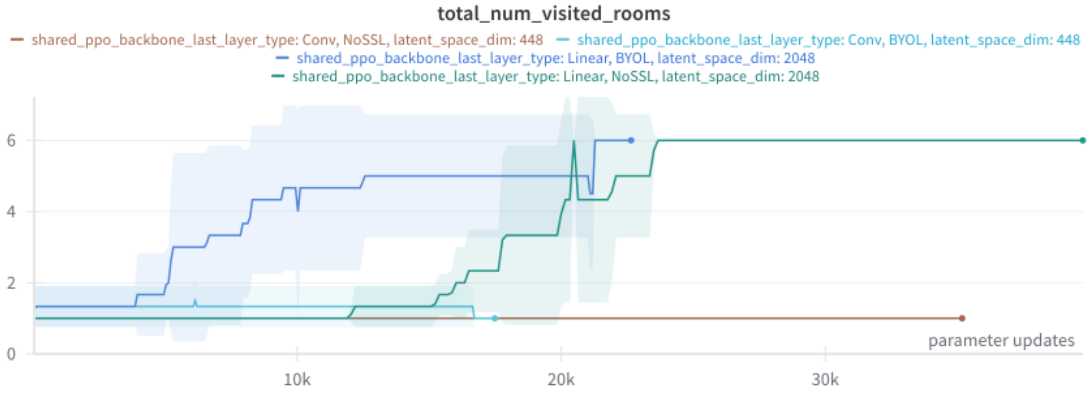


Figure 6.22: Total number of visited rooms vs parameter updates for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

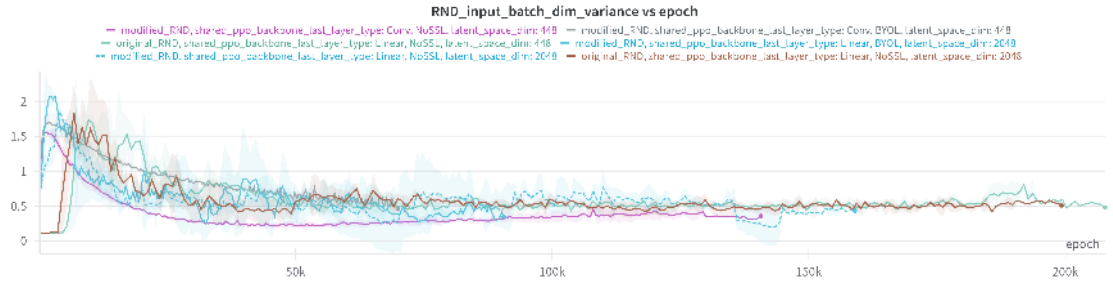


Figure 6.23: Batch dimension variance of the inputs to the RND module vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

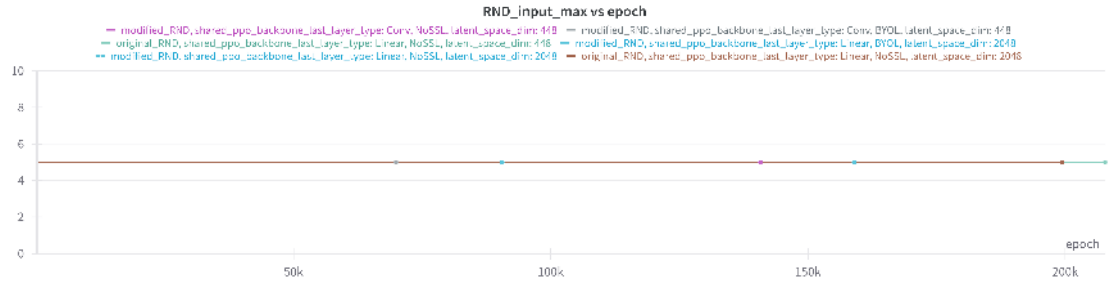


Figure 6.24: Maximum of the inputs to the RND module vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

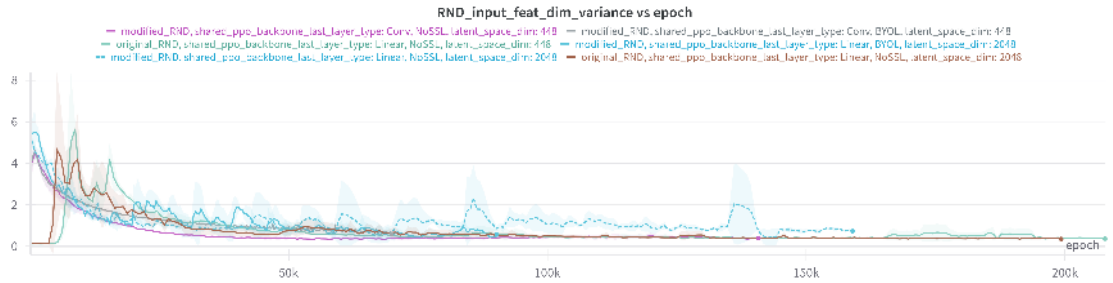


Figure 6.25: Feature dimension variance of the inputs to the RND module vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

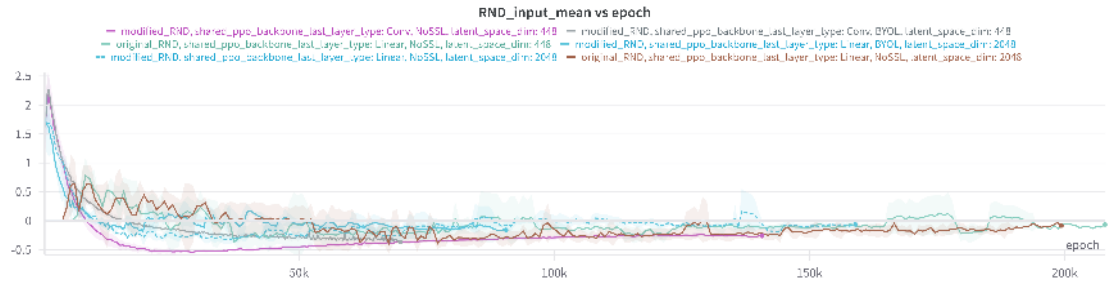


Figure 6.26: Mean of the inputs to the RND module vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

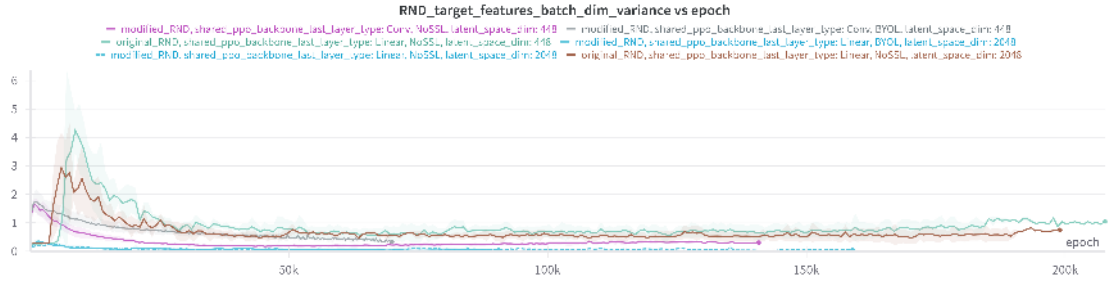


Figure 6.27: Batch dimension variance of the RND module targets vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

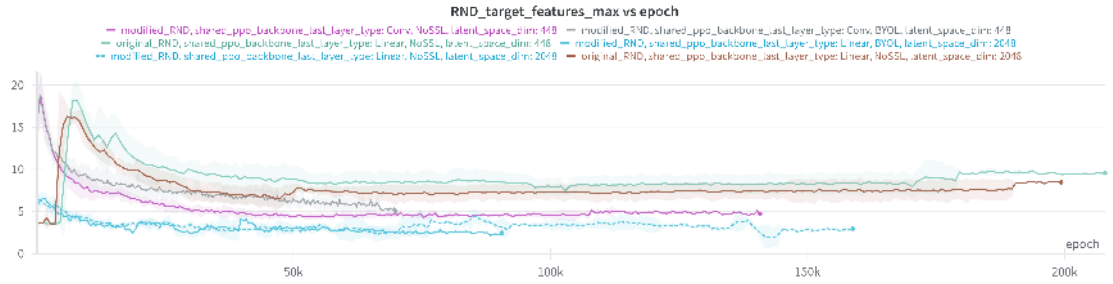


Figure 6.28: Maximum of the RND module targets vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

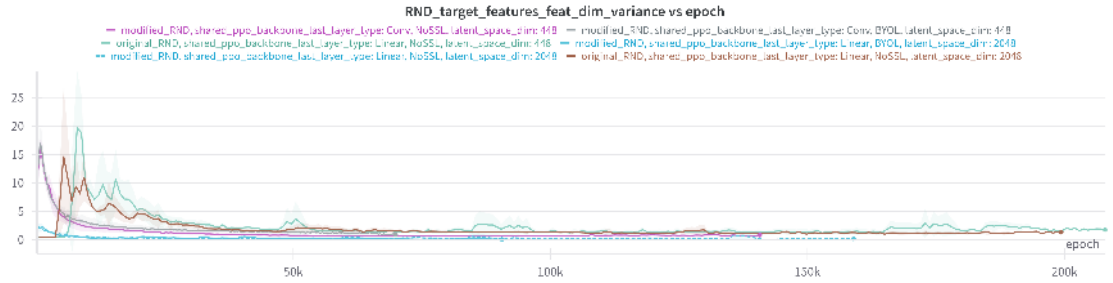


Figure 6.29: Feature dimension variance of the RND module targets vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

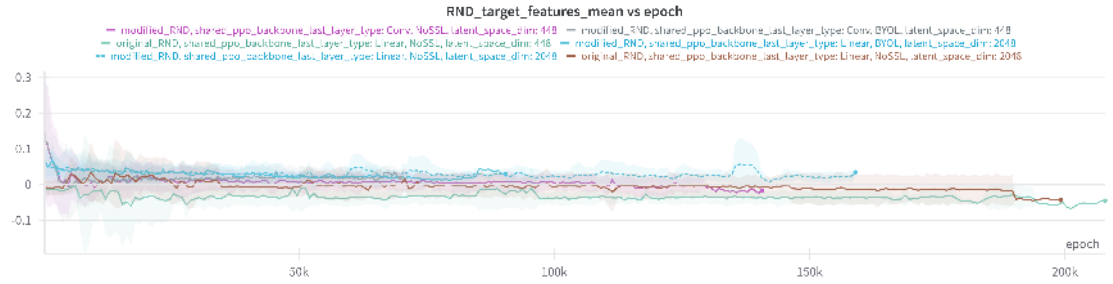


Figure 6.30: Mean of the RND module targets vs epoch for the Montezuma's Revenge environment for Modified RND with convolutional last layer and Modified RND with linear last layer from the original Modified RND experiments.

Chapter 7

VIT WITH EXPLORATIVE ATTENTION

So far, we have tried the Modified RND approach and its various modifications with convolutional backbones but failed to achieve justifiable results. Since these efforts did not yield success, we decided to investigate model architectures directly. Inspired by the success of transformers in vision and sequential decision making tasks, we aimed to harness their strengths for exploration problems in reinforcement learning. Our goal was to improve learned representations by learning two different representations specific to exploration and exploitation. We believed that successful implementation of this idea could lead to learning better representations through architectural changes alone, without the need for SSL. This approach can be viewed as representation learning through architectural changes for prediction-error based exploration methods in reinforcement learning. Among available transformer models for vision, we chose to build upon the Vision Transformer (ViT) architecture ([Dosovitskiy et al., 2021]) due to its widespread use in research and the availability of its code implementation.

We wanted to modify the ViT architecture so that it could learn where in the input images to pay attention for exploration and where to focus for exploitation. For instance, the explorative attention (exploration related attention) should learn to pay attention to exploration related features in a given input image, such as unexplored doors and ladders in the Montezuma's Revenge environment, as these can lead to the exploration of new rooms, indicative of the exploratory behaviour. On the other hand, the exploitative attention (exploitation related attention) should learn to pay attention to exploitation related features in a given input image, such as skulls and lava on the floor in the Montezuma's Revenge environment, as these can harm the agent and impact the extrinsic rewards (refer to Figure 7.2).

We wanted our proposed architecture to use the extracted explorative features for predicting intrinsic rewards using PPO's intrinsic value head, and to use the exploitative features for predicting extrinsic rewards using PPO's extrinsic value head. Both sets of extracted features (extracted explorative features and the extracted exploitative features) are aggregated (we used element-wise summation) to predict the next action using PPO's policy head. We aggregate the features because we want the policy to consider both exploration related features and exploitation related features when making a decision.

The explorative features are extracted by appending a learnable *exploration token* in front of the sequence of image patches and using the *exploration token*'s query vector to obtain the exploration features. Similarly, the exploitative features are extracted by appending a learnable *exploitation token* in front of the sequence of image patches and using the *exploitation token*'s query vector to obtain the exploitation features. This proposed ViT architecture, which we called *Vit with Explorative Attention*, serves as a backbone that replaces the convolutional PPO backbone in the RND approach. We tested this idea with RND, but it could be adapted for other methods that provide intrinsic rewards.

We performed experiments to investigate our proposed ViT with explorative attention idea. From the results in Figure 7.3, Explorative Attention appears to perform worse than the regular ViT backbone. Due to this, we believe that the Explorative Attention idea is not as helpful as we initially thought in Chapter 7. The regular Vit backbone seems to yield better performance earlier, but over a longer period, the Original RND with CNN backbone appears to be more preferable due to similar exploration performance with less computational requirement. We attribute this outcome to [Dosovitskiy et al., 2021]'s conclusion that transformers lack certain inherent biases found in CNNs, such as translation equivariance and locality, which can result in poor generalization when trained on limited data. We believe that our experiments with ViT with Explorative Attention fall into this category of training

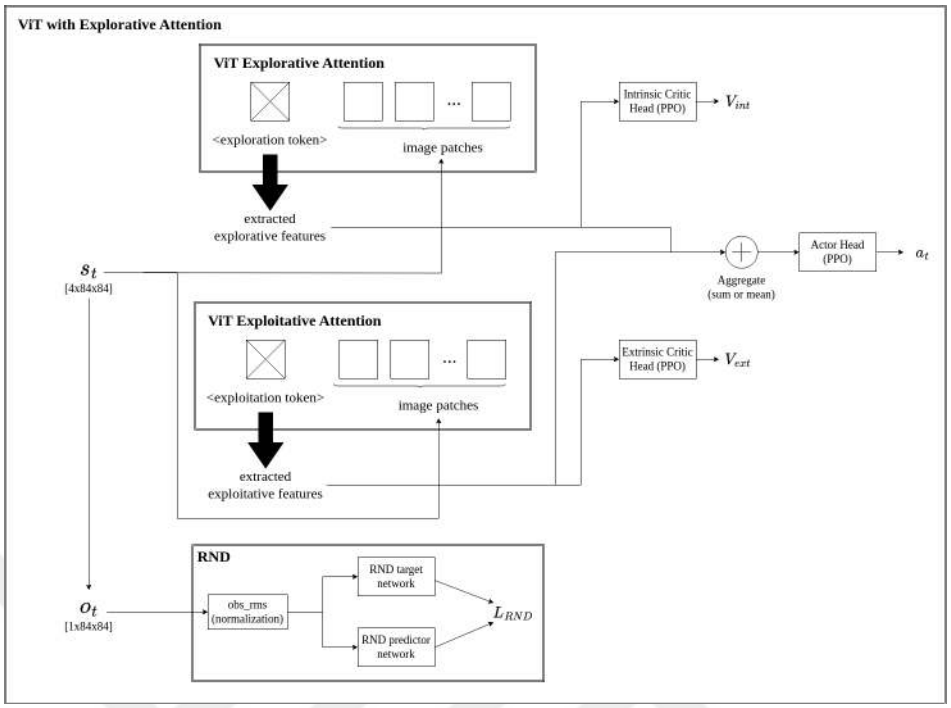


Figure 7.1: Overview of the ViT with Explorative Attention approach.

Figure 7.2: An intuitive visualization of how the Explorative Attention Module should ideally work is shown. The image on the left demonstrates that exploitative attention should pay attention to the skulls. The image in the middle shows that explorative attention should pay attention to the ladder. The image on the right shows the outcome of aggregating the explorative and exploitative attention features.

on limited data.

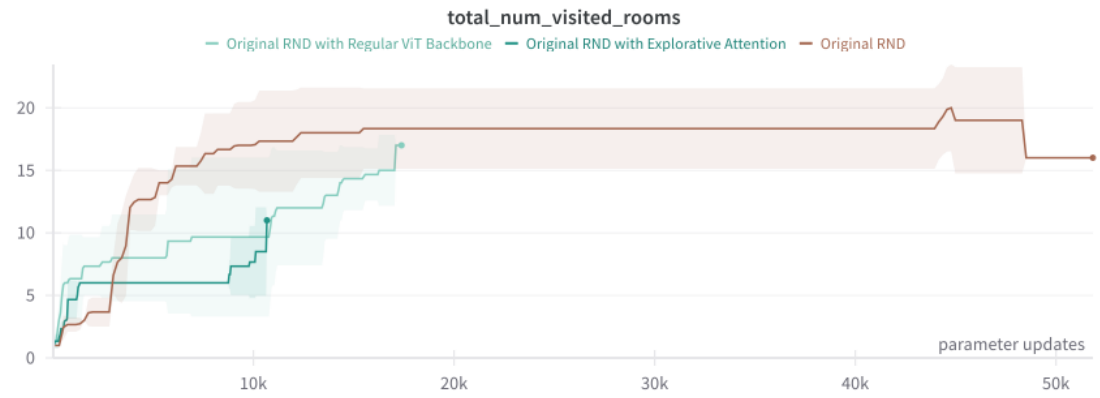


Figure 7.3: Total number of visited rooms vs parameter updates for the Montezuma's Revenge environment in the ViT experiments.

Chapter 8

POTENTIAL ISSUES

In this chapter, we touch upon some of the downsides which we think can be of concern depending on the application of our framework. We believe that some of our failed attempts during the experiments can be attributed to factors such as these.

Combining different losses as in equation 4.1 complicates the hyperparameter tuning process of the Modified RND approach by introducing more coefficients. This could hurt our framework's generalization by making it more sensitive to hyperparamaters and harder to tune.

Introducing representation learning for exploration in RL has a trade-off between learning better representations and bringing additional time and space complexities. This is due to needing additional models for SSL (SSL module of the Modified RND approach) and performing additional forward and backward passes using those models. This increased complexity could cause problems such as longer training times, increased overhead in distributed communications, and even all of the models not being able to fit into a single GPU.

Our Modified RND approach was built on top of the Original RND approach and as a result it inherited some of the inherent problems with the Original RND formulation. One of these issues is that the Original RND might detect things which are impertinent to the decision making process of the agent as novelty. [Pathak et al., 2017] explained the properties that a good exploration objective should possess. In light of those identified properties, we believe that RND has an inherent flaw.

The Modified RND approach combines different methods in RL, resulting in more

complex code. This increased complexity could lead to more bugs and require additional effort to ensure that the PPO's backbone is shared between its instances (i.e., all of them are pointing to the same instance).

In the Modified RND approach training the shared PPO backbone by optimizing multiple objectives simultaneously via accumulating their corresponding gradients might lead to unstable training and worst convergence properties. This is why we introduced the gradient projection method to help investigate and potentially mitigate this issue. In their work, [Kwon et al., 2024] argued that the gradients from the RL objective are noisy high-variance signals and preventing them to influence the representation learning modules of their architecture yields significant improvements. They simply discarded the RL gradients by using stop-gradients, whereas our gradient projection approach projected these gradients to an orthogonal space to the representation learning gradients.

In our Modified RND approach, the inputs to the RND module are not guaranteed to be stable. This could result in the target values generated by the RND's target network being non-stationary even though that network is frozen. This issue arises from the PPO's shared backbone being simultaneously trained while the exploration module is trying to identify novel states during the training phase. Consequently, this might result in meaningless exploration signals being generated and hinder the training performance.

In the pretraining method, the pretraining phase requires the gathered data to be expressive in order to learn good representations. As a result, the policy which gathers the data should be a good explorative policy. To assess the difference that this policy makes, [Schwarzer et al., 2021b] experimented with using a random policy and an explorative policy. In their work, using a random policy to gather pretraining data still showed improvement signals. Following from this, in our work, we started off by implementing a random policy and looked for indications of potential improvement signals. We believe that any lack of improvement in our pretraining approach might

stem from the trivial random policy approach. It might have resulted in incomprehensive data being gathered during the pretrainig phase and reduced the quality of the representations learned by the SSL Module. Also, in the pretraining method, having a separate pretraining step might reduce sample efficiency and increase training time.

In the ViT with Explorative Attention approach, due to the use of transformer models, it might result in larger models, vast amounts of training data being used, and longer training times. Furthermore, in our experiments, we used the Original RND approach to generate the intrinsic reward signals for the Explorative Attention Module. Due to the aforementioned inherent problems of the Original RND approach, our Explorative Attention Module might have learned to attend to things which are impertinent to the decision making process. This might have misguided our Explorative Attention approach and hindered its performance.

Chapter 9

FUTURE DIRECTIONS

Inspired by our experimental results and the related works shared by others, here we share some ideas for future directions to extend our work further.

[Cobbe et al., 2020] investigated the joint training of policy and value functions using a shared backbone. They pointed out that this formulation carries the risk of objectives clashing and it is not clear how the corresponding weights of these objectives should be picked. To tackle these issues, they proposed the Phasic Policy Gradient (PPG) method, which divides the optimization process into two stages: one for advancing the training and another for distilling the features. More importantly, the PPG algorithm can be employed for carrying out any auxiliary optimization task in conjunction with reinforcement learning. In their work, [Cobbe et al., 2020] used the value function error as the auxiliary optimization task. We believe that if we take the auxiliary optimization task as our Modified RND's SSL objective, we can develop another approach that might be able to handle the aforementioned clashing gradients problem.

Overall in general, one can perform more exhaustive hyperparameter tuning. Try adapting our frameworks to work with off-policy methods, and explore different exploration methods. Also, our approaches can be extended to different modalities other than images. For example, our Modified RND approach makes use of self-supervised learning methods from the computer vision domain because our target benchmark environment for explorative behaviour was Montezuma's Revenge and other OpenAI gym environments, which have high-dimensional state space in the form of images. Although, the RND Module and the PPO backbone can be easily modified to work with 1D non-image inputs, the current SSL module cannot be used.

To tackle this issue of scalability to non-image based tasks with small latent state spaces one can try to change the modality of the SSL method employed as an auxiliary loss. For example, in their work, [Fujimoto et al., 2023] devised representation learning over actions. This formulation operates over latent space predictions, which do not exploit the inherent structure in the images but the very built-in dynamics of the environment itself.

Chapter 10

CONCLUSION

The starting motivation of this thesis was that the potential of using a latent space that is ideally robust to noise, expressive, and relevant to the RL task would improve exploration performance, especially for high-dimensional, stochastic, and sparse reward RL environments. As a result, we sought out ways to learn such a latent space.

Using an intermediate representation of an RL model along with a representation learning objective seemed a good place to start. Based on this, we developed the *Modified RND* approach. Our results suggest that the *Modified RND* falls short of the Original RND method. However, we have observed that incorporating SSL does not deteriorate performance and may actually help with early convergence, especially when rewards are sparse and the latent space dimensions are low. Based on our results, we posit that latent space exploration is the main culprit behind the low performance. We performed extensive analysis on the potential reasons and hypothesize that the main issue is the non-stationary nature of the RND network inputs. The latent space changes as the agent learns, even when the full observations remain the same. In hindsight, we conclude that the nature of how RND explores is not suitable for use with a latent space, especially for on-policy methods. Another hypothesis is that the way we learn the latent space focuses on the RL task. Since we are using an on-policy approach, the agent tends to focus on local improvements, leading to a latent space that only concentrates on a short-sighted representation.

Keeping in line with the nature of RND while learning representations, we decided to try a different architecture and introduce the *ViT with Explorative Attention* method. The architectural change aims to learn exploration specific and exploitation specific representations while leveraging transformers. This method was also not able to surpass the performance of the baseline *Original RND* model.

Despite the lackluster outcomes, we believe we have shed some light on the behavior and potential of utilizing representation learning for predictive exploration methods in reinforcement learning. Based on our findings, we have made suggestions for future work in Chapter 9.

BIBLIOGRAPHY

[Amin et al., 2021] Amin, S., Gomrokchi, M., Satija, H., van Hoof, H., and Precup, D. (2021). A survey of exploration methods in reinforcement learning.

[Anderson et al., 2015] Anderson, C., Lee, M., and Elliott, D. (2015). Faster reinforcement learning after pretraining deep networks to predict state dynamics.

[Andrychowicz et al., 2020] Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O. (2020). What matters in on-policy reinforcement learning? a large-scale empirical study.

[Aytar et al., 2018] Aytar, Y., Pfaff, T., Budden, D., Paine, T. L., Wang, Z., and de Freitas, N. (2018). Playing hard exploration games by watching youtube.

[Burda et al., 2018] Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018). Exploration by random network distillation.

[Cai et al., 2023] Cai, Y., Zhang, C., Shen, W., Zhang, X., Ruan, W., and Huang, L. (2023). Reprem: Representation pre-training with masked model for reinforcement learning.

[Campero et al., 2021] Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J. B., Rocktäschel, T., and Grefenstette, E. (2021). Learning with amigo: Adversarially motivated intrinsic goals.

[Chen et al., 2021] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling.

[Chmura et al., 2023] Chmura, J., Burhani, H., and Shi, X. Q. (2023). Information content exploration.

[Cobbe et al., 2020] Cobbe, K., Hilton, J., Klimov, O., and Schulman, J. (2020). Phasic policy gradient.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

[Engstrom et al., 2020] Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. (2020). Implementation matters in deep policy gradients: A case study on ppo and trpo.

[Fetterman and Albrecth, 2020] Fetterman, A. and Albrecth, J. (2020). Understanding self-supervised and contrastive learning with "bootstrap your own latent" (byol).

[Fujimoto et al., 2023] Fujimoto, S., Chang, W.-D., Smith, E. J., Gu, S. S., Precup, D., and Meger, D. (2023). For sale: State-action representation learning for deep reinforcement learning.

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning.

[Gui et al., 2023] Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., and Tao, D. (2023). A survey on self-supervised learning: Algorithms, applications, and future trends.

[Guo et al., 2022] Guo, Z. D., Thakoor, S., Pîslar, M., Pires, B. A., Altché, F., Tallec, C., Saade, A., Calandriello, D., Grill, J.-B., Tang, Y., Valko, M., Munos,

R., Azar, M. G., and Piot, B. (2022). Byol-explore: Exploration by bootstrapped prediction.

[He et al., 2020] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning.

[Kwon et al., 2024] Kwon, J., Yang, L., Nowak, R., and Hanna, J. (2024). Future prediction can be a strong evidence of good history representation in partially observable environments.

[Lan et al., 2022] Lan, C. L., Tu, S., Oberman, A., Agarwal, R., and Bellemare, M. G. (2022). On the generalization of representations in reinforcement learning.

[Li et al., 2022] Li, A. C., Efros, A. A., and Pathak, D. (2022). Understanding collapse in non-contrastive siamese representation learning.

[Liu and Abbeel, 2021] Liu, H. and Abbeel, P. (2021). Behavior from the void: Unsupervised active pre-training.

[Nguyen et al., 2021] Nguyen, T., Luu, T. M., Vu, T., and Yoo, C. D. (2021). Sample-efficient reinforcement learning representation learning with curiosity contrastive forward dynamics model. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

[Ozbulak et al., 2023] Ozbulak, U., Lee, H. J., Boga, B., Anzaku, E. T., Park, H., Messem, A. V., Neve, W. D., and Vankerschaver, J. (2023). Know your self-supervised learning: A survey on image-based generative and discriminative training.

[Park et al., 2023] Park, S., Kim, J., Jeong, H.-Y., Kim, T.-K., and Yoo, J. (2023). C2rl: Convolutional-contrastive learning for reinforcement learning based on self-pretraining for strong augmentation. *Sensors*, 23(10).

[Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction.

[Pathak et al., 2019] Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-supervised exploration via disagreement.

[Prudencio et al., 2024] Prudencio, R. F., Maximo, M. R. O. A., and Colombini, E. L. (2024). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–0.

[Savinov et al., 2019] Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., and Gelly, S. (2019). Episodic curiosity through reachability.

[Schneider et al., 2023] Schneider, J., Schumacher, P., Häufle, D., Schölkopf, B., and Büchler, D. (2023). Investigating the impact of action representations in policy gradient algorithms.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

[Schwarzer et al., 2021a] Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. (2021a). Data-efficient reinforcement learning with self-predictive representations.

[Schwarzer et al., 2021b] Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, D., Bachman, P., and Courville, A. (2021b). Pretraining representations for data-efficient reinforcement learning.

[Seo et al., 2021] Seo, Y., Chen, L., Shin, J., Lee, H., Abbeel, P., and Lee, K. (2021). State entropy maximization with random encoders for efficient exploration.

[Seo et al., 2022] Seo, Y., Lee, K., James, S., and Abbeel, P. (2022). Reinforcement learning with action-free pre-training from videos.

[Srinivas et al., 2020] Srinivas, A., Laskin, M., and Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning.

[Sun et al., 2023] Sun, C., Qian, H., and Miao, C. (2023). Cudc: A curiosity-driven unsupervised data collection method with adaptive temporal distances for offline reinforcement learning.

[Thrun, 1992] Thrun, S. (1992). Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, PA.

[Wagner and Harmeling, 2024] Wagner, S. S. and Harmeling, S. (2024). Just cluster it: An approach for exploration in high-dimensions using clustering and pre-trained representations.

[Wu et al., 2018] Wu, Z., Xiong, Y., Yu, S., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance-level discrimination.

[Yarats et al., 2021] Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. (2021). Reinforcement learning with prototypical representations.

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction.

Appendix A

A.1 Exploration in RL

[Pathak et al., 2017] identified the properties that a good method to generate intrinsic rewards should possess. They agreed that the method should model both the things under the agent's control and those beyond its control yet capable of influencing the agent. As a consequence, they doubted that whether predicting pixels is a good objective. They proposed a forward dynamics objective which predicts the agent's action given the current and the next states to generate intrinsic rewards. They explain that their objective formulation is not motivated to represent things in the environment which bear no effect on the agent's decision taking process and hence fits the properties that they have identified for a good exploration method. It is interesting to note that in our work we used RND which does not fit the criteria that they have identified since RND can intrinsically reward things which are beyond the agent's influence and have no impact on the agent.

Distinct from most of the other exploration methods proposed in RL which decouples exploration from the agents current capabilities, [Campero et al., 2021] introduced a new approach which borrows from both the curriculum learning and adversarial training methodologies. In their work, they incorporate a goal-generating teacher network which aims to generate goals which are challenging enough for the student policy to reach. The goals generated serve as the intrinsic rewards which incentivize the agent to explore the environment. The student policy network and the goal-generating teacher network are trained adversarially.

[Pathak et al., 2019] emphasized that the stochasticity of the agent-environment interaction can have various reasons. When predictive world models are used for

generating intrinsic rewards, one has to handle these stochasticities well or else there can be problems. For example, using a deterministic prediction model which learns to predict the consequences of the agent's actions will invariably result in a prediction error, which can potentially trap the agent in the local minimum of exploration ([Pathak et al., 2019]). As a solution, [Pathak et al., 2019] proposed to learn a forward model which is itself stochastic by maximizing model-variance (disagreement) over an ensemble of dynamics models. We believe that our Modified RND approach does not suffer from these stochasticity issues because, in the RND module, we are not using any stochastic variables but only stacked image frames (states).

"Couch-potato" issue is a common problem which arises in RL exploration problems. The issue occurs when the agent discovers a means to immediately satisfy its desires by exploiting actions that result in outcomes that are difficult to foresee. To tackle the issue, [Savinov et al., 2019] introduced a new curiosity approach employing episodic memory for generating a novelty bonus. This bonus is computed by contrasting the current observation with those stored in memory. Significantly, the comparison relies on the number of environment steps required to transition from the stored observations to the current one, thereby integrating extensive details about environment dynamics.

[Seo et al., 2021] held the belief that commencing with a fully random encoder could still yield a valuable framework for gauging the extent of randomness across various states, given the inherent capability of convolutional models to capture similarities effectively. Their assertion proved accurate, as they discovered that even with a random encoder, the system adeptly discerned the similarities among diverse states. In a similar manner, [Park et al., 2023] have used a randomly initialized convolutional layer to better generalization in reinforcement learning tasks that involve vision. We think that their findings serve to further explain why the randomly initialized and frozen target RND network in the original RND paper worked so well.

The work of [Yarats et al., 2021] further connects the idea of representation learn-

ing with exploration by introducing prototypical representations which are utilized within a self-supervised framework. These prototypes function both as a summary of an agent's exploratory journey and as a foundation for representing observations ([Yarats et al., 2021]). In contrast to this, in our work there is a clear distinction between representation learning and exploration. These two ideas are also embodied by distinct modules in our Modified RND approach.

[Guo et al., 2022] proposed a framework in which a world model is learned using a self-supervised prediction task. The features learned by the world model can then be utilized by the RL policy, and the same self-supervised prediction loss generates intrinsic rewards to learn an exploration policy. This allows both the exploration and the representation learning to be done by a single self-supervised loss. Our motivation for the Modified RND was to improve the performance of the already existing prediction-error based exploration methods by using the features learned by the SSL loss as inputs while leveraging the learned representations with the RL heads to better the RL performance. On the other hand, the motivation of [Guo et al., 2022] was to both learn a world model and guide exploration using a single SSL loss. In other words, their approach suggests using SSL loss as an exploration method whereas our Modified RND builds on the existing exploration methods such as RND.

[Liu and Abbeel, 2021] introduced a method where unsupervised pretraining is used to learn representations for observations. These representations are then used to generate intrinsic rewards by performing particle-based entropy maximization in the learned representation space. Similar to how their work used the learned representations to derive intrinsic rewards, in our Modified RND approach we use the features learned by the shared backbone as inputs to the RND module to obtain intrinsic rewards.

A.2 Fixing the Moving Inputs to the RND Module Issue

For off-policy RL algorithms which make use of the Bellman error, there is a commonly applied trick. The trick is to update the target networks via an exponential moving average (EMA) operation. This is shown to stabilize the training and better its convergence. [Srinivas et al., 2020] used this trick to have more stable Q targets.

In their work, [Sun et al., 2023] decided to use target encoder networks which are updated by momentum based moving averaging. Overall, they had a state encoder, a target state encoder, an action encoder, and a target action encoder. [He et al., 2020] implemented a contrastive learning scheme which made use of two separate key and query encoder networks. They tried to update the key encoder by simply copying from the query encoder but it yielded poor results. They believed that the direct copying resulted in inconsistent and quickly changing representations. To solve this issue, they used momentum update to update the key encoder using the latest version of the query encoder. Through this mechanism, they managed to control the difference between the encoder networks and empirically achieved better performance results.

It is important to note that momentum updates are not only applied to model parameters but can also be applied to already extracted representations too. For example, [Wu et al., 2018], which takes a memory bank approach, performs momentum updates on the representations rather than on the encoder. In our work, we could not consider performing momentum updates on the representations because our goal is to stabilize the extracted state representations obtained via an online-policy, and maintaining representations for all possible states is intractable.

Following from their observations and given that our findings on the pretraining method (see Chapter 6) were not sufficient to conclude whether the moving inputs to the RND networks were the issue, we suggest applying approaches like EMA to the shared PPO backbone of the RND Module. These approaches could address the

issue of moving inputs while allowing the shared PPO backbone to be trained by the RL objective.

A.3 ViT with Explorative Attention

Transformer architectures shine at processing sequential data. They have initially emerged in NLP applications, and soon they made it to CV applications with models such as Vision Transformers ([Dosovitskiy et al., 2021]). Faced with the question of whether RL could be the next application of these models, the RL community have started to examine the possibility of transformers serving as a versatile solution for sequential decision-making ([Kwon et al., 2024]).

For example, to learn better representations which the RL policy can leverage, [Cai et al., 2023] proposed to pretrain a masked model transformer. The transformer learns representations that can capture long-term dynamics by learning to predict a trajectory's masked states or actions.

[Chen et al., 2021] presented a framework named the Decision Transformer, which conceptualizes reinforcement learning as a conditional sequence modeling task. The Decision Transformer uses a causally masked transformer model. By conditioning an autoregressive model on the target return, previous states, and actions, the Decision Transformer model can produce future actions that yield the desired return ([Chen et al., 2021]).

[Kwon et al., 2024] pointed out that the main issues in RL are policy optimization and exploration problems, which are separate from the remarkable sequence processing capabilities of the transformer family of models. In relation to this, we believe that our proposed ViT model, equipped with the Explorative Attention Module, takes a step forward in leveraging the transformers' capabilities to tackle the issue of exploration by learning separate exploration specific and exploitation specific feature representations.

A.4 Investigation of Observation and Intrinsic Reward Normalization Statistics

We investigated the running mean and variance statistics of the observations and intrinsic rewards through the objects used to normalize them during training. *obs_rms* refers to the object in the code which kept the moving statistics (mean and variance) of the inputs to the RND module networks. *reward_rms* refers to the object in the code which kept the running mean statistics (mean and variance) of the intrinsic rewards produced by the RND module. Both the *obs_rms* and the *reward_rms* were introduced in the original RND paper and we kept it in our Modified RND formulation.

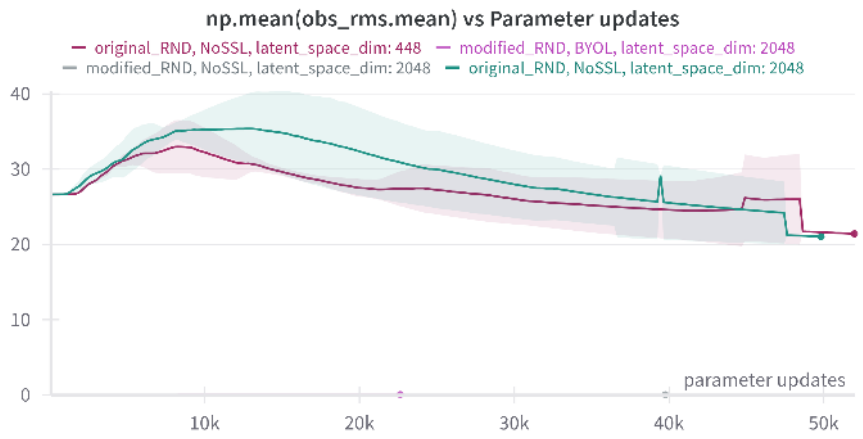


Figure A.1: Mean of the mean variable of obs_rms vs parameter updates in the original Modified RND experiments.

From Figure A.1 and Figure A.2, we observe that both the Original RND's and the Modified RND's *obs_rms* values converged. Original RND has obs_rms $\neq 0$, whereas Modified RND has obs_rms ~ 0. This makes sense because Original RND uses *obs_rms* to normalize raw states (frames). On the other hand, Modified RND

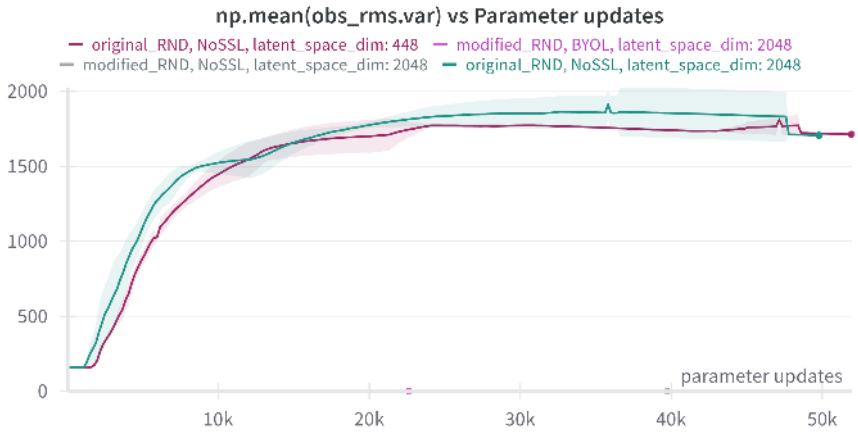


Figure A.2: Mean of the variance variable of obs_rms vs parameter updates in the original Modified RND experiments.

uses *obs_rms* to normalize the features extracted by the shared PPO backbone. The layers of this backbone are initialized to ideally output zero-mean, unit-variance values to address issues such as exploding or vanishing gradients. We think that when using Modified RND, we might eliminate the need for *obs_rms*. The original RND paper introduced *obs_rms* to ensure zero-mean, unit-variance inputs to the RND networks. However, in Modified RND, we are no longer using raw state observations but the outputs from the shared PPO backbone. As neural network layers in the shared PPO backbone are configured by default to operate with zero-mean, unit-variance values, we end up with inputs close to zero-mean, unit-variance for the RND Module when using the shared PPO backbone's last layer outputs as inputs to the RND Module.

Figures A.3, A.4, and A.5 show the statistics related to the reward normalization object, specifically the *reward_rms*. Unlike the Original RND, which has stable *reward_rms* variance, the Modified RND exhibits unstable *reward_rms* variance that sometimes explodes. We suspect this explosion reflects misleading intrinsic rewards and overall poor exploration. We believe this issue arises from the RND module's *target_network* extracting misleading features. While the *reward_rms.var* values of the Original RND converge, those of the Modified RND do not exhibit certain jumps. We think this discrepancy stems from the Modified RND's *target_network* yielding

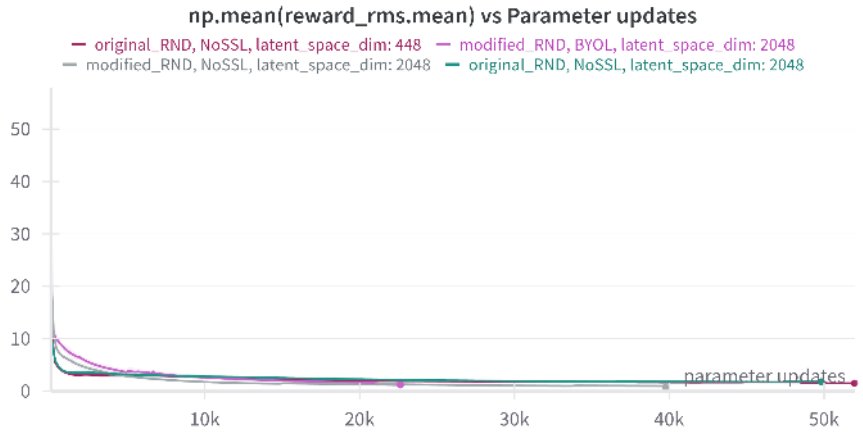


Figure A.3: Mean of the mean variable of reward_rms vs parameter updates in the original Modified RND experiments.

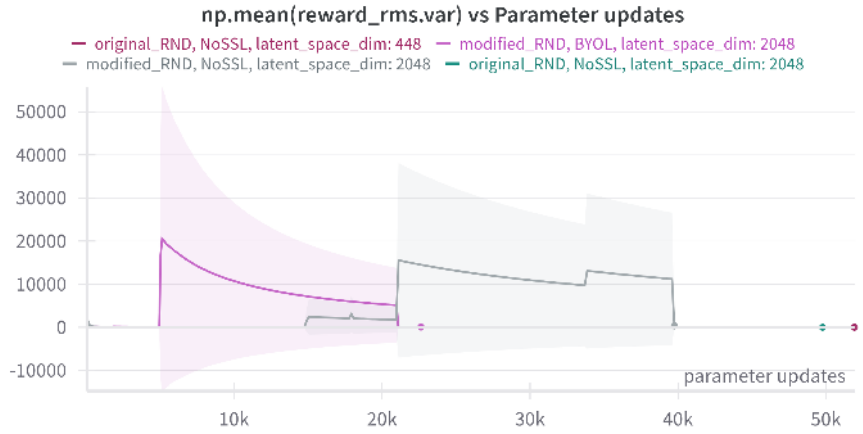


Figure A.4: Mean of the variance variable of reward_rms vs parameter updates in the original Modified RND experiments.

noisy and not useful features, which are then converted into intrinsic rewards. This hypothesis regarding the *target_network* yielding poor *target_features* was explored earlier in Chapter 6 when we investigated the RND Module's statistics.

A.5 Investigation of BYOL Losses

We analyse the BYOL loss values from our experiments to gain further insight. This helps us both to better evaluate the SSL performance and to better understand the

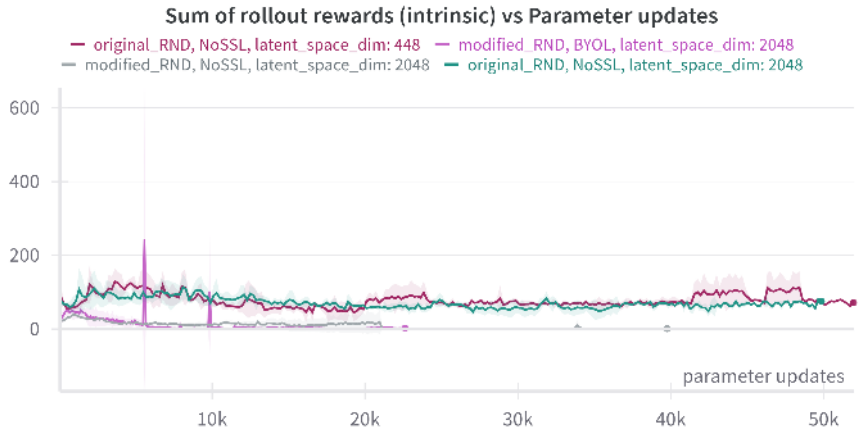


Figure A.5: Sum of the intrinsic rollout rewards vs parameter updates in the original Modified RND experiments.

behaviour of the introduced SSL method.

The BYOL loss is calculated as BYOL_loss = $2-2*(\vec{v_1}*\vec{v_2})$.sum(dim=-1), where $\vec{v_1}$and $\vec{v_2}$ are the two generated augmented views. Observe the BYOL loss values for the following two special cases. First, when $\vec{v_1} = \vec{v_2}$, and $\vec{v_1} \neq \vec{0}$ BYOL_loss = $2 - 2 * (\vec{v_1} * \vec{v_2})$.sum(dim=-1) = $2 - 2 * (\vec{v_1} * \vec{v_1})$.sum(dim=-1) = $2 - 2 * (1) = 0$. Here we used the fact that $\vec{v_1}$ and $\vec{v_2}$ are normalized vectors. Second case is when $\vec{v_1} = \vec{0}$ and $\vec{v_1} \neq \vec{0}$. Then BYOL_loss = $2 - 2 * (\vec{v_1} * \vec{v_2})$.sum(dim=-1) = $2 - 2 * (\vec{0} * \vec{v_2})$.sum(dim=-1) = $2 - 2 * (0) = 2$. Third case is when $\vec{v_1} \perp \vec{v_2}$ which yields BYOL_loss = $2 - 2 * (\vec{v_1} * \vec{v_2})$.sum(dim=-1) = $2 - 2 * (0) = 2$.

For the original Modified RND experiments (Figure A.6), towards the end, the BYOL loss is around 0.5, which yields BYOL_loss = $0.5 = 2-2*(1.5/2)$ then, $0.75 =< \vec{v_1}, \vec{v_2} >= \cos\theta * ||\vec{v_1}||_2 * ||\vec{v_2}||_2$. Since $\vec{v_1}$ and $\vec{v_2}$ are normalized vectors, we get $0.75 = \cos(\theta) \rightarrow \cos^{-1}(\theta) = 41.4°$. Despite the BYOL loss being an auxiliary loss, we see that it is small, which implies that $\vec{v_1}$ is generally similary to $\vec{v_2}$, which is desirable. Therefore, we believe that BYOL is learning to predict the target vectors and is not predicting the trivial solution of 0 for every input. We explain the small BYOL losses with the following reasons. First, the inputs to the BYOL networks

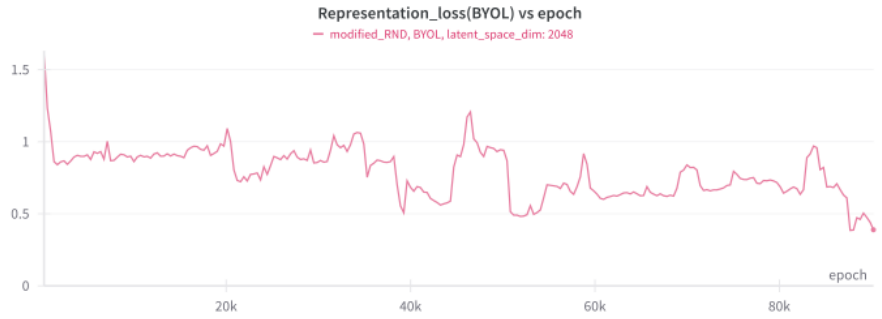


Figure A.6: Representation loss (BYOL) vs epoch for Modified RND in the original Modified RND experiments.

come from the Atari environments, which might look somewhat similar. Training over such similar states might make learning easier for the BYOL networks. Second, BYOL uses two different augmented views of the same state (stacked images) as inputs. It is possible that our image augmentations are not strong enough, resulting in two views that are not very different from each other. In such cases, providing similar views to BYOL's predictor and target networks would lead to similar outputs and thus small BYOL loss values.

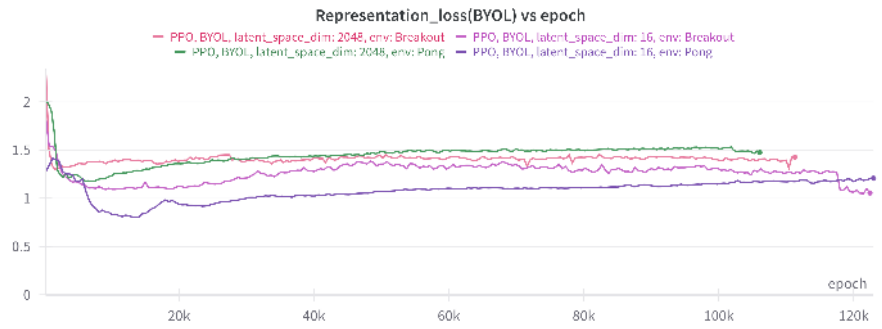


Figure A.7: Representation loss (BYOL) vs epoch for the effects of SSL without exploration experiments.

For the effects of SSL without exploration experiments (Figure A.7), the BYOL losses are around 1.3, which yields BYOL_loss $= 1.3 = 2 - 2 * (1.3/2)$, then $0.65 =< \vec{v_1},\ \vec{v_2} >= \cos\theta * ||\vec{v_1}||_2 * ||\vec{v_2}||_2$. Since $\vec{v_1}$ and $\vec{v_2}$ are normalized vectors, we get $0.65 = \cos(\theta) \rightarrow \cos^{-1}(\theta) = 49.5°$. This is consistent with our findings for the

original Modified RND experiments. We attribute the difference in BYOL loss values between this graph and the original Modified RND experiments' graph to the fact that this graph's runs were conducted on different Atari environments (Pong and Breakout) compared to the ones used in the other graph's runs (Montezuma's Revenge). The differences in these environments might have led to BYOL training on characteristically different looking images, which could have caused the variations in loss values.

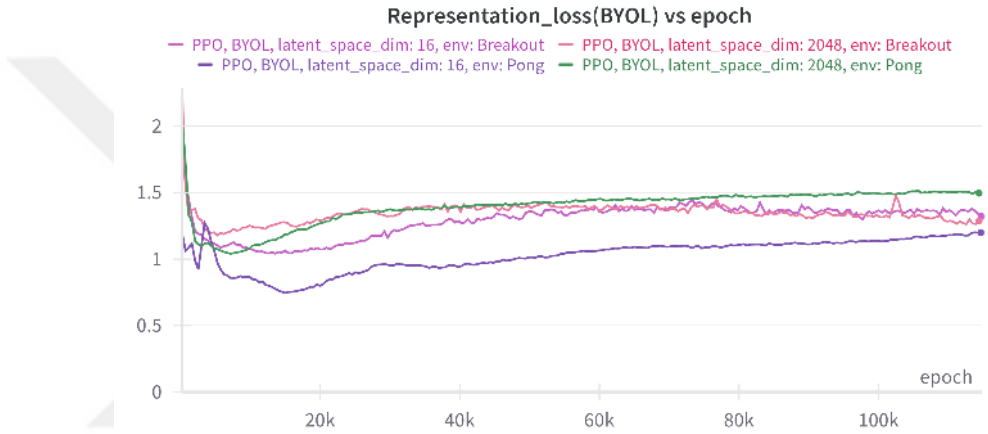


Figure A.8: Representation loss (BYOL) vs epoch for the effects of SSL and gradient projection without exploration experiments.

For the effects of SSL and gradient projection without exploration experiments, the BYOL losses are around 1.3. This yields BYOL_loss $= 1.3 = 2-2*(1.3/2)$,then $0.65 =< \vec{v_1},\ \vec{v_2} >= \cos\theta * ||\vec{v_1}||_2 * ||\vec{v_2}||_2$. Since $\vec{v_1}$ and $\vec{v_2}$ are normalized vectors, we get $0.65 = \cos(\theta) \rightarrow \cos^{-1}(\theta) = 49.5°$. When compared with the BYOL loss values from the effects of SSL without exploration experiments, we observe that using the Gradient Projection method did not cause BYOL's loss values to increase. This implies that projecting BYOL's gradients, which were clashing with the RL objective's gradients, did not hinder BYOL's training performance.

The Modified RND with pretraining experiments, which used the pretraining method, appears to have had higher BYOL losses compared to the runs in the

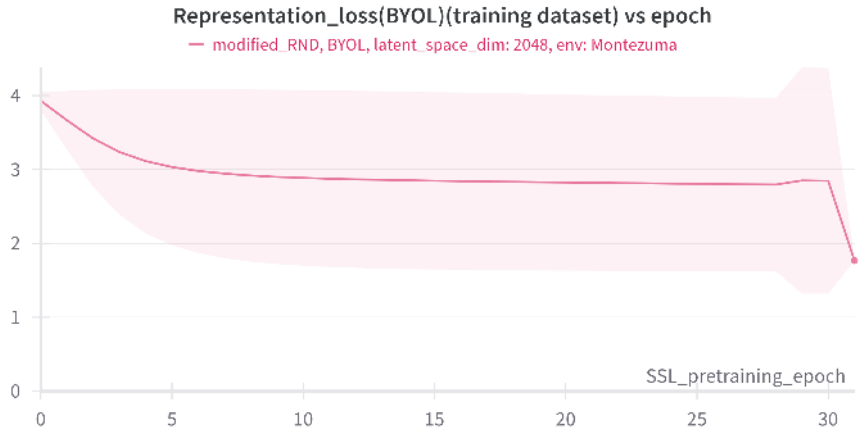


Figure A.9: Training representation loss (BYOL) vs SSL pretraining epoch for the Modified RND with pretraining experiments.

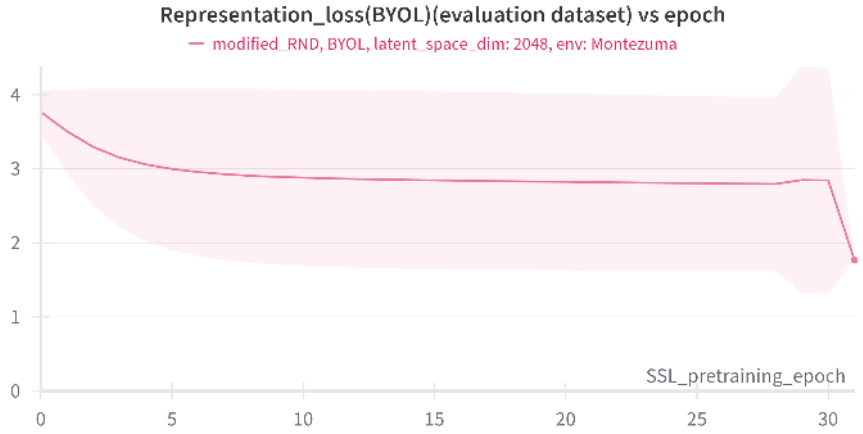


Figure A.10: Evaluation representation loss (BYOL) vs SSL pretraining epoch for the Modified RND with pretraining experiments.

original Modified RND experiments. This can be explained by the pretraining phase being run for a shorter duration and the use of a naive random policy for pretraining data collection, which is not effective for exploring Montezuma's Revenge. Among these explanations, the shorter training time seems to make the most sense, as the graphs for the Just PPO with SSL and pretraining experiments (Figure A.11, A.12) show similar BYOL loss values to their corresponding runs without the pretraining method. Based on this, we believe that using an auxiliary BYOL loss with on-policy

training is sufficient to learn good BYOL features and that a separate pretraining phase is not necessary.

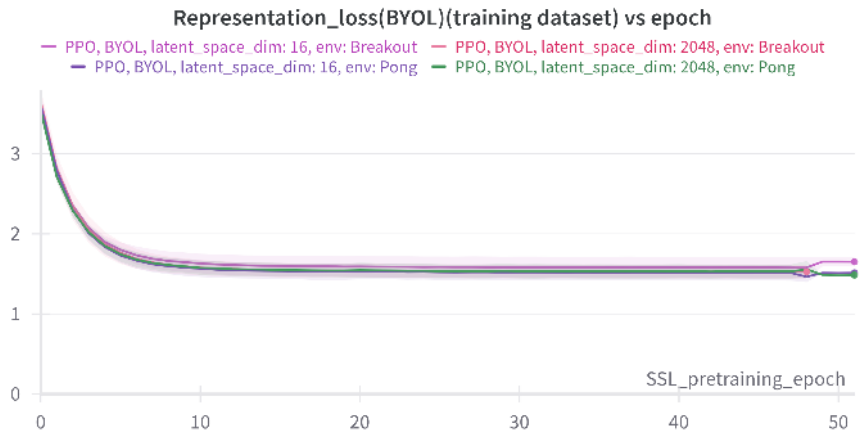


Figure A.11: Training representation loss (BYOL) vs SSL pretraining epoch for the Just PPO with SSL and pretraining experiments.

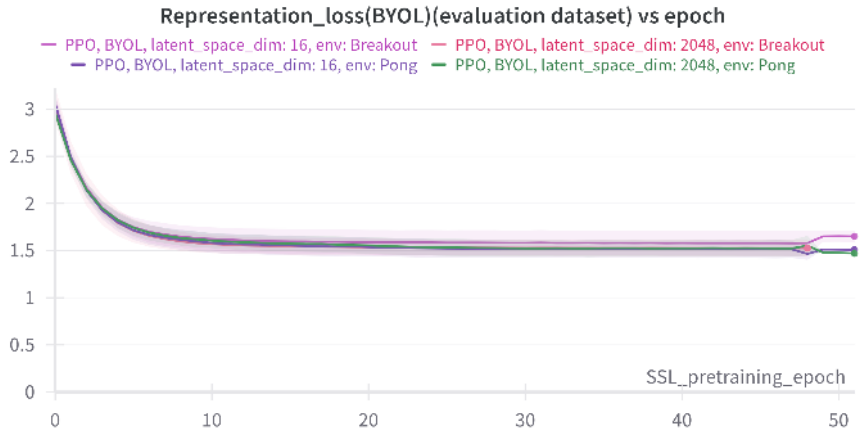


Figure A.12: Evaluation representation loss (BYOL) vs SSL pretraining epoch for the Just PPO with SSL and pretraining experiments.

A.6 Overview of the Experiment Configurations

Our experiments began by investigating the effectiveness of the proposed Modified RND approach. Motivated by the findings, we then introduced and tested various

modifications in subsequent runs. Finally, we assessed the performance of the proposed ViT with Explorative Attention method. These experiments were run using 3 different seeds (as [Guo et al., 2022] have also used 3 seeds) and 64 parallel environments. The key configurations are listed below in the order they are presented in this paper.

■ **Original Modified RND Experiments**

env: Montezuma's Revenge

latent space dimension: 2048

train_method: ['originalRND', 'modifiedRND_NoSSL', 'justPPO_NoSSL', 'modifiedRND_BYOL']

---

env: MontezumaRevenge

latent space dimension: 448

train_method: 'originalRND'

■ **Modified RND with Gradient Projection Experiments**

env: Montezuma's Revenge

latent space dimension: [448, 2048]

train_method: 'modifiedRND_BYOL'

use_gradient_projection: True

■ **Effects of SSL without Exploration Experiments**

env: [Pong, Breakout]

latent space dimension: [16, 2048]

train_method: ['justPPO_NoSSL', 'justPPO_BYOL']

■ **Effects of SSL and Gradient Projection without Exploration Experiments**

env: [Pong, Breakout]

latent space dimension: [16, 2048]

train_method: 'justPPO_BYOL'

use_gradient_projection: True

■ **Modified RND with Pretraining Experiments**

env: Montezuma's Revenge

latent space dimension: 2048

train_method: 'modifiedRND_BYOL'

SSL_pretraining: True

■ **Just PPO with SSL and Pretraining Experiments**

env: [Pong, Breakout]

latent space dimension: [16, 2048]

train_method: 'justPPO_BYOL'

SSL_pretraining: True

■ **EMA Updated Target PPO Backbone Experiments**

env: Montezuma's Revenge

latent space dimension: 2048

train_method: 'modifiedRND_BYOL'

smoothing factor (τ): [0.01, 0.05]

■ **Modified RND with Convolutional Last Layer Experiments**

env: Montezuma's Revenge

latent space dimension: 448

train_method: ['modifiedRND_NoSSL', 'modifiedRND_BYOL']

shared_ppo_backbone_last_layer_type: Conv

■ **ViT Experiments**

env: Montezuma's Revenge

latent space dimension: 448

train_method: 'originalRND_NoSSL'

attention_type: ['regularViTAttention', 'ExplorativeAttention']

A.7 Initial Experiments Comparing BYOL and Barlow Twins

Due to time and computational constraints during our research, we conducted our experiments using the BYOL method and were unable to repeat the same set of experiments with Barlow Twins method. We believed that both methods were interchangeable, provided they were properly tuned. To test this, we conducted an experiment varying the α_{SSL} (see equation 4.1) values. We tested with $\alpha_{\text{SSL}} = [10, 1e^{-3}, 1e^{-4}, 1e^{-5}]$ and shared the results in Figure A.13.

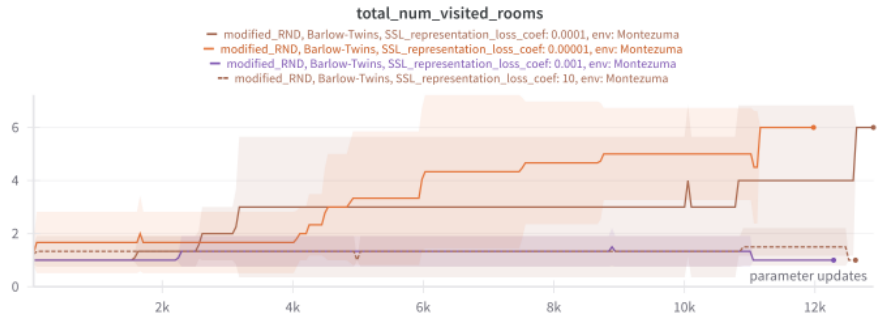


Figure A.13: Total number of visited rooms vs parameter updates in Montezuma's Revenge for the Barlow Twins tuning experiments. Note that *SSL_representation_loss_coef* in the legend corresponds to the α_{SSL} values.

Our results indicate that when the Barlow Twins method's α_{SSL} coefficient is tuned properly, its performance can match that of the BYOL method. Hence, we believe that our Modified RND approach's claim to work with different non-contrastive SSL methods holds. Moreover, the results suggest that the findings

from our other experiments using the BYOL method as the SSL objective should generalize, as the Barlow Twins method appears to yield similar performance.