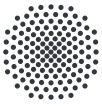


University of Stuttgart
Germany

Institute for Visualization and Interactive Systems
Universitätsstraße 38
70569 Stuttgart

Master Thesis

Evaluation of Different Image Representations for Reinforcement Learning Agents

Jayakumar Ramasamy Sundararaj

Course of Study:	INFOTECH
Examiner:	Prof. Dr. Andreas Bulling
Supervisor:	Anna Penzkofer, M.Sc. , Constantin Ruhdorfer, M.Sc.
Commenced:	November 15, 2023
Completed:	May 15, 2024

Abstract

Though Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm for training agents to perform complex tasks, it encounters challenges when confronted with raw sensory inputs. Despite using the deep neural network's prowess to generate meaningful internal representations, DRL approaches suffer from a high sample complexity. The effectiveness and scalability of DRL techniques are frequently hindered by the high-dimensional nature of input data, especially in methods utilizing image-based observations. To overcome this challenge, a promising approach is to start with improved input representations to enhance learning performance significantly. This work addresses this challenge using novel techniques to enhance DRL agents' training efficiency and performance. We propose using compact and structured image representations, namely object-centric and scene graph-based state representations, as intermediate state representations for training lightweight DRL agents. These representations facilitate extracting important features from raw observations, effectively reducing input space dimensionality. To assess the effectiveness of our proposed approaches, we conduct experiments on three Atari 2600 games: Space Invaders, Frostbite, and Freeway. Our findings reveal that models trained with intermediate state representations, while showing slightly lower performance than those trained from raw image pixels, achieved a notable performance by surpassing Human Normalized Score (HNS) in one game environment with fewer model parameters. Furthermore, we investigate alternative loss functions for value function estimation and explore strategies to mitigate the issue of diminishing entropy during training. Finally, through a systematic analysis of experimental findings, we provide valuable insights into the efficacy and drawbacks of these approaches, shedding light on promising avenues for future research in formulating suitable state spaces for training agents using DRL.

Contents

List of Figures

Acronyms

aGCN attentional Graph Convolutional Network. 16

ALE Arcade Learning Environment. 12

CNN Convolutional Neural Network. 19

DNN Deep Neural Network. 14

DQN Deep-Q Network. 36

DRL Deep Reinforcement Learning. 3, 11

GAT Graph Attention Network. 26

GCN Graph Convolution Network. 26

GNN Graph Neural Network. 16

HNS Human Normalized Score. 3, 17

HUD Heads-up Display. 25

MDP Markov Decision Process. 13

MSE Mean Squared Error. 36

OCR Object-centric representation. 15

PPO Proximal Policy Optimization. 13

ReLU Rectified Linear Unit. 27

RePN Relation Proposal Network. 16

RL Reinforcement Learning. 11

SPI Space Invaders. 20

TRPO Trust Region Policy Optimization. 14

VQA Visual Question Answering. 13

1 Introduction

Reinforcement Learning (RL) is a rapidly advancing field within machine learning that is dynamically shaping the landscape of artificial intelligence research. The core concept of RL is to enable agents to learn from their interactions with an environment. RL excels in scenarios necessitating sequential decision-making, spanning various applications, including robotic control, game-playing, recommendation systems, and autonomous navigation. Seminal studies conducted by [BPK+20; MKS+15; SAH+20; SHS+17] have demonstrated the remarkable capabilities of RL agents, often surpassing human performance in specialized domains. However, these accomplishments are accompanied by significant challenges, particularly regarding the efficiency of training RL agents. Conventional RL methods that rely on simple function approximations or tabular methods often require extensive data for training, making them computationally intensive and impractical for real-world applications.

An obstacle in Deep Reinforcement Learning (DRL) is determining how to represent the environment's state effectively [MRM11]. Some previous works have represented the state using single-vector representations, encoding the entire input image into a single vector [MKS+13], while others operate directly on image data [SAH+20; SHS+17]. However, these representations might not capture crucial relationships and interactions between objects in the scene. To address these challenges, we utilize novel and compact intermediate-state representations, specifically scene graphs and object-centric representations [ZSM20] derived from raw pixel data in complex RL environments. These representations are characterized by denser information structures, which have the potential to enhance the efficiency and effectiveness of RL training. By leveraging richer visual abstractions, we aim to accelerate the learning process and improve the performance of RL agents.

Furthermore, we acknowledge the inherent volatility of RL training, where minor perturbations can significantly disrupt the learning process [DD20]. Our approach seeks to cultivate stability and resilience within RL training methods, fostering more reliable and scalable solutions. Moreover, RL environments vary in the types of observation spaces they offer. Some environments present state representations as vectors, as seen in CartPole[1], while others feature image states where agents interact with raw pixel data, as exemplified by Atari[2]. Our research endeavours to bridge this gap by unifying the understanding and constructing intermediate state representations. These representations aim to address the challenges posed by disparate observation modalities, fostering a more integrated approach to training DRL agents.

We evaluate the performance of DRL agents trained on these intermediate-state representations by comparing them to a baseline model trained directly from raw pixel data. We use multiple metrics to evaluate our proposed methodologies comprehensively. Moreover, our research investigates

[1] https://gymnasium.farama.org/environments/classic_control/cart_pole/

[2] https://gymnasium.farama.org/environments/atari/

the interplay of underlying parameters affecting DRL agent performance. By examining these factors closely, we aim to understand the learning process dynamics better. To validate our findings and assess the generalizability of our approaches, we conduct experiments using the Atari 2600 games implemented in the Arcade Learning Environment (ALE) [BNVB13a]. The Atari games were chosen for their diverse nature. Additionally, the complexity of each game makes them representative of practical challenges, while their independent creation ensures unbiased results, enhancing the credibility and reliability of our study.

Ultimately, this research strives to develop streamlined image representations to bridge the gap between traditional RL training methods and more efficient, adaptable approaches. By utilizing compact visual representations and improving training stability, our objective is to advance reinforcement learning towards scalable, real-world applications, thereby making a significant contribution to the field of artificial intelligence.

1.0.1 Goal of the Thesis

This work investigates the impact of different state representation methods on the performance of DRL agents in complex environments. The work addresses the following research question.

How do different state representations influence the performance of DRL agents across various environments with distinct characteristics?

2 Related Work

RL is a machine learning paradigm where an agent learns an optimal policy through experiences rather than examples, as in typical machine learning methods. RL aims to find an optimal strategy, called a policy, to maximize cumulative rewards over time. The agent learns through trial and error, adjusting its actions based on the observed rewards. Current RL methods often demand extensive training times and significant computational resources. For example, a deep Q-network requires millions of samples to solve certain Atari games[MKS+15], resulting in considerable sample inefficiency. Studies [MBM21; YZK+20] propose a potential solution by advocating for the simultaneous learning of a latent representation of the environment states alongside a control policy to address this challenge. The following sections provide a comprehensive overview of the prior state-of-the-art literature on state representations, focusing on their impact on various downstream tasks, starting from robotic manipulation to Visual Question Answering (VQA) tasks.

2.1 Preliminaries

2.1.1 Markov Decision Process

We consider a RL agent that interacts with a Markov Decision Process (MDP). An MDP is described by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mu, \gamma \rangle$, where S is the state space, A is the action space whose size is bounded by a constant, $\mathcal{P}(s'|s,a)$ defines a Markovian transition probability density between the current state s and the next state s' under action a. The initial state distribution is defined as $\mu : \mathcal{S} \to [0,1]$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and a discount factor $\gamma \in [0,1)$. Given an MDP, the goal of RL agent is to learn a specific behaviour in an unknown environment. The policy π determines the agent's behaviour by selecting an action given its current state and is formally defined as a mapping from a state s to an action a, $\pi : \mathcal{S} \to \mathcal{A}$.

2.1.2 Actor-critic

Policy optimization methods are centred around the policy, a function that maps the agent's state to its next action. These approaches conceptualize RL as a numerical optimization task, focusing on optimizing the expected rewards relative to the parameters of the policy. Actor-critic algorithms such as A3C [MBM+16], and Proximal Policy Optimization (PPO) [SWD+17] serve as effective tools to optimise RL policies. In these algorithms, the actor component is responsible for learning the policy and determining which actions to take in different situations. Concurrently, the critic component learns a value function, providing feedback on the quality of the selected actions or states. Policy optimization in actor-critic algorithms typically involves updating the actor using policy gradient methods. These methods estimate the gradient of the expected return with respect

to the policy parameters, guiding the actor to adjust its policy in a direction that maximizes the expected return. This optimization process is informed by the critic's evaluations of the value function, aiding in refining the policy for better decision-making in the given environment.

2.1.3 Proximal Policy Optimization

PPO [SWD+17] is a specific variant of the actor-critic algorithm, building upon prior policy optimization methods like Trust Region Policy Optimization (TRPO) [SLM+17]. While TRPO prioritized data efficiency and prevented policy degradation through learning rate optimization, it introduced constraints to the objective function, thereby transforming it into a quadratic optimization problem. On the other hand, PPO adds significant adjustments to the objective function and the optimization algorithm, making it more sample-efficient and stable than earlier techniques.

The objective function of the PPO algorithm combines two terms: the objective term from the TRPO algorithm and a clipped policy gradient term. The policy gradient term is clipped based on the advantage value, representing the advantage of taking a particular action compared to others. PPO employs careful clipping to ensure the policy update is conservative yet effective. When the advantage value is positive, the policy gradient term is clipped by $1 + \varepsilon$, and when it is negative, it is clipped by $1 - \varepsilon$, where ε is a hyperparameter that controls the extent of the policy update. This means that updates to the policy for actions with positive advantages are limited to prevent large deviations, while actions with negative advantages have their updates restricted to avoid worsening the current policy. By adopting this approach, PPO guarantees that it takes small and safe steps towards identifying and reducing the impact of unfavourable actions while minimizing the influence of beneficial actions on the policy update. This cautious yet efficient updating strategy helps maintain the stability and performance of the learning process, ensuring that the algorithm avoids drastic changes that could harm the overall policy.

2.1.4 Deep Reinforcement Learning

DRL merges RL with Deep Neural Network (DNN) to approximate optimal action values or policy functions. It excels in complex environments with high-dimensional state and action spaces, where traditional RL methods may struggle. DRL algorithms leverage large datasets to maximize cumulative rewards and have been successfully applied across various domains. For instance, Silver et al. (2017) achieved mastery in the game of Go by combining supervised learning with RL steps to train DNN alongside a Monte-Carlo tree search algorithm.

2.2 State Representation

The effectiveness of state representation in a DRL task plays a crucial role in determining the performance and efficiency of the learning process. A good state representation should capture relevant environmental information that enables the RL agent to make informed decisions and effectively navigate the state space. The core idea is to map the high-dimensional state space to low-dimensional latent representations, which are more informative and facilitate the learning of an optimal policy. Bartlett et al. [BSO22] enhanced the agent performance by introducing a novel

state-space representation different from conventional methods. Utilizing biologically inspired grid cells and spatial semantic pointers in a Gym MiniGrid environment, they found that these representations reduced the number of learning trials needed to solve the environment compared to conventional one-hot encoded vectors. Hong et al. [HYS+18] applied semantic image segmentation to create a state representation from RGB images for training robots in obstacle avoidance and target following tasks. Their findings indicated that learning from these semantically segmented image state representations surpassed baseline models in both tasks.

Several prior methods have utilized single-vector representations [HWL+23; MKS+13; MKS+15] to encode the entire input image as input for the policy network. However, these representations may fail to capture crucial relationships and interactions among entities in the scene [SRB+17]. Zambaldi et al. [ZRS+18] developed a potential solution that adopted a region-based representation to overcome this constraint. In this approach, an image undergoes encoding into a grid of representations. These representations are subsequently fused through a transformer encoder, enabling the explicit modelling of interactions between various regions. In the subsequent sections, we will discuss prior studies that have employed object-centric and scene graph-based intermediate state representations to train agents in different task environments.

2.2.1 Object-centric Representation

Object-centric state representations have emerged as a promising approach in DRL tasks, particularly in environments characterized by complex and diverse object interactions. These representations offer a more intuitive understanding of the environment by explicitly modelling individual objects and their interactions. Ye et al. [YGGT19] utilized object-centric state representations, showcasing their effectiveness in guiding policies for robotic manipulation tasks. Their approach constructs a structured Object-centric representation (OCR) from images, facilitating model predictive control. Results indicate that this explicit representation captures object interactions more effectively, enhancing prediction using a forward model alongside an additional correction module. Yoon et al. [YWBA23] introduced a benchmark to systematically evaluate pre-training of OCR for RL tasks. Their experiments, spanning tasks like object interaction and relational reasoning, validated the effectiveness of OCR in navigating complex 2D and 3D visual scenes within image-based RL.

Several studies [CP19; LWP+20; VCC+20] have explored the unsupervised learning of OCR to encode the state of a scene. Unlike traditional methods relying on labelled data, these approaches aim to acquire a structured visual representation directly from images, treating each image as a composition of objects. Furthermore, similar to region-based models as mentioned in [ZRS+18], they can be integrated using a transformer encoder to capture object relationships. Delfosse et al. [DBG+24] introduced the OCAtari framework, which employs an object-centric approach to represent RL states in Atari games. Utilizing a colour-based object detection technique implemented in OpenCV, they detect objects within the game environment. Locatello et al. [LWU+20] presented an alternative approach for representing image states. They proposed the concept of slot attention to enhance the encoding of sets of embeddings, each capturing information about individual objects within a scene. This method offers more semantically explicit entity modelling within the scene than region-based representation models [YWBA23]. Therefore, we employ this slot attention method as one of our techniques for constructing intermediate state representations.

2.2.2 Scene Graph Representation

A different representation of the state space is done by generating scene graphs. A scene graph is a structured representation of a scene that can clearly express the objects, attributes, and relationships between objects in the scene. Koner et al. [KLH+21] employed scene graphs as a training methodology to enhance the performance of an RL agent in Visual Question Answering tasks. Creating a scene graph typically involves a bottom-up approach where entities are organized into triplets, and these triplets are linked to construct the complete scene graph. The primary goal of this task revolves around detecting visual relationships, denoted as < subject, relation, object > triplets, commonly abbreviated as $< s, r, o >$ [ZZJ+22].

In their work, Yang et al. [YLL+18] introduced a novel scene graph generation model named Graph R-CNN, demonstrating efficacy and efficiency in object and relation detection within images. Their model incorporates a Relation Proposal Network (RePN) specifically designed to handle the quadratic number of potential relations between objects present in an image. Graph R-CNN employs Faster R-CNN [RHGS16a] to detect objects within the image and subsequently investigates potential relationships among all identified objects. Following this, it employs a learned measure of 'relatedness' to filter out improbable relationships, resulting in a more concise graph structure. Finally, an attentional Graph Convolutional Network (aGCN) is applied to integrate global context and update labels for both object nodes and relationship edges.

Additionally, numerous studies have utilized reinforcement learning (RL) methodologies for either generating scene graphs [KEH22; LGLS22; LRC+21] or for reasoning over them to enhance performance in visual question-answering (VQA) tasks [HLK+20; KLH+21]. In contrast to these approaches, we adopt a different perspective by employing scene graphs as intermediate state representations to train our DRL agent, thereby empowering it to excel in mastering complex Atari games.

Recently, there has been a surge of interest in learning with graph-structured data, including knowledge graphs (KGs), biological networks, and scene graphs. Representing data as graphs offers numerous advantages, such as enabling systematic modelling of relationships and providing a simplified representation of complex problems. However, interpreting and evaluating graph-structured data using conventional DNN methods poses challenges. This is primarily due to the uneven structure, irregular size of unordered nodes, and dynamic neighbourhood composition inherent in graphs, making fundamental mathematical procedures like convolutions difficult to implement. Graph Neural Network (GNN) address these limitations by extending DNN techniques to graph-structured data. GNN architectures enable the joint modelling of both structural information and node attributes. Numerous review articles have emerged, focusing either on DRL [LRY+21; MSIB20] or GNN individually [LNF+19; ZCH+21; ZZH+22]. However, many of these reviews only briefly touch upon combining both approaches. For instance, Wu et al. [WPC+21] outlined various GNN architectures and their applications but did not delve into DRL beyond its potential use in generating graphs with desired properties. Similarly, in the study by Ji et al. [JPC+22], RL is briefly discussed in the context of path finding and relation extraction for knowledge graphs, primarily as a process or algorithmic step, without mentioning the integration of GNN with DRL. In contrast to the approaches mentioned above, our research harnesses GNN as function approximators, thereby enabling the implementation of DRL algorithms to enable policy estimation on scene graphs.

2.3 Atari Environments

The Atari environments provide a compelling testing ground for evaluating RL tasks, given their blend of diverse challenges, standardized benchmarks, rich visual input, and low-dimensional action spaces despite their high-dimensional observation spaces (210 × 160 RGB video at 60Hz). Bellemare et al. [BNVB13a] devised five distinct screen representations to support a shared Model-Free Reinforcement Learning (MFRL) algorithm called SARSA. Their empirical investigation showcased that learning progress can be achieved in Atari 2600 games. Numerous research endeavours have utilized the Atari 2600 platform within the ALE [BNVB13b] as a benchmark to assess the efficacy of their RL agents [BNVB13a; Fol; MKS+13]. The tasks presented in this test bed are intentionally designed to be intricate and present a diverse and engaging set of challenges that are challenging even for human players [MKS+13]. For example, the combination of game complexity and the necessity to make quick decisions make these games difficult for humans to master. Despite being deterministic, it is exceedingly challenging to develop a perfect model due to the vast number of possible game-play situations [Fol]. We have chosen three Atari environments for our work. Firstly, Space Invaders (SPI) [1] represents an environment conducive to easy exploration, characterized by optimal rewards achievable by human players. Secondly, Frostbite [2] presents a more challenging exploration scenario, offering dense rewards. Lastly, Freeway [3] poses another challenging exploration task featuring sparse rewards. Therefore, we utilize these three distinct Atari 2600 game environments to train and assess our RL agents, focusing on crafting the state-space representations as described above.

After training the models, we evaluate their performance. Each Atari game environment employs its own scoring system, resulting in diverse score scales. For instance, the high score in Space Invaders, as documented by Badia et al. (2020), reaches 74335, while in Frostbite, it extends to 631378. To make the evaluation of the games comparable, we employ a Human Normalized Score (HNS). Recent advancements in DRL algorithms [BPK+20; SAH+20] aim to develop agents capable of achieving superhuman performance. Therefore, utilising a metric that effectively reflects the algorithms' performance compared to human capabilities is crucial. Introduced by Bellemare et al. [BNVB13a], HNS has become widely adopted in DRL research. HNS can be calculated as follows:

$$\text{HNS}_{g,i} = \frac{G_{g,i} - G_{g,\text{ random}}}{G_{g,\text{ human average}} - G_{g,\text{ random}}} \tag{2.1}$$

where g denotes the gth game of Atari, i represents the algorithm i, $G_{g,\text{ human average}}$ represents the human average score baseline and $G_{g,\text{ random}}$ represents the performance of a random policy. An advantage of utilizing HNS is its provision of a simple and intuitive metric for comparing RL agents' performance to that of humans. $HNS_{g,i} \geq 100\%$ means algorithm i has surpassed the human average performance in game g.

[1] https://gymnasium.farama.org/environments/atari/space_invaders/

[2] https://gymnasium.farama.org/environments/atari/frostbite/

[3] https://gymnasium.farama.org/environments/atari/freeway/

3 Methodology

The methodology begins with the environment rendering observations in the RGB array mode, producing raw RGB arrays, which we refer to as frames. As depicted in Figure 3.1, these frames undergo pre-processing to transform them into one of two intermediate state representations: the object-centric and the scene graph representation. We hypothesise that this pre-processing step is crucial for extracting meaningful features from the raw visual data to reduce the high dimensional state space into low dimensional compact representation, enabling the agent to make informed decisions based on the environment's state. We selected these two intermediate state representations to evaluate them in assessing the most efficient one that would improve the performance of DRL agents. In the object-centric state representation, we focus on the individual objects in a given frame. We extract these objects along with their attributes and construct unique object descriptors. In the scene graph-based state representation, we focus on the objects and their relation to other objects in a frame, essentially forming a graph. These intermediate state representations are processed and fed into subsequent agents. The agent architecture comprises essential components such as the policy network, value network, and memory buffer designed to facilitate efficient learning and decision-making.

The following sections provide a comprehensive overview of the pre-processing pipeline and describe the methodologies utilized to transform the frames into object-centric and scene graph state representations. Additionally, we explore the architectures of the corresponding agents, explain the training procedure, and discuss the baseline model.

3.1 Object-centric State Representation

In the object-centric representation, we aim to utilize the objects present in an image as mentioned by [DBG+24; YWBA23; ZSM20]. The fundamental hypothesis entails formulating an intermediate state representation that exclusively encapsulates the defining attributes of objects, such as their shape, colour, and spatial positioning, thus yielding unique object descriptors. These descriptors serve as the foundational building blocks of our object-centric states. Once these descriptors are constructed, we combine individual object descriptors' features into a unified state representation. This is achieved by aggregating these individual object descriptors into a single vector. Subsequently, the agent utilizes this vector as input to generate actions.

As shown in Figure 3.2, the process starts with the environment generating observations in the form of RGB arrays/frames. We then process these frames with the model developed by [LWU+20]. This object-centric model comprises a Convolutional Neural Network (CNN)-based encoder augmented with positional embeddings and a slot attention module. The integration of positional embeddings within the encoder architecture is important here because of the internal operation of the slot attention module, which treats input elements as unordered vectors, even when derived from image

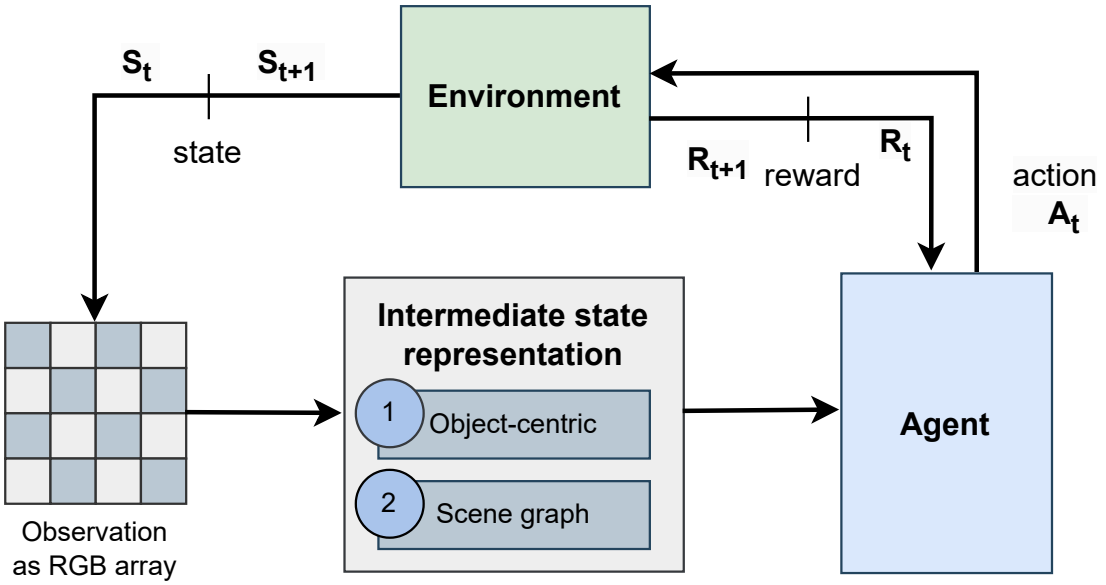


Figure 3.1: Methodology overview. The environment renders observations as RGB arrays, which are then transformed into one of two intermediate state representations before being fed as input to the agent.

data. Consequently, direct access to positional information gets lost. To circumvent this loss, as mentioned in [LWU+20], we augment the input features to the slot attention module, represented by CNN feature maps, with positional embeddings, thereby enabling the incorporation of positional information. Thereafter, the output of the encoder feeds into the slot attention module, where the set of N input feature vectors transforms into K output vectors, denoted as slots. These slots serve as unique descriptors, each representing certain types of objects, capturing attributes such as shape, colour, and position. Therefore, each object type in a given image is grouped into one of the available slots.

As described in [LWU+20], employing an iterative attention mechanism, the slot attention module iteratively maps its inputs to the designated slots. These slots are randomly initialized as independent samples from a Gaussian distribution, with shared and learnable parameters $\mu \in \mathbb{R}^{D_{\text{slots}}}$ and $\sigma \in \mathbb{R}^{D_{\text{slots}}}$ and fine-tuned with each iteration $t = 1...T$ to align with specific portions or groupings of the input features. We have set the dimension D, i.e. the vector size of the slot, to be 64, and the number of iterations for the slot attention module to $T = 3$, following the original paper [LWU+20]. During each iteration, slots compete to group input segments through a softmax-based attention mechanism, updating their representations via a recurrent update function. In our work, we have maintained a consistent number of five output slots across all environments. However, the number of slots can be adjusted depending on the specific downstream task requirements. Our selection of five slots is based on the fact that the objects in our chosen environments can be reasonably categorized into five distinct types. For example, the five distinct types of objects in the Space

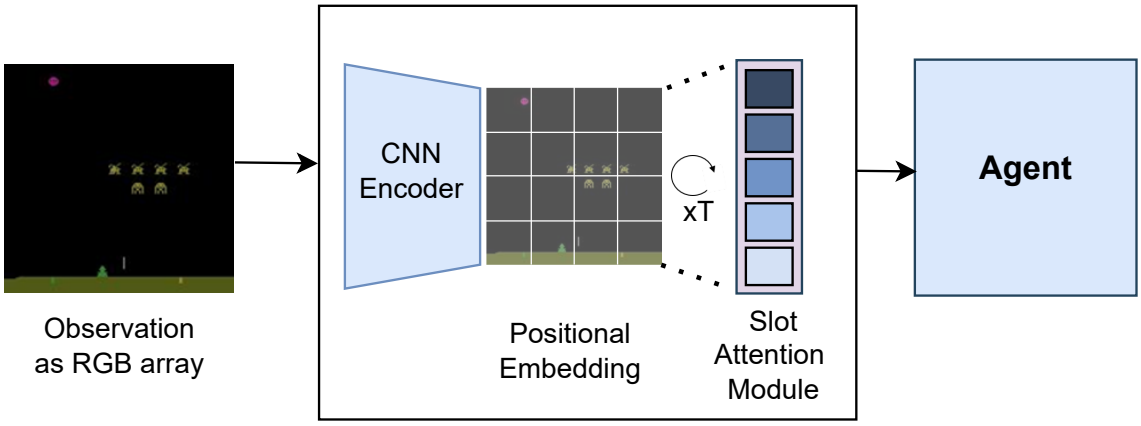


Figure 3.2: An overview of object-centric state representation: The environment renders observations as RGB arrays, subsequently directed into an object-centric model containing a CNN-based encoder augmented with positional embeddings, succeeded by a slot attention module. This framework generates slots, further utilized by the agent for action generation.

Invaders (SPI) game environment are aliens, bullets, satellites, shields and a player. Once the slots are obtained, we proceed to process and combine them into a unified vector representation, which acts as the intermediate state representation for the agent

Before using the above-specified object-centric state formulation technique, we utilized a pre-trained Faster R-CNN model [RHGS16a] with a ResNet backbone to detect and localize objects within the frames. Initially, for the SPI game environment, we conducted fine-tuning on the Faster R-CNN model for 50 epochs, leveraging a custom dataset comprising 400 samples distributed across seven distinct object classes sourced from the Atari-Head dataset [ZWL+19]. The obtained mean average precision (mAP) score of 0.86 denotes the model's competence in object detection tasks. The Faster R-CNN model provides the detected objects' bounding boxes, class labels, and confidence scores. Subsequently, we encoded the bounding boxes for each object and colour values to create unique object descriptors. These descriptors were then aggregated to form a single vector, serving as our state space representation for the respective frame. However, we opted against continuing with this approach due to its high computational overhead and prolonged training times. Specifically, using the Faster R-CNN model led to a notable reduction in the steps per second (SPS) during RL training, averaging at 6, corresponding to approximately 13.3% of the current SPS rate. Hence, we neglected the Faster R-CNN approach and used the slot-attention-based technique to construct object-centric states.

3.1.1 Agent Architecture

The agent employed in our object-centric state representation comprises policy and value networks and a replay buffer. At its core lies an actor-critic architecture integrated with the Proximal Policy Optimization (PPO) algorithm [SWD+17]. This architecture features shared neural network parameters for both the actor (policy network) and the critic (value network). Structurally, the network consists of a sequence of three linear layers followed by a mean aggregation function, consolidating information from individual slots processed by these linear layers. Each slot undergoes processing through three linear layers before being concatenated into a single tensor along the slot dimension (number of slots). Subsequently, the resulting tensor undergoes mean pooling to generate a consolidated representation, which is then forwarded to the actor and critic layers. In the actor network, we calculate action logits, which outline the raw probabilities linked to each action in the action space before normalization. These logits are then used to create a categorical distribution, allowing us to sample actions. At the same time, we compute the log probability and entropy of the chosen action, which are crucial for determining losses later on. Meanwhile, the critic network focuses on estimating the value function, which predicts the total reward we can expect from a specific state.

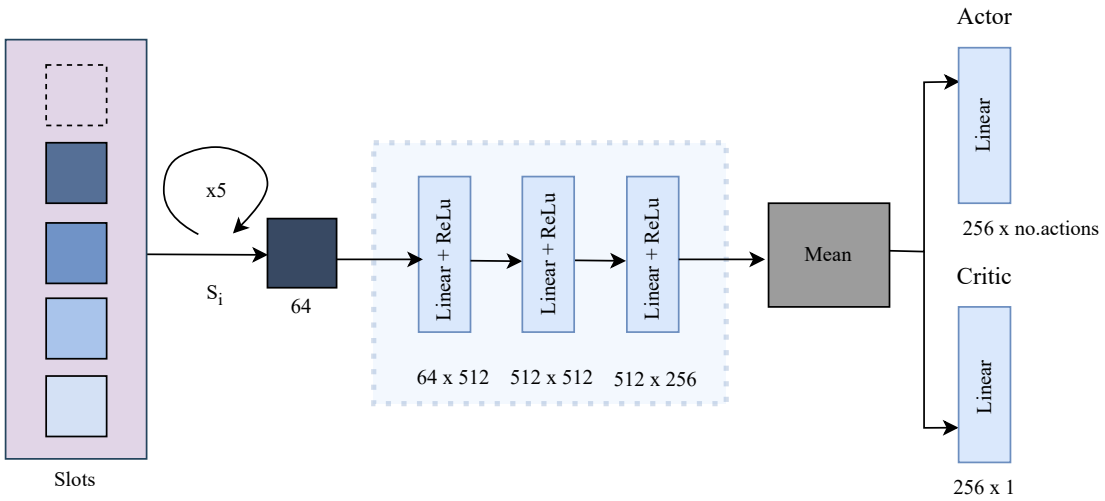


Figure 3.3: Policy and value network architecture of agent for the object-centric state representation. We individually process each slot by passing it through a series of linear layers before concatenating it into a single tensor along the slot dimension (number of slots). This resulting tensor is then aggregated using a mean function, resulting in a single vector representation. This resultant vector is subsequently fed through the corresponding fully connected layers of both the actor and critic components to generate action probabilities and a value estimate.

3.1.2 Training Paradigm

The DRL agent's training begins with PPO initializing a vectorized environment capable of running multiple independent yet similar game environments in parallel through multi-processing. This parallel execution is represented by the number of environments (N) configured as N = 8 for our

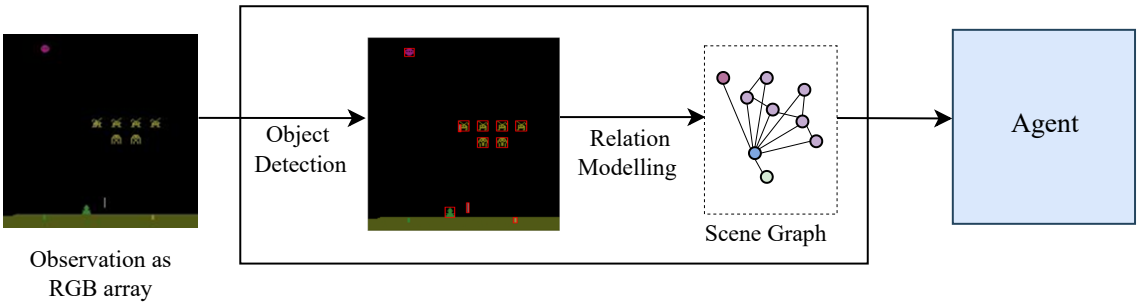


Figure 3.4: An overview of scene graph-based state representation: Initially, the environment captures observations as RGB arrays fed into a scene graph generation function. This function encompasses an object detection phase inspired from[DBG+24], succeeded by rule-based relation modelling to construct scene graphs. The resulting scene graphs serve as input for the agent, enabling action generation and state manipulation within the environment. Subsequently, the environment responds with rewards corresponding to the agent's actions.

training. The vectorized environment provided by PPO offers a synchronous interface, consistently delivering a batch of N observations from N environments and accepting a batch of N actions to take a step in these environments simultaneously.

The vectorized environments operate in two distinct phases: the rollout phase and the learning phase. During the rollout phase, the agent samples actions for the N environments and continues to step them for a predefined number of steps (M). The values of N and M can be found in the Table 4.1. Throughout these M steps, the agent collects valuable data into the replay buffer, including actions, observations, termination flags, and rewards, which are collectively termed experiences. The usage of the replay buffer mechanism stores these experiences and addresses issues such as correlated data and non-stationary distributions by allowing the agent to sample randomly from past interactions, thereby ensuring a more stable training process. Subsequently, during the learning phase, the agent utilizes the stored data from the replay buffer for learning. In detail, the collected data is shuffled and segmented into mini-batches, each containing 256 samples, to calculate gradients and update the policy accordingly. The PPO algorithm employs the mini-batch data to estimate advantage values and returns, which in turn are utilized for gradient computation and subsequent policy updates. Additionally, we have adhered to other hyperparameters as outlined in [HDR+22]. In total, the agent undergoes training for 10 million timesteps.

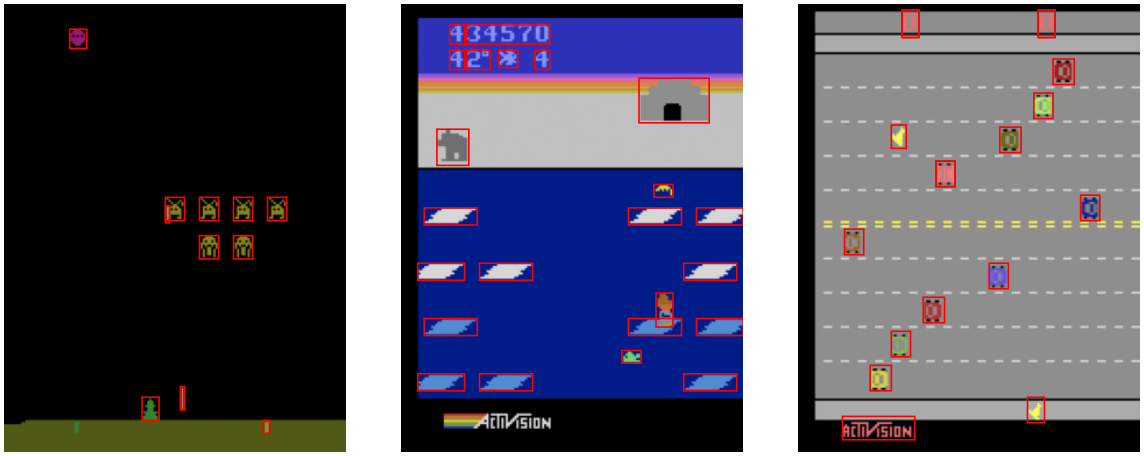
3.2 Scene Graph State Representation

For the second intermediate state representation, we adopted the scene graph-based approach. Initially, we aimed to use existing models to construct a scene graph from an image. One such model is the Graph R-CNN by [YLL+18]. Graph R-CNN employs Faster R-CNN [RHGS16b] to detect objects within the image and subsequently investigates potential relationships among all identified nodes. Following this, it employs a learned measure of 'relatedness' to filter out

improbable relationships, resulting in a more concise graph structure. Finally, an aGCN is applied to integrate global context and update labels for object nodes and relationship edges. However, a challenge arises with the RePN component, which assesses whether the bounding boxes of two nodes overlap. Positive 'relatedness' scores are only assigned to nodes and edges if an overlap is detected; otherwise, the score is set to 0. This works well on natural images or images of global context, where there is a high possibility of overlapping of bounding boxes. In the context of Atari game environments, the likelihood of overlapping bounding boxes between two objects is minimal, as depicted in Figure 3.5, resulting in the absence of viable graphs. Hence, we devised a different spatial rule-based scene graph construction approach as depicted in Figure 3.4 to ensure the generation of a graph structure with meaningful relationships for the policy network.

The process of generating scene graph state representations begins with the environment, rendering observations as RGB arrays/frames. These frames undergo a two-step scene graph construction approach. Initially, the frames are inputted into an object detection phase. Subsequently, they proceed to a rule-based relation modelling phase, where spatial constraints are applied to organize the detected objects into a scene graph structure. Following this construction, the scene graphs are fed into a GNN integrated within the PPO agent model. We believe that this scene graph representation would endow the agent with the ability to perceive and comprehend the underlying structure of its environment.

3.2.1 Object Detection Phase



(a) Space Invaders **(b)** Frostbite **(c)** Freeway

Figure 3.5: Object detection in Atari environments using the OCAtari framework.

Object detection is pivotal in constructing a scene graph. We first utilized the Faster R-CNN model [RHGS16b] to detect objects. However, due to the high computational time as mentioned in Section 3.1, we then utilised the OCAtari framework [DBG+24] to detect objects in a given frame. As developed by [DBG+24], this object detection pipeline uses OpenCV functions and integrates colour segmentation, morphological operations, and contour detection to identify objects precisely. Initially, the frames undergo colour-based segmentation, isolating objects with specific colour attributes from the background. Subsequent morphological operations like 'closing' refine the segmentation outcomes, reducing noise and enhancing object definition. Finally, contour detection

extracts object contours, facilitating precise bounding box determination for the comprehensive scene graph generation. Our methodology excludes non-essential visual elements, such as Heads-up Display (HUD) components (e.g., scores, number of lives). Figure 3.5 illustrates the result of the object detection process applied to frame samples from selected Atari environments such as Space Invaders, Frostbite, and Freeway.

3.2.2 Rule-based Relation Modelling

The construction of a scene graph relies on the formulation of $< subject, relation, object >$ triplets, representing the core entities and their relationships within a given scene. These triplets, commonly denoted as $< s, r, o >$, are the backbone for organizing and interpreting visual information. Upon completing the object detection phase, the scene graph construction process commences by leveraging the bounding box coordinates and object labels obtained from the object detection phase. These elements serve as the foundational components for building the scene graph, with each bounding box representing a distinct object and its label providing semantic context. A critical aspect of scene graph construction involves establishing spatial relationships between objects. All detected objects except for HUD elements are primarily designated as nodes within the scene graph. Within this framework, a special node, often termed the 'player' node, represents the entity under the agent's direct control and is instrumental in action execution.

We constructed the scene graph utilizing a mesh topology, commencing with the player node. The player node is connected to all other identified nodes within the frame. This design choice stems from the agent's control over the player node, suggesting that encapsulating information from other nodes into the player node may enhance the agent's performance. Spatial relationships between the player node and other nodes are delineated based on their relative positions within the frame, such as 'to the left of,' 'to the right of,' 'in front of,' and 'behind.'

We implement a distance-based threshold mechanism to establish spatial relationships between the remaining nodes. The Graph R-CNN model inspires this design [YLL+18], which defines relations between nodes based on bounding box overlap. We wanted to stay close to this constraint and employed a distance-based relation modelling. This threshold dictates that nodes must be within a certain Euclidean distance of each other to establish connections as formulated in 3.1

$$\text{Euclidean (centroid (node i)), centroid(node j))} \leq \text{ threshold} \tag{3.1}$$

After experimenting with various threshold values, we determined that setting the threshold to 10% of the total frame size, corresponding to a maximum Euclidean distance of 33 units, resulted in improved performance compared to other thresholds.

3.2.3 Scene Graph Processing

After obtaining the scene graph, the next step is to process it to make it compatible with the agent's utilization. To this end, we employ PyTorch Geometric[1], a library specifically designed for processing geometric structures including graphs.

We incorporate node attributes to enrich the information available to the agent. Specifically, we generate node attributes based on selected features. Drawing inspiration from Bortoletto et al. [BSB23], we preprocess the features before encoding them as node attributes to improve performance. This involves utilizing fully connected layers to process these three node features: bounding box values, colour attributes, and node types, which are then concatenated into a unified representation. We define the 'node type' input feature as the nature of the detected node, distinguishing between friendly and enemy nodes. We hypothesize that these input features enhance the scene graph representation, offering the agent a comprehensive understanding of the environment. Detailed exploration of different methodologies for incorporating input features is discussed in Chapter 4. Upon completing these pre-processing steps, the resulting PyTorch Geometric objects encapsulate a heterogeneous graph representation of the scene. This standardized format ensures compatibility with the agent's learning framework, facilitating seamless integration into the training process.

3.2.4 Agent Architecture

The agent architecture presented in Figure 3.6 closely resembles that described in Section 3.1.1, with modifications to the policy and value networks. Each node within the graph is enriched with an 8-dimensional feature vector, capturing details regarding its position, colour, and type. Subsequently, the graph and its feature set are processed by a shared GNN architecture. GNNs are transformable operations applied to all graph attributes (nodes, edges, global context) while maintaining graph symmetries, as introduced by Scarselli et al. [SGT+09]. We examined different GNN architectures to assess their suitability as feature extractors within our DRL framework. Among the models explored were GCNConv [KW17] and GATv2Conv [BAY22]. Compared to Graph Convolution Network (GCN), which utilizes the complete graph for learning node representations, Graph Attention Network (GAT) offers a more efficient approach by utilizing an attention-based mechanism to learn node representations. While GCN may face inefficiencies due to varying numbers of neighbours for each node [KW17], GAT addresses this challenge by determining the relative weights between interconnected nodes during the convolutional operation. However, for our work, after testing the different GNN architectures, we chose to utilize the GCNConv-based architecture for the policy network due to the reasons discussed in Section 4.2. Following the two GCNConv layers, we use a max-pooling function to downsample the feature maps and reduce their spatial dimensions. The output of the max-pooling function is then passed through two linear layers before being channelled into fully connected layers dedicated to the actor and critic components.

The training of the DRL agent for scene graph-based state representation approach mirrors that outlined in Section 3.1.2.

[1] https://pytorch-geometric.readthedocs.io/en/latest/

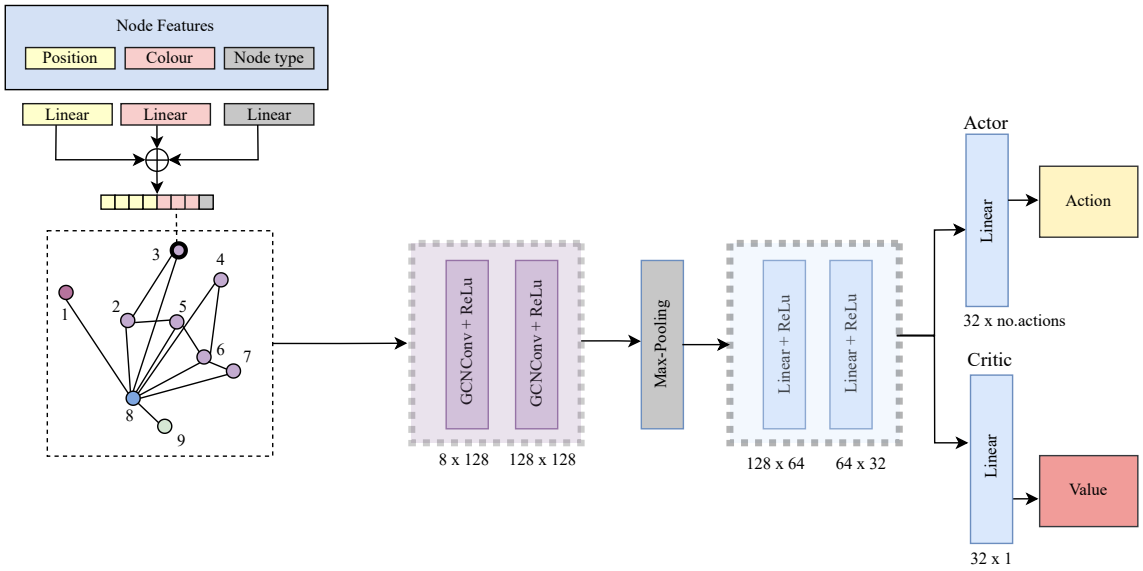


Figure 3.6: Policy and value network architecture of agent for the scene graph state representation. The input graph nodes are augmented with an 8-dimensional feature vector containing information about their position, colour, and type. These features undergo separate linear transformations and are concatenated into a unified 8-dimensional vector. The resulting graph is processed by a shared GNN architecture (GCNConv) for policy and value networks, followed by max-pooling across feature dimensions. The pooled vector then passes through a sequence of two linear layers before inputting into fully connected layers of the actor and critic components.

Our baseline is adapted from the model presented by Hung et al. [HDR+22]. This model employs the PPO algorithm to train agents across various Atari environments. Operating on input frames, the model utilizes a CNN-based function approximator, succeeded by fully connected layers, to sample actions and estimate values. The input to the CNN-based neural network is $84x84x4$ image. The network consists of three convolutional layers, each followed by a Rectified Linear Unit (ReLU) activation function. The first convolutional layer has 32 filters of size 8x8 with a stride of 4. The second convolutional layer has 64 filters of size 4x4 with a stride of 2, and the third convolutional layer has 64 filters of size 3x3 with a stride of 1. After the convolutional layers, the output is flattened into a one-dimensional tensor using the flatten operation. The flattened output is then passed through a fully connected layer, followed by a ReLU activation function before being fed into separate actor and critic components. We retained the hyperparameters as reported in the baseline model without modification. Our models closely resembles this baseline, with the primary distinction in choosing function approximators tailored to address diverse state representations.

4 Experiments and Analysis

This section will discuss the series of experiments conducted to analyze and improve the GNN agent performance. Initially, preliminary experiments were conducted to refine the model architecture. Once the architecture was finalised, we utilized it to conduct further experiments, such as exploring different formulations of scene graphs and varying input features to enhance the performance. These experiments were performed on a system equipped with an Intel Xeon E5-2637 chip, housing 16 cores operating at a frequency of 3.50 GHz. Furthermore, the system was equipped with an NVIDIA Titan X (Pascal) graphics card featuring 12 GB of memory with a bandwidth of 480.4 GB/s.

4.1 Training Parameters

We utilized the same hyperparameters as the baseline model[HDR+22] for all experiments. However, to provide comprehensive details, we present them in Table 4.1.

Hyperparameter	Value
Total training steps	10M
Optimizer	Adam Optimizer
Learning Rate (annealed)	$2.5e^{-4}$
Parallel Environments(N)	8
Steps per policy rollout(M)	128
Discount factor	0.99
Number of mini-batches	4
Policy update epochs	4
PPO general advantage estimation coefficient	0.95
PPO clipping coefficient	0.1
PPO entropy coefficient	0.01
PPO value function clipping coefficient	0.5
PPO gradient clipping norm	0.5

Table 4.1: Model hyperparameters

4.2 Evaluating GNN Architectures and Input Node Features

The experiment aims to assess various GNN architectures in conjunction with input node features to identify the most effective configuration to train agents in scene graph state representations. We assessed two distinct sets of node features across GNN architectures, GCNConv[KW17] and Gatv2Conv [BAY22], to identify the optimal combination. In the first experiment, we utilized raw bounding box values obtained from object detection and node types (friendly or enemy) as depicted in Figure 4.1a. These unprocessed features, presented as a single vector of length 5, were concatenated and encoded as node features. In the second experiment, as shown in Figure 4.1b, the node features comprised processed information about the node's bounding box, colour, and type. Instead of directly integrating these features into the graph, we initially passed each feature type through fully connected layers to extract processed representations. Subsequently, we concatenated these processed features into a unified vector of size 8, which was then encoded into the nodes. The two GNN architectures were GCNConv and GATv2Conv. While we ensured a modest parameter count, our main focus was assessing the models based on their reward performance.

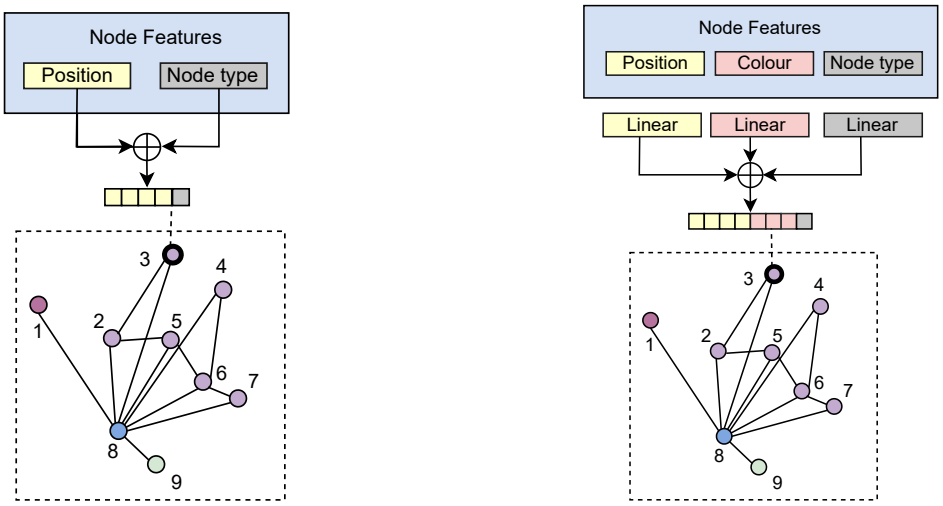
		GCNConv		**GATv2Conv**	
Games	**HNS**	**Feature 1**	**Feature 2**	**Feature 1**	**Feature 2**
SPI	1668.70	356 ± 34	410 ± 13	380 ± 12	**415 ± 7**
Frostbite	4334.70	1070 ± 83	1531 ± 558	1150 ± 147	**1690 ± 261**
Freeway	29.5	20 ± 0.3	**22.6 ± 2.3**	20 ± 0.71	20 ± 0.13

Table 4.2: Assessing the efficacy of GCNConv and GATv2Conv model architectures with both unprocessed (Feature 1) and processed (Feature 2) node features across three Atari2600 game environments with episodic reward as the metric, alongside human-normalized scores (HNS).

The results in Table 4.2 demonstrate that processed features (Feature 2) consistently yield higher rewards compared to unprocessed features (Feature 1) across all environments. This observation aligns with the findings reported by [BSB23]. Table 4.3 displays the parameter count for both GNN architectures. Despite having fewer parameters, the GCNConv architecture performs comparably to the GATv2Conv architecture. Therefore, based on these findings, we select the GCNConv model with processed features (Feature 2) as our final choice for subsequent experiments.

Model	Parameter Count
GCNConv	28627
GATv2Conv	96051

Table 4.3: Comparison of the number of parameters between GCNConv and GATv2Conv.

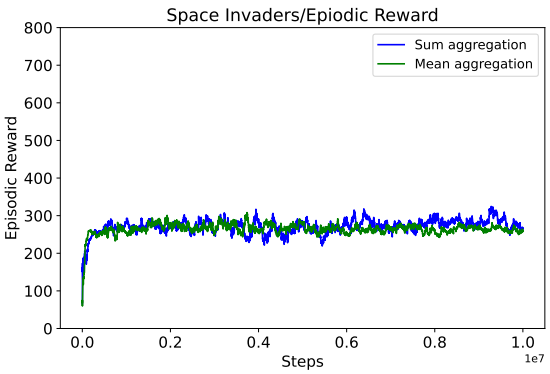


(a) Unprocessed node features of size 5 (Feature 1) **(b)** Processed node features of size 8 (Feature 2)

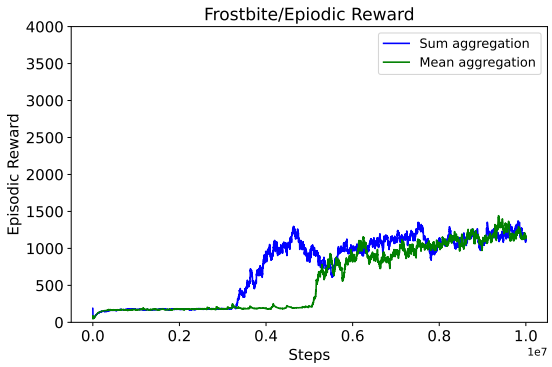
Figure 4.1: Representation of different node features. Figure 4.1a represents the unprocessed raw object information such as position and node type. Figure 4.1b represents a processed node feature containing information about object position, colour value and node type.

4.3 Evaluation of Aggregation Techniques on Nodes

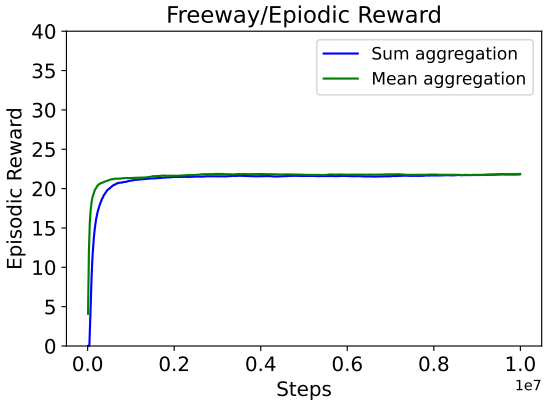
After selecting GCNConv as the optimal GNN feature extractor, we conduct an ablation study on node aggregation functions to assess their impact. Specifically, we compare two techniques: sum and mean aggregation. Initially, we employed 'sum' aggregation for GCN layers, aggregating neighbouring node features through summation without considering their relative significance. However, we believe that this approach proved sensitive to node degree distribution. Consequently, we experiment with 'mean' aggregation, where neighbouring node features are averaged for aggregation. The expectation was that this approach would ensure a balanced influence from all neighbours, irrespective of their degrees, thereby fostering more uniform information exchange [XHLJ19]. We conducted this experiment in all three game environments. Despite these adjustments, the post-training analysis revealed comparable performance between the two aggregation methods, as depicted in Figure 4.2. However, the model trained using the 'mean' aggregation function indicated a marginally faster convergence behaviour. Based on this finding, we propose that the exploration behaviour and the agent's ability to adapt to environmental changes could potentially overshadow any advantages of a more intricate aggregation technique. Following the negligible divergence between the two aggregation techniques, we maintained the 'sum' aggregation technique for all subsequent experiments.



(a) Space Invaders



(b) Frostbite

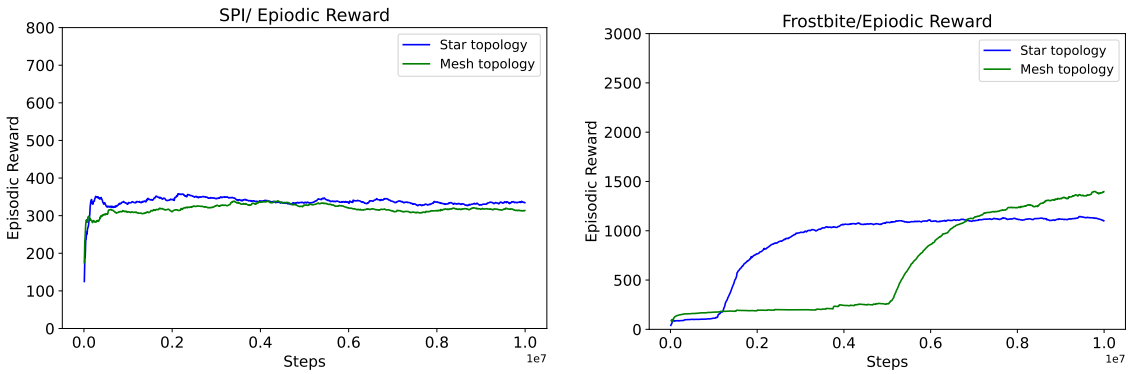


(c) Freeway

Figure 4.2: The performance between 'Mean' and 'Sum' aggregation techniques in message passing layers of the GNN is compared. The graph illustrates episodic returns during training against total steps. It is evident from the plot that both aggregation methods produce similar results across all three game environments.

4.4 Evaluation of Different Topology of Scene Graph

Modifying the node aggregation functions in the previous experiment did not improve the performance of the scene graph-based agent. Hence, we investigated the source of the scene graph structure. Initially, we adopted a mesh topology, wherein the connections were established based on spatial rule-based modelling as mentioned in section 3.2. The motivation to use mesh topology is that each node is typically connected to several other nodes, forming a dense and interconnected network. We hypothesize that this mesh topology allows for rich information exchange among nodes, facilitating complex decision-making processes. Nevertheless, we investigated a star topology to explore alternative graph topologies and their impact on agent performance. In contrast to the mesh topology, the star topology featured a more centralized network structure, with all nodes directly connected to a central player node. This architectural variation raised concerns regarding



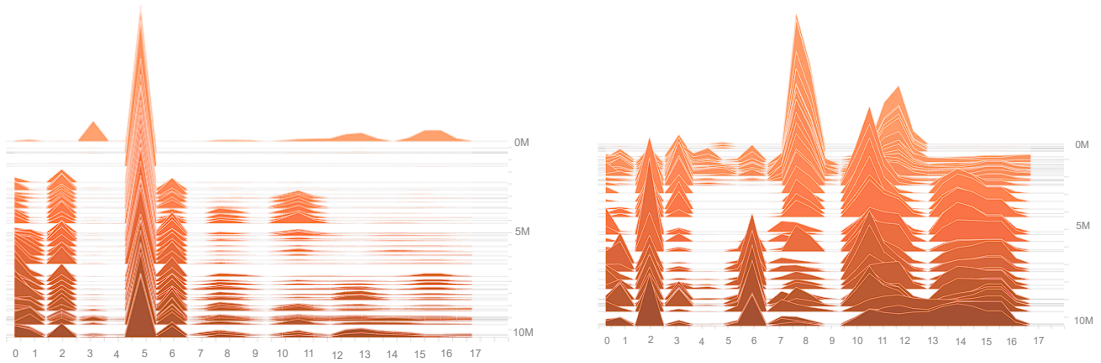
(a) Space Invaders **(b)** Frostbite

Figure 4.3: Episodic reward comparison of models trained on star and mesh topology in Space Invaders4.3a and Frostbite environment 4.3b. It is evident that the models trained with different topologies exhibited similar performance with a slight difference in the convergence behaviour.

potential limitations in information flow and the agent's ability to gather diverse insights from its environment. Surprisingly, our experimental findings revealed minimal performance differences between the two topologies.

Despite the inherent differences in network structure, the agent demonstrated consistent performance across both topologies, suggesting robustness to changes in graph topology. Interestingly, while both topologies exhibited comparable performance, we observed a slightly higher magnitude of value loss in the star topology. However, this discrepancy did not significantly impact overall performance, indicating the RL agent's resilience to topology variations. We hypothesized that introducing a global max-pooling layer after the output of GCNConv layers may have contributed to this resilience. Given the limited graph size and information aggregation into the player node in a single message-passing step, the spatial information loss resulting from max pooling could have minimized the noticeable impact of topology variations. To investigate this hypothesis further, we extracted features only from the player node in each graph constructed using the star topology. By feeding these features into fully connected layers, we eliminated the need for a max-pooling layer, thereby preserving spatial information throughout the network. We hypothesized this modification would have an impact on the performance of our model. However, both topologies yielded a similar performance, except that the star topology model exhibited a faster convergence behaviour. This experiment was conducted only on the Space Invaders and Frostbite game environment.

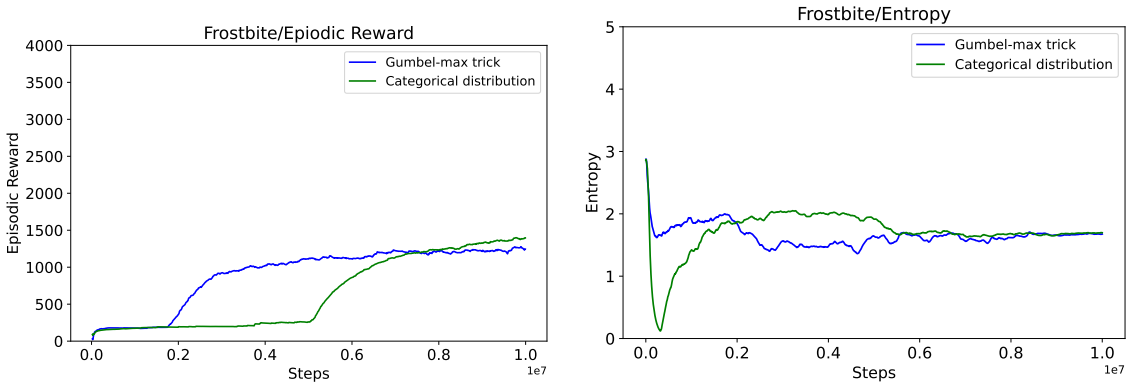
To identify the underlying reason for this performance discrepancy, we generated action histograms for models employing star and mesh topologies, as depicted in Figure 4.4. Despite the star topology model showcasing superior exploration capabilities 4.4b, than the model utilizing mesh topology 4.4a, this advantage does not lead to overall performance enhancements for the model.



(a) Action distribution in model using mesh topology **(b)** Action distribution in model using star topology

Figure 4.4: Action distribution comparison between models utilizing star and mesh topologies in the Frostbite game environment. While the star topology model demonstrates superior exploration capabilities, this enhancement does not translate into overall performance improvements for the models.

4.5 Assessing the Exploration Problem in Scene Graph-based Agents

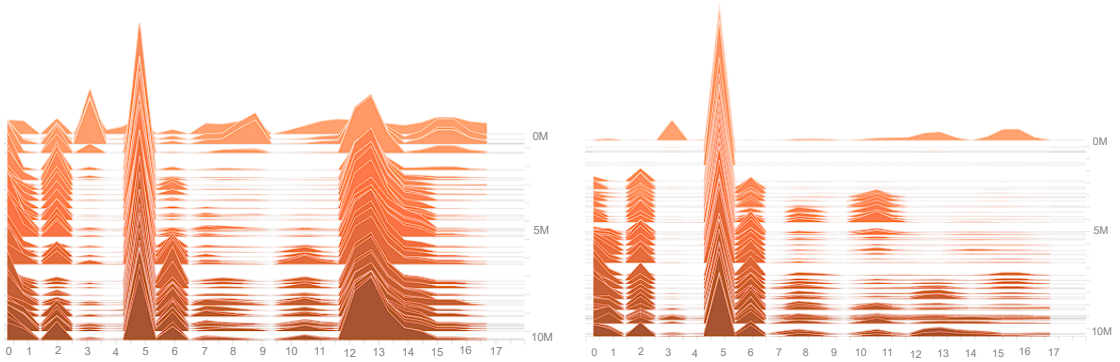


(a) Episodic reward comparison **(b)** Entropy loss comparison

Figure 4.5: Comparison of the model's performance using the Gumbel-max trick and sampling from a categorical distribution in the Frostbite game environment.

The exploration problem refers to the challenge of finding a balance between exploiting the current knowledge to maximize immediate rewards and exploring new actions or states to discover potentially better strategies in the long run. As evident from the histograms in the preceding section, a significant imbalance exists in action probabilities within the policy distribution. This imbalance often leads to premature convergence to sub-optimal policies due to inadequate exploration of the environment. An entropy coefficient is employed in the PPO algorithm's overall loss function to address this issue. Entropy, derived from the actor's logits, serves as a metric for quantifying the uncertainty of the policy. Maximum entropy is achieved when all actions are equally probable, while minimum entropy occurs when a single action dominates the policy. Higher entropy is desirable

in training DRL agents as it promotes policy diversity and helps prevent premature convergence to sub-optimal solutions. However, the anticipated behaviour of an agent entails initially high entropies that gradually decrease over time, eventually approaching zero. This decrease in entropy signifies the agent's increasing proficiency and preference for specific actions as it gains experience. Nevertheless, abrupt and drastic fluctuations in the entropy curve, without a gradual downward trend, indicate instability and may adversely affect training performance.



(a) Action distribution using Gumbel-max trick **(b)** Action distribution using categorical distribution

Figure 4.6: Comparison of the model's performance using the Gumbel-max trick and sampling from a categorical distribution in the Frostbite game environment.

As depicted in Figure 4.5b, our entropy curve displayed significant variability. Still, it failed to gradually converge to zero throughout the training phase, indicating sub-optimal learning by our agent. We hypothesize the reason to be under-fitting of the model, where the model fails to capture the underlying data structure, resulting in poor performance. We sought to enhance exploration by modifying the action sampling process to address this issue. Traditionally [MKS+13], actions are sampled from a categorical distribution using the softmax function. However, we investigated the action sampling process using the Gumbel-max trick. This technique involves sampling noise ε from the Gumbel distribution:

$$\varepsilon = -\log(-\log(U)) \tag{4.1}$$

where U is sampled from the uniform distribution in the range (0, 1). This noise is then added to the logits z before applying the 'argmax' function to select the action:

$$a = \text{argmax}_i \left(z_i + \varepsilon_i\right) \tag{4.2}$$

This modification aims to introduce randomness and exploration into the action selection process, potentially leading to improved learning performance by the agent.

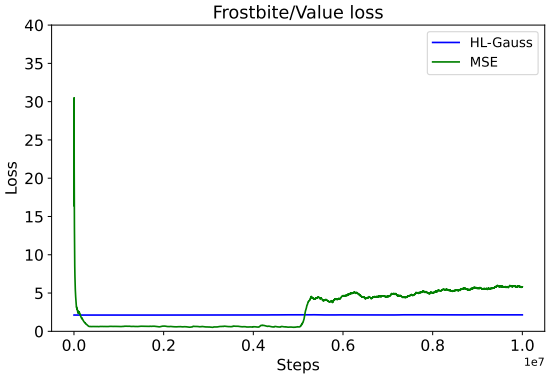
A comparative examination of action distributions in the Frostbite game environment highlights the effectiveness of the Gumbel-max trick in enhancing exploration when contrasted with the conventional categorical distribution-based sampling method. Figure 4.6a illustrates that the model employing the Gumbel-max trick explores additional actions, specifically actions 13 and 14, compared to the model that samples action from categorical distribution. Despite this improvement in exploration, it is observed that the enhanced exploration does not translate into improved agent

performance. Notably, the entropy curve, as shown in figure 5.4b, crucial for assessing exploration efficacy, did not gradually decrease in both cases, suggesting that excessive exploration may hinder learning. Hence, while the Gumbel-max trick shows promise in addressing exploration challenges, further refinement is necessary to reconcile enhanced exploration with improved agent performance. Despite implementing a specialized action sampling process, no improvement was observed in the performance of the scene graph-based model. Consequently, we investigate alternative techniques, such as value loss scaling in the following experiment, to enhance the models' performance.

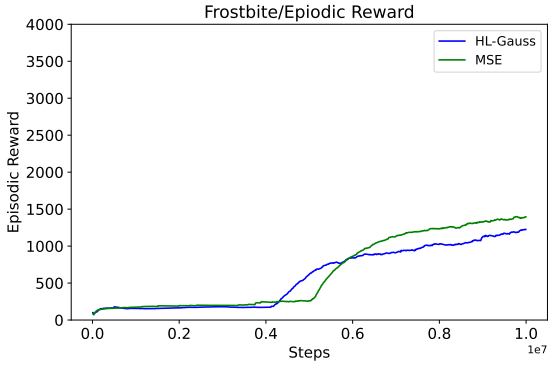
4.6 Evaluation of Value loss Scaling

As the previous experiments did not improve the performance of the scene graph model, we investigated an experiment where value functions are trained using classification loss as mentioned in [FOV+24]. Value function scaling aims to stabilize and improve the training process by constraining value estimates within an appropriate range. This mitigation strategy addresses issues like vanishing or exploding gradients, which can impede the convergence of RL algorithms. Central to DRL, value functions are typically trained using a Mean Squared Error (MSE) regression objective [SWD+17], aligning estimates with bootstrapped target values. A study by Farebrother et al. [FOV+24] proposed an alternative approach, suggesting that a classification objective for training value functions could enhance RL agent performance in Atari 2600 games. This method involved substituting the traditional MSE loss with categorical cross-entropy loss, which yielded more expressive representations and better management of noise and non-stationarity in value-based RL. While they applied value loss scaling to the Deep-Q Network (DQN) [MKS+15] algorithm, we implemented it within the PPO algorithm in our study. Adopting this methodology, our research aimed to validate its applicability to GNN-based RL agents. The transformation of scalar values into categorical distributions was facilitated using the HL-Gauss method advocated by [FOV+24].

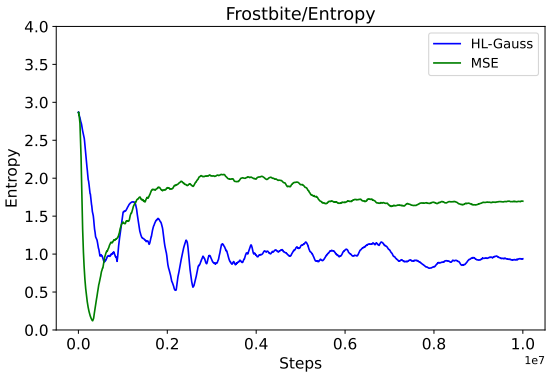
Training the model for 10 million steps with the classification loss, the study observed its effects and plotted the results. Figure 4.7a showcases the value loss magnitude obtained using both MSE and HL-Gauss loss functions, with the experiment conducted in the Frostbite environment. While the classification loss scaled the value loss consistently, it did not improve agent performance. Figure 4.7b illustrates that agents trained with the regression objective outperformed those trained with classification loss, albeit with a slightly delayed learning onset in the latter. The influence of value loss scaling extended to entropy loss, as illustrated in Figure 4.7c, where a decrease in magnitude was evident. We believe this reduction in entropy magnitude contributed to balancing the trade-off between exploration and exploitation, facilitating more efficient exploitation of rewarding actions by the agent. In summary, while adopting classification loss for value function training did not enhance the performance of GNN-based agents, it exhibited a faster learning onset.



(a) Value loss comparison



(b) Episodic reward comparison



(c) Entropy comparison

Figure 4.7: A comparison of performance between the HL-Gauss (classification) and MSE (regression) loss functions. The magnitude of value losses is depicted in Figure 4.7a, while Figure 4.7b illustrates the comparison of episodic returns during training. Additionally, Figure 4.7c showcases the comparison of entropy losses. Upon analyzing these figures, it can be concluded that, for our GNN-based agent, the regression loss performs slightly better than the classification loss.

5 Evaluation and Result

In the previous chapter, we looked at different experiments to improve the performance of scene graph agents. Next, we evaluate our best-performing object-centric and scene graph models. We compare the performance of the models in the chosen Atari environments with the baseline model against the HNS taken from [BPK+20].

	Scene Graph		**Object-centric**		**Image**	
Games	**Reward**	**HNS(%)**	**Reward**	**HNS(%)**	**Reward**	**HNS(%)**
SPI	410 ± 13	25%	598 ± 37	36%	**809 ± 285**	49%
Frostbite	1531 ± 558	35%	5470 ± 1674	126%	**7822 ± 461**	181%
Freeway	22.6 ± 2.3	77%	24.1 ± 0.5454	83%	**30.2 ± 0.6**	103%

Table 5.1: Evaluation of performance across different state representations (scene graph, object-centric, and default image) in Atari games. The table displays the mean scores obtained from 1000 episodes of evaluation across Atari games for each state representation, along with their corresponding percentages of HNS

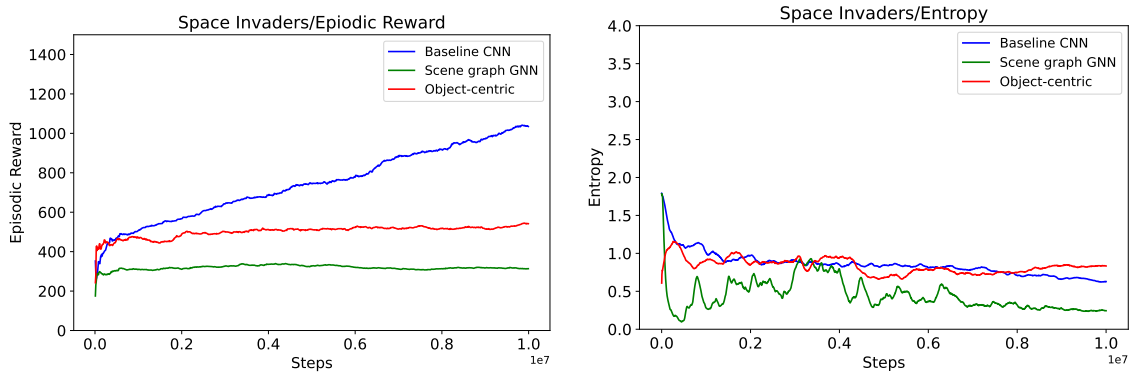
Table 5.1 presents the reward score comparison among models trained in object-centric and scene graph-based state representations alongside the baseline model trained in the conventional image states. Across all environments, the baseline model demonstrates superior performance compared to the other models. Learning from the object-centric state representation yields better reward scores in comparing the scene graph-based state representation and object-centric state representation. Notably, the models trained on these state representations are half the size of those used in learning from the image states, as shown in table 5.2. In the following sections, we will dissect the performance of these models in individual environments to identify the underlying factors contributing to their performance discrepancies.

Model	Number of Parameters
GCNConv	28K
Object-centric model	800K
Baseline CNN model	1.68M

Table 5.2: Number of parameters in GCNConv, Object-Centric Model, and CNN Model.

5.1 Space Invaders

Space Invaders is a valuable test bed for evaluating the performance of DRL models, providing a challenging yet accessible environment for testing various state space representations. Figure 3.5a shows an example frame from Space Invaders. In this game, players control a spaceship positioned at the bottom of the screen, tasked with defending against descending waves of alien invaders. The player aims to shoot down as many aliens as possible while avoiding enemy fire. Additionally, the player can seek temporary refuge under shields strategically positioned above the spaceship. These barriers provide limited cover but can be destroyed by enemy projectiles, adding an element of risk and strategy to the gameplay.



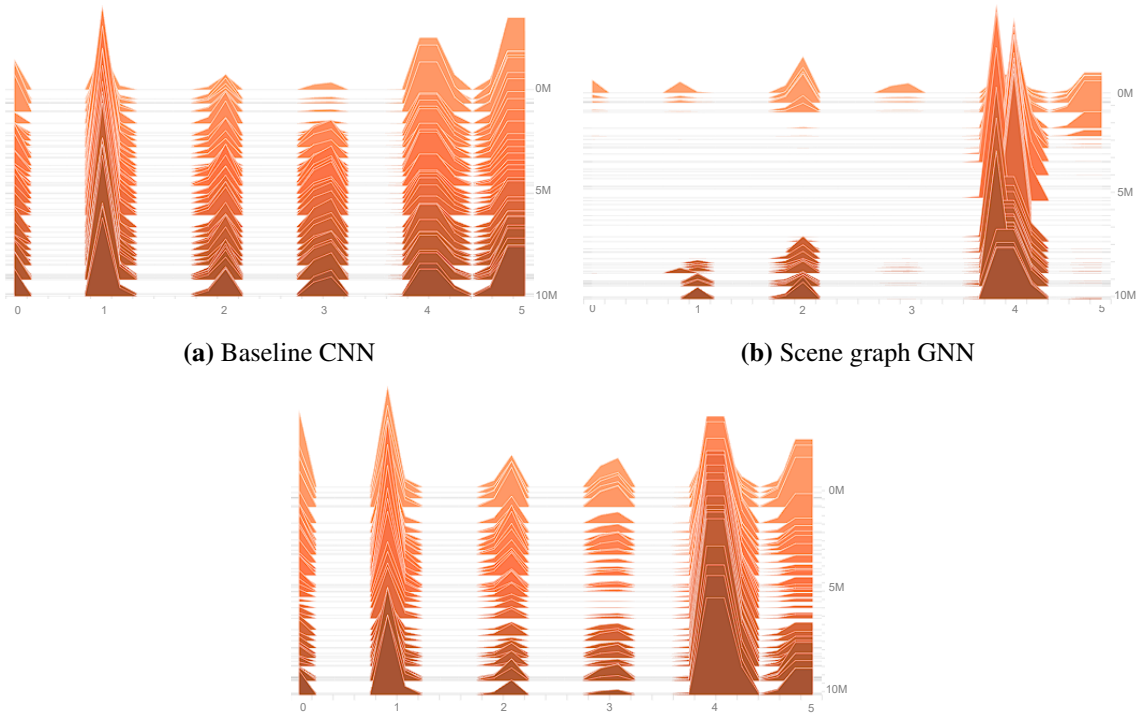
(a) Episodic reward comparison

(b) Entropy curve comparison

Figure 5.1: A comparison of post-training performance between the different state representations in Space Invaders game environment after evaluation for 1000 episodes.

To evaluate the performance of DRL models in Space Invaders, we employed three different state representations: scene graph, object-centric, and conventional image-based representations. Upon conducting experiments with these different representations, we observed notable disparities in the performance of the DRL models. The baseline model, trained using the conventional image-based representation, consistently outperformed the scene graph and object-centric models across various metrics, including reward scores and exploration efficiency. Notably, the scene graph model exhibited the poorest performance, struggling to achieve satisfactory results even after extensive training.

We conducted a detailed analysis of the state representations and decision-making processes to investigate the underlying reasons for the inferior performance of the scene graph and object-centric models. One major observation identified for the object-centric model was the insufficient diversity of nodes or objects in the Space Invaders game environment, leading to a skewed distribution of object information into slots. The initial hypothesis in the scene graph representation is that a higher frequency of player-alien connections would significantly influence the agent's action selection process. However, employing the GNNExplainer technique [YBY+19] yielded contrasting results. An analysis of a trained scene graph model was conducted using a set of randomly selected samples from [ZWL+19] dataset, depicting various Space Invaders scenarios. The trained scene graph agent consistently selected action '4 - (RIGHTFIRE)' across all samples. Figure 5.3 illustrates this process, with left images 5.3a, 5.3c, 5.3e displaying input samples featuring detected nodes and their IDs and right images 5.3b, 5.3d, 5.3f depicting corresponding reasoning subgraphs. Darker



(a) Baseline CNN

(b) Scene graph GNN

(c) Object-centric

Figure 5.2: In the Space Invaders environment, action sampling across different models is compared, with actions plotted on the X-axis and steps on the Y-axis. Notably, the Space Invaders game environment comprises 6 discrete action spaces. Upon examining these histograms, it becomes evident that the scene graph-based GNN model has consistently favoured a single action, indicating a lack of exploration.

edge connections signify greater importance in action selection, while transparent connections indicate lesser significance. Interestingly, the analysis revealed that the agent's action selection depended predominantly on the player-bullet connection rather than the anticipated player-alien connection, contradicting the initial hypothesis. This explainability test was specifically conducted only within the Space Invaders game environment.

Similarly, we hypothesize that the object-centric representation struggled to capture the subtle dynamics of the game due to the limited variety of object attributes. Furthermore, we analyzed the action sampling patterns of the models to assess their exploration and exploitation strategies as depicted in figure 5.2. Both the scene graph and object-centric models showed a strong bias towards a single action, typically firing in a specific direction, indicating a lack of exploration and adaptability. This deficiency suggests that the agents failed to learn an optimal strategy, likely due to insufficient data provided during training. Despite attempts to address this issue through techniques such as entropy regularization and alternative action sampling methods, the models did not exhibit significant performance improvements. Thus, it is concluded that the original cause of the poor performance was likely due to under-fitting caused by inadequate data generation.

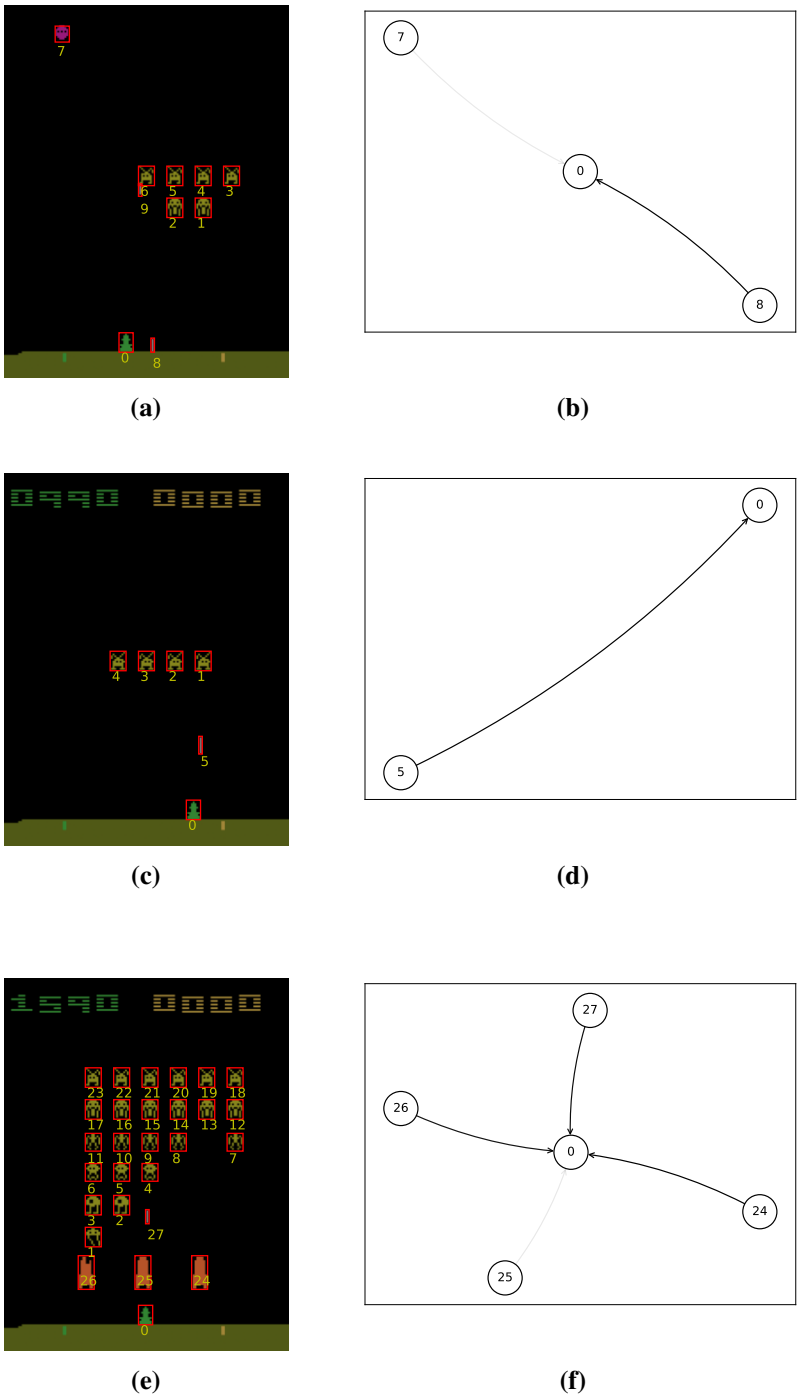
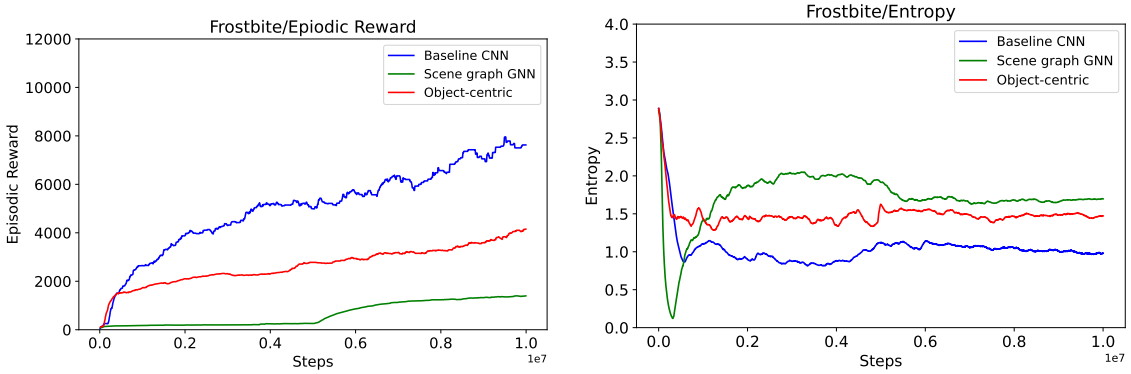


Figure 5.3: Explainability in the agent's action selection was investigated using the GNNExplainer technique [YBY+19] in Space Invaders. Detected nodes and their IDs are depicted on the left side, while respective reasoning subgraphs generated by GNNExplainer are shown on the right. Despite the availability of other nodes, the trained model consistently selected action '4' based on the 'bullet' node, as evident from the darker edges indicating higher importance in the action selection process.

5.2 Frostbite



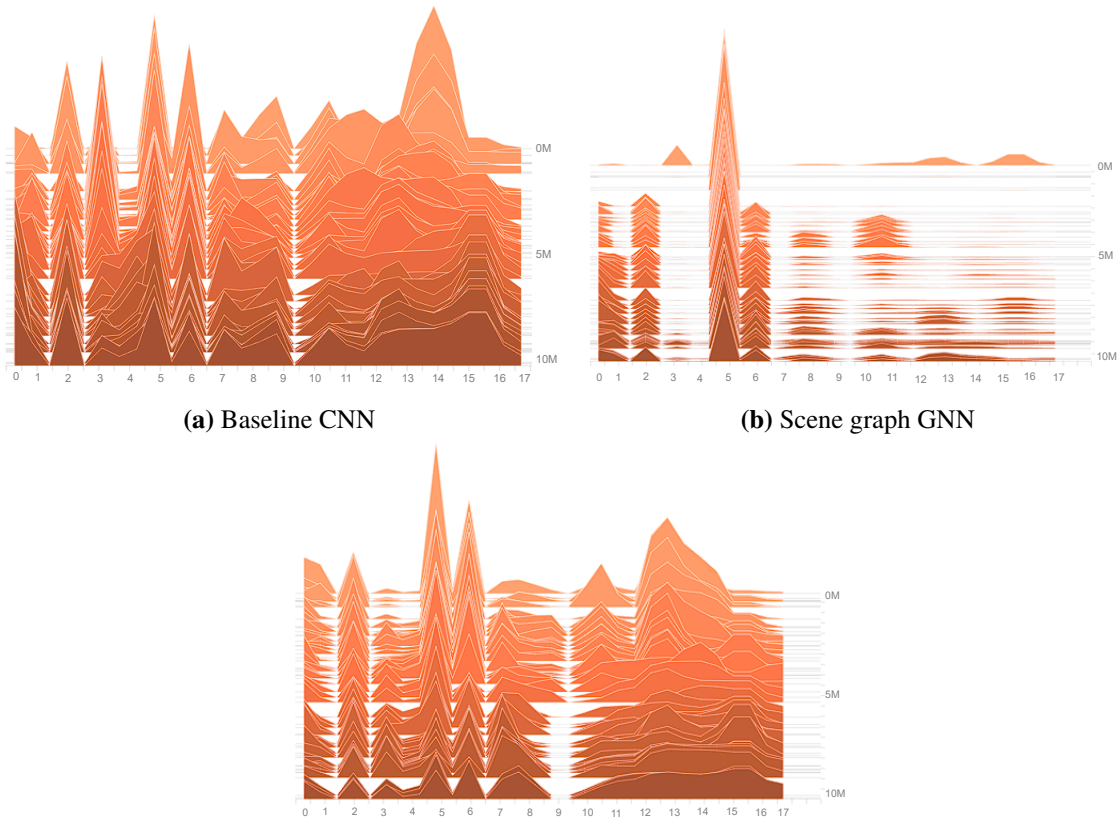
(a) Episodic reward comparison

(b) Entropy curve comparison

Figure 5.4: A comparison of post-training performance between the different state representations in Frostbite game environment after evaluation for 1000 episodes.

In the Frostbite game environment, players control an agent tasked with constructing an igloo before succumbing to freezing temperatures once the time limit elapses. To build the igloo, the agent must navigate between ice floes, earning progress for each unvisited set of white-coloured ice floes encountered in the current round. The ice floes move horizontally across the screen, each row moving in a different direction and increasing in speed as the player progresses. Along the way, the agent must evade hostile animals (coloured in orange and yellow), which results in a loss of life upon contact. The agent may gain bonus points by consuming fish (coloured green). Optimal performance requires strategic planning to achieve sub-goals such as visiting ice floes, avoiding animals, and collecting fish while focusing on completing the igloo and advancing to the next level. Frostbite is classified as a hard exploration game environment by [BSO+16] and offers 18 discrete action spaces, providing dense rewards.

In our evaluation of model performance, although we fell short of matching the scores achieved by the baseline model, we surpassed the HNS with our object-centric model. Notably, this model boasts a lightweight design compared to the baseline regarding model parameters. We attribute the performance of this model to the diverse range of object types within the environment, enabling the generation of meaningful object-centric representations (slots). The Frostbite environment features objects with rich attributes compared to the Space Invaders environment, potentially explaining the superior performance of the object-centric model in Frostbite. Contrarily, the scene graph model exhibited the lowest performance across our experiments. Despite modifications to the input graph topology and experimentation with different action sampling methods, we observed minimal improvement in the scene graph model's overall score. It is possible that inadequate data preprocessing or insufficient training steps contributed to the subpar performance. Notably, in Atari game environments, contemporary RL models are typically trained for over 100 million timesteps; for example, Agent 57[BPK+20]; using GNN in DRL may require longer training periods than other deep neural network models. However, we are clear that the primary issue appears to be related to agent exploration. Analysis of action histograms (Figure 5.5) indicates a bias towards a single action in the scene graph model, suggesting exploitation rather than exploration. This



(a) Baseline CNN

(b) Scene graph GNN

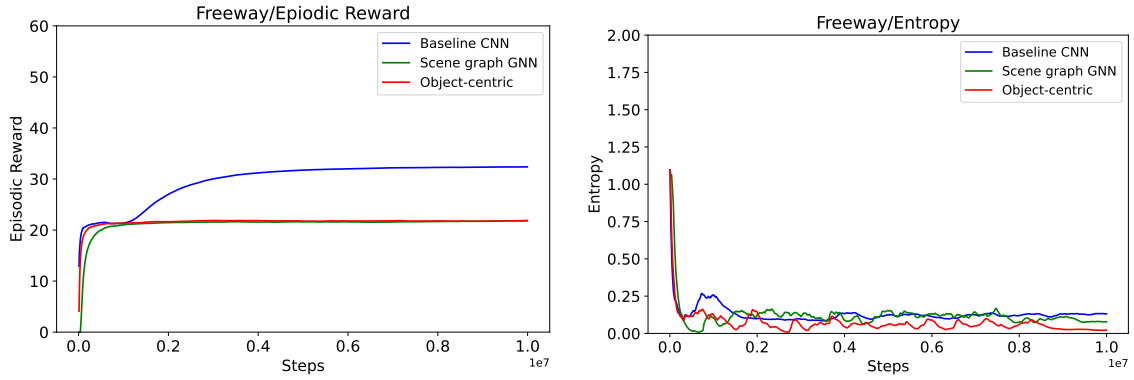
(c) Object-centric

Figure 5.5: Comparison of action sampling across different models in the Frostbite environment, with actions plotted on the X-axis and steps on the Y-axis. Frostbite game has an action space of 18 discrete actions. It is evident that the baseline model has explored different actions that ultimately lead to better performance.

tendency may stem from insufficient data provided to the scene graph model, as evidenced by its poor performance compared to the object-centric model. Thus, we attribute the scene graph model's under-performance to an under-fitting issue with inadequate data for effective learning.

5.3 Freeway

In the Freeway game environment, agents are tasked with manoeuvring chickens through a congested ten-lane highway, where the goal is to guide them upward to avoid colliding with oncoming traffic. Collisions with vehicles force the chicken backwards, hindering progress, while successful traversal to the opposite side rewards the agent with a score of 1; otherwise, the reward remains at 0. Despite its apparent simplicity, Freeway presents a notable exploration challenge due to the chickens' slow ascent and frequent vehicle collisions during random exploration, limiting their chances of achieving a positive reward. With only 3 discrete actions available, navigating the Freeway poses a sparse reward problem, further complicating exploration for the agent.



(a) Episodic reward comparison

(b) Entropy curve comparison

Figure 5.6: A comparison of post-training performance between the different state representations in Freeway game environment after evaluation for 1000 episodes.

Upon comparison, the baseline model consistently outperformed our models, likely due to ineffective data representation techniques and limited exploration capabilities. Given the sparse reward structure and small action space, all models struggled to explore the environment effectively, as indicated by consistently low entropy curves. Initially, rewards for all models ranged between 20 and 22 as they approached 100K timesteps. However, while the baseline model adapted and discovered a more effective policy around the 1M timestep mark, resulting in increased rewards, our models failed to adjust and continued with their existing policies, yielding consistent rewards throughout the 10M timesteps. This lack of adaptation highlights the inadequate exploration exhibited by all three models, as evidenced by their repetitive selection of the same actions.

Our analysis highlights the challenges of employing different state representations in training DRL models in complex Atari games. While scene graphs and object-centric representations offer promising avenues for capturing structured information, their effectiveness heavily depends on the diversity and richness of the underlying state. While our models failed to outperform the baseline model, it is worth noting that the object-centric model successfully surpassed the HNS in the Frostbite game environment.

6 Discussion and Limitations

Based on the findings presented earlier, we aim to provide conclusive answers to the research question posed in the Introduction: We will begin by addressing the impact of different state representations, discussing potential enhancements to this study, and examining any limitations.

6.1 Impact of State Representations on RL Agents.

From this work, we can observe that the selection of state representation plays a pivotal role in shaping the performance of RL agents. We hypothesized that representations capturing relevant features of the environment concisely and informatively could expedite learning by reducing the dimensionality of the state space and directing the agent's focus toward the most salient aspects of the environment. Our study revealed distinct performance trends across different state representations, with notable implications for RL agents' learning efficiency and adaptability.

Our investigation observed that the object-centric state representation facilitated marginally faster learning across all three environments than pixel-based and scene graph-based representations. This expedited learning process can be attributed to the object-centric representation's ability to encapsulate specific environmental features effectively. However, while representations utilizing compact details may offer initial learning advantages, they may encounter limitations in long-term performance. This phenomenon is exemplified by the superior performance of the baseline model, which is learned directly from pixel values. Abstract representations, such as those derived from pixel values, capture the underlying structure of the environment, enabling the agent to recognize patterns and similarities across diverse states.

The object-centric state representation, for instance, organizes individual objects within a frame into distinct slots based on their attributes. However, we hypothesize that the slot attention model fails to account for the quantity of each object type, grouping similar objects into a single slot. Consequently, while the agent may learn rapidly from specific object attributes, it may struggle to generalize knowledge across states with varying object quantities. On the other hand, the scene graph-based representation offers insights into object quantities via node representation but may encounter challenges related to insufficient feature complexity. We experimented with different node embeddings across varying dimensions and ultimately settled on an embedding dimension of size 8. However, this adjustment did not yield optimal results for the graph representation. Despite the relatively compact size of our graph structure, it featured a higher degree of variability in nodes between observations. We hypothesize that this complexity may have hindered the ability of the GNN-based agent to learn from this representation effectively.

The training paradigm employed is another factor influencing the impact of different state representations. Our study adhered to a common training paradigm established by the baseline representation without tailoring hyperparameters to accommodate alternative representations. This uniform approach may have favoured pixel-based learning paradigms, potentially limiting the performance of alternative representations.

Our findings underscore the impact of different state representations on RL agent performance, each presenting unique advantages and disadvantages. While representations emphasizing specific details may facilitate rapid learning, they may encounter long-term performance and generalization challenges. Contrarily, abstract representations offer broader generalization capabilities but may require sophisticated learning mechanisms to leverage effectively.

6.2 Limitations

Our methodology has its limitations, which we must acknowledge and address. One significant limitation is the computational overhead of preprocessing raw images required to convert them into specific state representations. In particular, the construction of scene graphs imposes a substantial processing burden, significantly increasing the time required to complete a single update step. On average, utilizing our hardware setup, a single update step (one rollout and one learning step) in the RL training took 34 seconds. The object-centric model takes around 19 seconds to complete a single update step. These processing times are notably higher than the baseline model, which takes approximately 7 seconds for a single update step. This is because the baseline model directly operates on raw pixel data without complex preprocessing. The processing time was doubled when we used Faster R-CNN [RHGS16b] as our object detection model.

Additionally, our approach faces challenges in incorporating temporal information between observations. Our state representation methodology primarily focuses on capturing spatial information within individual frames. As a result, we encountered difficulties integrating temporal context into our state representations. This limitation restricts our agent's ability to learn from sequential observations over time, potentially hindering its performance in tasks that require an understanding of temporal dynamics and dependencies.

Since the Atari environment provides images that differ significantly from real-world images, existing models designed for constructing scene graphs in real-world contexts, such as those by Yang et al. [YLL+18], Xu et al. [XZCF17], and Zellers et al. [ZYTC18], were not suitable for our purposes. Consequently, we resorted to employing rule-based scene graph construction techniques, using some of our domain knowledge to tailor the approach to the unique characteristics of the Atari observations.

6.3 Future Work

In considering future directions, the current study's findings and limitations present several promising aspects for enhancing understanding, addressing existing challenges, and expanding research scope in the domain of state representations in RL.

One promising direction involves the integration of alternative intermediate state representations for training RL agents in Atari game environments. These untested ideas include the construction of a hybrid state representation by combining multiple modalities, such as pixel values, edge maps generated by edge detection algorithms like canny edge detection algorithm [1], and object detection. This approach could provide a comprehensive view of the game environment, capturing both low-level visual features and high-level semantic information. While our current work focuses solely on spatial information within image frames, future efforts could explore incorporating temporal information between consecutive frames. Leveraging the 'Framestack' wrapper in the Atari environment could offer access to this information, although integrating it effectively into intermediate state representations presents a challenge that warrants further investigation.

Exploring different training setups is another avenue worth exploring. While our experiments were limited to training for 10M steps, contemporary RL agents often require substantially longer training times, typically exceeding 100M steps [BPK+20; SAH+20]. Therefore, extending the training duration or exploring alternative training setups could unveil decisive advantages that may not be apparent within our current experimental framework. Experimentation with different graph topologies and feature embeddings holds promise for the scene graph-based state representation. While our study utilized fully meshed and star topologies and encoded restricted information into nodes, significant room exists for exploring various graph structures and feature representations. This exploration could lead to better performances, extending the field of state representations in RL.

Furthermore, a critical area for future work involves developing suitable evaluation techniques for assessing the efficacy of these state representations. For instance, our construction of an object-centric state representation using a slot attention mechanism raises questions regarding whether the resulting slots effectively capture the desired information about objects in the image. Establishing robust evaluation methods will be paramount in providing meaningful insights into the performance and suitability of different state representations.

In summary, the future research aspects we presented are comprehensive and well-considered. They could focus on integrating diverse state representations, exploring alternative training setups, refining existing representation techniques, and developing robust evaluation methodologies. By pursuing these avenues, we can deepen our understanding of state representations in RL and advance the capabilities of RL agents in complex environments, providing a solid foundation for further research in the field.

[1] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

7 Conclusion

This work commenced with the objective of assessing the efficacy of various intermediate-state representations for training DRL agents. We utilized Atari 2600 game environments to train the agents and evaluate the performance of intermediate state representations. Object-centric and scene graph-based intermediate state representations were constructed from the image data provided by Atari game environments. To achieve this, we utilized a slot-attention mechanism for object-centric state representation construction and spatial rule-based relation modelling for scene graph-based state representation construction. Following construction, we devised appropriate agent architectures, including object-centric agents and graph neural network GNN-based agents, to extract features from the constructed intermediate state representations. Subsequently, these models underwent training for 10 million steps utilizing the PPO algorithm. Various experiments were conducted to enhance the agents' performance, culminating in evaluating their performance against a baseline model trained on raw pixel data. While the results indicate that the baseline model outperformed our models, we are contended in achieving a result: surpassing the HNS in one of the Atari 2600 game environments with our lightweight model architecture and intermediate state representations. In summary, this study evaluated the impact of different state representations on the performance of DRL agents.

Bibliography

[BAY22] S. Brody, U. Alon, E. Yahav. *How Attentive are Graph Attention Networks?* 2022. arXiv: 2105.14491 [cs.LG] (cit. on pp. 26, 30).

[BNVB13a] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* 47 (June 14, 2013), pp. 253–279. ISSN: 1076-9757. DOI: 10.1613/jair.3912. URL: https://www.jair.org/index.php/jair/article/view/10819 (visited on 11/12/2023) (cit. on pp. 12, 17).

[BNVB13b] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279. ISSN: 1076-9757. DOI: 10.1613/jair.3912. URL: http://dx.doi.org/10.1613/jair.3912 (cit. on p. 17).

[BPK+20] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, C. Blundell. "Agent57: Outperforming the Atari Human Benchmark". In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, Nov. 21, 2020, pp. 507–517. URL: https://proceedings.mlr.press/v119/badia20a.html (visited on 11/11/2023) (cit. on pp. 11, 17, 39, 43, 49).

[BSB23] M. Bortoletto, L. Shi, A. Bulling. *Neural Reasoning About Agents' Goals, Preferences, and Actions*. 2023. arXiv: 2312.07122 [cs.AI] (cit. on pp. 26, 30).

[BSO+16] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos. *Unifying Count-Based Exploration and Intrinsic Motivation*. 2016. arXiv: 1606.01868 [cs.AI] (cit. on p. 43).

[BSO22] M. Bartlett, T. C. Stewart, J. Orchard. "Biologically-Based Neural Representations Enable Fast Online Shallow Reinforcement Learning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society* 44.44 (2022). URL: https://escholarship.org/uc/item/49v0x3rz (visited on 11/04/2023) (cit. on p. 14).

[CP19] E. Crawford, J. Pineau. "Spatially invariant unsupervised object detection with convolutional neural networks". In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA: AAAI Press, Jan. 27, 2019, pp. 3412–3420. ISBN: 978-1-57735-809-1. DOI: 10.1609/aaai.v33i01.33013412. URL: https://dl.acm.org/doi/10.1609/aaai.v33i01.33013412 (visited on 11/13/2023) (cit. on p. 15).

[DBG+24] Q. Delfosse, J. Blüml, B. Gregori, S. Sztwiertnia, K. Kersting. *OCAtari: Object-Centric Atari 2600 Reinforcement Learning Environments*. 2024. arXiv: 2306.08649 [cs.LG] (cit. on pp. 15, 19, 23, 24).

[DD20] Z. Ding, H. Dong. "Challenges of Reinforcement Learning". In: *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Ed. by H. Dong, Z. Ding, S. Zhang. Singapore: Springer, 2020, pp. 249–272. ISBN: 9789811540950. DOI: 10.1007/978-981-15-4095-0_7. URL: https://doi.org/10.1007/978-981-15-4095-0_7 (visited on 11/10/2023) (cit. on p. 11).

[Fol] D. J. Foley. "Model-Based Reinforcement Learning in Atari 2600 Games". In: () (cit. on p. 17).

[FOV+24] J. Farebrother, J. Orbay, Q. Vuong, A. A. Taïga, Y. Chebotar, T. Xiao, A. Irpan, S. Levine, P. S. Castro, A. Faust, A. Kumar, R. Agarwal. *Stop Regressing: Training Value Functions via Classification for Scalable Deep RL*. 2024. arXiv: 2403.03950 [cs.LG] (cit. on p. 36).

[HDR+22] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, W. Wang. "The 37 Implementation Details of Proximal Policy Optimization". In: *ICLR Blog Track*. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. 2022. URL: https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/ (cit. on pp. 23, 27, 29).

[HLK+20] M. Hildebrandt, H. Li, R. Koner, V. Tresp, S. Günnemann. *Scene Graph Reasoning for Visual Question Answering*. 2020. arXiv: 2007.01072 [cs.LG] (cit. on p. 16).

[HWL+23] N. Heravi, A. Wahid, C. Lynch, P. Florence, T. Armstrong, J. Tompson, P. Sermanet, J. Bohg, D. Dwibedi. *Visuomotor Control in Multi-Object Scenes Using Object-Aware Representations*. Mar. 12, 2023. DOI: 10.48550/arXiv.2205.06333. arXiv: 2205.06333[cs]. URL: http://arxiv.org/abs/2205.06333 (visited on 11/13/2023) (cit. on p. 15).

[HYS+18] Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. H.-L. Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao, H.-W. Hsiao, S.-P. Lai, C.-Y. Lee. *Virtual-to-Real: Learning to Control in Visual Semantic Segmentation*. 2018. arXiv: 1802.00285 [cs.CV] (cit. on p. 15).

[JPC+22] S. Ji, S. Pan, E. Cambria, P. Marttinen, P. S. Yu. "A Survey on Knowledge Graphs: Representation, Acquisition, and Applications". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.2 (Feb. 2022), pp. 494–514. ISSN: 2162-2388. DOI: 10.1109/tnnls.2021.3070843. URL: http://dx.doi.org/10.1109/TNNLS.2021.3070843 (cit. on p. 16).

[KEH22] K. N. Kumar, I. Essa, S. Ha. "Graph-based Cluttered Scene Generation and Interactive Exploration using Deep Reinforcement Learning". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 7521–7527. DOI: 10.1109/ICRA46639.2022.9811874 (cit. on p. 16).

[KLH+21] R. Koner, H. Li, M. Hildebrandt, D. Das, V. Tresp, S. Günnemann. "Graphhopper: Multi-hop Scene Graph Reasoning for Visual Question Answering". In: *The Semantic Web – ISWC 2021*. Ed. by A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P. Barnaghi, A. Haller, M. Dragoni, H. Alani. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 111–127. ISBN: 978-3-030-88361-4. DOI: 10.1007/978-3-030-88361-4_7 (cit. on p. 16).

[KW17] T. N. Kipf, M. Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG] (cit. on pp. 26, 30).

[LGLS22] X. Li, D. Guo, H. Liu, F. Sun. "Embodied Semantic Scene Graph Generation". In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by A. Faust, D. Hsu, G. Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, Aug. 2022, pp. 1585–1594. URL: https://proceedings.mlr.press/v164/li22e.html (cit. on p. 16).

[LNF+19] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, T. Rocktäschel. *A Survey of Reinforcement Learning Informed by Natural Language*. 2019. arXiv: 1906.03926 [cs.LG] (cit. on p. 16).

[LRC+21] Y. Lu, H. Rai, J. Chang, B. Knyazev, G. Yu, S. Shekhar, G. W. Taylor, M. Volkovs. "Context-Aware Scene Graph Generation With Seq2Seq Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 15931–15941 (cit. on p. 16).

[LRY+21] N. Le, V. S. Rathour, K. Yamazaki, K. Luu, M. Savvides. *Deep Reinforcement Learning in Computer Vision: A Comprehensive Survey*. 2021. arXiv: 2108.11510 [cs.CV] (cit. on p. 16).

[LWP+20] Z. Lin, Y.-F. Wu, S. Peri, B. Fu, J. Jiang, S. Ahn. *Improving Generative Imagination in Object-Centric World Models*. Oct. 5, 2020. DOI: 10.48550/arXiv.2010.02054. arXiv: 2010.02054[cs,stat]. URL: http://arxiv.org/abs/2010.02054 (visited on 11/13/2023) (cit. on p. 15).

[LWU+20] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, T. Kipf. *Object-Centric Learning with Slot Attention*. Oct. 14, 2020. DOI: 10.48550/arXiv.2006.15055. arXiv: 2006.15055[cs,stat]. URL: http://arxiv.org/abs/2006.15055 (visited on 11/14/2023) (cit. on pp. 15, 19, 20).

[MBM+16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. arXiv: 1602.01783 [cs.LG] (cit. on p. 13).

[MBM21] M. Mahe, P. Belamri, J. B. Martin. *Towards a Sample Efficient Reinforcement Learning Pipeline for Vision Based Robotics*. May 20, 2021. DOI: 10.48550/arXiv.2105.09719. arXiv: 2105.09719[cs]. URL: http://arxiv.org/abs/2105.09719 (visited on 11/14/2023) (cit. on p. 13).

[MKS+13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. *Playing Atari with Deep Reinforcement Learning*. Dec. 19, 2013. DOI: 10.48550/arXiv.1312.5602. arXiv: 1312.5602[cs]. URL: http://arxiv.org/abs/1312.5602 (visited on 11/12/2023) (cit. on pp. 11, 15, 17, 35).

[MKS+15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015). Number: 7540 Publisher: Nature Publishing Group, pp. 529–533. ISSN: 1476-4687. DOI: 10.1038/nature14236. URL: https://www.nature.com/articles/nature14236 (visited on 11/10/2023) (cit. on pp. 11, 13, 15, 36).

[MRM11] O.-a. Maillard, D. Ryabko, R. Munos. “Selecting the State-Representation in Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/970af30e481057c48f87e101b61e6994-Paper.pdf (cit. on p. 11).

[MSIB20] N. Mazyavkina, S. Sviridov, S. Ivanov, E. Burnaev. *Reinforcement Learning for Combinatorial Optimization: A Survey*. 2020. arXiv: 2003.03600 [cs.LG] (cit. on p. 16).

[RHGS16a] S. Ren, K. He, R. Girshick, J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV] (cit. on pp. 16, 21).

[RHGS16b] S. Ren, K. He, R. Girshick, J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Jan. 6, 2016. DOI: 10.48550/arXiv.1506.01497. arXiv: 1506.01497[cs]. URL: http://arxiv.org/abs/1506.01497 (visited on 11/13/2023) (cit. on pp. 23, 24, 48).

[SAH+20] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver. “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model”. In: *Nature* 588.7839 (Dec. 24, 2020), pp. 604–609. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-020-03051-4. arXiv: 1911.08265[cs, stat]. URL: http://arxiv.org/abs/1911.08265 (visited on 11/10/2023) (cit. on pp. 11, 17, 49).

[SGT+09] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini. “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605 (cit. on p. 26).

[SHS+17] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. Dec. 5, 2017. DOI: 10.48550/arXiv.1712.01815. arXiv: 1712.01815[cs]. URL: http://arxiv.org/abs/1712.01815 (visited on 11/10/2023) (cit. on p. 11).

[SLM+17] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel. *Trust Region Policy Optimization*. Apr. 20, 2017. DOI: 10.48550/arXiv.1502.05477. arXiv: 1502.05477[cs]. URL: http://arxiv.org/abs/1502.05477 (visited on 08/07/2023) (cit. on p. 14).

[SRB+17] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, T. Lillicrap. *A simple neural network module for relational reasoning*. June 5, 2017. DOI: 10.48550/arXiv.1706.01427. arXiv: 1706.01427[cs]. URL: http://arxiv.org/abs/1706.01427 (visited on 11/13/2023) (cit. on p. 15).

[SWD+17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. *Proximal Policy Optimization Algorithms*. Aug. 28, 2017. DOI: 10.48550/arXiv.1707.06347. arXiv: 1707.06347[cs]. URL: http://arxiv.org/abs/1707.06347 (visited on 11/17/2023) (cit. on pp. 13, 14, 22, 36).

[VCC+20] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. B. Tenenbaum, S. Levine. *Entity Abstraction in Visual Model-Based Reinforcement Learning*. May 6, 2020. DOI: 10.48550/arXiv.1910.12827. arXiv: 1910.12827[cs,stat]. URL: http://arxiv.org/abs/1910.12827 (visited on 11/13/2023) (cit. on p. 15).

[WPC+21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.2978386. URL: http://dx.doi.org/10.1109/TNNLS.2020.2978386 (cit. on p. 16).

[XHLJ19] K. Xu, W. Hu, J. Leskovec, S. Jegelka. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG] (cit. on p. 31).

[XZCF17] D. Xu, Y. Zhu, C. B. Choy, L. Fei-Fei. *Scene Graph Generation by Iterative Message Passing*. 2017. arXiv: 1701.02426 [cs.CV] (cit. on p. 48).

[YBY+19] R. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec. *GNNExplainer: Generating Explanations for Graph Neural Networks*. 2019. arXiv: 1903.03894 [cs.LG] (cit. on pp. 40, 42).

[YGGT19] Y. Ye, D. Gandhi, A. Gupta, S. Tulsiani. *Object-centric Forward Modeling for Model Predictive Control*. Oct. 8, 2019. DOI: 10.48550/arXiv.1910.03568. arXiv: 1910.03568[cs]. URL: http://arxiv.org/abs/1910.03568 (visited on 11/04/2023) (cit. on p. 15).

[YLL+18] J. Yang, J. Lu, S. Lee, D. Batra, D. Parikh. *Graph R-CNN for Scene Graph Generation*. 2018. arXiv: 1808.00191 [cs.CV] (cit. on pp. 16, 23, 25, 48).

[YWBA23] J. Yoon, Y.-F. Wu, H. Bae, S. Ahn. *An Investigation into Pre-Training Object-Centric Representations for Reinforcement Learning*. June 12, 2023. DOI: 10.48550/arXiv.2302.04419. arXiv: 2302.04419[cs]. URL: http://arxiv.org/abs/2302.04419 (visited on 11/04/2023) (cit. on pp. 15, 19).

[YZK+20] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, R. Fergus. *Improving Sample Efficiency in Model-Free Reinforcement Learning from Images*. July 9, 2020. DOI: 10.48550/arXiv.1910.01741. arXiv: 1910.01741[cs,stat]. URL: http://arxiv.org/abs/1910.01741 (visited on 11/14/2023) (cit. on p. 13).

[ZCH+21] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun. *Graph Neural Networks: A Review of Methods and Applications*. 2021. arXiv: 1812.08434 [cs.LG] (cit. on p. 16).

[ZRS+18] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, M. Shanahan, V. Langston, R. Pascanu, M. Botvinick, O. Vinyals, P. Battaglia. *Relational Deep Reinforcement Learning*. June 28, 2018. DOI: 10.48550/arXiv.1806.01830. arXiv: 1806.01830[cs,stat]. URL: http://arxiv.org/abs/1806.01830 (visited on 11/13/2023) (cit. on p. 15).

[ZSM20] A. Zadaianchuk, M. Seitzer, G. Martius. *Self-supervised Visual Reinforcement Learning with Object-centric Representations*. 2020. arXiv: 2011.14381 [cs.LG] (cit. on pp. 11, 19).

[ZWL+19] R. Zhang, C. Walshe, Z. Liu, L. Guan, K. S. Muller, J. A. Whritner, L. Zhang, M. M. Hayhoe, D. H. Ballard. *Atari-HEAD: Atari Human Eye-Tracking and Demonstration Dataset*. 2019. arXiv: 1903.06754 [cs.LG] (cit. on pp. 21, 40).

[ZYTC18] R. Zellers, M. Yatskar, S. Thomson, Y. Choi. *Neural Motifs: Scene Graph Parsing with Global Context*. 2018. arXiv: 1711.06640 [cs.CV] (cit. on p. 48).

[ZZH+22] Y. Zhou, H. Zheng, X. Huang, S. Hao, D. Li, J. Zhao. "Graph Neural Networks: Taxonomy, Advances, and Trends". In: *ACM Transactions on Intelligent Systems and Technology* 13.1 (Jan. 2022), pp. 1–54. ISSN: 2157-6912. DOI: 10.1145/3495161. URL: http://dx.doi.org/10.1145/3495161 (cit. on p. 16).

[ZZJ+22] G. Zhu, L. Zhang, Y. Jiang, Y. Dang, H. Hou, P. Shen, M. Feng, X. Zhao, Q. Miao, S. A. A. Shah, M. Bennamoun. *Scene Graph Generation: A Comprehensive Survey*. June 22, 2022. DOI: 10.48550/arXiv.2201.00443. arXiv: 2201.00443[cs]. URL: http://arxiv.org/abs/2201.00443 (visited on 11/14/2023) (cit. on p. 16).

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted hard copies.

---

place, date, signature