UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering
Department of Industrial Design & Production Engineering
MSc in Intelligence and Deep Learning

Master of Science Thesis

Transferable State and Action Embeddings in Deep Reinforcement Learning

Student: Eleftheriou Dimitrios
Registration Number: AIDL-0038

MSc Thesis Supervisor

Dr. Kasnesis Panagiotis
Lecturer

ATHENS-EGALEO, June, 2024

This MSc Thesis has been accepted, evaluated and graded by the following committee:

Supervisor	Member	Member
Panagiotis Kasnesis Digitally signed by Panagiotis Kasnesis Date: 2024.07.29 16:24:23 +03'00'	Digitally signed by CHARALAMPOS PATRIKAKIS Date: 2024.07.30 14:20:40 +03'00'	ELENI AIKATERINI LELIGKOU Digitally signed by ELENI AIKATERINI LELIGKOU Date: 2024.07.29 10:52:19 +03'00'
Kasnesis, Panagiotis	Patrikakis, Charalampos	Leligou, Nelly
Lecturer	Professor	Professor
Department of Electrical & Electronics Engineering	Department of Electrical & Electronics Engineering	Department of Industrial Design & Production Engineering
Univeristy of West Attica	Univeristy of West Attica	Univeristy of West Attica

Declaration of the author of this MSc thesis

I, Eleftheriou Vasileios Dimitrios with the following student registration number: aidl-0038, postgraduate student of the MSc program in "Artificial Intelligence and Deep Learning", which is organized by the Department of Electrical and Electronic Engineering and the Department of Industrial Design and Production Engineering of the Faculty of Engineering of the University of West Attica, hereby declare that:

I am the author of this MSc thesis and any help I may have received is clearly mentioned in the thesis. Additionally, all the sources I have used (e.g., to extract data, ideas, words or phrases) are cited with full reference to the corresponding authors, the publishing house or the journal; this also applies to the Internet sources that I have used. I also confirm that I have personally written this thesis and the intellectual property rights belong to myself and to the University of West Attica. This work has not been submitted for any other degree or professional qualification except as specified in it.

Any violations of my academic responsibilities, as stated above, constitutes substantial reason for the cancellation of the conferred MSc degree.

I wish to deny access to the full text of my MSc thesis until 30/6/2024 following my application to the Library of UNIWA and the approval from my supervisor.

The author

Eleftheriou Dimitrios

To my mother Valentina

Abstract

Reinforcement Learning (RL) represents a crucial and rapidly advancing domain within Machine Learning, focusing on the resolution of sequential decision-making challenges. It plays a pivotal role in the development of contemporary technologies, evidenced by its application in various sophisticated fields. For instance, RL algorithms are integral to the functioning of autonomous vehicles, enabling self-driving cars to navigate complex environments safely and efficiently. In industrial robotics, RL is utilized to optimize processes and improve precision, enhancing the automation of manufacturing and operational tasks. Moreover, in the realm of Natural Language Processing (NLP), RL contributes to advancements in systems designed for question answering, where the algorithms learn to understand and generate human-like responses. The gaming industry also benefits significantly from RL, with algorithms capable of mastering complex games, demonstrating strategic thinking and adaptability.

This thesis delves into two primary areas: the impact of transfer learning on the performance of RL algorithms and the implementation of a transformers-based architecture for learning representations of both states and actions. The proposed architecture, named JASE-DQN (Joint Action and State Embeddings Deep Q-Network), introduces a novel approach by concurrently learning latent representations of state and action attributes, as well as their interdependencies. Transfer learning process is facilitated through the integration of pretrained models for state and action representations within a cross-attention transformer framework.

The use of pretrained models in JASE-DQN significantly enhances the learning efficiency compared to traditional RL algorithms. In empirical studies, particularly focusing on the game Pac-Man, the JASE-DQN architecture demonstrated a remarkable reduction in training time by nearly 50% when compared with the standard Deep Q-Network (DQN) algorithm. This reduction in training time without sacrificing performance showcases the effectiveness of integrating transfer learning with cross-attention mechanisms.

Additionally, the thesis explores the transferability of pretrained embeddings to different environments. For example, when applying the embeddings learned from the Ms. Pac-Man game to a new environment, specifically the game Alien, the JASE-DQN algorithm not only matched the performance of the DQN algorithm but did so in approximately 75% less training time. Moreover, the JASE-DQN achieved an approximate 25% improvement in performance, measured by the average reward, highlighting its capability in leveraging prior knowledge for enhanced learning and adaptation.

These findings underscore the transformative potential of combining pretrained embeddings within a cross-attention transformers framework in RL. The JASE-DQN architecture seems able to not only accelerate the learning process but also to improve the performance when compared to DQN algorithm. This dual benefit is particularly significant for applications requiring quick adaptation but also high efficiency, such as dynamic and unpredictable environments.

In conclusion, this thesis presents a compelling case for the adoption of advanced architectural strategies in RL, such as the JASE-DQN. By leveraging the strengths of transfer learning and transformer-based models, it is possible to achieve substantial gains in both learning speed and effectiveness. The results achieved in this research illustrate a promising direction for future work in RL, emphasizing the importance of sophisticated representation learning techniques in enhancing the capability of RL systems.

Keywords

Reinforcement Learning, Transfer Learning, Deep Learning, Transformers, Atari Games, Embeddings

Table of contents

List of figures

Figure 21: Comparison between rolling average rewards of JASE-DQN without using pre-trained encoders along with DQN

Figure 22: Comparison between rolling average rewards of JASE-DQN and DQN with different durations of fine tuning for state and action encoders.

Figure 23: Comparison between rolling average rewards of JASE-DQN using pre-trained encoders along with DQN in playing Alien

Figure 24: Comparison between rolling average rewards of JASE-DQN using pre-trained encoders along with DQN in playing Bank Heist

Acronym Index

AI: Artificial Intelligence

CNN: Convolutional Neural Networks

DL: Deep Learning

DQN: Deep Q-Network

DRL: Deep Reinforcement Learning

ICM: Intrinsic Curiosity Module

JASE: Joint Action and State Emebeddings

MSE: Mean Squared Error

NLP: Natural Language Processing

ReLU: Rectified Linear Unit

RL: Reinforcement Learning

RNN: Recurrent Neural Network

TD: Temporal Difference

TL: Transfer Learning

ViT: Vision Transformer

INTRODUCTION

Reinforcement Learning (RL) is a dynamic and rapidly evolving sector of Machine Learning that has garnered substantial attention for its ability to solve complex sequential decision-making problems. From powering self-driving cars and enabling sophisticated industrial robotics to enhancing Natural Language Processing (NLP) tasks such as question answering and transforming the landscape of gaming, RL stands at the forefront of technological innovation. The advancements in RL over recent years underscore its potential to revolutionize various fields, making it a cornerstone of modern AI research and application.

The subject of this thesis

This thesis explores the integration of transfer learning into RL algorithms to address the critical issue of lengthy training times and suboptimal performance in new environments. As RL models often require extensive training to achieve proficiency, the ability to transfer knowledge from pretrained models presents a compelling opportunity to enhance efficiency. This is particularly important given the growing complexity and scale of RL tasks in real-world applications. The focus of this research is on developing a method that leverages pretrained state and action embeddings within a cross-attention transformer architecture to significantly accelerate learning and improve performance across diverse RL environments.

Aim and objectives

The primary aim of this thesis is to demonstrate how the integration of transfer learning with a cross-attention mechanism can enhance the performance and learning speed of RL algorithms. The specific objectives are:

1. To investigate the effectiveness of pretrained state and action embeddings in RL.
2. To develop a cross-attention transformer architecture that fuses these embeddings.
3. To evaluate the impact of this approach on the learning time and performance compared with a Deep Q-Network (DQN) algorithm in playing atari games (eg. Ms Pac-Man).
4. To assess the generalizability of the pretrained embeddings in new gaming environments, specifically the game Alien.

These objectives will be pursued through a series of research questions, including:

- How do pretrained embeddings influence the learning efficiency of RL algorithms?
- What improvements in performance can be observed when applying the cross-attention mechanism?
- Can the benefits observed in one environment (Pac-Man) be transferred to another (Alien)?

Methodology

The methodology adopted in this thesis involves several key steps:

- **Pretrained Embeddings**: State and action embeddings are pretrained on diverse tasks to capture a wide range of features.
- **Developing the Architecture**: A cross-attention transformer architecture is designed to integrate these pretrained embeddings into RL models.
- **Implementing the Model**: The integrated model is implemented using the Intrinsic Curiosity Module.
- **Evaluation and Testing**: The model is tested on the game Pac-Man to measure reductions in learning time and improvements in performance.
- **Generalization Assessment**: The pretrained embeddings are further evaluated in a different environment, the game Alien, to assess their generalizability and effectiveness.

Innovation

This thesis introduces several innovative elements:

1. Cross-Attention Mechanism: The use of a cross-attention mechanism to integrate pretrained state and action embeddings in RL is novel and has shown to significantly enhance learning efficiency.
2. Transfer Learning Application: Applying transfer learning in this specific manner to RL tasks demonstrates a new approach to overcoming the challenges of lengthy training periods and poor performance in novel environments.
3. Empirical Evaluation: By providing empirical evidence of reduced training times and improved performance in different gaming environments, this research offers practical insights into the benefits of this innovative approach.

Structure

The structure of this thesis is organized into the following chapters:

1. Chapter 1: Background and Literature Review – Introduces the basic principles of reinforcement learning as well as some of the main methodological frameworks. Surveys existing research on reinforcement learning (RL), transfer learning, and attention mechanisms.
2. Chapter 2: Atari Games in the Gym Library – Presents an overview of the games Ms Pac-Man, Alien and Bank Heist which will be used to evaluate the performance of reinforcement learning (RL) algorithms.
3. Chapter 3: Methodology - Details the methodological framework, including the pretraining of embeddings and the design of the cross-attention transformer architecture.
4. Chapter 4: Results - Presents the results of training both DQN and JASE-DQN algorithms on the Ms. Pac-Man game. This chapter also includes a comparative analysis of the two algorithms, explores the application of transfer learning to the Alien and Bank Heist environments, and provides a detailed discussion of the performance metrics.

5. Chapter 5: Conclusions - Summarizes the findings, discusses the implications of the research, and suggests directions for future work.

This comprehensive structure ensures a thorough exploration of the research questions and a clear presentation of the innovative contributions of this thesis.

1 CHAPTER 1: Background and Literature Review

This chapter provides a comprehensive overview of the key concepts and methodologies that form the foundation of this thesis. It begins with an introduction to Reinforcement Learning (RL) and its advanced form, Deep Reinforcement Learning (DRL). The chapter then explores the architecture and functionality of Deep Q-Networks (DQNs) and the innovative application of transformers in machine learning. Finally, it discusses the principles of transfer learning and presents a literature review of relevant research, highlighting the contributions and gaps that this thesis aims to address.

1.1 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of Machine Learning concerned with how agents ought to take actions in an environment to maximize cumulative reward. It is distinguished by its focus on learning from interaction and the trade-off between exploration (of uncharted territory) and exploitation (of current knowledge). RL has been instrumental in solving complex decision-making problems where the optimal solution is not immediately clear and must be discovered through trial and error.

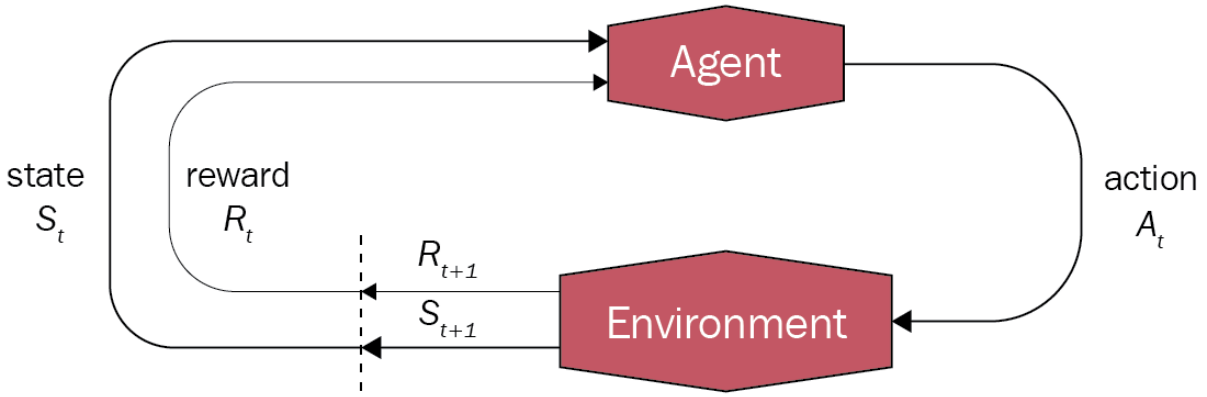

Figure 1: Fundamental interaction loop in Reinforcement Learning

Figure 1 illustrates the fundamental interaction loop between an agent and its environment in Reinforcement Learning (RL). Here is a detailed explanation:

1. **State (St)**: At any given time step t, the agent observes the current state of the environment, denoted as St.
2. **Action (At)**: Based on the observed state St, the agent selects and performs an action At.
3. **Environment Response**: The action At affects the environment, resulting in a new state St+1 and a reward Rt+1 at the next time step t+1.

4. **Reward (Rt)**: The reward Rt is a scalar feedback signal received by the agent from the environment at time step tt, which evaluates the immediate impact of the action At.
5. **Next State (St+1)**: The environment transitions to a new state St+1 as a consequence of the action At.

The loop can be described mathematically with the following notation:

1. At time t:
   a. The agent is in state St.
   b. The agent takes action At.
2. At time t+1:
   c. The agent receives reward Rt+1 based on the previous state St and action At.
   d. The environment transitions to a new state St+1.

Formally: $S_t \xrightarrow{A_t} (R_{t+1}, S_{t+1})$

The process repeats, with the agent using the reward and new state information to inform its next action, continuing the cycle of interaction. This loop encapsulates the agent's goal to learn a policy π that maximizes the cumulative reward over time.

1.2 Q-Learning

Q-Learning is a model-free reinforcement learning algorithm designed to learn the value of actions in a given state. It is an off-policy algorithm, meaning it learns the value of the optimal policy independently of the agent's actions. The core objective of Q-Learning is to estimate the optimal action-value function $Q^*(s, a)$, which indicates the maximum expected cumulative reward achievable from state s by taking action a and following the optimal policy thereafter.

The Q-Learning process involves initializing a Q-table to store estimated values for each state-action pair. The agent interacts with the environment by selecting actions (often using an epsilon-greedy policy to balance exploration and exploitation), receiving rewards, and transitioning to new states. The Q-values are updated using the Q-learning formula:

$$Q_{new}(s, a) = Q(s, a) + \alpha\,(R(s, a) + \gamma\, max_{a'} Q'(s', a') - Q(s, a))$$

where:

- s is the current state
- a is the action taken in the current state
- $Q(s, a)$ expected cumulative reward by taking action a from state s
- $R(s, a)$ is the actual reward received after taking action a from state s
- s' is the next state
- a' is the next action taken in state s'

- $max_{a'}Q'(s', a')$ is the maximum Q-value for the next state s' over all possible actions a'
- α is the learning rate
- γ is the discount factor, a parameter between 0 and 1. It determines the importance of future rewards with values closer to 1 makes future rewards more important.

The main task in Q-Learning is to develop a function $f_\theta(s, a)$ that accurately predicts Q-values. The problem formulation is, use $f_\theta(s, a)$ to predict the terms $\hat{Q}(s, a)$ and $\hat{Q}'(s', a')$ with respect to the minimization problem:

$$minimize_\theta \; L_\theta(\hat{Q}(s, a), \hat{Q}_{new}(s, a))$$

where:

- L_θ can be any loss function for regression problems (e.g. MSE, RMSE, MAE)
- $\hat{Q}(s, a) = f_\theta(s, a)$
- $\hat{Q}_{new}(s, a) = \hat{Q}(s, a) + \alpha \, (R(s, a) + \gamma \, max_{a'}\widehat{Q'}(s', a') - \hat{Q}(s, a))$

1.3 Alternatives to Q-Learning

In Reinforcement Learning (RL), there are several alternatives to value-based methods (Q-Learning), each with its own approach to learning optimal policies. The main alternatives include:

Policy-based methods which focus directly on learning the policy that maps states to actions without necessarily estimating value functions. They optimize the policy by maximizing the expected cumulative reward. The most well known algorithm is Proximal Policy Optimization (PPO).

Actor-critic methods which combine aspects of both value-based and policy-based methods. The actor learns the policy, while the critic estimates the value function. An example of these methods is the algorithm Deep Deterministic Policy Gradient (DDPG), an off-policy algorithm that can operate over continuous action spaces, combining Q-learning with policy gradients.

Model-based methods which involve creating a model of the environment's dynamics (i.e., transition and reward functions). The agent uses this model to plan and make decisions. An example of this class of algorithms is Monte Carlo Tree Search (MCTS) that builds a search tree using simulated outcomes to make decisions, often used in game playing.

1.4 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines the principles of RL with Deep Learning, using neural networks to approximate the decision-making policy and value functions. This combination allows DRL to handle high-dimensional state spaces and complex environments, making it suitable for a wide range of applications from robotics to game playing.

1.4.1 Deep Q-Networks DQN

Deep Q-Networks (DQNs) represent a groundbreaking approach in deep reinforcement learning (DRL), utilizing a deep neural network to approximate the Q-function, which estimates the expected reward for taking specific actions in given states. Introduced by DeepMind in their seminal work "Playing Atari with Deep Reinforcement Learning" [1], DQNs achieved a significant milestone in the field of reinforcement learning by enabling agents to master complex video games. This was exemplified by the mastery of seven Atari 2600 games from the Arcade Learning Environment, where the extensive state space renders traditional methods ineffective.

1.4.2 Advancements and Variations in the DQN Framework

Within the Deep Q-Network (DQN) framework, numerous advancements and variations have been devised to enhance learning stability and efficiency. Double DQNs, as presented by Van Hasselt, Guez, and Silver in their influential paper "Deep Reinforcement Learning with Double Q-learning" [2], mitigate the overestimation bias inherent in the Q-learning algorithm by decoupling action selection from action evaluation. This approach yields more accurate value estimates and improved performance.

Dueling DQNs, introduced by Wang et al. in "Dueling Network Architectures for Deep Reinforcement Learning" [3], propose a novel network architecture that separately estimates the state value function and the advantage function for each action. This separation allows the network to learn which states are valuable independently of the effects of each action within those states, resulting in more efficient and reliable learning.

Prioritized Experience Replay, detailed by Schaul et al. in "Prioritized Experience Replay" [4], modifies the standard experience replay mechanism by prioritizing significant transitions. This ensures the agent learns more effectively from experiences that have a substantial impact on learning. By sampling more frequently from these prioritized experiences, the agent focuses on learning from crucial states and actions, thereby accelerating the training process and enhancing convergence.

Each of these innovations addresses specific challenges within the DQN framework, contributing to the evolution of more robust and efficient reinforcement learning algorithms.

1.5 Transformers

Transformers, as originally introduced by Vaswani et al. in "Attention Is All You Need" [5], have revolutionized various machine learning tasks beyond their initial application in Natural Language Processing (NLP). Their ability to handle sequential data without the constraints inherent in recurrent architectures marks a significant advancement. The core innovation of transformers lies in their self-attention mechanism, which dynamically weighs the importance of different elements within the input sequence. This mechanism enables transformers to capture long-range dependencies and contextual relationships more effectively, leading to superior performance across a wide range of tasks.

1.5.1 Transformer Architecture: Key Components

The transformer architecture has revolutionized machine learning tasks, particularly in Natural Language Processing (NLP), due to its ability to handle sequential data efficiently. At its core, the transformer consists of an encoder-decoder structure as shown in Figure 2. The encoder maps an input sequence to a continuous representation, while the decoder uses this representation to generate an output sequence.

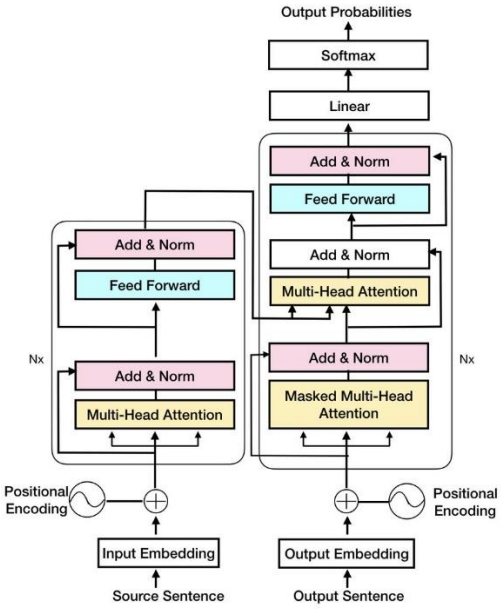

Figure 2: Encoder-Decoder Structure [5]

Multi-head attention mechanisms are crucial in this architecture, allowing the model to focus on different parts of the input sequence simultaneously. Each attention head operates independently, computing scaled dot-product attention, and their outputs are concatenated and linearly transformed as shown in Figure 3. This multi-faceted attention process helps the model capture various aspects of the data's context.

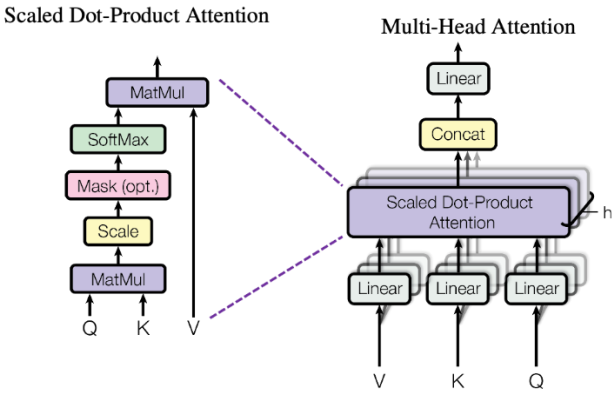

Figure 3: The multi-head attention mechanism, illustrating how multiple attention heads operate in parallel [5]

Additionally, the transformer employs positional encoding to inject information about the position of each token in the sequence, since the architecture itself does not inherently consider the order of tokens. This encoding uses sine and cosine functions to generate unique positional vectors added to the input embeddings, enabling the model to learn positional relationships as shown in Figure 4. Together, these components form a powerful and flexible architecture capable of learning complex dependencies in data.

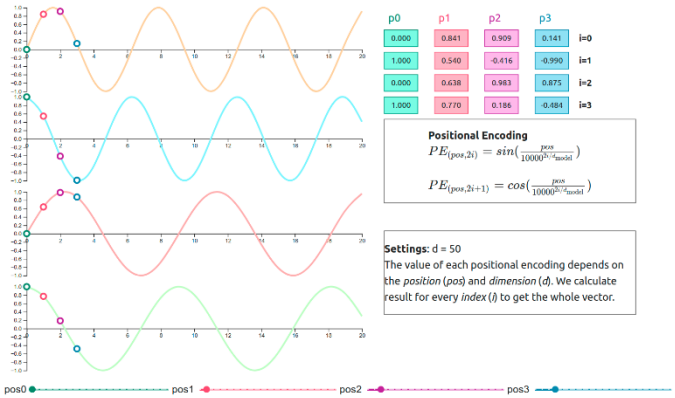

Figure 4: Visualization of positional encoding, where unique positional information is added to input embeddings

1.5.2 Beyond NLP: Applications of Transformers

Transformers have transcended their initial application in Natural Language Processing (NLP) and are now being utilized across a variety of domains. A notable adaptation is in computer vision with the development of Vision Transformers (ViTs) as described by Dosovitskiy et al. [6]. ViTs treat image patches as sequence data, applying the transformer architecture to process these patches and achieve state-of-the-art results in image classification tasks. A typical example of the Vision Transformer is shown in Figure 5. This approach leverages the transformer's ability

to model long-range dependencies, providing a powerful alternative to convolutional neural networks (CNNs).

Furthermore, transformers have found applications in Reinforcement Learning (RL), where they enhance the learning process by improving context understanding and sequence processing, as explored by Agarwal et al. [7]. By integrating transformers into RL frameworks, agents can better manage temporal dependencies and long-term planning, leading to more efficient learning and superior performance in complex environments. The versatility of transformers in handling sequential data and their ability to model intricate relationships have made them a valuable tool across various machine learning tasks beyond NLP.

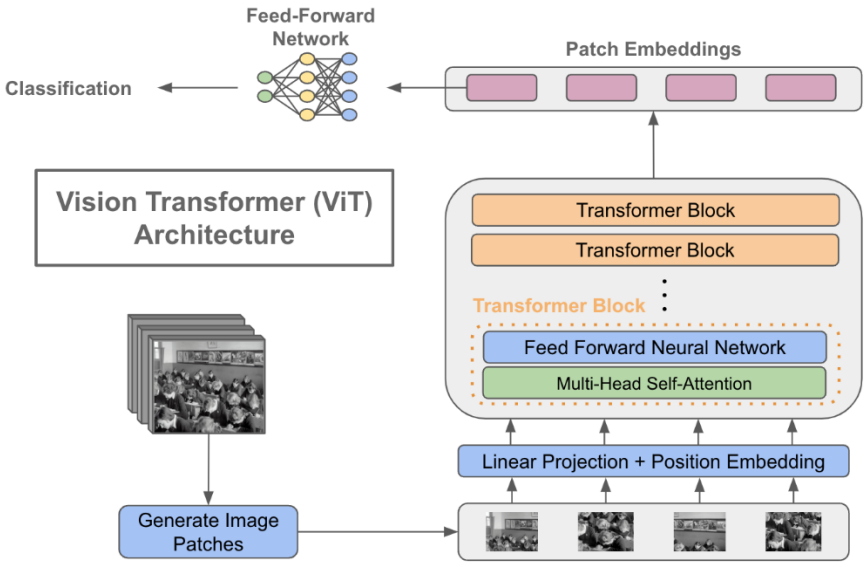

Figure 5: The structure of Vision Transformers architecture

1.6 Transfer Learning

Transfer Learning involves leveraging knowledge gained from one domain or task to improve learning performance in a different but related domain or task. This approach is particularly useful in scenarios where the target task has limited data, making it difficult to train a model

from scratch. By transferring knowledge from a pretrained model, we can achieve faster convergence and better performance. In the context of Reinforcement Learning (RL), transfer learning can dramatically reduce training time and enhance the effectiveness of algorithms, making it a crucial technique for modern AI applications.

1.6.1 Importance of Transfer Learning

Transfer Learning is important for several reasons:

- Efficiency: By utilizing pretrained models, Transfer Learning significantly reduces the amount of data and time required to train new models. This is especially beneficial in RL, where training from scratch can be time-consuming and computationally expensive.
- Performance: Transfer Learning can improve the performance of models on target tasks by incorporating knowledge learned from related tasks. This leads to more robust and accurate models, which are critical for applications like robotics, self-driving cars, and complex games.
- Overcoming Data Scarcity: In many real-world scenarios, obtaining a large amount of labeled data for training is challenging. Transfer Learning addresses this issue by allowing models to leverage existing data from similar tasks, thus mitigating the problem of data scarcity.
- Generalization: Models trained with Transfer Learning often generalize better to new tasks and environments, as they build upon a broader base of knowledge. This adaptability is crucial for applications that require models to perform well in diverse and dynamic settings.

1.4.2 Applications of Transfer Learning in RL

In Reinforcement Learning (RL), Transfer Learning has been applied in various ways to enhance the learning process:

- Pretrained Policy Networks: Initializing RL agents with policy networks pretrained on related tasks can significantly accelerate the learning process. As explored by Mini et al. [8], this approach enables agents to achieve higher performance more quickly, effectively leveraging the knowledge gained from previous tasks to improve efficiency in new environments.
- Feature Transfer: Transfer Learning allows for the reuse of features learned from one environment to another, which is particularly advantageous in visual-based RL tasks where feature extraction can be computationally intensive. Yin et al. [9] demonstrated the effectiveness of this approach in reducing the need to learn these features from scratch in new environments, thus saving computational resources and time.
- Reward Shaping: Transferring knowledge about reward structures from one task to another can help shape the reward function of the target task. This method, discussed by Doncieux [10], leads to more efficient learning and better alignment with desired

outcomes, enabling agents to quickly adapt to new tasks by utilizing prior knowledge about rewards.

The ability of Transfer Learning to enhance efficiency, performance, and generalization makes it a vital tool in the development of advanced RL systems. By providing a foundation for the rapid advancement and deployment of intelligent agents in diverse applications, Transfer Learning continues to drive progress in the field, enabling more robust and adaptable RL solutions.

1.7 Literature Review

1.7.1 Transfer Learning

The literature on transfer learning in deep reinforcement learning (DRL) provides substantial evidence of its impact on improving the efficiency and performance of reinforcement learning algorithms. A common theme across the reviewed papers is the utilization of transfer learning to address the challenges of data scarcity, computational expense, and lengthy training times in RL.

Firstly, the integration of transfer learning into DRL frameworks has been shown to enhance learning efficiency. By leveraging pretrained models or knowledge from related tasks, these methods significantly reduce the need for extensive data and computation, leading to faster convergence. For instance, Zhu et al. [11] highlight how pretrained models expedite the learning process by providing a robust starting point. This aligns with the goals of this thesis, which employs pretrained state and action embeddings to accelerate learning in new environments.

Secondly, the literature underscores the role of transfer learning in improving model performance. Transferring knowledge from previously learned tasks enables RL agents to achieve higher performance on target tasks. Gao et al. [12] discuss how this transfer of knowledge helps agents perform better by building on past experiences. In our approach, cross-attention mechanisms between pretrained state and action embeddings are employed to refine the learning process and achieve superior results.

Another significant aspect highlighted in the reviewed studies is the focus on generalization. Transfer learning methods enable RL agents to apply their knowledge to new tasks and domains, which is critical for developing adaptable and robust models. This thesis similarly emphasizes the ability of cross-attention mechanisms to enhance the generalization capabilities of RL agents, facilitating better adaptation to novel environments.

Furthermore, several studies explore the combination of transfer learning with other techniques to boost the learning process. For example, the use of rigorous simulation methods and multitask learning frameworks in conjunction with transfer learning has shown to improve overall learning outcomes. Parisotto et al. [13] illustrate how combining these techniques leads to more effective learning. Our methodology incorporates a cross-attention transformer architecture to integrate pretrained embeddings, leveraging the strengths of both transfer learning and advanced neural network architectures.

In conclusion, the reviewed literature supports the effectiveness of transfer learning in enhancing the efficiency, performance, and generalization of reinforcement learning algorithms.

The commonalities between these studies and our thesis highlight the relevance and potential of using cross-attention mechanisms between pretrained state and action embeddings to improve RL outcomes in new environments.

1.7.2 Action Representations

Action representations in deep reinforcement learning (DRL) have been attracting significant attention in enhancing learning efficiency and overall performance. One notable example is provided by Chen et al. [14], who propose a method for learning action embeddings that enable the transfer of knowledge across tasks with differing state-action spaces. Their innovative approach not only creates informative action embeddings but also significantly accelerates the policy learning process, leading to notable improvements across a variety of tasks. This method addresses the challenges associated with task variability and demonstrates the potential for more efficient policy learning in diverse environments.

Similarly, Chandak et al. [15] highlight the substantial benefits of learning action representations. By decomposing a policy into components that operate in low-dimensional action spaces, their technique significantly enhances generalization over large sets of actions. This decomposition allows reinforcement learning agents to infer the outcomes of actions that are similar to those previously taken, thereby improving their performance in real-world scenarios.

Furthermore, Kim et al. [16] present an action-driven contrastive representation method that prioritizes essential features for decision-making while disregarding irrelevant details. This approach, which is particularly effective in environments such as Atari and OpenAI ProcGen benchmarks, substantially boosts both sample efficiency and generalization. By focusing on the

critical features necessary for action decisions, this method ensures that the learning process is more robust and adaptable to new observations, thereby enhancing the agent's overall performance and generalization capabilities.

Moreover, the work of Fujimoto et al. [17] introduces SALE, a novel approach for learning state-action embeddings that model the intricate interaction between state and action. This method significantly enhances the performance of continuous control algorithms. By integrating SALE into the TD3 algorithm, resulting in the development of the TD7 algorithm, they achieve substantial performance gains on OpenAI gym benchmark tasks. The success of SALE in improving the TD7 algorithm highlights the potential of advanced state-action representations to dramatically enhance learning efficiency and outcomes in DRL. The modeling of state-action interactions allows for more precise and effective policy learning, particularly in continuous control tasks.

The reviewed literature robustly supports the effectiveness of action representations in enhancing the efficiency, performance, and generalization of reinforcement learning algorithms. It aligns with the goals of this thesis, which aims to employ advanced state and action embeddings to improve RL outcomes in Atari environments.

2 CHAPTER 2: Atari Games in the Gym Library

The OpenAI Gym library provides a rich set of environments designed to develop and evaluate reinforcement learning (RL) algorithms. Among these environments, the collection of Atari 2600 games has become a standard benchmark for testing and comparing the performance of RL agents. This chapter provides an overview of the Atari games available in the Gym library [26], with a focus on three specific games: Pac-Man, Alien, and Bank Heist. We will compare and evaluate their similarities and differences, highlighting the unique challenges and features each game presents.

2.1 Overview of Atari Games in the Gym Library

The Atari 2600 games provided by the Gym library serve as an ideal testbed for RL research due to their diverse gameplay mechanics, varying levels of complexity, and the requirement for strategic decision-making. These games range from simple single-screen action games to more complex multi-level adventures. Each game offers distinct challenges in terms of state representation, action space, and reward structures, making them suitable for evaluating different aspects of RL algorithms.

Some popular Atari games included in the Gym library are:

- **Breakout**: A game where the player controls a paddle to bounce a ball and break bricks.
- **Space Invaders**: A shooting game where the player must defend against waves of alien invaders.
- **Pong**: A table tennis simulation where the player competes against an AI opponent.
- **Frostbite**: A game where the player builds igloos while avoiding enemies.

Among these, we will focus on Pac-Man, Alien, and Bank Heist due to their popularity and the unique challenges they present.

2.2 Description of Pac-Man

Pac-Man is a classic arcade game where the player controls Pac-Man through a maze, eating pellets while avoiding ghosts, as shown in Figure 6. The primary objectives of Pac-Man are:

- Collecting Pellets: Pac-Man must eat all the pellets in the maze to advance to the next level.
- Avoiding Ghosts: Four ghosts (Blinky, Pinky, Inky, and Clyde) roam the maze and attempt to capture Pac-Man. If caught, Pac-Man loses a life.
- Eating Power Pellets: Eating power pellets gives Pac-Man temporary invincibility, allowing him to eat the ghosts for extra points.
- Fruit Bonuses: Fruits periodically appear near the center of the maze and can be eaten for extra points.

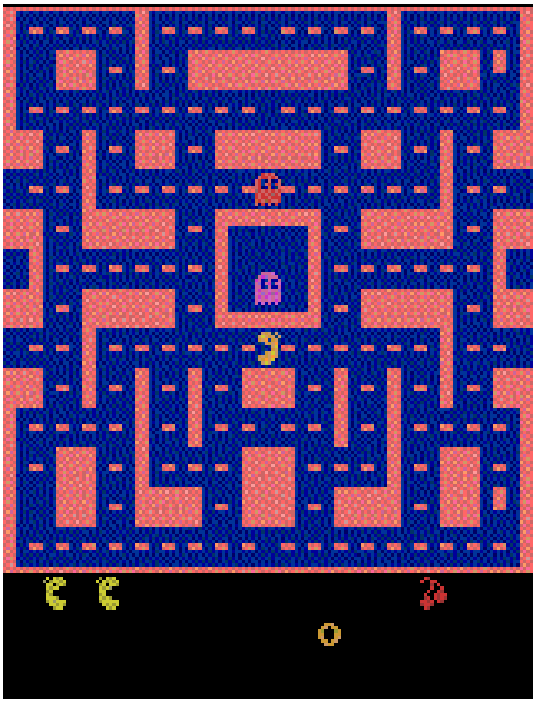

Figure 6: The maze layout of Pac-Man, illustrating the pathways and locations of pellets and ghosts [26]

Possible Actions:

- NOOP: No operation; Ms. Pac-Man remains stationary.
- UP: Directs Ms. Pac-Man to move upwards in the maze.
- RIGHT: Directs Ms. Pac-Man to move right in the maze.
- LEFT: Directs Ms. Pac-Man to move left in the maze.
- DOWN: Directs Ms. Pac-Man to move downwards in the maze.
- UPRIGHT: Directs Ms. Pac-Man to move diagonally up and to the right.
- UPLEFT: Directs Ms. Pac-Man to move diagonally up and to the left.
- DOWNRIGHT: Directs Ms. Pac-Man to move diagonally down and to the right.
- DOWNLEFT: Directs Ms. Pac-Man to move diagonally down and to the left.

Points and Bonuses:

- Small Pellets: 10 points each.
- Power Pellets: 50 points each.
- Ghosts: Eating ghosts after consuming a power pellet gives points in a progressive manner (200, 400, 800, 1600).
- Fruits: Points vary by fruit type and level, ranging from 100 to 5000 points.
    - Strawberry: 200 points.
    - Pretzel: 700 points.
    - Apple: 1000 points.
    - Pear: 2000 points.
    - Banana: 5000 points.

Key features of Pac-Man include:

- Maze Navigation: The game involves navigating through a fixed maze with various pathways and dead ends.
- Strategic Planning: Players must devise strategies to collect pellets efficiently while avoiding or confronting ghosts.
- Dynamic Difficulty: The behavior of the ghosts changes as the player progresses, increasing the game's difficulty.

Pac-Man's combination of navigation, evasion, and occasional aggression provides a rich environment for testing RL algorithms.

2.3 Description of Alien

Alien is a survival-based game where the player navigates through different levels, avoiding or destroying aliens, as shown in Figure 7. The primary objectives of Alien are:

- Surviving Alien Attacks: Players must avoid or eliminate aliens that appear on the screen.
- Navigating Levels: Each level has a different layout, requiring players to adapt their strategies accordingly.
- Collecting Power-ups: Power-ups may appear, providing temporary advantages such as increased firepower or invincibility.

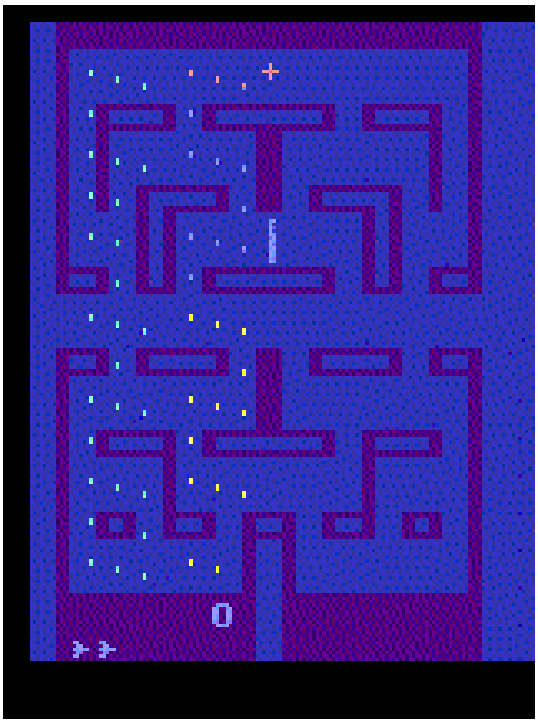

Figure 7: A gameplay screen from Alien, showing the player character, enemies, and level layout [26]

Possible Actions:

- NOOP: No operation; the player character remains stationary.
- FIRE: Fires a weapon in the direction the player is facing.
- UP: Directs the player character to move upwards in the level.
- RIGHT: Directs the player character to move right in the level.
- LEFT: Directs the player character to move left in the level.
- DOWN: Directs the player character to move downwards in the level.
- UPRIGHT: Directs the player character to move diagonally up and to the right.
- UPLEFT: Directs the player character to move diagonally up and to the left.
- DOWNRIGHT: Directs the player character to move diagonally down and to the right.
- DOWNLEFT: Directs the player character to move diagonally down and to the left.
- UPFIRE: Moves up and fires simultaneously.
- RIGHTFIRE: Moves right and fires simultaneously.
- LEFTFIRE: Moves left and fires simultaneously.
- DOWNFIRE: Moves down and fires simultaneously.
- UPRIGHTFIRE: Moves diagonally up-right and fires simultaneously.
- UPLEFTFIRE: Moves diagonally up-left and fires simultaneously.
- DOWNRIGHTFIRE: Moves diagonally down-right and fires simultaneously.
- DOWNLEFTFIRE: Moves diagonally down-left and fires simultaneously.

Points and Bonuses:

- Eggs: 10 points each.
- Pulsar: 100 points.
- 1st Alien: 500 points.
- 2nd Alien: 1,000 points.
- 3rd Alien: 2,000 points.
- Completed Screen: 1 point.
- Rocket: 500 points.
- Saturn: 1,000 points.
- Star Ship: 2,000 points.
- 1st Surprise: 2,000 points.
- 2nd Surprise: 3,000 points.
- 3rd Surprise: 5,000 points.

Key features of Alien include:

- Level Variety: Each level introduces new challenges and enemy types, requiring continuous adaptation.
- Combat Mechanics: The game involves a significant amount of shooting and dodging, testing the player's reflexes and tactical planning.
-

- Resource Management: Players must manage their limited ammunition and power-ups effectively.

Alien's emphasis on survival, level navigation, and resource management makes it a challenging environment for RL agents.

2.4 Description of Bank Heist

Bank Heist is a strategy game where the player controls a character attempting to rob banks while avoiding police capture, as shown in Figure 8. The primary objectives of Bank Heist are:

- Robbing Banks: The player must enter banks, collect money, and escape without getting caught.
- Avoiding Police: Police patrol the streets and attempt to capture the player.
- Resource Collection: Players can collect keys to open banks and other useful items.

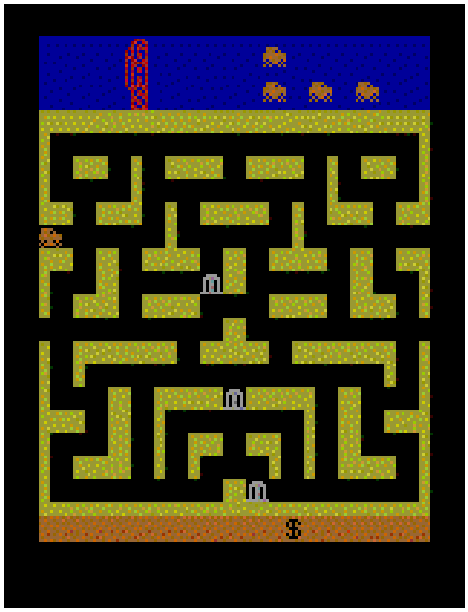

Figure 8: The city map in Bank Heist, displaying banks, streets, and police patrols [26]

Possible Actions:

- NOOP: No operation; the player character remains stationary.
- FIRE: Fires a weapon in the direction the player is facing.
- UP: Directs the player character to move upwards in the level.
- RIGHT: Directs the player character to move right in the level.
- LEFT: Directs the player character to move left in the level.
- DOWN: Directs the player character to move downwards in the level.
- UPRIGHT: Directs the player character to move diagonally up and to the right.

- UPLEFT: Directs the player character to move diagonally up and to the left.
- DOWNRIGHT: Directs the player character to move diagonally down and to the right.
- DOWNLEFT: Directs the player character to move diagonally down and to the left.
- UPFIRE: Moves up and fires simultaneously.
- RIGHTFIRE: Moves right and fires simultaneously.
- LEFTFIRE: Moves left and fires simultaneously.
- DOWNFIRE: Moves down and fires simultaneously.
- UPRIGHTFIRE: Moves diagonally up-right and fires simultaneously.
- UPLEFTFIRE: Moves diagonally up-left and fires simultaneously.
- DOWNRIGHTFIRE: Moves diagonally down-right and fires simultaneously.
- DOWNLEFTFIRE: Moves diagonally down-left and fires simultaneously.

Points and Bonuses:

- Bank Values:
  - First Bank: $10.00
  - Second Bank: $20.00
  - Third Bank: $30.00
  - Fourth Bank: $40.00
  - Fifth Bank: $50.00
  - Sixth Bank: $60.00
  - Seventh Bank: $70.00
  - Eighth Bank: $80.00
  - Ninth Bank: $90.00
- Cop Car Values:
  - One Cop Car in Pursuit: $10.00
  - Two Cop Cars in Pursuit: $30.00
  - Three Cop Cars in Pursuit: $50.00
- Bonus Points for Level 1:
  - Bankersfield: $93.00
  - Silver Dollar: $186.00
  - Flat Broke: $279.00
  - Heistown: $372.00
- Add $372.00 per level to these values for the other levels.

Key features of Bank Heist include:

- Open-World Navigation: The game features a city map with multiple banks and hiding spots.
- Stealth and Strategy: Players must use stealth and strategic planning to avoid police and successfully rob banks.
- Dynamic Enemies: Police behavior changes based on the player's actions, increasing the game's complexity.

Bank Heist's combination of open-world exploration, stealth, and strategic planning offers a unique challenge for RL algorithms.

2.5 Comparison of Pac-Man, Alien, and Bank Heist

While Pac-Man, Alien, and Bank Heist are all part of the Atari 2600 game collection, they present distinct gameplay mechanics and challenges:

Similarities:

- Objective-Driven Gameplay: All three games have clear objectives (collecting pellets, surviving attacks, robbing banks) that guide the player's actions.
- Obstacle Avoidance: Each game requires players to avoid enemies (ghosts, aliens, police) to succeed.
- Resource Management: Players must manage resources (power pellets, ammunition, keys) effectively to achieve their goals.

Differences:

- Game Structure:
  - Pac-Man features a fixed maze with repetitive structure across levels.
  - Alien offers varied levels with different layouts and increasing difficulty.
  - Bank Heist provides an open-world environment with a city map and multiple objectives.
- Gameplay Mechanics:
  - Pac-Man focuses on navigation and evasion within a confined space.
  - Alien emphasizes survival and combat, requiring quick reflexes and tactical decisions.
  - Bank Heist combines stealth and strategy, involving careful planning and resource collection.
- Enemy Behavior:
  - In Pac-Man, ghosts have predictable patterns but can adapt to the player's actions.
  - In Alien, enemies appear and move dynamically, presenting continuous threats.
  - In Bank Heist, police patrols react to the player's actions, necessitating strategic evasion.

2.6 Evaluation of Similarities and Differences

The comparison of Pac-Man, Alien, and Bank Heist highlights the diversity of challenges and gameplay mechanics these games offer. This diversity makes them suitable for evaluating different aspects of RL algorithms:

- Pac-Man: Ideal for testing navigation, strategy, and dynamic adaptation to enemy behavior within a structured environment.

- Alien: Suitable for evaluating survival tactics, combat efficiency, and adaptability to varied level designs.
- Bank Heist: Excellent for assessing strategic planning, stealth, and resource management in an open-world setting.

Each game presents unique opportunities and challenges for RL research, making them valuable benchmarks for developing and testing advanced RL algorithms. By comparing these games, we gain insights into the strengths and limitations of various RL approaches, ultimately guiding the development of more robust and efficient algorithms.

By understanding the similarities and differences among these games, we can better design and evaluate RL algorithms tailored to specific types of challenges, enhancing their applicability and performance across diverse environments.

3 CHAPTER 3: Methodology

This chapter details the methodology employed in our study, focusing on the application of Deep Q-Networks (DQN) to the Ms. Pac-Man game and the new proposed algorithm JASE-DQN. An in-depth description of the preprocessing steps, network architecture, and hyperparameters used in our experiments is provided.

3.1 DQN on Ms. Pac-Man

The DQN algorithm serves as the primary benchmark for this analysis. Initially developed by DeepMind, DQN has demonstrated significant success in various Atari games, making it an ideal candidate for our study on Ms. Pac-Man.

3.1.1 Pre-Processing of states

Preprocessing the states is a crucial step to ensure the input data is manageable and relevant for the learning algorithm. For the Ms. Pac-Man game, the following preprocessing steps were applied:

- Resizing and Grayscaling: The observation space was transformed from its original dimensions of (210, 160, 3) to (84, 84) by converting the image to grayscale and downsampling. This reduces the complexity of the input data while retaining essential features for gameplay.
- Normalization: All frames were normalized by dividing pixel values by 255, scaling the values to the range [0, 1]. This standardization helps in stabilizing and accelerating the learning process.
- Frame Skipping: By default, 4 frames were skipped when an action was selected. This technique reduces the frequency of actions and focuses on significant state changes.
- Stacking Frames: Four sequential frames were stacked to form the input state, providing temporal context to the agent. This method, introduced in DeepMind's paper, helps the agent understand the motion and dynamics within the game.

3.1.2 Network Architecture

The network architecture for the DQN used in Ms. Pac-Man is designed to process the input frames and output Q-values for each possible action, as shown in Figure 9.

3.1.2.1 Deep Q-Network

The architecture consists of the following layers:

- Input Layer: The input consists of 4 stacked frames from the Ms. Pac-Man game, with dimensions typically being (4, Height, Width), where Height and Width are determined by the game resolution (84, 84 in this case).

- Conv2D Layer 1: This convolutional layer has 32 channels, with a kernel size of 8x8, stride 4x4, and ReLU activation function. It processes the input frames to generate a feature map.
- Conv2D Layer 2: This layer has 64 channels, with a kernel size of 4x4, stride 2x2, and ReLU activation function. It further processes the feature maps produced by the first convolutional layer.
- Conv2D Layer 3: This layer has 64 channels, with a kernel size of 3x3, stride 1x1, and ReLU activation function. It continues processing the feature maps from the second convolutional layer.
- Flatten Layer: The 3D output from the last convolutional layer is flattened into a 1D array, preparing it for the fully connected layers.
- Dense Layer: A fully connected layer with 512 neurons and ReLU activation function processes the flattened array to learn non-linear combinations of high-level features.
- Output Layer: This fully connected layer has 9 neurons, corresponding to the 9 possible actions in the Ms. Pac-Man game. It outputs the Q-values for each action.

This architecture yields a total of 3,377,490 trainable parameters.

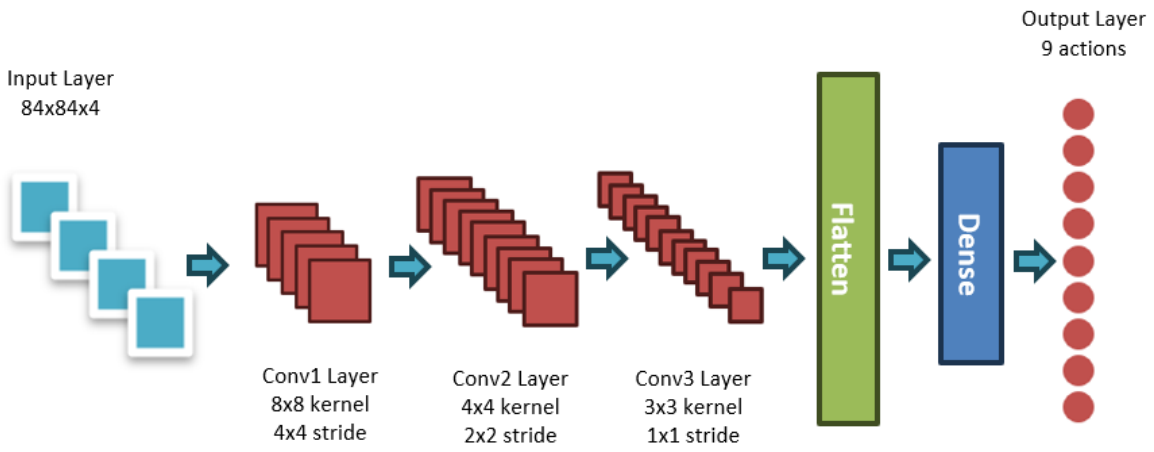

Figure 9: Deep Q-Network Architecture

3.1.2.2 Double Q Learning

In the context of Deep Q-Networks (DQN), the use of online and target networks has proven to be a critical advancement according to Van Hasselt et al. [18]. This dual-network approach offers several benefits that enhance the stability and efficiency of the learning process.

Primarily, the target network addresses the moving target problem inherent in Q-learning by providing a stable reference for generating target Q-values. This stability is crucial for mitigating the risk of divergence or oscillation in Q-values, which can otherwise destabilize the learning process. By periodically updating the target network to match the online network, we ensure that the learning remains smooth and converges more reliably.

Moreover, the use of target networks significantly reduces overestimation bias, a common issue in Q-learning, leading to more accurate policy decisions. This is especially beneficial when

combined with techniques such as Double DQN, which further separate action selection and evaluation to refine the learning process. Additionally, the stability provided by the target network enhances learning efficiency, reducing the number of updates required and thereby accelerating convergence. This efficiency is particularly valuable in complex environments with high-dimensional state and action spaces, where stable targets enable the agent to develop more sophisticated strategies.

Overall, the inclusion of online and target networks in our methodology not only improves the stability and efficiency of the learning process but also ensures robust performance across different tasks, aligning perfectly with the objectives of our research.

3.1.2.3 Experience Replay

In addition, a replay buffer is utilized to effectively mitigate the issue of catastrophic forgetting. A replay buffer stores a diverse set of past experiences (an experience is considered a quadruple of state, action, reward, next state), which are randomly sampled during training. This approach ensures that the agent learns from a broad range of states and actions, rather than overfitting to recent experiences.

By maintaining a diverse and representative sample of the environment's dynamics, the replay buffer helps in stabilizing the learning process and promotes more robust policy development. This method not only enhances the stability and efficiency of the learning process but also ensures that the agent retains valuable knowledge over time, preventing the rapid loss of previously acquired skills that can occur in sequential learning without such a mechanism as indicated by Wang, Z. et al. [21].

Advancements in replay buffer techniques, such as Prioritized Experience Replay (PER) developed by Schaul, T. et al. in the paper “Prioritized experience replay”, offer additional benefits. PER prioritizes experiences based on their significance, often measured by the

magnitude of their temporal-difference (TD) error. This prioritization allows the agent to focus on learning from the most informative experiences, leading to faster and more efficient learning. By sampling important experiences more frequently, PER enhances the overall performance and robustness of reinforcement learning models.

However, for simplicity, a simple replay buffer is used in our framework. While PER provides significant advantages, a simple replay buffer is easier to implement and still effectively prevents catastrophic forgetting by maintaining a diverse set of experiences for the agent to learn from. This approach ensured the stability and efficiency necessary for our research objectives.

3.1.2.4 Optimizer

The Adam optimizer, introduced by Kingma and Ba [19], is utilized for updating the weights of the network due to its numerous advantages, making it a popular choice for training neural networks.

First, it combines the benefits of two other popular optimizers, AdaGrad and RMSProp, by using adaptive learning rates for each parameter. This adaptivity results in faster convergence and greater stability during training, as Adam adjusts the learning rates based on the gradients' magnitudes.

Second, Adam maintains separate learning rates for each parameter, which can lead to improved performance on models with sparse gradients or in high-dimensional spaces. Additionally, Adam efficiently computes the moving averages of gradients and their squared gradients, which helps in handling noisy or sparse gradients typical in complex models.

Overall, these features make Adam a robust choice for optimizing neural networks, offering quicker convergence, better handling of varied learning rates, and improved performance across a range of tasks.

3.1.2.5 Loss function

SmoothL1Loss is used as the loss function of this network as it offers significant benefits, particularly in scenarios where outlier data points could otherwise distort training. Unlike Mean Squared Error (MSE), SmoothL1Loss is less sensitive to outliers due to its quadratic and linear components, which prevent it from magnifying errors as drastically as MSE does. It combines the advantages of both L1 and L2 loss because it behaves like the L2 loss when the absolute error is small (less than 1) and like the L1 loss when the absolute error is large. SmoothL1Loss is defined as follows:

$$SmoothL1Loss(x) = \begin{cases} 0.5x^2, & if\ |x| < 1 \\ |x| - 0.5, & otherwise \end{cases}$$

This characteristic not only aids in enhancing the stability of training but also reduces the likelihood of encountering exploding gradients (Girshick, 2015) [20] a common issue in deep learning when gradients become excessively large and disrupt the learning process. By smoothing out extreme errors, SmoothL1Loss promotes stable gradient values, thereby enhancing the overall training stability and convergence of your neural network.

3.1.3 Hyperparameters

The following hyperparameters were used in the DQN training process:

- Gamma (Discount Factor): 0.9
- Exploration Rate (Epsilon): Initially set to 1.0

- Exploration Rate Decay: 0.99999
- Minimum Exploration Rate: 0.1
- Replay Buffer Size: 100,000
- Batch Size: 32
- Number of Episodes: 5,000
- Learning Rate: 0.00025
- Online Network Update Frequency: Every 3 new experiences
- Target Network Update Frequency: Every 10,000 experiences
- Minimum Number of Experiences Before Training: 10,000

These hyperparameters were tailored to the specific requirements of the Ms. Pac-Man game.

3.2 Training State & Action Encoders

In this section, we describe the methodology for training state and action encoders, essential components for enhancing the efficiency and effectiveness of reinforcement learning through intrinsic motivation.

3.2.1 The Intrinsic Curiosity Module

The Intrinsic Curiosity Module (ICM) is a framework designed to improve exploration in reinforcement learning by incorporating intrinsic motivation. Intrinsic curiosity is driven by the agent's own experience and the desire to reduce prediction errors about its environment, rather than relying solely on extrinsic rewards provided by the environment. Figure 10 presents the entire mechanism of this framework. This approach helps the agent explore more efficiently, particularly in environments with sparse or deceptive rewards. We will describe the main components of ICM module, but we encourage you to learn more about this module in Brown and Zai's book Deep Reinforcement Learning in Action [23].

The ICM consists of several components:

Forward-Prediction Model

The forward-prediction model, f, is defined as:

$$f: (S_t, a_t) \rightarrow \widehat{S_{t+1}}$$

Here, S_t is the state at time t, a_t is the action taken, and S_{t+1} is the predicted next state. This model predicts the next state based on the current state and the action performed.

Inverse Model

The inverse model, g, is defined as:

$$g: (S_t, S_{t+1}) \rightarrow \widehat{a_t}$$

This function takes a state S_t and the subsequent state S_{t+1}, and returns a prediction for the action $\widehat{a_t}$ that led to the transition from S_t to S_{t+1}.

Encoder Model

On its own, the inverse model is not particularly useful. It is tightly coupled with the encoder model, denoted φ.

The encoder function φ is defined as:

$$\varphi: S_t \rightarrow \widetilde{S_t}$$

This function takes a state S_t and returns an encoded state $\widetilde{S_t}$, where the dimensionality of $\widetilde{S_t}$ is significantly lower than the raw state S_t. The encoded state $\widetilde{S_t}$ captures the essential features of the original state in a more compact form.

Coupled Models

With the encoder model, the forward model and the inverse model are reformulated to work on the encoded states. The forward model becomes a function that predicts the encoded next state:

$$f: \varphi(S_t) \times a_t \rightarrow \varphi(\hat{S}_{t+1})$$

Here, $\varphi(\hat{S}_{t+1})$ refers to the prediction of the encoded next state. Similarly, the inverse model is reformulated to predict the action based on the encoded states:

$$g: \varphi(S_t) \times \varphi(S_{t+1}) \rightarrow \widehat{a_t}$$

These reformulations ensure that the predictions and actions are based on the more compact and informative encoded states rather than the raw states.

The combination of these models within the ICM framework facilitates the training of state encoder. By focusing on intrinsic curiosity, the agent learns to explore its environment more thoroughly and efficiently, driven by the need to reduce the prediction error of its own internal models. This approach enhances the agent's ability to generalize knowledge across different environments and tasks, ultimately leading to more robust and effective learning outcomes.

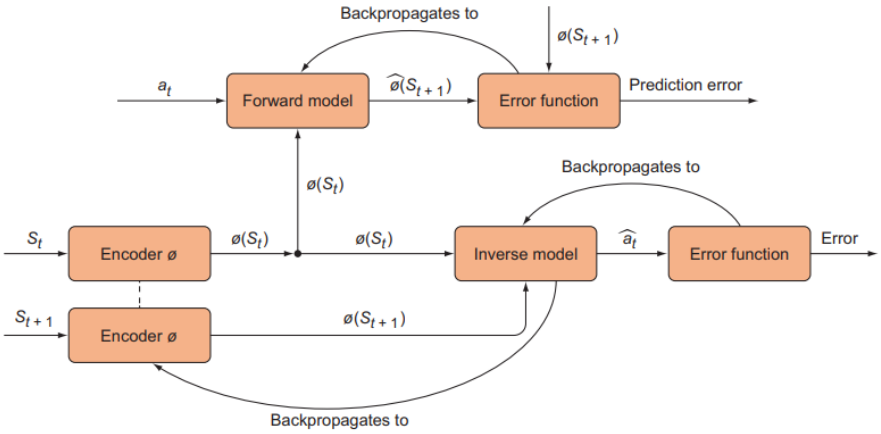

Figure 10: Diagram of Intrinsic Curiosity Module [19]

3.2.2 Alternatives

In this section, we explore alternatives to the standard DQN algorithm by incorporating encoded states and actions (embeddings) into the Q-Learning algorithm. Our goal is to create a more efficient learning process by utilizing these embeddings. Specifically, we aim to:

- Use the encoded states (embeddings) in the Q-Learning algorithm.
- Create action embeddings to represent each action of the game.
- Train the state encoder and action encoder in an offline manner by gathering experiences of DQN.

3.2.2.1 State Encoder & Inverse Model

The state encoder and inverse model are essential components for embedding the states. The state encoder is implemented as a series of convolutional layers, batch normalization, max-pooling, and dropout layers to effectively reduce the dimensionality of the input states and capture relevant features. A sequence of these layers together form a convolutional block, as per Figure 11. Below is a detailed explanation of the architecture:

1. **Convolutional Block 1**:
    - Two convolutional layers each have 64 filters, a kernel size of 3x3, and a stride of 1. ReLU activation function is applied to both convolutional layers. These layers help in feature extraction from the input state.
    - Batch Normalization Layer to stabilize and speed up the training process by normalizing the inputs of each layer.
    - Max-pooling layer with a pool size of 2x2 are used to downsample the feature maps, reducing their spatial dimensions and computational load.
    - Dropout layer with a dropout rate of 0.1 is used to prevent overfitting by randomly dropping units during training.
2. **Convolutional Block 2:**
    - Same as Convolutional Block 1

3. **Convolutional Block 3:**
    - Same as Convolutional Block 1, but with a kernel size 2x2 in convolutional layers.
4. **Convolutional Block 4:**
    - Same as Convolutional Block 3, but with 128 filters in convolutional layers.

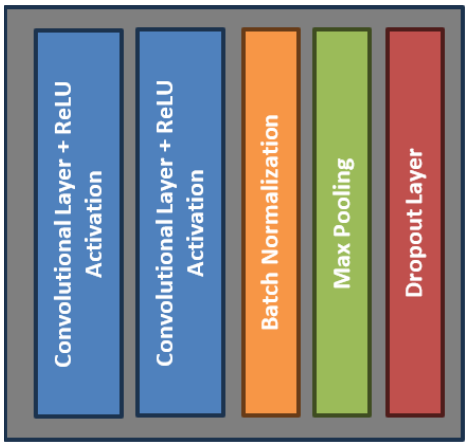

Figure 11: Convolutional Block of State Encoder

The output of the state encoder is an embedding vector of 1152 elements, which is created by adding a flatten layer after the 4 convolutional blocks, as per Figure 12.

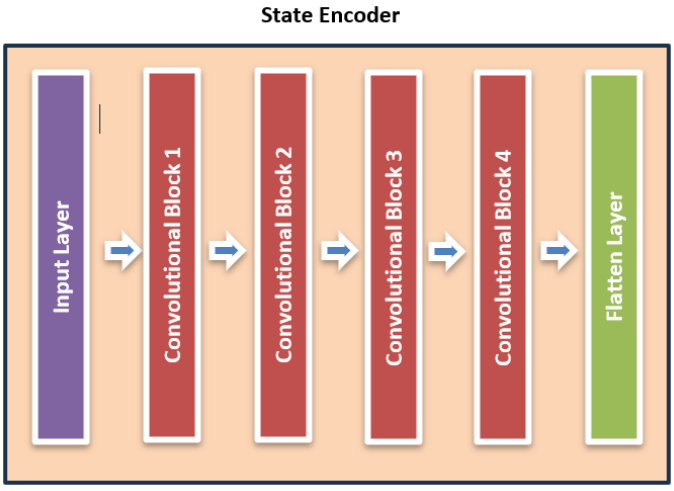

Figure 12: State Encoder

The inverse model is designed to predict the action taken between two consecutive state embeddings, as shown in Figure 13. It includes:

1. **Encoder**:
   - The encoder model, is used to encode the states into a lower-dimensional representation (embedding).
2. **Fully Connected Layers**:
   - The inverse model consists of fully connected layers that take the concatenated encoded states as input and predict the action taken. The layers are configured with 512 neurons and a final layer corresponding to the number of possible actions.

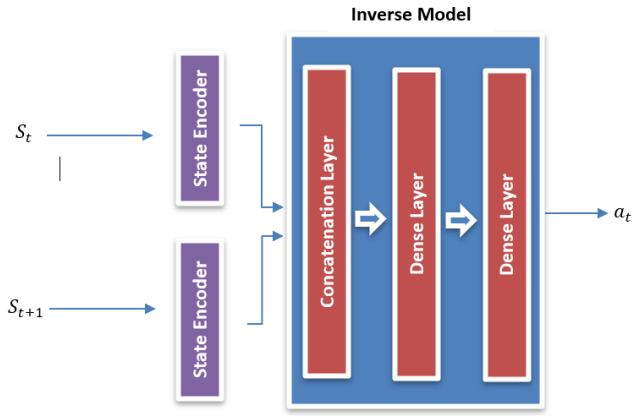

Figure 13: Inverse Model

Training the State Encoder by Inverse Model

As described earlier, training of the state encoder is done by the inverse model. The training of the inverse model involves the following steps:

1. **Loss Function**: Cross Entropy Loss is used, which is suitable for classification tasks where the model predicts a probability distribution over classes (actions in this case).
2. **Optimizer**: Adam optimizer is employed with a learning rate of 10^{-4} and a weight decay of 0.01. Adam is an adaptive learning rate optimization algorithm that is efficient for large datasets and high-dimensional parameter spaces.
3. **Learning Rate Scheduler**: A linear learning rate scheduler, is used to adjust the learning rate periodically. The step size is set to 30 epochs, and the learning rate is reduced by a factor of 0.8 every 30 epochs. This helps in fine-tuning the learning rate and preventing it from becoming too high or too low during training.

3.2.2.2 Action Encoder & Forward Model

The action encoder and forward model work together to predict the next state given the current state and action, as shown in Figure 14. The action encoder is an embedding layer, which helps represent each action in a continuous vector space.

1. **Action Encoder**:
   - The action encoder is implemented as an embedding layer within the forward model. This layer converts discrete actions into continuous embeddings, which can then be used in conjunction with state embeddings.
2. **State Encoder**:
   - The state encoder, is pre-trained and loaded with its trained weights.
3. **Fully Connected Layers**:
   - The forward model uses fully connected layers to combine the state and action embeddings. The input to the first layer is the concatenated state and action embeddings, followed by layers of 1152 neurons that refine this combined representation.

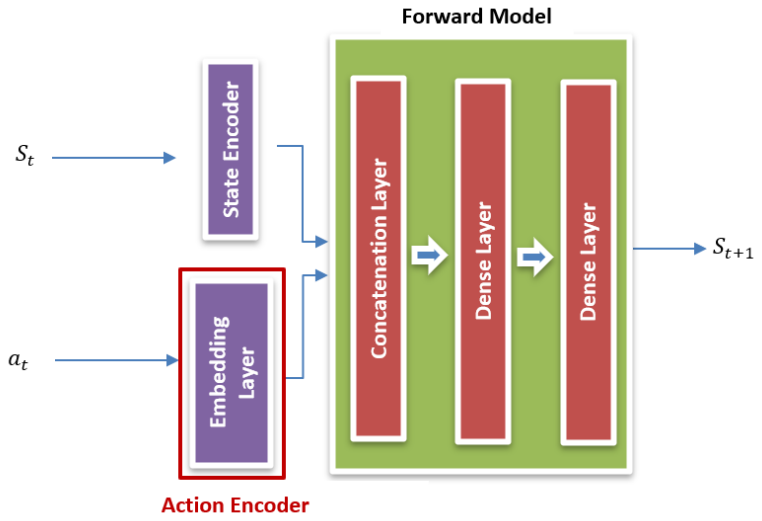

Figure 14: Action Encoder and Forward Model

Training the Action Encoder by Forward Model

Training of the action encoder is done by the forward model. The training of the forward model involves the following steps:

1. **Loss Function**: Smooth L1 Loss is used, which is suitable for regression tasks and is not sensitive to outliers.
2. **Optimizer**: Adam optimizer is employed with a learning rate of 10^{-6} and a weight decay of 0.05.

These architectural choices enable the effective training of state and action encoders, which can be utilized in the Q-Learning algorithm to enhance learning efficiency and speed. By incorporating these embeddings, we aim to generalize the learned knowledge across different environments and tasks more effectively.

3.3 JASE-DQN

This section introduces the JASE-DQN algorithm, which aims to enhance the traditional DQN framework by incorporating cross-attention mechanisms between states and actions to compute the expected rewards (Q-values) of state-action pairs. This approach is able to leverage pre-trained state and action encoders to utilize "previous knowledge" from either the same or different environments, thereby improving the learning efficiency and effectiveness.

3.3.1 Architecture

The architecture of the JASE-DQN consists of three primary components: a state encoder, an action encoder, and an attention mechanism, as per Figure 15. A detailed description of the architecture is provided below:

Input:

- The algorithm receives four stacked frames from the environment as input, representing the current state. These frames are similar to those used in the standard DQN algorithm, capturing temporal context and motion information essential for understanding the environment dynamics.

State Encoder:

- The input state is transformed into an embedding via the state encoder. The architecture of the state encoder aligns with that described in Section 3.2.2.1 (State Encoder & Inverse Model), featuring multiple convolutional layers designed to extract spatial features from the raw input frames.
- The sequence of layers includes:
  - **Conv2D Layer 1**: 64 channels, kernel size of 3x3, and stride of 1.
  - **Conv2D Layer 2**: 64 channels, kernel size of 3x3, and stride of 1.
  - **Batch Normalization Layer**: Applied after each convolutional layer to stabilize the learning process.
  - **Max Pooling Layer**: Reduces spatial dimensions by taking the maximum value over 2x2 regions.
  - **Dropout Layer**: Applied with a probability of 0.1 to prevent overfitting.
- This architecture is repeated with varying configurations, progressively reducing spatial dimensions and increasing the depth of feature maps until a final encoded state representation is obtained.

Action Encoder:

- The action encoder is represented by an embedding matrix of size 9×256, where each of the 9 possible actions is encoded into a 256-dimensional vector. This matrix functions as a lookup table, with each row corresponding to the embedding of a specific action.
- This method allows for abstract representation of actions in a high-dimensional space, facilitating nuanced distinctions based on their effects on the environment.

Cross-Attention Mechanism:

- The encoded state and action embeddings are processed through multiple attention mechanisms to compute the expected rewards of state-action pairs.
- The cross-attention mechanism functions as follows:
    - **Attention Modules**: Nine cross-attention modules, each configured with query (q), key (k), and value (v) dimensions of 512, 512, and 100, respectively, are employed. Layer normalization is applied to the outputs to ensure stable gradients and efficient learning.
        - The attention mechanism enables the model to focus on different aspects of the state-action pairs, learning the dependencies and interactions between them.
    - **Concatenation**: The output vectors from the cross-attention modules are concatenated to form a comprehensive representation. This aggregated vector captures a holistic view of the state-action interactions.
- **Dense Layer**: A fully connected layer with 512 neurons processes the concatenated vector, learning complex, non-linear relationships between the state and action embeddings, further refining the state-action value estimations.
- **Output Layer**: The final fully connected layer consists of 9 neurons, each corresponding to one of the possible actions in the Ms. Pac-Man game. This layer outputs the Q-values for each action, representing the expected future rewards for taking each action in the given state.

By integrating the attention mechanism, the JASE-DQN algorithm extends the traditional DQN framework, incorporating action representations in q values calculation and also the ability to utilize prior knowledge through pre-trained state and action encoders. This architecture results in a total of 7,432,145 trainable parameters, with 7,184,593 generated by the Q-network, 245,248 by the state encoder, and 2,304 by the action embeddings.

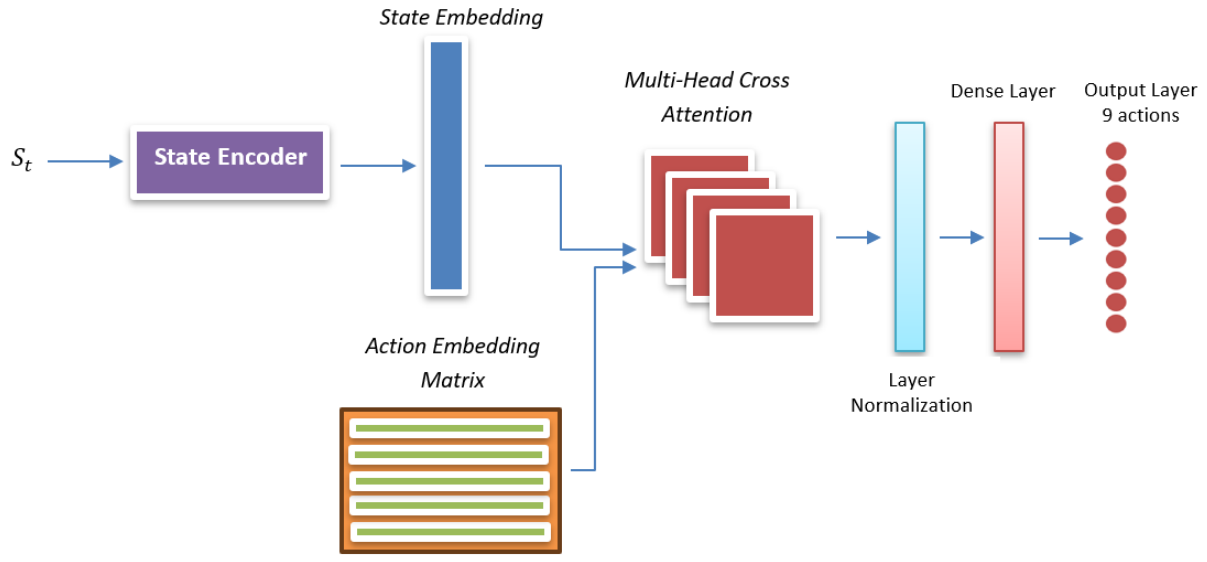

Figure 15: JASE-DQN Architecture

The Double Q-Learning approach was utilized similarly to the standard DQN as well as the experience replay mechanism. The same data pre-processing of states as DQN is applied before feeding states to state encoder. Additionally, SmoothL1Loss loss function and Adam optimizer are selected for training the algorithm, similarly with DQN. This ensures that the observed differences in performance are attributable to the algorithmic enhancements rather than variations in these fundamental components. However, due to the architectural differences between the networks, certain hyperparameters, such as the learning rate, were adjusted to optimize the performance of the JASE-DQN algorithm. These changes were necessary to accommodate the unique requirements and characteristics of the network, ensuring stable and efficient learning.

3.3.2 Hyperparameters

The following hyperparameters were used in the JASE-DQN training process:

- Gamma (Discount Factor): 0.9
- Exploration Rate (Epsilon): Initially set to 1.0
- Exploration Rate Decay: 0.99999
- Minimum Exploration Rate: 0.1
- Replay Buffer Size: 100,000
- Batch Size: 32
- Number of Episodes: 5,000
- Q Network (Multi-Head Cross Attention) Learning Rate: 0.000055
- State Encoder Learning Rate: 0.00001
- Action Encoder Learning Rate: 0.000001
- Online Network Update Frequency: Every 3 new experiences
- Target Network Update Frequency: Every 10,000 experiences
- Minimum Number of Experiences Before Training: 10,000

These hyperparameters were tailored to the specific requirements of the Ms. Pac-Man game and JASE-DQN algorithm.

4 CHAPTER 4: Results

This chapter presents the results of training both the DQN and JASE-DQN algorithms on the game Ms. Pac-Man. Subsequently, a comparative analysis of these algorithms' performances is conducted. Furthermore, the chapter explores the application of transfer learning, utilizing pre-trained state and action encoders from the Ms. Pac-Man environment within the JASE-DQN architecture, to the Alien and Bank Heist environments. The performance of each algorithm is evaluated based on the rolling average reward over the last 100 episodes, with each episode being a complete game session until game over.

4.1 DQN on Ms. Pac-Man

In the context of our experiments, the DQN algorithm demonstrated significant proficiency when applied to the Ms. Pac-Man game. Over the training period, DQN achieved an average episode reward of 2,600, as illustrated in Figure 17. This performance metric serves as a robust indicator of the algorithm's ability to learn and adapt to the complexities of the game environment.

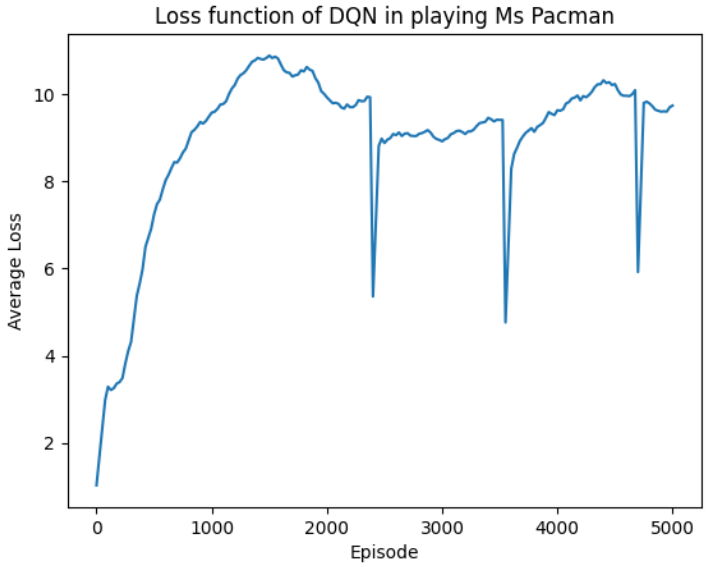

Figure 16: Rolling average loss function values of DQN for 5000 episodes

While the reward outcomes were promising, an examination of the loss function revealed a less straightforward pattern. As depicted in Figure 16, the loss function exhibited fluctuating behavior throughout the training process. Unlike the reward metric, which showed a clear trend of improvement, the loss function's values oscillated without a consistent directional pattern, sometimes increasing and other times decreasing. This variability suggests that the loss function

alone may not be a reliable indicator of the learning progress or the ultimate performance of the DQN algorithm.

The results obtained from the DQN algorithm on Ms. Pac-Man establish a critical baseline for comparative analysis with the JASE-DQN algorithm. By setting a benchmark with an average episode reward of 2,600, these findings enable a meaningful evaluation of the enhancements or deficiencies introduced by the JASE-DQN approach. This baseline is pivotal for assessing the efficacy of employing pre-trained state and action encoders within the JASE-DQN framework, particularly in terms of their contribution to performance improvements in the Ms. Pac-Man game environment.

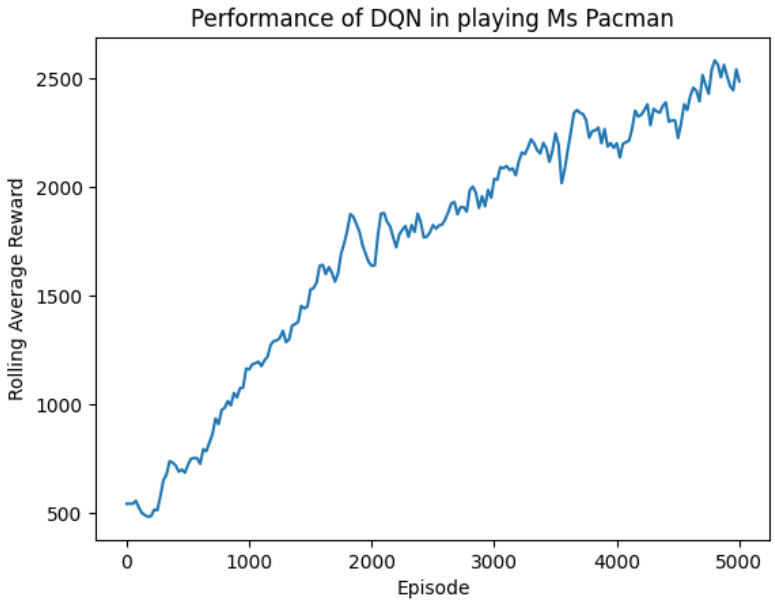

Figure 17: Rolling average reward of DQN in playing Ms Pac-Man for 5000 episodes

4.2 JASE-DQN on Ms Pac-Man

This section details the performance results of the JASE-DQN algorithm on Ms. Pac-Man, both with and without the use of pre-trained encoders for states and actions.

4.2.1 Without pre-trained encoders

When the JASE-DQN algorithm was employed without the benefit of pre-trained encoders, it managed to achieve an average episode reward of 2,100, as shown in Figure 19. This outcome, while notable, did not surpass the performance of the standard DQN algorithm. The behavior of the loss function during training was characteristic of reinforcement learning processes, exhibiting a general decreasing trend interspersed with numerous spikes, depicted in Figure 18. Such fluctuations are typical in reinforcement learning, reflecting the trial-and-error nature of the algorithm as it explores and learns from the game environment.

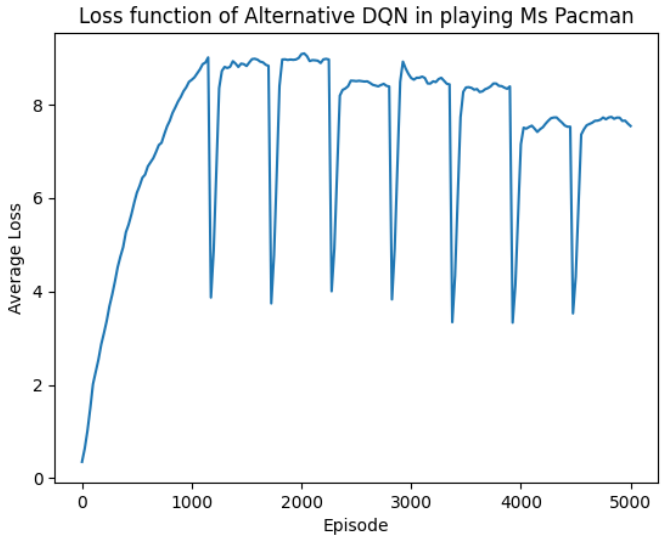

Figure 18: Rolling average loss function values of JASE-DQN without using pre-trained encoders for 5000 episodes

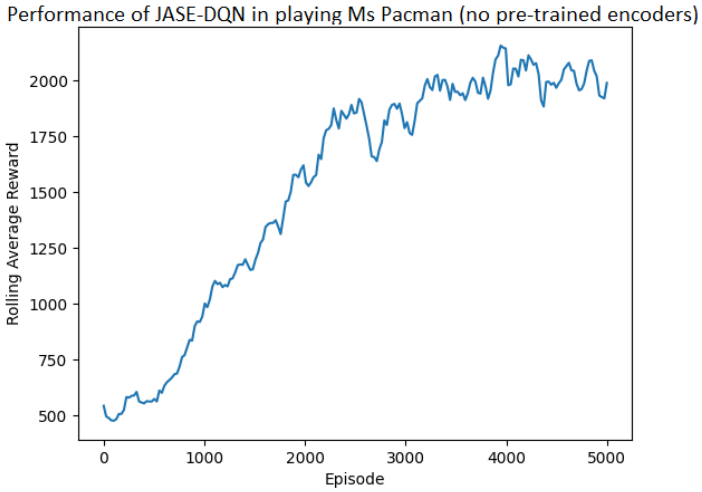

Figure 19: Rolling average reward of JASE-DQN without using pre-trained encoders in playing Ms Pac-Man for 5000 episodes

4.2.2 With pre-trained encoders

To enhance the learning efficiency of the JASE-DQN algorithm, state and action encoders were pre-trained using the most recent 20,000 experiences (state, action, reward, and next state) collected from the DQN agent. The state encoder was initially trained using the inverse model, followed by training the action encoder with the forward model and the already trained state encoder, as detailed in Section 3.2.2. The data were split into 80%, 10%, and 10% for training, validation, and test sets, respectively, and the training spanned 200 epochs. The F1-score of the inverse model on the test set was 0.5, while the RMSE of the forward model was 0.8. Despite these results not being highly impressive and acknowledging significant room for improvement, the objective was not to develop highly sophisticated encoders due to the extensive training time required. Consequently, these pre-trained encoders were integrated into the JASE-DQN architecture, with their weights fine-tuned during the training of the Q-network. This strategy aimed to expedite the learning process by utilizing the prior knowledge embedded in the encoders.

The use of pre-trained state and action encoders resulted in a marked improvement in the algorithm's performance. Specifically, the JASE-DQN algorithm achieved an average episode reward of 2,250, as illustrated in Figure 20. This performance gain underscores the efficacy of incorporating pre-trained encoders, which facilitated faster adaptation and learning within the game environment. The fine-tuning of encoder weights during training proved beneficial, allowing the algorithm to refine its understanding of the game dynamics more rapidly compared to the non-pre-trained version.

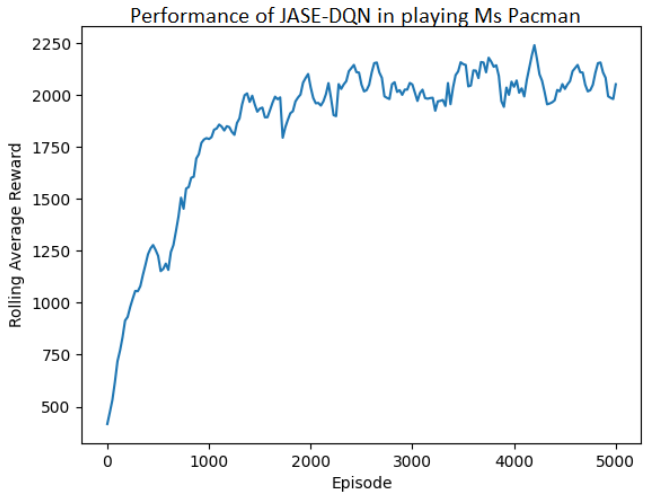

Figure 20: Rolling average reward of JASE-DQN with pre-trained encoders in playing Ms Pac-Man for 5000 episodes

Overall, the inclusion of pre-trained encoders in the JASE-DQN framework not only enhanced the learning speed but also improved the ultimate reward outcomes, demonstrating a valuable strategy for optimizing reinforcement learning algorithms.

4.3 DQN vs JASE-DQN on Ms Pac-Man

The comparison between the DQN and JASE-DQN algorithms on Ms. Pac-Man reveals insightful distinctions in their learning behaviors and performance outcomes. Initially, the JASE-DQN algorithm, when deployed without the aid of pre-trained encoders, could not surpass the average reward achieved by the standard DQN algorithm. Up to the first 3000 episodes, both algorithms exhibited similar trends, with DQN maintaining a slight performance edge. However, post the 3,000-episode mark, the DQN algorithm clearly outperformed the JASE-DQN, as depicted in Figure 21.

Conversely, the inclusion of pre-trained encoders significantly altered the performance dynamics of the JASE-DQN. Fine-tuning of the state and action encoders was tested over two different durations: up to 2,000 episodes and up to 5,000 episodes. It was observed that extending fine-tuning to 5,000 episodes resulted in better performance compared to fine-tuning for only 2,000 episodes, although the improvement was not substantially significant.

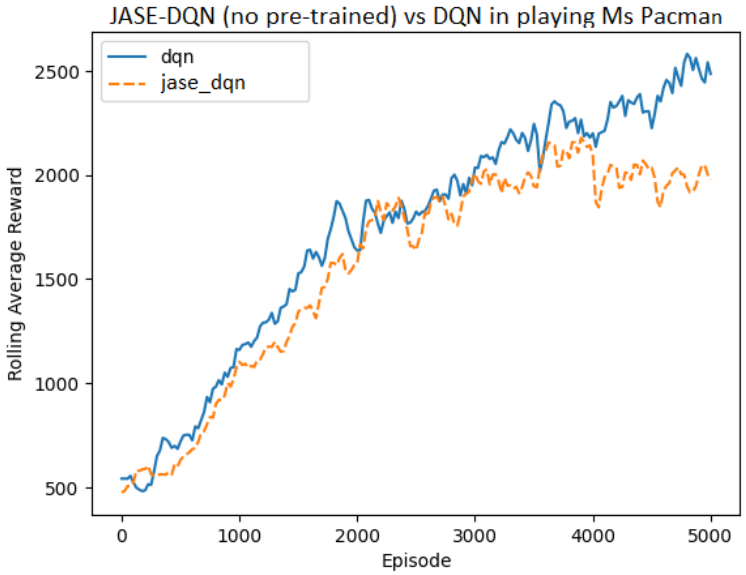

Figure 21: Comparison between rolling average rewards of JASE-DQN without using pre-trained encoders along with DQN

A critical observation is that the JASE-DQN, when equipped with pre-trained encoders, demonstrated a markedly faster learning curve. For instance, the JASE-DQN achieved an average episode reward of 2,000 within the first 1300 episodes, whereas the DQN required approximately 2,800 episodes to reach a similar reward level, as illustrated in Figure 22. This accelerated learning indicates that the pre-trained encoders enabled the JASE-DQN to grasp the game characteristics more swiftly, providing a notable advantage in the early stages of training. In summary, while the JASE-DQN without pre-trained encoders struggled to match the performance of the standard DQN, the integration of pre-trained encoders not only improved its overall performance but also significantly hastened the learning process. This highlights the potential of leveraging pre-trained models to enhance the efficiency and effectiveness of reinforcement learning algorithms.

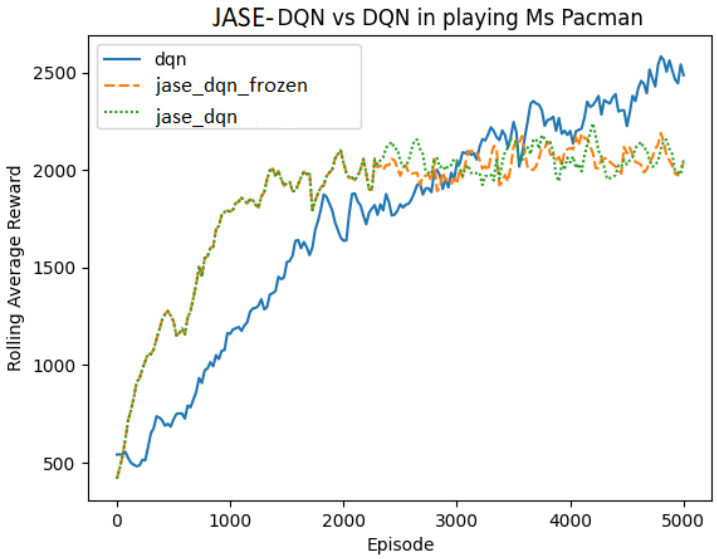

Figure 22: Comparison between rolling average rewards of JASE-DQN and DQN with different durations of fine tuning for state and action encoders.

4.4 DQN vs JASE-DQN on Alien

In this section, we explore the performance of DQN and JASE-DQN algorithms when applied to the game Alien. For this evaluation, state and action encoders pre-trained from the JASE-DQN agent in the Ms. Pac-Man environment were utilized.

The results clearly demonstrate that the JASE-DQN, equipped with pre-trained encoders from the Ms. Pac-Man environment, significantly outperformed the standard DQN in playing Alien.

As illustrated in Figure 23, the JASE-DQN quickly adapted to the new environment, achieving an average reward exceeding 900 within the first 1,500 episodes. Over the course of 5,000 episodes, it reached a maximum average reward of 950. In contrast, the DQN algorithm managed to achieve an average reward of only 800 after 5,000 episodes.

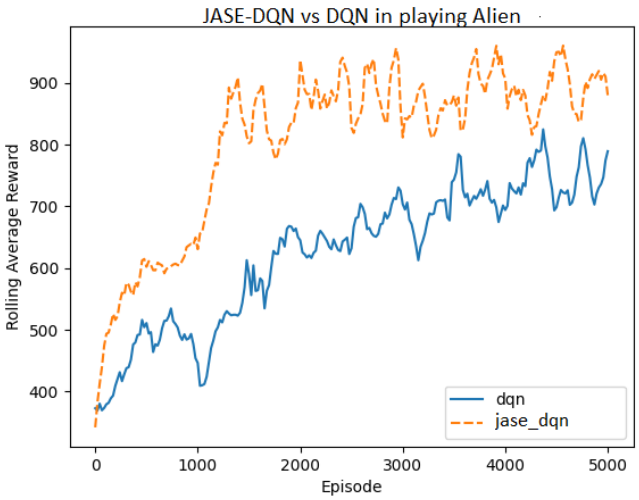

Figure 23: Comparison between rolling average rewards of JASE-DQN using pre-trained encoders along with DQN in playing Alien

To facilitate adaptation to the new environment, the state and action encoders were not frozen. This allowed the JASE-DQN to fine-tune these encoders, thereby capturing the unique characteristics and dynamics of the Alien game. The substantial improvement in performance highlights the efficacy of leveraging pre-trained encoders from one environment and adapting them to another, underscoring the potential of transfer learning in reinforcement learning.

Overall, the JASE-DQN's ability to utilize previously acquired knowledge from the Ms. Pac-Man environment and successfully apply it to the Alien environment resulted in superior performance and faster learning compared to the standard DQN. This experiment showcases the versatility and effectiveness of pre-trained models in enhancing the capabilities of reinforcement learning algorithms across different tasks.

4.5 DQN vs JASE-DQN on Bank Heist

In this section, we analyze the performance of the DQN and JASE-DQN algorithms in the game Bank Heist. For this comparison, state and action encoders trained from the JASE-DQN agent in the Alien environment were used. This selection was made based on the reasoning that although Bank Heist has many different characteristics compared to both Alien and Ms. Pac-Man, it is perhaps more similar to Alien than Ms. Pac-Man.

Initially, the JASE-DQN with pre-trained encoders from the Alien environment performed comparably to the DQN in playing Bank Heist. As shown in Figure 24, both algorithms had similar performance up to approximately 1800 episodes. The state and action encoders were not frozen during this period, allowing them to adapt to the new environment's characteristics.

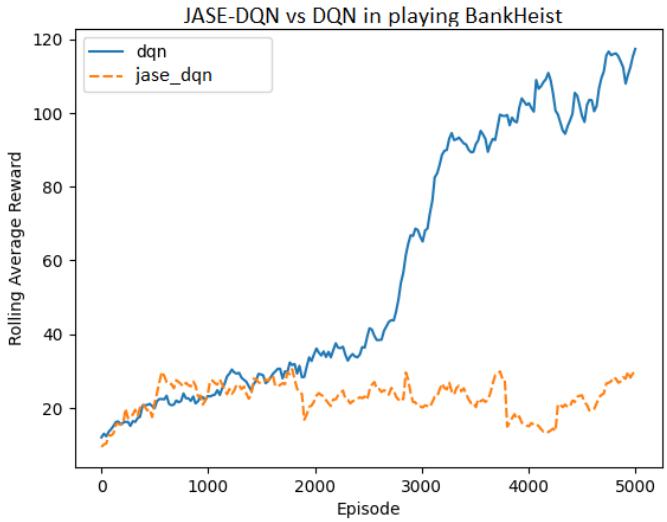

Figure 24: Comparison between rolling average rewards of JASE-DQN using pre-trained encoders along with DQN in playing Bank Heist

However, as training progressed, the JASE-DQN with pre-trained embeddings struggled to adapt to the game characteristics as effectively as the DQN. By the end of 5000 episodes, the DQN achieved an average reward of 120, whereas the JASE-DQN only reached an average score of 30. This significant disparity highlights the challenge of transferring learned representations between environments with distinct dynamics and characteristics.

The stark difference in performance underscores the fact that Bank Heist is considerably different from both Ms. Pac-Man and Alien. Despite the initial similarities that might exist between Alien and Bank Heist, the unique aspects of Bank Heist necessitate environment-specific training to achieve optimal performance.

These findings emphasize the limitations of transfer learning in reinforcement learning, particularly when applied across vastly different environments. While pre-trained models can provide a head start in some scenarios, as demonstrated by the initial performance parity, they may not always suffice for long-term adaptation and mastery of new, distinct environments.

5 CHAPTER 5: CONCLUSIONS

In this chapter, we synthesize the findings and insights gained from our extensive experimentation with the DQN and JASE-DQN algorithms across different game environments. Our research aimed to investigate the efficacy of these algorithms in various contexts, focusing particularly on the integration of state and action encoders, and the potential benefits of leveraging pre-trained models through transfer learning.

5.1 Overview of Experiments

Our experiments were conducted across three primary environments: Ms. Pac-Man, Alien, and Bank Heist. Initially, we trained both DQN and JASE-DQN algorithms on the Ms. Pac-Man environment to establish a baseline performance and compare the effectiveness of the two approaches. Subsequently, we explored the application of transfer learning by utilizing pre-trained encoders from Ms. Pac-Man in the Alien and Bank Heist environments, examining how well the knowledge acquired in one game could be transferred to another.

5.2 Performance in Ms. Pac-Man

The DQN algorithm demonstrated robust performance in the Ms. Pac-Man environment, achieving an average episode reward of 2600. This served as a strong baseline for comparison with the JASE-DQN algorithm. The loss function for DQN exhibited expected behavior typical of reinforcement learning processes, with fluctuations that did not provide significant insights into the learning dynamics but indicated ongoing adjustments and learning.

In contrast, the JASE-DQN algorithm achieved an average episode reward of 2100 without using pre-trained encoders. Although slightly lower than DQN, this performance was noteworthy given the additional complexity introduced by the state and action encoders. The loss function for the JASE-DQN without pre-trained encoders showed a decreasing trend with frequent spikes, a common occurrence in reinforcement learning due to the stochastic nature of the training process.

When pre-trained encoders were introduced, the performance of the JASE-DQN algorithm improved, achieving an average episode reward of 2250. This indicates that the pre-trained encoders facilitated faster learning and adaptation to the game environment, leveraging the knowledge encoded during the initial training phase. Fine-tuning these encoders during the training of the Q-network further enhanced the algorithm's ability to adapt and optimize its strategy.

5.3 Comparative Analysis on Ms. Pac-Man

Comparing the performance of DQN and JASE-DQN on Ms. Pac-Man revealed several key insights. The JASE-DQN without pre-trained encoders initially mirrored the trend of DQN up to approximately 3000 episodes, with DQN performing slightly better. Beyond this point, DQN outperformed JASE-DQN, underscoring the challenges of optimizing complex architectures without prior knowledge.

However, the scenario changed significantly when pre-trained encoders were utilized. The JASE-DQN with pre-trained encoders demonstrated a markedly different learning trajectory, particularly benefiting from fine-tuning up to 5000 episodes. The most significant advantage was observed in the early stages of training; the JASE-DQN achieved an average episode reward of 2000 within the first 1300 episodes, whereas DQN required 2800 episodes to reach a similar reward level. This rapid learning highlights the efficacy of transfer learning in expediting the training process by leveraging previously acquired knowledge.

5.4 Transfer Learning in Alien Environment

In the Alien environment, the JASE-DQN with pre-trained encoders from the Ms. Pac-Man environment exhibited superior performance compared to the DQN. The state and action encoders were allowed to adapt to the new environment, which proved beneficial. The JASE-DQN achieved an average reward exceeding 900 within the first 1500 episodes and peaked at 950 by 5000 episodes. In contrast, the DQN attained an average reward of only 800 over the same duration. These results underscore the potential of transfer learning to enhance performance in related but distinct environments, leveraging pre-trained models to accelerate learning and achieve higher rewards more rapidly.

5.5 Transfer Learning in Bank Heist Environment

The application of transfer learning in the Bank Heist environment presented a different set of challenges. Using state and action encoders pre-trained in the Alien environment, the JASE-DQN initially performed similarly with the DQN up to 1800 episodes. However, as training continued, the JASE-DQN struggled to adapt to the unique characteristics of the Bank Heist game. By 5000 episodes, the DQN had achieved an average reward of 120, significantly outperforming the JASE-DQN, which managed an average score of only 30.

This stark contrast highlights the limitations of transfer learning when applied to environments with substantially different dynamics. While the pre-trained encoders provided a temporary advantage, they were insufficient for long-term adaptation and mastery of the Bank Heist environment. This outcome underscores the importance of environment-specific training and the challenges inherent in transferring knowledge across disparate contexts.

5.6 Final Thoughts and Future Work

The findings from our experiments reveal a nuanced understanding of the strengths and limitations of DQN and JASE-DQN algorithms, particularly in the context of transfer learning. While pre-trained encoders can significantly expedite the learning process and enhance performance in related environments, their effectiveness is related within the similarity of the source and target environments. In cases where the environments are vastly different, as demonstrated by the Bank Heist experiments, the benefits of transfer learning diminish, necessitating tailored training approaches.

Future research could explore hybrid approaches that combine the strengths of both methods, perhaps by dynamically adjusting the reliance on pre-trained encoders based on the observed performance in the target environment. Additionally, investigating more sophisticated techniques for adapting pre-trained models to new environments, such as meta-learning or continual learning frameworks, could further enhance the versatility and effectiveness of transfer learning in reinforcement learning applications. Furthermore, simultaneously training the encoder components along with the Q-network could lead to more efficient feature extraction and representation learning. By jointly optimizing these parts, the agent might better adapt to varying environments, improving the overall robustness and performance of the RL algorithms. Lastly, investigating policy gradient methods, such as Proximal Policy Optimization (PPO) can provide insights into their potential advantages in handling high-dimensional or continuous action spaces. Comparing these methods with value-based approaches could uncover scenarios where policy gradient methods offer superior performance and stability.

In conclusion, while the JASE-DQN algorithm with pre-trained encoders shows promising potential, especially in accelerating learning and achieving higher rewards quickly, the choice of source and target environments plays a crucial role in determining the overall success of transfer learning strategies. This research lays the groundwork for further exploration into optimizing and generalizing reinforcement learning algorithms across diverse and complex environments.

Bibliography – References – Online sources

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. arXiv preprint arXiv:1312.5602.

[2] Van Hasselt, H., Guez, A., & Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. arXiv preprint arXiv:1509.06461.

[3] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. arXiv preprint arXiv:1511.06581.

[4] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized Experience Replay. arXiv preprint arXiv:1511.05952.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2023). Attention Is All You Need. arXiv preprint arXiv:1706.03762.

[6] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929.

[7] Agarwal, P., Abdul Rahman, A., St-Charles, P.-L., Prince, S. J. D., & Ebrahimi Kahou, S. (2023). Transformers in Reinforcement Learning: A Survey. arXiv preprint arXiv:2307.05979.

[8] Mini, U., Grietzer, P., Sharma, M., Meek, A., MacDiarmid, M., & Turner, A. M. (2023). Understanding and Controlling a Maze-Solving Policy Network. arXiv preprint arXiv:2310.08043.

[9] Yin, X., Yu, X., Sohn, K., Liu, X., & Chandraker, M. (2019). Feature Transfer Learning for Deep Face Recognition with Under-Represented Data. arXiv preprint arXiv:1803.09014.

[10] S. Doncieux, "Transfer learning for direct policy search: A reward shaping approach," 2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL), Osaka, Japan, 2013, pp. 1-6, doi: 10.1109/DevLrn.2013.6652568. keywords: {Knowledge based systems;Robot sensing systems;Switches;Trajectory;Evolutionary computation;Conferences},

[11] Zhu, Z., Lin, K., Jain, A. K., & Zhou, J. (2023). Transfer Learning in Deep Reinforcement Learning: A Survey. arXiv preprint arXiv:2009.07888.

[12] Gao, Q., Yang, H., Shanbhag, S. M., & Schweidtmann, A. M. (2023). Transfer learning for process design with reinforcement learning. arXiv preprint arXiv:2302.03375.

[13] Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2016). Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. arXiv preprint arXiv:1511.06342.

[14] Chen, Y., Chen, Y., Hu, Z., Yang, T., Fan, C., Yu, Y., & Hao, J. (2021). Learning Action-Transferable Policy with Action Embedding. arXiv preprint arXiv:1909.02291

[15] Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., & Thomas, P. S. (2019). Learning Action Representations for Reinforcement Learning. arXiv preprint arXiv:1902.00183.

[16] Minbeom, K., Kyeongha, R., Yong-duk, K., Kyomin, J. (2022). Action-driven contrastive representation for reinforcement learning. Graduate School of Artificial Intelligence, Seoul National University, Department of Electrical and Computer Engineering, Seoul National University, Defense AI Technology Center, Agency for Defense Development.

[17] Fujimoto, S., Chang, W.-D., Smith, E. J., Gu, S. S., Precup, D., & Meger, D. (2023). For SALE: State-Action Representation Learning for Deep Reinforcement Learning. arXiv preprint arXiv:2306.02451.

[18] Van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-learning. arXiv preprint arXiv:1509.06461.

[19] Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[20] Girshick, R. (2015). Fast R-CNN. arXiv preprint arXiv:1504.08083.

[21] Wang, Z., Yang, E., Shen, L., & Huang, H. (2023). A Comprehensive Survey of Forgetting in Deep Learning Beyond Continual Learning. arXiv preprint arXiv:2307.09218.

[22] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. arXiv preprint arXiv:1511.05952.

[23] Brown, B., & Zai, A. (2020). Deep Reinforcement Learning in Action. Manning Publications Co.

[24] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

[25] Agarap, A. F. (2019). Deep Learning using Rectified Linear Units (ReLU). arXiv preprint arXiv:1803.08375.

[26] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. ArXiv Preprint ArXiv:1606.01540.