Chair of Computer Science 6 (Machine Learning and Reasoning)

RWTH AACHEN UNIVERSITY

The present work was submitted to the Chair of Machine Learning and Reasoning.

Learning Symbolic State Descriptions From Images Using Scene Graph Generation

Lernen symbolischer Zustandsbeschreibungen aus Bildern mithilfe von Szenengraphen

Master's Thesis

Presented by / Vorgelegt von

Nirmal Maheshwari
414337

Supervised by / Betreut von Dr. Till Hofmann

Examiner/ 1. Prüfer Prof. Hector Geffner, Ph.D.

Examiner/ 2. Prüfer Prof. Gerhard Lakemeyer, Ph.D

Aachen, June 20, 2025

Abstract

Symbolic planning, a powerful paradigm for long-horizon reasoning, requires structured state descriptions in the form of logical predicates. This requirement creates a significant knowledge-acquisition bottleneck when applying planners to visually rich environments, such as video games, as symbolic states are not directly available from raw pixel data. Traditional approaches that rely on manual annotation or brittle, hand-crafted geometric rules do not scale to these dynamic environments.

This thesis addresses this challenge by developing and evaluating a novel pipeline that automatically learns symbolic state descriptions directly from images. Our method leverages Scene Graph Generation (SGG) to extract objects and their spatial relationships from raw Atari game frames. The pipeline's perception module utilizes a fine-tuned Faster R-CNN for object detection, coupled with a domain-adapted Neural Motifs model, to generate comprehensive scene graphs. These graphs are then translated into the formal predicates of Object-Oriented Domains (O2D), providing a symbolic representation suitable for a planner.

The primary contributions of this work are an automated pipeline that bridges visual perception and symbolic state representation, enabling downstream symbolic planning and a domain adaptation strategy that tailors SGG to the unique visual characteristics of Atari games.

Contents

1 Introduction

Modern video games, robotic simulations, and other interactive applications present complex, ever-changing visual worlds. How can an autonomous agent interpret complex visual environments and plan actions to achieve its goals? The central challenge lies in a fundamental disconnect: powerful AI planning algorithms operate on abstract, symbolic descriptions of the world, such as "player is `left_of` enemy" or "ball is Above paddle"—while the environments themselves provide only a stream of raw, unstructured pixels. A classical symbolic planner cannot operate directly on raw visual data; it requires a structured, symbolic perception of its world to devise a winning strategy. Planning fundamentally involves devising a sequence of actions, or a plan, that enables an autonomous agent to transition effectively from an initial state to a desired goal state within its environment [1]. The entire process critically depends on the availability of robust and accurate representations of that environment, particularly the identities of all relevant objects and their intricate interrelationships. STRIPS [2] introduced the first predicate-based action formalism. PDDL [3] has since become the de facto domain definition language, extending STRIPS with typing, quantifiers, and conditional effects. These logic-based models formalize planning problems as state-transition systems where actions alter the world state through well-defined rules. While these formalisms are highly effective in structured and predictable environments, a critical challenge emerges when applying them to visually complex scenarios. The planning process itself does not inherently require manually defined rules; the difficulty arises when one must first determine the symbolic state from unstructured perceptual data, such as images. This grounding step creates a significant "knowledge-acquisition bottleneck" since it has traditionally relied on brittle, hand-crafted rules and annotations that do not scale to diverse, real-time settings [1].

The ability to extract structured, symbolic information from visual environments is important in many applications—from planning to post-game analysis. In classic Atari games, pixel-based scenes provide no direct symbolic structure. Understanding which objects are present and how they relate spatially (e.g., whether the ball is above the paddle) is essential for building higher-level reasoning tools. While this thesis does not control game agents, it demonstrates how symbolic scene structure can be automatically derived from raw visual input, forming a foundation for future symbolic reasoning or planning systems.

Historically, methods for achieving this symbolic grounding have been limited. Manual annotation [4], where human experts label objects and relations, is precise and has been foundational in creating large-scale vision datasets such as Visual Genome [4]. However, this approach is highly labor-intensive and fundamentally unscalable for real-time applications. A more

automated approach involves rule-based systems [5, 6], which use the output of object detectors (e.g., YOLO [7**?**]) and apply predefined geometric heuristics to infer relationships. These systems, however, remain brittle; they often fail when faced with visual variation, occlusion, or noise, and they lack the generalizability required for diverse and dynamic scenarios. Neither approach scales to high-variation, low-latency contexts: manual annotation is too slow for online play, and rule-based heuristics break under varied sprite art or unexpected occlusions.

This thesis confronts the critical perception-to-planning gap by proposing and developing a novel, end-to-end pipeline that automatically generates symbolic state descriptions directly from raw visual data. We focus on the dynamic environments found in the Atari 2600 suite [8], whose diverse mechanics, visually distinct sprites, and well-established benchmarks make it an ideal setting for studying perception-to-symbolic pipelines in real-time scenarios. To facilitate this translation from pixels to symbolic predicates, we leverage OC-Atari [9], a recent object-centric extension of the Arcade Learning Environment that provides bounding box annotations and class labels for key game entities. By using OC-Atari as our perception foundation, we create a bridge from low-level visual inputs to high-level symbolic predicates, enabling classical planners to reason and operate in visually rich domains that were previously inaccessible to them.

We leverage modern Scene Graph Generation (SGG) techniques to bridge the gap between raw pixels and symbolic reasoning. First, a fine-tuned Faster R-CNN model [10] detects and classifies objects. Next, a domain-adapted Neural Motifs model [11] infers pairwise relationships, yielding a structured scene graph of nodes (objects) and labeled edges (relations), which we then translate into a formal symbolic representation using Object-Oriented Domains (O2D) [12]—a predicate language for expressing spatial relations. The resulting scene graph is then translated into formal O2D predicates—binary spatial relations, such as `left_of`, `below`, and `overlap`—that classical planners can understand. To address this, we ask the central research question: How can we automatically and accurately extract, formalize, and translate the geometric and semantic relationships in dynamic visual inputs (e.g., Atari frames) into symbolic Object-Oriented Domain predicates suitable for planning?

Our solution is a fully automated, end-to-end perception-to-planning pipeline. The pipeline utilizes state-of-the-art deep learning models — specifically Faster R-CNN [10] for object detection and Neural Motifs [11], which employs Long Short-Term Memory (LSTMs) to capture global contextual dependencies between objects — to generate detailed scene graphs from raw visual inputs. This structured visual information is then translated into formal Object-Oriented Domain (O2D) predicates. O2D is a language explicitly designed to model the spatial and geometric relationships among objects using binary predicates, such as `left_of`, Below, and Overlap [12]. This process bridges the gap between low-level visual perception and high-level symbolic reasoning, enabling symbolic planners to reason over structured perceptions derived from visual environments.

In the remainder of this thesis, we detail the background, related work, approach, and evaluation of this perception-to-planning pipeline, demonstrating its applicability across diverse Atari environments.

2 Background

This chapter provides the foundational background for understanding the research presented in this thesis. It covers key topics at the intersection of artificial intelligence and computer vision that inform the task of learning symbolic state representations from raw visual input for planning and decision-making. The chapter begins with an overview of classical symbolic planning paradigms, followed by a discussion of object-oriented representations, with a particular emphasis on Object-Oriented Domains (O2D) [12], which forms the core formalism of our system. Next, it surveys modern visual perception techniques, including object detection and scene graph generation (SGG), with a detailed discussion of the Neural Motifs architecture [11]. The chapter concludes by introducing OC-Atari [9], the framework used for environment simulation, data annotation, and empirical evaluation.

2.1 Planning Paradigms

AI planning is a foundational area of research focused on generating action sequences that enable an agent to reach a specified goal from a given initial state. It has widespread applications in robotics, logistics, automated gameplay, and decision-making systems. The effectiveness of any planning system depends critically on how the environment, goals, and available actions are represented. In classical planning, a world is represented as a set of logical predicates that describe the initial state and one or more goal conditions. A planner searches over discrete actions—each with known preconditions and effects—to find a sequence that transforms the initial state into one satisfying the goals. This model presumes a static, fully observable environment and that actions always succeed as specified. The planner's task is purely symbolic: derive the action sequence that bridges the symbolic gap between start and goal. In this thesis, we demonstrate how raw Atari frames can be automatically converted into the propositional predicates and PDDL descriptions required by such planners, enabling high-level symbolic reasoning over visual inputs.

2.1.1 Classical Planning:

STRIPS and PDDL Classical planning paradigms rely on logic-based formalisms to model states and transitions explicitly.

STRIPS Formalism: The Stanford Research Institute Problem Solver (STRIPS) [2] formalism introduced by Fikes and Nilsson was one of the earliest and most influential planning

frameworks. It models the environment using logical predicates, and actions are defined in terms of how they transform these predicates.

- **States** are represented as sets of ground, function-free literals (e.g., {At(robot, roomA), Holding(robot, blockB)}). The closed-world assumption applies: anything not explicitly stated to be true is assumed to be false.
- **Actions (also known as operators)** consist of three components:
  - **Preconditions:** A set of literals that must be true for the action to be applicable.
  - **Add List:** Literals that become true after the action is executed.
  - **Delete List:** Literals that are removed (become false) after action execution.
- **Goals** are defined as a set of literals that the planner attempts to make accurate by applying a sequence of actions.

PDDL Extension: The Planning Domain Definition Language (PDDL), developed by Ghallab et al. [3], extends the STRIPS formalism and has become the standard for describing planning problems in AI. It introduces greater expressivity, enabling planners to represent more complex scenarios.

- **Typed Objects:** Structured domain definitions using object classes (e.g., `Block`, `Location`).
- **Quantified Preconditions and Goals:** Universal and existential quantification.
- **Conditional Effects:** Effects applied conditionally based on runtime state.
- **ADL Fragment:** Adds negative and disjunctive preconditions (plus conditional effects).
- **Extensions:** Numeric reasoning (PDDL2.1), temporal planning (PDDL2.2, PDDL3), user-defined preferences and soft constraints.

2.1.2 Planning Algorithms

Once a planning problem is formalized in a symbolic language such as PDDL, various algorithmic strategies can be employed to search for a solution. Most modern planners use heuristic-guided search to efficiently explore the vast space of possible world states, identifying a valid sequence of actions that satisfies the goal conditions. One widely used system, Fast Downward, applies domain-independent heuristics to enable scalable classical planning across a range of benchmarks [13].

While the specific choice of planning algorithm can influence solution quality and runtime, this thesis does not focus on the internals of the planner. Instead, the planner is treated as a black-box module that takes symbolic state descriptions as input and returns a plan if one exists.

2.2 Planning with Object-Oriented Domains

To bridge the gap between high-level symbolic reasoning and unstructured visual data, specialized symbolic languages are required. One such formalism is O2D [12], proposed by Occhipinti , Bonet, and Geffner, that extends planning with a formal language explicitly designed to support perceptually grounded symbolic representations. O2D provides a concise and expressive notation for modeling spatial and geometric relationships between objects in a scene. Initially developed for domains like robotics and block-based environments, O2D was not originally designed for Atari-style visual data. In this thesis, we adapt it to the pixel-art context of Atari games, demonstrating its versatility and applicability beyond its original scope. As illustrated in Figure 2.1, O2D allows for encoding a variety of geometric and spatial relations, including predicates such as `left_of`, `below`, `overlaps`, and `smaller`. These representations mirror the kind of perceptual structure encountered in visually complex tasks.

The relevance of O2D to this thesis is twofold: first, it enables symbolic representation of perceptual input; second, it forms the basis for symbolic planning once such representations are constructed. In this thesis, we apply O2D to Atari game environments, where reasoning about spatial relationships between entities—such as players, bullets, and obstacles—is critical for successful planning. As Kaelbling and Lozano-Pérez [14] argue, effective robotic planning requires reasoning about geometric constraints and spatial configurations. While their work focuses on task and motion planning in physical environments, our use of spatial predicates (e.g., `left_of`, `below`) plays a similar role in enabling symbolic reasoning grounded in visual scenes.

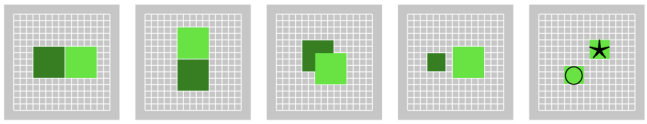

Figure 2.1: Example spatial relationships in the O2D formalism [12], `left_of`, `below`, `Overlaps`, and basic object attributes like smaller and shape. These predicates form the symbolic vocabulary used to reason about object configurations in perceptual environments.

2.2.1 Formal Structure of O2D

An O2D state is defined as a set of logical atoms, each expressed as:

$$R(o_1, o_2, \ldots, o_k)$$

Where:

- R is a k-ary relation symbol (e.g., `left_of`, `below`, `Overlaps`).
- $o_1, \ldots, o_k$ are object symbols (e.g., `ball_A`, `enemy`, `player_paddle`).

O2D supports a range of relational structures, including:

- **Unary Relations (Properties):** Describe intrinsic attributes of individual objects, such as $\mathsf{IsBall}(o_1)$ or $\mathsf{IsPlayer}(o_2)$.
- **Binary Relations (Spatial Relationships):** Capture pairwise geometric configurations such as $\mathsf{above}(o_1, o_2)$, $\mathsf{overlaps}(o_3, o_4)$, or $\mathsf{left\_of}(o_1, o_2)$.
- **Higher-Arity Relations:** Represent complex configurations involving more than two objects. For example, $\mathsf{between}(o_1, o_2, o_3)$ indicates that o_2 lies spatially between o_1 and o_3.

2.2.2 Grounding and Planning with O2D

A key strength of O2D is its perceptual grounding: symbolic atoms are not hand-crafted but are derived automatically from raw visual input using intermediate perception modules—typically object detectors or SGG models. In this thesis, we adopt SGG as the primary mechanism for extracting structured visual information from Atari frames, which is then translated into O2D predicates. In a planning context, an O2D-based domain consists of:

- **Initial State:** A set of grounded O2D atoms representing the current spatial configuration of objects in the environment.
- **Goal State:** A set of target predicates that define the desired outcome.
- **Actions:** Symbolic operators defined by preconditions and effects expressed over O2D predicates.

This formulation enables the planner to operate entirely over symbolic representations grounded in perceptual input. By structuring the domain in terms of O2D, the pipeline bridges low-level visual perception and high-level symbolic reasoning in a modular and interpretable way.

2.3 Foundations of Visual Scene Understanding

Visual scene understanding is a central pursuit in both computer vision and artificial intelligence [15]. It aims to enable machines to interpret and reason about visual input in a way that approximates human perception. This broad task spans several subdomains, including object detection, recognition of spatial configurations, and semantic interpretation of object relationships within a scene. This thesis specifically focuses on object detection and spatial relation recognition as core components of the visual perception module.

2.3.1 Object Detection

Object detection refers to the identification and localization of visual entities within an image or video frame. Each detected object is typically enclosed by a bounding box and assigned a semantic class label corresponding to its semantic category (e.g., ball, paddle, wall). As shown in Figure 2.2, object detection outputs bounding boxes and class labels for all identified objects in the scene, serving as the essential input for higher-level tasks such as Scene Graph Generation.

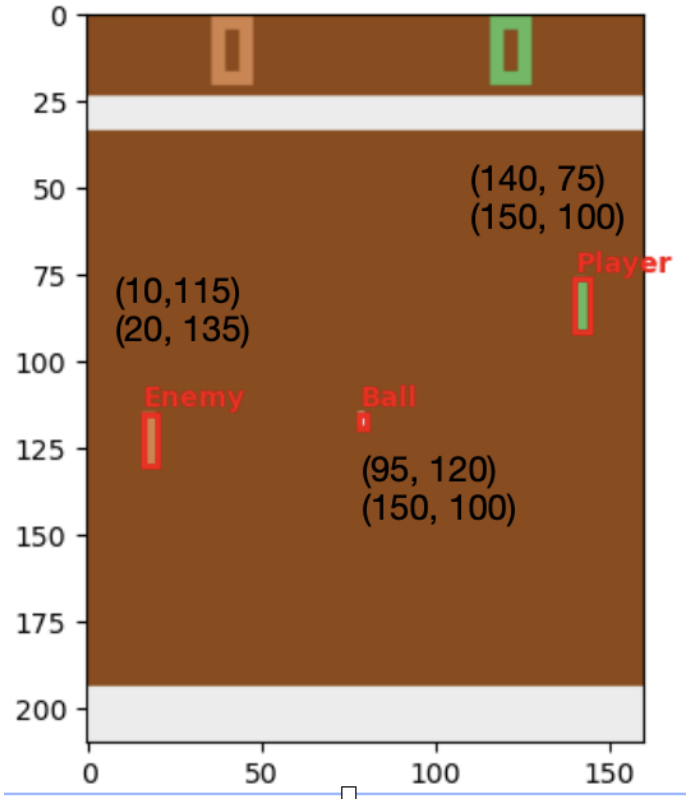

Figure 2.2: Example Atari frame with detected objects (Player, Enemy, Ball) and their corresponding bounding boxes and coordinates. These structured detections serve as inputs for downstream symbolic reasoning modules such as SGG.

Traditional Approaches Before the emergence of deep learning, object detection relied heavily on manually engineered features and classical machine learning classifiers. Common methods included sliding-window detectors using visual descriptors such as SIFT [16], SURF [17], and HOG [18], as well as model-based approaches like DPM [19]. While these methods achieved moderate success, they were computationally expensive and often failed to generalize across diverse visual scenes [16, 19].

Deep Learning–Based Detectors The introduction of Convolutional Neural Network (CNN) transformed object detection. Several landmark models include:

- **R-CNN:** Introduced by Girshick et al. [20] Regions with Convolutional Neural Network features (R-CNN) generated region proposals using Selective Search, and passed each proposal through a CNN for feature extraction, followed by classification and bounding box regression. Although accurate, R-CNN was computationally slow due to its per-region forward passes.
- **Fast R-CNN:** Girshick et al. [21]improved upon R-CNN by computing a shared convolutional feature map for the entire image, then using ROI pooling to extract features for each proposal. This significantly improved speed and accuracy.
- **Faster R-CNN:** Ren et al. [10] introduced the Region Proposal Network (RPN), which generates proposals directly from the shared feature map, making the architecture fully end-to-end trainable.
- **YOLO:** Proposed by Redmon et al. [22], You Only Look Once (YOLO) reframed object detection as a single-pass regression problem over a grid. It achieved real-time performance, with subsequent versions (YOLOv2, YOLOv3, YOLOv4) improving accuracy and robustness.
- **SSD:** Introduced by Liu et al. [23], Single Shot MultiBox Detector (SSD) predicts bounding boxes and class scores directly from multiple feature maps in one pass, balancing speed and detection accuracy across object scales.

These models now form the foundation of modern object detection pipelines and are commonly employed as the initial stage in more complex visual reasoning systems. In this thesis, we adopt a fine-tuned Faster R-CNN as the core object detection module.

2.3.2 Scene Graph Generation

Scene Graph Generation (Scene Graph Generation (SGG)) builds upon object detection by producing a structured, relational representation of a visual scene. A scene graph consists of object nodes and labeled edges representing relationships between them, typically encoded as ⟨subject, predicate, object⟩ triplets [24, 11].

Definition and Structure Formally, a scene graph $G = (V, E)$ comprises:

- A set of objects $V = \{v_1, v_2, \dots, v_n\}$, each associated with class labels and bounding boxes.
- A set of directed edges $E = \{e_{ij}\}$, where each e_{ij} denotes a predicted relationship between object v_i and object v_j, such as ⟨player, above, ball⟩.

Scene graphs thus encode both the presence of objects and their spatial or semantic interrelationships.

Importance and Applications Scene graphs enhance visual understanding and support a range of downstream tasks:

- **Image Captioning:** Generating descriptive sentences based on structured visual representations.
- **VQA:** Answering semantic questions about images.
- **Semantic Image Retrieval:** Locating images that contain specific object-relationship queries.
- **Robotics and Interaction:** Enabling agents to reason about spatial arrangements of objects for planning and manipulation.

In this thesis, SGG serves as the critical intermediate step between low-level perception and high-level symbolic planning. It transforms pixel-level information into relational predicates suitable for conversion into Object-Oriented Domain (O2D) representations. Figure 2.3 illustrates an example of a scene graph derived from a Pong frame, showing object nodes (e.g., Ball, Player, Enemy) and their spatial predicates. This structured graph acts as the foundation for symbolic reasoning and planning.

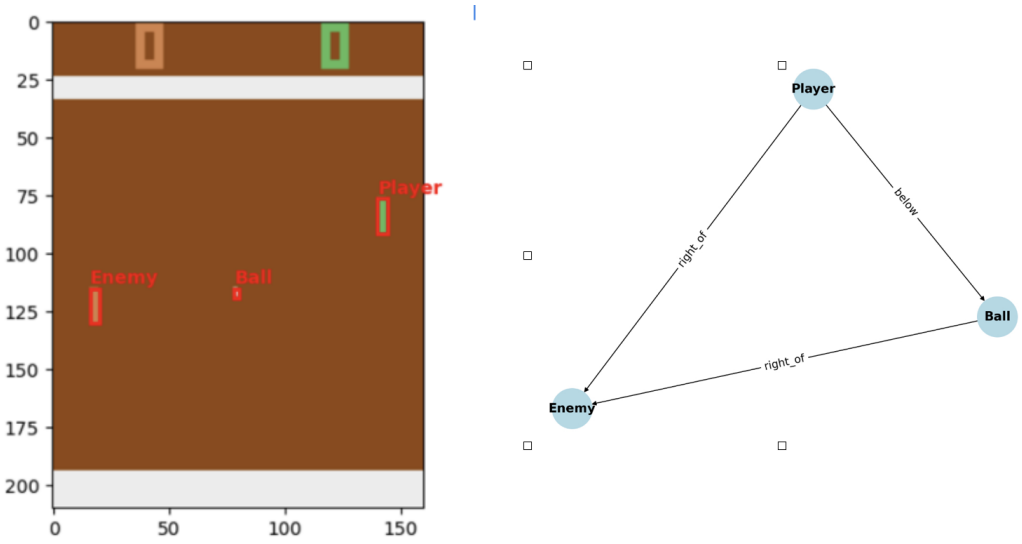

Figure 2.3: Scene graph generation in Pong. Left: the detected objects in a game frame. Right: the corresponding scene graph, with predicted relationships such as `right_of(Ball, Enemy)` and `below(Player, Ball)`.

Deep Learning Approaches to SGG Modern SGG pipelines are built on deep learning architectures and typically consist of three core stages:

- **Object Detection:** Detect and localize objects in an image using models such as Faster R-CNN, which outputs object proposals, bounding boxes, and class labels.

- **Feature Extraction:** Extract appearance features from detected object regions and geometric features from object pairs (e.g., relative positions, box deltas). These features are used to represent both individual nodes and potential edges in the scene graph.
- **Relationship Prediction:** Predict relationships between object pairs using context-aware modules. Common strategies include:
  - **LSTMs:** Used to encode global context across object pairs (e.g., in Neural Motifs) [11].
  - **Graph Neural Networks (GNNs):** Used to iteratively update node and edge features via message passing [24].
  - **Attention Mechanisms:** Employed to focus on the most relevant object pairs or contextual features during relationship classification [25].

This modular design enables end-to-end training and allows the model to learn complex spatial and semantic interactions directly from annotated image data.

Neural Motifs Model The Neural Motifs model [11] introduces the idea of learning frequent relational patterns ("motifs") across images. Motifs refer to recurring object–relationship structures (e.g., person-wearing-shirt, car-parked-on-street) that occur across many scenes. It consists of:

- **Stage 1:** Object detection using a CNN-based backbone.
- **Stage 2:** Relationship prediction using BiLSTMs to model global object context.
- **Stage 3:** Scene graph construction by combining predicted object classes and their pairwise predicates.

Neural Motifs are trained on large-scale datasets such as Visual Genome [4], with loss functions tailored to address the long-tailed distribution of relationships in real scenes. Its strength lies in leveraging global scene context to predict more coherent relationships, making it a strong candidate for structured domains, such as Atari games.

Other Scene Graph Generation Models Beyond Neural Motifs, several alternative models have introduced important ideas to the field of scene graph generation:

- **IMP (Iterative Message Passing)** [24]: Refines relationship predictions by passing messages iteratively between object and predicate nodes, modeling contextual dependencies across the graph.
- **VTransE** [26]: Proposes a translational embedding framework for visual relationships, inspired by techniques from knowledge graph completion.
- **Graph R-CNN** [27]: Incorporates a relation proposal module and employs Graph Convolutional Networks (GCNs) to capture interactions between object pairs.

- **Transformer-Based SGG** [28]: Leverage self-attention mechanisms to capture long-range dependencies and holistic context across the entire scene, achieving state-of-the-art results on benchmark datasets.

These models reflect the growing diversity in architectural choices for scene graph generation, each emphasizing different aspects of visual context, relational structure, and computational efficiency.

Benchmarks and Evaluation Metrics Common benchmark datasets used for SGG include:

- **Visual Genome** [4]: A large-scale dataset with over 100,000 images annotated with objects, attributes, and relationships. It is the most widely used benchmark for training and evaluating SGG models.
- **VRD Dataset** [29]: Designed specifically for visual relationship detection, it focuses on ⟨subject, predicate, object⟩ triplet-level annotations.
- **COCO-Stuff** [30]: An extension of the MS COCO dataset that includes "stuff" categories (e.g., grass, sky, road), enabling a more holistic scene representation.

Evaluation metrics typically used in SGG include:

- **RecallK (R@K):** Measures whether a ground-truth triplet appears in the top-K predicted triplets. Recall@20, Recall@50, and Recall@100 are standard evaluation thresholds.
- **Mean Average Precision (mAP):** Calculates precision across all object or predicate categories and averages the precision-recall curves.
- **F1-Score:** BThe harmonic mean of precision and recall, providing a balanced measure of triplet prediction quality, especially in predcls or sgcls settings.

In this thesis, we primarily report Recall@50 and F1-score for predicate classification in predcls mode.

Limitations of Generic SGG Models While state-of-the-art Scene Graph Generators perform well on frequent relations in natural images, they often struggle with long-tail predicates—those that occur rarely in the training data (e.g., "bridges," "supports," or domain-specific spatial relations). As Zellers et al. [11] demonstrate, this skewed predicate distribution significantly reduces recall on infrequent but semantically important relations, which undermines downstream reasoning tasks.

In the Atari domain, object interactions involve a limited and domain-specific vocabulary of relations that differ significantly from natural image datasets like Visual Genome. Sprites such as paddles, balls, bullets, and enemies appear in spatial configurations that demand precise geometric reasoning (e.g., `left_of`, `below`, `overlaps`). Off-the-shelf SGG models, trained on general datasets, are not optimized for this structure.

To address this limitation, we fine-tune the SGG model on OC-Atari data, adapting it to the visual and relational patterns of Atari games. This domain adaptation step (explored in Chapter 4) is essential for enabling symbolic reasoning in such structured, pixel-based environments.

2.4 The Object-Centric Atari Environment

The Atari 2600 game suite—popularized in AI research through the Arcade Learning Environment (ALE)) [8]—has become a standard benchmark for evaluating reinforcement learning agents. More recently, ALE has been increasingly adopted in symbolic reasoning and planning research, where the focus shifts from policy learning to interpreting game states symbolically.

Atari games offer a unique combination of visual diversity, discrete action spaces, and deterministic mechanics, making them ideal for studying perception-to-symbolic translation. Their pixel-art graphics, limited object vocabulary, and well-defined interaction rules allow researchers to ground symbolic representations in visual data without the full complexity of real-world imagery.

As such, Atari domains serve as a valuable intermediate challenge between toy domains like Blocks World and high-resolution environments like robotics or driving simulators. They support experiments that explore how object detection, scene graph generation, and symbolic planning can be combined in a single pipeline.

2.4.1 Overview of the ALE

The Arcade Learning Environment (ALE) [8] provides an emulation interface for dozens of Atari 2600 games. It exposes each game's screen as raw pixel input and offers programmatic access to low-level RAM states and action controls. Originally designed to benchmark reinforcement learning algorithms, ALE has played a pivotal role in the development of modern deep RL methods.

A landmark example is the Deep Q-Network (DQN) by Mnih et al. [31], which demonstrated that agents could learn directly from visual input and outperform human players in many games.

However, from the standpoint of symbolic reasoning and structured visual understanding, ALE presents a critical limitation:

While it delivers raw visual streams, it does not expose object-level semantics such as categories, bounding boxes, or spatial relationships.

As a result, any system that aims to build symbolic models or scene representations must first infer this information through external perception modules (e.g., object detectors or scene graph

models). This gap motivates the need for intermediate environments—such as OC-Atari—that extend ALE with object-centric annotations and facilitate perception-driven symbolic planning.

2.4.2 Object-Centric Extensions and Their Significance

To address the lack of object-level semantics in ALE, several object-centric extensions have been proposed. One of the most notable is OC-Atari, introduced by Delfosse et al. [9], which augments Atari gameplay frames with object-level annotations derived from memory states.

OC-Atari: Contributions and Relevance

- **Object-Level Annotations:** OC-Atari provides bounding boxes and class labels for key entities (e.g., `Player`, `Enemy`, `Ball`, `Missile`) across multiple Atari games.
- **Game Coverage:** It supports a diverse set of classic titles such as Pong, Space Invaders, Ms. Pac-Man, and Seaquest, among others.
- **Structured Representation:** The annotations facilitate the generation of structured, symbolic state descriptions from raw frames, forming the foundation for downstream tasks such as symbolic planning and decision-making.

As shown in Figure 2.4, OC-Atari provides annotated game frames with bounding boxes and class labels for game objects, enabling symbolic systems to ground their reasoning in perceptual data.

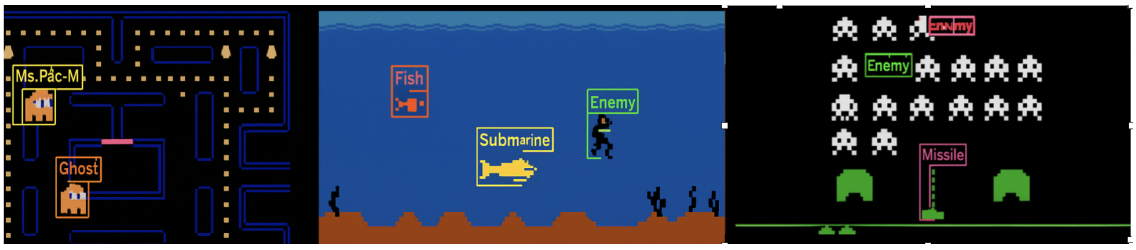

Figure 2.4: Example OC-Atari [9] annotations for Ms. Pac-Man, Seaquest, and Space Invaders. Object bounding boxes highlight entities such as Ghost, Submarine, Enemy, and Missile, providing the structured visual input required for supervised scene graph generation and symbolic representation.

Significance for This Thesis OC-Atari plays a critical role in the implementation and evaluation of this thesis. Specifically:

- *Supervised Fine-Tuning of SGG Models:* OC-Atari provides the object annotations required to adapt general-purpose SGG models (e.g., Neural Motifs) to the visual structure and style of Atari games.
- *Relationship Annotation:* While OC-Atari does not natively include relationships between objects, the availability of object types and positions allows the semi-automated derivation of spatial relationships (e.g., `left_of`, `Below`) for training relational classifiers.

- *Evaluation Framework:* OC-Atari enables the creation of ground-truth symbolic states in the O2D format. These can be used for evaluating both the accuracy of perception modules and the effectiveness of the planning system downstream.

2.4.3 Available Data Modalities

OC-Atari plays a central role in the implementation, training, and evaluation of the perception-to-planning pipeline developed in this thesis. Specifically, it enables the following:

- **Supervised Fine-Tuning of SGG Models:**
  OC-Atari provides object-level annotations (bounding boxes and class labels) necessary to fine-tune general-purpose scene graph generation models—such as Neural Motifs—on Atari game frames. This domain-specific fine-tuning is essential for adapting models trained on datasets like Visual Genome to the visual structure of pixel-art environments.
- **Relationship Annotation via Geometry:**
  While OC-Atari does not include native relationship labels, the annotated object positions allow for the derivation of spatial relationships (e.g., `left_of`, `below`, `overlap`) using geometric rules. These automatically generated relation labels serve as training targets for the predicate classification module.
- **Evaluation of Symbolic Accuracy:**
  OC-Atari enables the generation of ground-truth symbolic state descriptions in the O2D predicate format. These structured representations are essential for:
  (a) evaluating the symbolic accuracy of the perception pipeline, and
  (b) validating whether symbolic planners (e.g., Fast Downward) can generate coherent plans using the learned states.

  OC-Atari is therefore indispensable for both perception-side training and end-to-end evaluation of the system.
- **Game Coverage in This Thesis**
  The following ten games were selected to span a range of mechanics and visual structures, allowing robust evaluation of detection, relationship modeling, and symbolic planning:
  - **Reflex-Based Paddle Games:**
    * Pong
    * Breakout
  - **Arcade Shooters:**
    * Space Invaders
    * Assault
    * Alien
  - **Maze and Navigation:**
    * Ms. Pac-Man
    * Amidar

- **Platforming and Interaction:**
  - Frostbite
  - Berzerk
  - Chopper Command

These games provide a diverse testbed to validate the pipeline's ability to detect objects, infer spatial relations, and generate symbolic states compatible with planning environments.

2.5 Mathematical Formalisms and Notations

A rigorous mathematical representation of objects, relationships, and symbolic structures is essential for translating visual perception into actionable symbolic knowledge. This section presents the formal notations used throughout this thesis to define the structure of scene graphs, object attributes, and O2D descriptions, laying the groundwork for symbolic planning from images. These representations bridge the gap between outputs of deep learning-based visual modules and the symbolic structures required by classical planners. The notations defined in this section will be used consistently throughout the method and evaluation chapters to describe symbolic states derived from visual input.

2.5.1 Object Representation

In a given frame F_t, each detected object o_i is represented by a bounding box $\mathbf{b}_i \in \mathbb{R}^4$, defined as:

$$\mathbf{b}_i = [x_i, y_i, w_i, h_i]$$

Where:

- x_i, y_i: Coordinates of the top-left corner of the bounding box
- w_i, h_i: Width and height of the box

Each object o_i is also associated with a class label $c_i \in C$, where C is the set of known object classes (e.g., `Player`, `Ball`, `Wall`). This representation follows the standard object detection format introduced by Faster R-CNN [10]. This basic spatial and semantic encoding serves as the input for generating a scene graph.

2.5.2 Scene Graph Formalism

A scene graph $G_t = (V_t, E_t)$ is a directed graph representing the structure of a visual scene at time t, where:

- **Vertices (V_t):** Each vertex $v_i \in V_t$ corresponds to an object o_i, described by its bounding box $\mathbf{b}_i$ and class label c_i.

- **Edges (E_t):** Each edge $e_{ij} \in E_t$ represents a directed relationship from object o_i to object o_j, labeled by a predicate r_k.

Formally, a scene graph edge can be expressed as:

$$e_{ij} = (o_i, r_k, o_j), \quad r_k \in \{\text{above}, \text{below}, \text{left\_of}, \text{right\_of}, \text{overlap}\}.$$

The complete scene graph G_t for a frame F_t thus captures not only the object identities but also their pairwise spatial and semantic relationships. This follows standard representations in scene graph generation [11, 32].

2.5.3 O2D Formalism

The Object-Oriented Domain (O2D) [12] formalism transforms the relational data from the scene graph into a symbolic representation suitable for high-level planning and reasoning. Given a scene graph G_t, its corresponding O2D state representation D_t is defined as the set of actual logical atoms derived from the scene graph's edges.

Each edge in the scene graph—representing a triplet ⟨subject, predicate, object⟩—maps directly to an O2D logical atom. For example, if the scene graph generator detects the relationship (Player, left_of, Ball), this corresponds directly to the O2D predicate `left_of(Player, Ball)`.

The complete state description D_t is the set of all such actual atoms for a given scene. For instance, a state in an Atari game might be represented as the set:

```
{IsPlayer(p1), IsBall(b1), IsEnemy(e1), left_of(p1, b1), below(b1, e1)}
```

This formal,symbolic O2D state can then be programmatically translated into a PDDL `:init` state or used to define a `::goal` condition, enabling compatibility with standard symbolic planners.

2.5.4 Relevance to Planning

The symbolic state description D_t, generated from visual input, is central to our perception-to-planning pipeline. Its primary purpose is to serve as the grounded initial state for symbolic planning. In our implementation, we translate the set of O2D atoms from D_t into the `:init` section of a `problem.pddl` file. This is paired with a manually defined `domain.pddl` file, which specifies the action schemas (e.g., `MoveUp`, `HitBall`).

By feeding these PDDL files into an off-the-shelf classical planner such as *Fast Downward*, we can verify whether the system produces planner-compatible symbolic states from raw visual scenes. This step completes the loop from pixels to symbolic action plans—not by learning

the planning model, but by demonstrating that perceptual outputs can be directly used in automated reasoning once the domain model is supplied.

2.6 Summary of Background

This chapter established the theoretical and technical foundation for the research presented in this thesis. We traced the evolution of symbolic planning—from classical propositional approaches to object-oriented formalisms—highlighting the role of the O2D language in modeling geometric and spatial relationships necessary for decision-making.

We then examined key components of visual scene understanding. This included a review of object detection models, from early feature-based methods to modern CNN-based detectors, and a detailed exploration of SGG, with particular focus on the Neural Motifs model as an effective tool for relational scene representation.

Finally, we introduced the OC-Atari framework, which provides object-centric annotations over Atari gameplay frames. OC-Atari plays a crucial role in this thesis by enabling supervised fine-tuning of visual models and providing a structured environment for validating the proposed perception-to-symbolic planning pipeline.

Together, these foundations provide the necessary tools and context for the chapters that follow, which develop and evaluate methods for learning symbolic state representations directly from raw visual data.

3 Related Work

This chapter surveys existing research on learning symbolic representations from visual data for use in planning systems. It covers foundational work in symbolic reasoning, advances in object detection and scene graph generation, and recent attempts to integrate perception with formal planning frameworks.

The focus is on identifying the key challenges of applying these techniques in dynamic, visually complex domains—such as video games—where symbolic states are not directly observable. While significant progress has been made in perception and reasoning separately, we find that current methods often fail to robustly automate the full transformation from raw images to structured, planner-compatible symbolic states.

This review motivates the approach taken in this thesis: a unified pipeline that directly bridges low-level visual input with high-level symbolic reasoning.

3.1 Symbolic Planning and Representation

Symbolic planning has long served as a fundamental paradigm in AI, enabling agents to reason about sequences of actions using logical abstractions of the world. Foundational frameworks such as STRIPS [2] and PDDL [3] laid the groundwork for classical planning systems by introducing representations based on logical preconditions, actions, and effects. These models formalize planning problems as state-transition systems, where actions modify symbolic world states according to well-defined logical rules.

While highly effective in domains where a complete symbolic model can be manually crafted, these classical systems rely on hand-engineered models, which limit their scalability and generalization to real-world or dynamic environments. The design of predicates, object types, and action schemas in PDDL often requires extensive domain knowledge, creating a knowledge-acquisition bottleneck in tasks where the environment is visually rich or rapidly changing, such as video games or robotics. To address these limitations, researchers have increasingly focused on learning symbolic representations directly from interaction data. For example, Konidaris et al. [33] proposed a framework that grounds symbolic representations in sensorimotor experience, enabling an agent to derive abstract states and action models from observed effects autonomously. However, even these modern approaches typically rely on preprocessed input features or engineered abstractions, rather than raw perceptual signals, such as images. However, these models frequently assume access to pre-segmented objects or engineered features,

limiting their generalizability to raw visual inputs. This creates a gap between perception and planning: while symbolic models offer powerful reasoning capabilities, they cannot operate directly on unstructured visual data.

Attempts to bridge the gap between perception and planning have followed several paradigms. Earlier approaches often involved rule-based symbolic extraction, where symbolic predicates are derived from sensor data using fixed, hand-crafted heuristics (e.g., inferring `Above(A, B)` from bounding box positions). Systems in this category often rely on geometric preconditions to connect symbolic actions to the continuous world [14]. While partially automated, these methods are often domain-specific, fragile, and not easily generalizable to new scenarios. To overcome these limitations, more recent attention has shifted toward integrating visual learning with symbolic reasoning. This diverse body of work includes Learning object-centric representations from images for planning [34, 12], developing methods to map geometric features to task predicates [6], creating end-to-end neural-symbolic planning models [35], and learning relational state abstractions directly from interaction [36]. These modern efforts emphasize that for symbolic planning to scale, the representations it relies on must be automatically derivable from visual input—a challenge that this thesis addresses through a pipeline from SGG to O2D.

3.2 Traditional Approaches to Learning Symbolic State Descriptions from Images

The challenge of automatically obtaining symbolic state descriptions from raw visual data has historically been addressed through methods that rely heavily on either manual effort or hard-coded domain knowledge. These traditional techniques, while foundational, often lack scalability and generalization, especially in environments characterized by high visual variability and temporal dynamics.

Manual Annotation of Symbolic States One of the earliest and most accurate strategies for generating symbolic descriptions involved human experts manually labeling objects, their properties, and spatial relations from visual inputs. Large-scale crowdsourced datasets, such as Visual Genome [4], and scene-graph studies, like those by Johnson et al. [32], rely on dense manual annotations of objects and relations. While these handcrafted annotations are highly precise and interpretable, they are fundamentally *non-scalable*. Creating symbolic state descriptions manually is time-consuming and impractical for large-scale datasets, continuous video streams, or real-time applications. Moreover, such annotations often encode a human's implicit understanding of the scene and cannot be directly generalized across domains without considerable re-engineering.

Rule-Based Inference from Object Detections A natural step toward automation is to apply deterministic *rule-based inference* over the output of object detection algorithms. Here, predefined spatial heuristics are used to infer symbolic relations between detected objects. For example, if the top edge of bounding box A is above the bottom edge of box B, a rule might infer the predicate `Above(A, B)`. Modern detectors like You Only Look Once (YOLO) [22] or Faster R-CNN [10] can localize and classify objects in each frame, after which simple geometric rules assign relations. While these systems reduce manual effort by automating predicate generation, their reliance on hard-coded geometric rules makes them fragile in the face of visual ambiguity, noise, or occlusion. As Russell & Norvig [1] discuss, rule-based methods lack the semantic depth and robustness needed for complex, dynamic domains and must often be rebuilt from scratch for each new environment.

Learning from Feature Vectors (Not Raw Pixels) A more generalizable approach emerged with the idea of *learning symbolic representations from feature vectors* rather than direct pixel inputs. Konidaris et al. [33] proposed a framework where agents observe how actions affect high-level features (e.g., positions, velocities) and learn preconditions and effects in that abstract space. This eliminates manual annotation but assumes access to structured sensory features. Consequently, these methods falter when faced with raw images, since grounding perception into discrete symbols remains unaddressed. Asai & Fukunaga's LatPlan [34] takes a step further by learning both state encodings and action models end-to-end directly from image sequences, enabling classical A* planning in a latent symbolic space—but still relies on an intermediate latent-space encoder rather than operating directly on pixels.

This distinction is crucial to this thesis, which aims to construct an *end-to-end pipeline* that starts from raw pixel data and produces formal, symbolic representations compatible with object-oriented planning.

3.3 Advances in Visual Perception for AI Planning

The integration of visual perception into AI planning systems has undergone significant evolution with the rise of deep learning. Traditional symbolic planners relied on manually curated representations or rule-based inferences, often struggling to scale or adapt to dynamic environments. The advent of deep visual models—such as Faster R-CNN [10] and Neural Motifs [11] —particularly in object detection and structured scene understanding, has opened the door for learning symbolic state abstractions directly from images. However, the effective use of these perceptual tools in symbolic reasoning remains a significant challenge.

3.3.1 Object Detection

Modern object detectors such as Faster R-CNN [10], YOLO [7], and SSD [23] have redefined visual recognition by achieving impressive performance in object localization and classification. These models enable the real-time detection of multiple objects, providing bounding boxes and class labels with high confidence across a wide variety of scenes. Their ability to process visual information at scale has made them indispensable in computer vision applications, from autonomous driving to robotic manipulation.

However, while these models are powerful for perception, they fall short when used in isolation for symbolic planning:

- **Lack of Relational Semantics:** Detectors provide no direct means to infer relationships between objects (e.g., `left_of(Paddle, Ball)`). Yet such relational knowledge is crucial for planning tasks. As emphasized by Johnson et al. [32], visual understanding for reasoning tasks requires not only recognizing objects but also modeling their interactions. Consequently, additional post-processing—via rule-based heuristics or scene graph generation—is often needed.
- **Flat, Non-Compositional Output:** Object detectors return flat lists of detected entities, which lack structural organization or contextual information. This becomes a bottleneck for AI planners, which rely on structured inputs, such as symbolic predicates or logical atoms, to perform causal reasoning. Battaglia et al. [37] argue that relational inductive biases are essential for structured reasoning, which flat perception models inherently lack.
- **Generalization to Abstract Domains:** Object detectors trained on natural-image datasets (e.g., MS COCO [38] or Pascal VOC [39]) often suffer significant performance drops—sometimes exceeding 30%—when applied to out-of-domain inputs like Atari frames [40].
- **Symbolic Mismatch:** Symbolic planning operates on logical predicates (e.g., `Above`, `Near`, `Blocked`) rather than bounding boxes. There is a mismatch between the continuous, numeric output of object detectors and the discrete, symbolic input expected by classical planners. This gap has been noted by Konidaris et al. [33], who emphasize the need to ground perceptual data in structured symbolic forms.

These limitations motivate the need for models that can explicitly capture structured relationships—namely, scene graph generators—which we discuss in the next section.

3.3.2 Scene Graph Generation

SGG aims to extract structured relational representations from visual scenes. In contrast to plain object detection, SGG models output a graph of ⟨subject, predicate, object⟩ triplets that capture spatial or semantic relationships between entities. This relational structure is precisely

what symbolic planners need as input in the form of predicates like `Above(obj1, obj2)` or `Near(obj1, obj2)`.

Despite progress in general-purpose SGG, most models are not designed with *planning* in mind [34]. For example, while Visual Genome–based SGG models like Neural Motifs [11], Iterative Message Passing (IMP) [24], and Graph R-CNN [27] demonstrate strong performance on triplet detection benchmarks (e.g., Recall@K), they often fall short in producing symbolic abstractions that are both task-relevant and compatible with automated reasoning systems like classical planners.

General-Purpose SGG Models

- **Neural Motifs** [11] remains one of the most widely used SGG baselines. It employs BiLSTM encoders to model frequent co-occurrence patterns ("motifs") in object–relation structures, enabling global contextual reasoning that improves triplet prediction.
- **IMP** [24] refines relation predictions via repeated information exchange across the graph, improving relational coherence.
- **Graph R-CNN** [27] uses a relation proposal module and attentional Graph Convolutional Network (GCN) to generate compact and interpretable graphs.
- **Transformer-based SGG** [28] models show promise by capturing long-range dependencies through self-attention, but often require massive datasets and complex pretraining.

However, these models are typically trained on natural image datasets like Visual Genome and require domain adaptation to perform well in structured domains such as Atari.

Limitations in Planning Context

- **Predicate Mismatch:** Most general-purpose SGGs detect high-level or semantically vague relations(e.g., "holding," "on," "has") that are not directly usable by planners requiring precise geometric predicates like `left_of`, `Overlap`, or `AlignedWith`.
- **Static Descriptions, No Dynamics:** Symbolic planning requires modeling state transitions (e.g., "After `move()`, Ball will be Above Paddle"), but SGGs produce only static scene graphs.
- **Poor Generalization to Stylized Domains:** Models trained on natural images (e.g., Visual Genome) struggle with abstract, pixel-art visuals found in environments like Atari without extensive fine-tuning.
- **Incomplete Grounding:** Without mapping detected objects to symbolic identifiers, such as mapping obj1 to a symbolic identifier like PlayerPaddle, a semantic disconnect remains that hinders downstream planning.

These limitations underscore the need for domain-adapted models and structured post-processing steps, as implemented in our SGG-to-O2D pipeline (see Chapter 4).

3.4 Object-Oriented Domains

Object-Oriented Domains (O2D) have emerged in the past two years as a lightweight, first-order formalism for representing spatial configurations in planning. Rather than hand-crafting an exhaustive propositional state space, O2D encodes scenes via a small set of object-centric predicates, enabling planners to operate on relational abstractions. Research on O2D can be broadly categorized into three directions: the foundational formalism, its application in planning domains, and ongoing work on predicate expressivity and extensions.

- **Foundational O2D Formalism:** Occhipinti et al. [12] introduced O2D within the DReaM framework, demonstrating that a handful of binary spatial predicates—`Above`, `Below`, `left_of`, `Overlap`—suffice to reconstruct Blocks World–style domains without exponential blowup. They showed that planning operators defined over O2D states exhibit comparable expressiveness to classical STRIPS encodings, yet are orders of magnitude smaller.
- **O2D in Games and Robotics:** Occhipinti et al. [12] manually annotated Atari frames with O2D facts (e.g., `Above(Player,Ball)`) to learn high-level models of Pong and Breakout. Their work established that even pixelated, low-resolution game visuals can be abstracted into O2D and fed into off-the-shelf STRIPS planners. They also applied O2D to a cluttered tabletop manipulation domain, using predicates such as `overlap` and `clear(Object)` to sequence pick-and-place actions with a symbolic planner.
- **Predicate Taxonomies and Extensions:** Konidaris et al. [33] highlighted the role of unary typing predicates (e.g., `IsMovable(o)`, `IsContainer(o)`) in reducing operator arity and learning action models from trajectory data. Cohn and Hazarika [41] surveyed qualitative spatial calculi that include higher-arity relations (e.g., `Between(`o_1`, `o_2`, `o_3`)`) for navigation and spatial reasoning, suggesting that richer relational vocabularies can capture more complex domain constraints.

Remaining Challenges Despite its promise, the O2D framework presents several open challenges that represent promising directions for future research:

- **Perceptual Grounding:** Prior work has relied on manual annotations (e.g., in Atari) or analytical geometric rules (e.g., in robotics), limiting the feasibility of fully automated, end-to-end pipelines from raw visual inputs.
- **Temporal Integration:** Current O2D systems typically operate on isolated frames. Extending the framework to handle temporal sequences, dynamic object states, and long-term dependencies remains an open research problem.
- **Predicate Vocabulary Expansion:** Existing implementations focus on a limited set of spatial predicates (e.g., `left_of`, `above`, `overlap`). New domains may require additional

relations such as `adjacent_to`, `contained_in`, or `contact`, which are not currently handled by automated pipelines.

- **Robustness and Generalization:** The resilience of O2D grounding under noise, occlusion, or domain shift has not been systematically evaluated. Future work should benchmark performance under such realistic and challenging conditions.

How This Thesis Advances O2D This thesis addresses the key limitations in prior O2D research by introducing the first fully automated pipeline for generating O2D predicates directly from raw visual input. By fine-tuning a learned SGG model on Atari game frames, we eliminate the reliance on manual annotation or rule-based heuristics. The resulting system is capable of producing grounded, symbolic representations in real time—enabling classical symbolic planners to operate in fast-moving, visually rich environments like Atari games.

3.5 Integration of Visual Perception with Symbolic Planning

Bridging raw visual input with formal symbolic planning is a key research frontier, which has seen a progression from early concepts to more integrated, modern systems.

Early work by Garnelo et al. [35] showed promise by using convolutional encoders to produce object-centric features for a reasoning module, but did not connect these to a full symbolic planner. Asai & Fukunaga's LatPlan [34] advanced this by learning state encodings and action models from static image pairs, though this approach does not operate on continuous video streams. Other work, such as Dearden & Burbridge [6], integrated sample-based motion planners with learned symbolic abstractions by automatically lifting continuous geometric constraints into PDDL preconditions.

In modern robotics, influential models have demonstrated impressive end-to-end capabilities. CLIPort (Shridhar et al. [42]) and the subsequent Perceiver-Actor (Shridhar et al. [43]) learn manipulation policies directly from language and vision. However, these systems typically learn implicit, reactive policies rather than the explicit, generalizable symbolic action models used by classical planners.

Separately, research has focused on learning the symbolic operators themselves. Silver et al. [44], for instance, demonstrated success in learning operators for task and motion planning, but their work assumes a structured, symbolic state representation is already available as input.

Limitations of Prior Approaches

While these state-of-the-art approaches are powerful, they highlight several key limitations in the context of creating a fully autonomous, symbolic planning agent:

- They often rely on custom or pre-defined symbolic schemas rather than deriving relations from a general-purpose perception module like Scene Graph Generation.
- They tend to operate on static snapshots or goal images rather than processing dynamic or streaming visual information.
- They are often demonstrated in controlled manipulation domains and not tested in fast-paced, visually diverse environments like video games.

None of these prior works leverage modern Scene Graph Generation (SGG) to extract rich, relational structure directly from pixel data and use it as input to classical symbolic planners. Unlike approaches that assume structured symbolic states, our pipeline generates symbolic predicates automatically from images and connects them to a manually defined PDDL domain, enabling symbolic planning in dynamic visual environments.

3.6 Symbol Representation Across Time (Temporal SGG)

Most SGG methods assume that each frame is independent, but dynamic planning domains—from Atari games to robotic pick-and-place tasks—require temporally coherent symbol streams. Without persistent object identities, relation labels can "jump" arbitrarily between frames, breaking any notion of evolving preconditions or effects. Several works have begun to bridge this gap. For instance, Kipf et al. [45] introduce Slot Attention for Video (SAVi), which learns stable per-object representations over time. Other works, such as STTran by Cong et al. [46] and research in Video Relation Detection (Video Visual Relationship Detection (VidVRD)), utilize transformers or graph networks to enhance relational consistency across frames.

In symbolic planning, temporal logics (e.g., `HeldAt(obj, t)`, `Moved(obj_1,obj_2, t)`) are well established for encoding change along action sequences [47], but grounding those predicates from raw video remains largely unexplored. The chief challenges are (a) occlusion and re-identification, where fast-moving sprites in Atari can vanish and reappear, and (b) relation jitter, where slight noise in an object's bounding box can toggle a predicate—for example, a 2-pixel shift could flip `left_of`(ball, paddle) from true to false and back again, causing spurious plan failures.

We take a step toward achieving temporal consistency by augmenting Neural Motifs with lightweight smoothing across consecutive frames. This yields a more stable and temporally coherent graph stream, which can be mapped into time-indexed O2D predicates (e.g., `Above(ball, paddle, t)`). While full object re-identification and event tracking remain open challenges, this work demonstrates the feasibility of generating symbolic streams from raw video for downstream symbolic planning.

3.7 Predicate Space Design and Domain Vocabulary

A core challenge in perception-to-planning systems is choosing a predicate vocabulary that is both expressive enough to distinguish critical scene configurations and compact enough to keep the planning search tractable. Recent work has shown that a small set of purely spatial predicates—such as `left_of`(o_1, o_2), `Above`(o_1, o_2), and `Overlap`(o_1, o_2)—strikes this balance by grounding directly in simple geometric tests on object bounding boxes (e.g., comparing centroid x-coordinates for `left_of` or using an Intersection-over-Union threshold for Overlap), thus avoiding the combinatorial explosion associated with richer relational vocabularies [12]. Empirical evaluations have shown that symbolic planners like Fast Downward perform significantly better when operating over a compact spatial vocabulary that aligns with the domain's action affordances [13]. In contrast, general-purpose scene-graph generators trained on Visual Genome produce hundreds of relation types—"holding," "wearing," "riding," etc., many of which are irrelevant or even harmful for planning because they introduce noisy predicates that bloat the search space [4]. While this core set of spatial predicates is sufficient for many manipulation and game domains, other problems may require richer vocabularies, including unary state predicates (e.g., IsMovable) or numeric fluents. In this thesis, we adopt the minimalist approach, deliberately selecting a handful of binary spatial relations closely tied to the underlying physics of our Atari environments to achieve robust symbol grounding while preserving planner efficiency.

3.8 Research Gaps and Positioning of this Thesis

Despite progress in perception and planning, key limitations remain. Classical planners require symbolic inputs, yet existing pipelines often rely on manual annotations or brittle geometric rules. Prior work on O2D [12] demonstrates symbolic reasoning over spatial relations but does not automate extraction from raw pixels.

General-purpose SGG models produce generic relations and lack adaptation to planning contexts or stylized domains like Atari. No existing work offers an end-to-end pipeline that connects raw game frames to symbolic planning via learned scene graphs.

This thesis addresses these gaps by introducing a fully automated perception-to-planning pipeline. It adapts Neural Motifs for Atari, generates symbolic O2D predicates from video frames, and enables classical planners to operate in dynamic, visually complex environments. Lightweight temporal smoothing further improves consistency across frames, moving toward more robust symbolic reasoning over time.

4 Approach

In this chapter, we describe our end-to-end vision-to-planning pipeline, that transforms raw Atari video frames into Object-Oriented Domain (O2D) state descriptions suitable for symbolic planners. Our system is designed to bridge the long-standing gap between low-level pixel inputs and high-level relational predicates without relying on hand-crafted rules at inference time.

We achieve this by (i) fine-tuning a Scene Graph Generation (SGG) model based on Neural Motifs, to detect game sprites and their spatial relationships, and (ii) translating the resulting scene graphs into a compact and extensible O2D vocabulary.

Our key contributions, as detailed by the methodology in this chapter, are:

- **An End-to-End Automated Pipeline:** A complete and working framework that takes raw Atari frames as input and produces formal O2D state descriptions ready for a planner.
- **A Domain-Adapted SGG Model:** A robust fine-tuning strategy for adapting the complex Neural Motifs model to the unique, pixel-art domain of Atari games, successfully overcoming challenges like overfitting.
- **A Principled Evaluation Framework:** A methodology that includes both end-to-end qualitative analysis and a focused, quantitative evaluation of the model's relational reasoning capabilities.

Throughout this chapter, we justify our design choices, enumerate input and output formats, and highlight how each block integrates into a coherent, automated perception-to-planning system.

4.1 System Architecture Overview

We designed the symbolic state generation pipeline as a modular system that transforms raw visual input (Atari game frames) into structured, symbolic state descriptions suitable for a planner (O2D predicates). The entire perception-to-planning process is illustrated in Figure 4.1.

The pipeline operates sequentially through four main stages:

- **Input:** A single 210×160 Atari frame (RGB) serves as the initial raw visual input for the system.

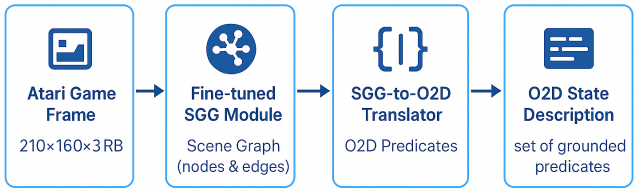

Figure 4.1: Perception-to-Planning Pipeline for OC-Atari

.

- **Scene Graph Generation:** The frame is fed into the fine-tuned SGG module. This module, built upon a domain-adapted Neural Motifs model [11], detects all relevant objects, with bounding boxes, class labels (e.g., Player, Ball), and ⟨subject, predicate, object⟩ edges.
- **Symbolic Translation:** The resulting scene graph is passed to the *SGG-to-O2D Translation Module*. This component maps the graph's nodes to formal O2D object symbols and its edges to logical predicates (e.g., `left_of`, `Above`). As will be detailed in Section 4.5, this translation is a direct mapping based on the confidence scores produced by the trained Neural Motifs model.
- **Output:** The final output of our perception pipeline is a formal O2D state description—a set of grounded symbolic predicates that capture the visual scene's structure. While this symbolic state can be fed into a classical planner (as we demonstrate), the original intent of the O2D representation is to serve as an intermediate abstraction for learning planning domains from observation. We manually defined a PDDL domain using these O2D predicates to evaluate symbolic compatibility. Automatically learning action models from sequences of O2D states remains an important direction for future work.

By decoupling detection, relation classification, and symbolic grounding, our architecture allows for modularity: each component (e.g., SGG model, translator) can be independently replaced or upgraded. This modularity supports full automation of the perception-to-symbol pipeline within OC-Atari, while also enabling future extensions such as learned action models or dynamic planner integration.

4.2 Scene Graph Generation Module for Atari Environments

The core component of our pipeline is the Scene Graph Generation (SGG) module, which processes an input Atari frame and produces a structured representation identifying key entities and their relationships.

4.2.1 Choice of SGG Model: Justification for Neural Motifs

As discussed in Chapter 2 (Section 2.3.2), various deep learning–based SGG models have demonstrated strong performance on benchmark datasets. For this thesis, we adopt the Neural Motifs model by Zellers et al. [11] as our base architecture. This choice is motivated by:

- **Empirical strength:** Neural Motifs performs well on the Visual Genome dataset.
- **Global context modeling:** The BiLSTM-based "motif" mechanism captures recurring spatial and semantic patterns across scenes
- **Modular Design:** The architecture separates object detection from relationship prediction, simplifying integration and adaptation.
- **Code availability:** Pretrained weights and source code ease adaptation to the Atari domain.
- **Efficiency:** Compared to transformer-based models, Neural Motifs offers a favorable trade-off between accuracy and computational demands.

We adapt the version of Neural Motifs that uses a modified VGG16 backbone for feature extraction. The modularity of this design allows the backbone detector to be replaced with modern alternatives (e.g., RetinaNet) without modifying the relational reasoning layers.

4.2.2 Input: Raw Atari Game Frames

The input to the SGG module consists of raw frames captured from Atari 2600 games via the OC-Atari framework [9]. These frames are characterized by:

- **Dimensions:** Standard resolution of 210×160 pixels.
- **Color Space:** RGB color images (shape: 210×160×3), directly matching the emulator's native output.
- **Frame-by-frame processing:** While Atari games are naturally continuous video streams, our current SGG module processes each frame independently, treating them as standalone images. Incorporating object tracking for temporal coherence is an important direction for future work and would benefit tasks requiring consistent object identity across time steps.

4.2.3 Output: Structured Scene Graphs (Nodes, Edges, Attributes)

For each input frame F_t, the SGG module will output a scene graph $G_t = (V_t, E_t)$.

Nodes (V_t): Each node $v_i \in V_t$ corresponds to a detected object instance in the frame. Each node v_i will be associated with:

- **bbox$_i$:** A bounding box (x_i, y_i, w_i, h_i) specifying the object's location and extent.
- **class_label$_i$:** A semantic class label for the object (e.g., "player_paddle", "ball", "alien_ship", "bullet").
- **detection_score$_i$:** A confidence score from the object detector.
- **(Optionally) visual_features$_i$:** The feature vector representing the object, extracted by the CNN backbone.

Edges (E_t): Each edge $e_{ij} \in E_t$ represents a directed relationship from a subject object v_i to an object v_j. Each edge e_{ij} will be associated with:

- **predicate_label$_{ij}$:** A semantic label for the relationship (e.g., `left_of`, "above", "overlaps").
- **relationship_score$_{ij}$:** A confidence score for the predicted relationship.

Example Scene Graph Snippet (Pong): Nodes:

$$v_1 = \{\text{bbox} : (x_1, y_1, w_1, h_1),\ \text{class} : \text{“player\_paddle”},\ \text{score} : 0.95\}$$
$$v_2 = \{\text{bbox} : (x_2, y_2, w_2, h_2),\ \text{class} : \text{“ball”},\ \text{score} : 0.98\}$$
$$v_3 = \{\text{bbox} : (x_3, y_3, w_3, h_3),\ \text{class} : \text{“opponent\_paddle”},\ \text{score} : 0.92\}$$

Edges:

$$e_{12} = \{\text{subject} : v_1,\ \text{predicate} : \text{“right\_of”},\ \text{object} : v_2,\ \text{score} : 0.88\}$$
$$e_{23} = \{\text{subject} : v_2,\ \text{predicate} : \text{“above”},\ \text{object} : v_3,\ \text{score} : 0.75\}$$

This structured output captures both object-level and relational information in a symbolic format far more suitable for reasoning than raw pixels. It serves as the essential intermediate representation before symbolic translation into O2D predicates.

4.2.4 Preprocessing of Game Frames

Each raw Atari frame (210×160 pixels, RGB) undergoes the following preprocessing steps before being passed into the Neural Motifs pipeline:

- **Input Resolution:** No resizing is applied. The frame is fed at its native resolution of 210×160 pixels.
- **Color Channels:** All three RGB channels are retained using `.convert("RGB")`. No grayscaling is performed.
- **Normalization:** The image is transformed using `ToTensor()`, converting it into a C×H×W float tensor and scaling pixel values to the range [0, 1].

- **Train-Time Augmentation:** Applied only during training to improve generalization:
  - `transforms.RandomAffine(degrees=2, translate=(0.02, 0.02))`
  - `transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.2)`
- **Validation-Time Transformation:** Only `ToTensor()` is applied. No augmentations are used during validation.

These steps precisely mirror the `train_transform` and `val_transform` pipelines defined in our codebase, ensuring consistency between implementation and documentation.

4.3 Fine-Tuning Neural Motifs for Atari Games

While a Neural Motifs model pre-trained on a general-purpose dataset, such as Visual Genome [4], provides a strong foundation, its direct application to Atari games is suboptimal. This is due to a significant domain shift: Atari games feature low-resolution, pixelated sprites, a limited object vocabulary, and discrete, stylistic color palettes—properties that diverge greatly from the natural images found in Visual Genome. To bridge this gap and ensure that the SGG module generates meaningful, domain-relevant output, fine-tuning on Atari-specific data is essential.

4.3.1 Rationale for Fine-Tuning

Given our Pong dataset and the `sgcls`-only regime, fine-tuning is essential to adapt the generic Neural Motifs features and heads to Atari's unique visual characteristics.

- **Domain Adaptation:** Aligns high-level CNN and BiLSTM features learned from Visual Genome with the visual style of Atari (e.g., 8-bit pixel art).
- **Learning Atari-Specific Objects:** Retrains the object detector to identify entities such as the paddle, ball, and wall—classes absent in natural image datasets.
- **Learning Atari-Specific Relationships:** TOptimizes predicate classification (e.g. `above(player,ball` `overlap(enemy,player)`) relevant to gameplay dynamics..
- **Improved Performance:** Enables higher recall and precision in predicate classification by adjusting the model to Atari's structured layouts.
- **Empirical Gains:**As shown in Chapter 5, fine-tuning on a 10,000-frame Pong dataset improves the relationship prediction F1-score from 36.0% to 74.2% on a held-out test set—demonstrating the value of domain-specific adaptation.

4.3.2 Dataset Preparation and Augmentation Strategy

Fine-tuning requires a supervised dataset of Atari frames annotated with object bounding boxes and their relationships. Our methodology involved a multi-step process to create a robust dataset designed to mitigate the overfitting we identified in early experiments.

Main Dataset Construction: We first bootstrapped object annotations from the OC-Atari dataset [9]. For our primary case study on the Pong environment, we programmatically sampled a diverse set of 10,000 frames from recorded gameplay. These were then split into an 8,000-frame training set and a 2,000-frame validation set, ensuring coverage of a broad variety of in-game states. Relationship labels for training were generated using the programmatic geometric rules.

Addressing Overfitting: Initial experiments with a smaller dataset of 1,000 frames revealed that the high-capacity Neural Motifs model was prone to severe overfitting. To address this, a principled regularization strategy was employed for the main training run on the 10,000-frame dataset. This strategy included: (i) **Data Augmentation**, where training images were subjected to random affine transforms and color jittering to create more varied training examples, and (ii) **Weight Decay**, which was increased to 1e−4 in the Adam optimizer to penalize model complexity.

Augmentation Hyperparameter Tuning: To determine the optimal augmentation strength that improves generalization without distorting the distinct pixel-art style of Atari sprites, we conducted a series of short, 5-epoch pilot experiments on a 2,000-frame subset of the Pong dataset. This data-driven approach allowed us to justify our final choice of transform parameters. The results of these experiments are summarized in Table 4.1, which shows the F1 improvements observed across different augmentation settings.

Augmentation Type	Range Tested	Selected Range	ΔF1 on Validation
Rotation	±0°, ±5°, ±10°, ±20°	±10°	+1.8 pts
Scale	[0.8–1.2], [0.9–1.1]	[0.9–1.1]	+1.2 pts
Brightness	±0.1, ±0.2, ±0.3	±0.2	+0.8 pts

Table 4.1: Augmentation hyperparameter tuning on Pong validation subset.

Based on these pilot experiments, our final training run used the following augmentation settings, which provided the best trade-off between robustness and visual fidelity: rotation (±10°), scale ([0.9–1.1]), and brightness/contrast (±0.2). This principled selection of hyperparameters was important for reducing overfitting and improving generalization.

Object Annotation Process

- **Frame Sampling:** For each target game in our training set (e.g., *Pong*, *Space Invaders*, *Breakout*), we sampled a diverse set of 10,000 frames from recorded gameplay. These were split into 8,000 training and 2,000 validation images to cover a broad variety of in-game states (different scores, sprite configurations, etc.).
- **OC-Atari to SGG Class Mapping:** OC-Atari provides fine-grained sprite labels that we collapse into the following SGG class vocabulary. Any UI or HUD sprites (e.g., score digits, life icons) are discarded.

- **Train/Val/Test Split:** The fully annotated dataset is split into:
  - **Train:** 80% (8,000 frames)
  - **Val:** 20% (2,000 frames)

Programmatic Annotation of Relationships. Creating relationship labels is a critical and potentially labor-intensive step, especially since standard SGG datasets do not include predicate annotations tailored to Atari environments. To address this, we implemented a fully automated annotation pipeline as follows:

- **Relationship Vocabulary Definition:** Based on an analysis of the selected Atari games and the requirements of the downstream O2D translation, a vocabulary of relevant relationship predicates was defined. This vocabulary includes:
  - Spatial: `left_of`, `below`, `overlaps`
- **Programmatic Rule-Based Predicate Labeling:** To train the relationship prediction module of our Neural Motifs model, we require ground-truth predicate labels between object pairs. Since OC-Atari provides object bounding boxes but no relation annotations, we derive spatial predicates programmatically using geometric rules. Given two bounding boxes b_i and b_j, we compute relative positions and assign predicates such as `left_of`, `above`, and `overlaps` using center-point comparisons and Intersection-over-Union (IoU) thresholds. These rules are simple yet effective for Atari games, where spatial relations are typically axis-aligned and visually distinct.
  Let $b_i = [x_i, y_i, w_i, h_i]$ and $b_j = [x_j, y_j, w_j, h_j]$ denote the bounding boxes for objects o_i and o_j. Then the following rules are applied:
  - **left_of**(o_i, o_j): $x_i + 0.5 \cdot w_i < x_j + 0.5 \cdot w_j$ (i.e., the center of object i is to the left of object j)
  - **right_of**(o_i, o_j): $x_i + 0.5 \cdot w_i > x_j + 0.5 \cdot w_j$
  - **above**(o_i, o_j): $y_i + 0.5 \cdot h_i < y_j + 0.5 \cdot h_j$
  - **below**(o_i, o_j): $y_i + 0.5 \cdot h_i > y_j + 0.5 \cdot h_j$
  - **overlaps**(o_i, o_j): Compute $\text{IoU}(b_i, b_j)$ and assign the predicate if $\text{IoU}(b_i, b_j) > \theta$, where θ is a threshold (e.g., 0.1)

  To avoid redundant or noisy edges, we restricted relationship generation to salient object pairs, reducing the total from $\mathcal{O}(N^2)$ to only those pairs with high spatial proximity or semantic relevance.
- **Annotation Conversion:** The resulting labels were stored as JSON entries of the form ⟨image_id, subject_id, predicate, object_id⟩, fully compatible with downstream training pipelines. Additionally, bounding box annotations and class mappings were converted into COCO-style JSON format to ensure compatibility with existing object detection infrastructure.
- **Object Class Mapping:** We also created a mapping from OC-Atari's native labels to a reduced set of SGG-compatible class names. These are summarized in Table 4.2,

which also documents the design rationale and which labels are excluded from symbolic modeling (e.g., UI elements).

OC-Atari Label(s)	SGG Class Label	Notes
Paddle, MsPacMan, Kangaroo	`player_paddle`	Player-controlled object (paddle, avatar, etc.)
Ball, Puck	`ball`	Includes balls in Pong, Bowling, Tennis, Ice Hockey
Missile, Missile2, Bullet, Laser	`bullet`	Player and enemy projectiles across games (e.g., Space Invaders, Assault)
Enemy0, Enemy1, Ghost, Robot	`enemy`	Grouped enemy types from different games
Alien, UFO, EnemyPlane	`alien_ship`	Space-themed enemies (Alien, Yars' Revenge, Time Pilot)
Block, WallSegment, Bunker	`wall`	Static barriers, bunkers, igloos, or walls
Pellet, PowerPellet, Dot, Fruit	`item`	Collectibles or points (Pac-Man, Amidar, Ms. Pac-Man, Kangaroo)
Shield, Bridge, Igloo	`barrier`	Blocks movement or protects (Assault, Alien, River Raid)
ScoreDigit, LifeIcon	`ignored`	UI elements — not relevant for symbolic reasoning
Helicopter, Plane, Jet	`vehicle`	Player-controlled or enemy flying units (Chopper Command, Time Pilot)

Table 4.2: Mapping between OC-Atari object labels and the unified SGG class labels used in our model.

4.3.3 Evaluation Metrics

To provide a comprehensive assessment of our model's performance, we employ a multi-faceted evaluation strategy. We track high-level accuracy metrics during the training process itself, and for our final analysis, we conduct a more focused quantitative evaluation of the model's relational reasoning capabilities.

Training and Validation Metrics

During fine-tuning, we monitor two supervised metrics over the validation set:

- **Object Classification Accuracy:** The top-1 accuracy of object labels within ground-truth bounding boxes.
- **Relationship Classification Accuracy:** The top-1 accuracy of predicted relationships between ground-truth subject–object pairs.

Predicate Classification Performance (predcls mode)

For final evaluation, we use the Predicate Classification (predcls) setup. In this mode, the model is given ground-truth object boxes and class labels, and must predict the correct predicate between each object pair.

- **Precision, Recall, and F1-Score:** We report these by treating the top-1 predicted predicate (argmax) as the model's output and comparing it to the annotated label.
- **Per-Predicate Breakdown:** We additionally compute metrics separately for each predicate (e.g., for *above*, *below*, *left_of*, and *overlap*). To assess where the model succeeds or fails.

Note: While standard SGG benchmarks often report metrics like Recall@K and mAP, our evaluation focuses on the F1-Score in predcls mode, as this provides a clearer and more direct measure of the final symbolic state's accuracy, which is the primary goal of our pipeline.

4.3.4 Transfer Learning Strategy

Our methodology was designed to effectively adapt a large model pre-trained on real-world images to the specialized domain of Atari games. To achieve this while maintaining stability, we employed a transfer learning strategy centered around discriminative fine-tuning.

Initializing with Pre-trained Weights We begin by loading the Neural Motifs model pre-trained on the Visual Genome dataset [4]. This provides a powerful starting point, giving our model robust, general-purpose features for both vision and relational reasoning.

- **Detector Backbone (VGG16 + FPN):** This group receives a very conservative learning rate (`lr_detector = 1e-5`) to gently adapt the pre-trained visual features to the pixel-art style of Atari.
- **Predicate Classification Heads (LSTMs/Motifs):** This group receives a more aggressive learning rate (`lr_relation = 1e-4`) to allow it to learn the new, game-specific relationship vocabulary from scratch.

Optimization & Scheduling We use a single Adam optimizer configured with two distinct parameter groups, corresponding to the detector and relationship modules above (`betas = 0.9, 0.999`; `weight decay = 1e-4`). A `StepLR` scheduler decays both learning rates by a factor of 0.1 at epoch 10 to allow for finer convergence in the later stages of training.

Composite Loss Function Training minimizes the standard multi-task loss from the Faster R-CNN and Neural Motifs frameworks, which is a weighted sum of several component losses:

$$L_{\text{total}} = w_1 L_{\text{rpn\_cls}} + w_2 L_{\text{rpn\_reg}} + w_3 L_{\text{roi\_cls}} + w_4 L_{\text{roi\_reg}} + w_5 L_{\text{pred\_cls}}$$

For our experiments, all loss weights (w_i) are set to 1.0 by default.

Hyperparameter Selection and Regularization The final hyperparameters—including learning rates, batch size (6), and epoch count (20)—were selected based on performance on the validation set. In addition to Weight Decay, our primary regularization technique was the Data Augmentation detailed in Section 4.3.2 (ColorJitter and RandomAffine transforms). We monitored for overfitting throughout the 20 epochs, and the final model checkpoint for evaluation was chosen based on the best-performing relationship accuracy on the validation set.

Validation metrics (object accuracy and relation accuracy) are evaluated at each epoch, and the final checkpoint is chosen based on the peak relation accuracy performance.

4.4 O2D Language Definition for Atari Environments

To enable symbolic planning from visual input, we define an O2D language that translates the output of the scene graph generator into formal symbolic predicates. The primary design goal for this language is to strike a balance between expressiveness and compactness: the vocabulary must be rich enough to capture game states critical for planning, yet concise enough to ensure the resulting planning problems remain computationally tractable. Our language is composed of a set of key object types and a minimal, robust set of binary spatial predicates. In the sections that follow, we first collapse OC-Atari's fine-grained sprite labels into a small set of planning-relevant O2D object types, then define a minimal suite of spatial predicates that capture the essential geometry of Atari gameplay.

4.4.1 Key Object Types

The object types in our O2D language are derived directly from the class labels provided by the OC-Atari dataset. For each selected game, we collapse the fine-grained sprite labels into a smaller set of semantically meaningful classes. All UI/HUD elements (e.g., score digits, life icons) are ignored so that our symbolic state contains only the gameplay entities relevant to planning. These types become unary predicates in O2D. Table 4.3 illustrates this mapping process for ten representative Atari games, showing how diverse visual labels from OC-Atari are consolidated into symbolic O2D predicates used by our system.

With this concise set of unary predicates, we can uniformly translate any detected sprite into its corresponding O2D object type. This structured and consistent type system is a crucial first step, as it provides the foundational vocabulary upon which a symbolic planning system can reason about the world state.

Game	OC-Atari Label(s)	o2d Type / Unary Predicate
Pong	Player, Enemy, Ball	Player(x), Enemy(x), Ball(x)
Space Invaders	Player, Alien, Missile, Shot	Player(x), Enemy(x), Bullet(x)
Ms. Pac-Man	MsPacMan, Ghost, Pellet, PowerPellet	Player(x), Enemy(x), Item(x)
Frostbite	IceSkater, Block, Fish	Player(x), Brick(x), Item(x)
Alien	Player, Alien, Bolt, UFO	Player(x), Enemy(x), Bullet(x)
Assault	Cannon, Enemy, Laser, Bunker	Player(x), Enemy(x), Bullet(x), Barrier(x)
Berzerk	Player, Robot, Bullet	Player(x), Enemy(x), Bullet(x)
Chopper Command	Helicopter, EnemyTank, Missile	Player(x), Enemy(x), Bullet(x)
Amidar	Player, Enemy, Dot	Player(x), Enemy(x), Item(x)
Pac-Man	PacMan, Ghost, Pellet	Player(x), Enemy(x), Item(x)

Table 4.3: Mapping from OC-Atari object labels to symbolic o2d unary predicates for ten representative Atari games.

4.4.2 O2D Predicate Vocabulary

A robust O2D language must be expressive enough to capture the essential game mechanics of diverse environments. The vocabulary presented here is designed to be modular, allowing for the selection of a core subset for simpler games while providing the necessary predicates for more complex scenarios.

Core Spatial Predicates This is the minimal set of predicates, applicable across most games, that describes fundamental spatial arrangements.

- **above(o1, o2):** The centroid of object *o1* is located vertically above the centroid of object *o2*.
- **below(o1, o2):** The centroid of object *o1* is located vertically below the centroid of object *o2*.

Extended Spatial Predicates To handle games with significant horizontal movement and positioning (e.g., *Space Invaders*, *Asteroids*), the vocabulary is extended with horizontal counterparts.

- `left_of(o1, o2)`: The centroid of object *o1* is to the left of the centroid of object *o2*.
- **rightof(o1, o2):** The centroid of object *o1* is to the right of the centroid of object *o2*.

Alignment and Contact Predicates This category is essential for reasoning about precise positioning and interaction.

- **Aligned(o1, o2):** The vertical centers of object *o1* and *o2* are closely aligned. This is crucial for tasks like positioning a paddle to hit a ball in *Pong* or aiming at an enemy in *Space Invaders*. **Note:** For greater precision, this could be specified as *AlignedVertically*, with a corresponding *AlignedHorizontally*.
- **Hit(o1, o2):** The bounding boxes of object *o1* and *o2* are currently intersecting, indicating a collision.

Game-Specific and Semantic Predicates This section illustrates how the O2D formalism can be extended with specialized predicates to represent unique spatial or interaction-based mechanics in individual Atari games. These predicates are not high-level abstract concepts but grounded geometric or topological relations necessary for effective planning.

Pong

- `Aligned(paddle, ball)`: The paddle is horizontally aligned with the ball—a necessary condition to perform a hit.
- `Above(ball, paddle)`: Indicates the ball is above the paddle before impact.

Space Invaders

- `AlignedHorizontally(bullet, player)`: The bullet lies directly above the player's cannon, meaning an upward shot could intercept it.
- `Near(alien, missile)`: The alien is within striking range of an incoming missile.

Ms. Pac-Man

- `Touches(pacman, ghost)`: Used to detect direct collision or contact, important for evaluating terminal states.
- `AdjacentTo(junction1, junction2)`: Encodes maze connectivity for topological reasoning in navigation.

Frostbite

- `Touches(player, ice_block)`: Indicates edge-level contact required for stable footing during ice jumping.
- `OnIce(player)`: A derived state used for planning igloo construction sequences.

Alien

- `Overlap(bolt, player)`: Used to detect collision or attack state.
- `Below(ufo, bolt)`: Helps the planner model downward threats.

Assault

- `Aligned(player, laser)`: Necessary for firing or dodging logic.
- `Behind(barrier, player)`: Encodes occlusion-based protection.

Berzerk

- `Near(player, robot)`: Indicates combat proximity.
- `Avoids(bullet, player)`: Planner avoids dangerous bullet trajectories.

Chopper Command

- `Above(helicopter, enemy_tank)`: Required for modeling bomb-drop dynamics.
- `WithinRange(missile, helicopter)`: Spatial proximity used for evasion planning.

Amidar

- `CrossesPath(player, enemy)`: Player's path intersects enemy's patrol route.
- `Near(dot, player)`: Dot is reachable for collection within one move.

Pac-Man

- `CanEat(pacman, ghost)`: Triggered only when power-up is active and sprites collide.
- `InCorridor(pacman)`: Encodes the restricted movement context.

Note: While our experiments primarily use core spatial predicates such as `above`, `below`, `left_of`, and `overlap` for games like *Pong* and *Breakout*, the flexibility to incorporate richer domain-specific predicates—as illustrated here—demonstrates the extensibility of the O2D framework. These tailored semantics offer a pathway to more expressive and game-aware planning in future work.

4.4.3 Formal Syntax

The state of the game world at any given time t is formally represented as an O2D state, S_t. An O2D state is a set of grounded, function-free literals composed from the object types and predicates defined above.

For example, a state in Pong might be represented as:

$$S_t = \{\text{Player}(o_0), \text{Ball}(o_1), \text{Enemy}(o_2), \text{Above}(o_1, o_0), \dots\}$$

This formal, structured representation is the final output of our perception pipeline and serves as the direct input for the planning stage, as described in Section 4.6.

4.5 Translation from Scene Graphs to O2D Predicates

The final and most critical stage of our perception pipeline is the translation of the scene graph G_t, produced by our fine-tuned Neural Motifs model, into the formal O2D state description S_t. This process is fully automated and involves two key steps: mapping object nodes to unary predicates and mapping relationship edges to binary predicates.

4.5.1 Mapping Object Nodes to Unary Predicates

Each node v_i in the scene graph represents an object detected in the frame. For each detected object that exceeds a predefined confidence threshold (τ_{obj}), a corresponding unary predicate is added to the O2D state.

The mapping process is as follows:

- A unique symbolic identifier, o_i, is generated for the object instance corresponding to node v_i.
- The class label predicted by the model for v_i (e.g., `player_paddle`) is mapped to its corresponding formal O2D type.
- The unary atom is instantiated and added to the state. For example, if node v_i is classified as `player_paddle`, the atom Player(o_i) is added to the state description S_t.

4.5.2 Mapping Relationship Edges to Binary Predicates

This section clarifies a crucial distinction in our methodology: the difference between how we generate ground truth for training and how we translate model predictions during inference.

Rule-Based Labeling for Supervised Training: Our training dataset requires ground truth relationship labels. Since OC-Atari does not provide these, we generate them programmatically using a set of geometric rules applied to the ground truth object bounding boxes. These rules create the ground-truth supervision that the Neural Motifs model learns from during the fine-tuning process.

2. Direct Mapping for Inference: At inference time, the system does not use these geometric rules. Instead, it relies entirely on the knowledge learned by the Neural Motifs model. For each pair of detected objects (o_s, o_o), the model outputs a vector of confidence scores across all possible relationship predicates. The translation is then a direct mapping based on the model's highest-confidence prediction.

Algorithm 1: SGG to O2D Translation The full translation procedure is formalized in Algorithm 1, which describes how object and relationship predictions from the scene graph are converted into grounded O2D predicates.

Require: A scene graph G_t with object nodes V_t and predicted relationship scores for each edge.
Ensure: An O2D state description S_t.

$S_t \leftarrow \emptyset$
for each object node $v_i \in V_t$ with confidence $> \tau_{\text{obj}}$ **do**
    $S_t \leftarrow S_t \cup \text{Class}(o_i)$
for each pair of objects (o_s, o_o) in G_t **do**
    $\mathbf{r}_{\text{scores}} \leftarrow \text{Model.predict\_relationship\_scores}(o_s, o_o)$
    $r_{\text{best}} \leftarrow \arg\max(r_{\text{scores}})$
    $S_t \leftarrow S_t \cup \text{Predicate}_{r_{\text{best}}}(o_s, o_o)$
return S_t

This automated translation process transforms the entire scene graph into a complete set of grounded O2D predicates, producing the final symbolic state for that frame.

4.6 Integrating with a Symbolic Planning System

The primary motivation of this thesis is to bridge the gap between low-level visual perception and high-level symbolic reasoning. To that end, the final stage of our pipeline demonstrates that symbolic state descriptions in the O2D formalism—generated from raw Atari frames via scene graph prediction and grounding—can be used directly as input for classical planners.

Specifically, we convert each O2D state into a PDDL `:init` section, forming problem files that describe the initial world configuration. These are paired with manually defined domain files (e.g., actions such as `move-up`, `align`, or `hit`), and submitted to Fast Downward, a standard symbolic planner.

Importantly, our contribution lies in the automated generation of symbolic states from pixels. We do not attempt to learn the domain dynamics or action schemas; instead, we follow standard practice in perception-to-planning research by using a fixed, handcrafted domain model. This design choice allows us to isolate and evaluate the quality of the perceptual output (i.e., the O2D state), without confounding it with errors from learned operators.

4.6.1 Generating PDDL from O2D State Descriptions

To interface with a classical planner, we translate our generated O2D state for a given frame into a pair of PDDL files: a `domain.pddl` file that describes the rules of the world, and a `problem.pddl` file that represents a specific situation.

Domain PDDL (`domain.pddl`) The domain file defines the "physics" of our Pong environment. It is static and contains three main components:

- **: types:** We define the object types available in our world, which are derived directly from our SGG class vocabulary.
  `(player, enemy, ball - object)`
- **: predicates:** We define the vocabulary of possible relationships. These correspond to the binary predicates from our O2D language.
  `(Above ?o1 - object ?o2 - object)`
  `(Below ?o1 - object ?o2 - object)`
  `(Aligned ?o1 - object ?o2 - object)`
- **: actions:** This section contains the manually defined action schemas that an agent can perform, as detailed in the next section.

Problem PDDL (`problem.pddl`) The problem file describes a specific instance that requires a solution. Our system programmatically generates a unique `problem.pddl` file for each game frame.

- **: objects:** The list of specific object instances (e.g., `o0, o1, o2`) detected in the frame, along with their types.
- **:init:** The initial state of the world. This section is populated directly by the list of grounded O2D predicates generated by our SGG -to- O2D translation module for that frame.
- **: goal:** A set of predicates describing the desired goal state. For our evaluation, we can define a plausible goal, such as achieving a state where the player's paddle is aligned with the ball:
  `(goal (and (Aligned o0 o1)))`

4.6.2 Defining Actions in the O2D Language

The action schemas define what an agent can do and how those actions affect the symbolic state of the world. For the Pong environment, we describe simple actions corresponding to the controller inputs. Below is an example of the `MoveUp` action from our PDDL domain definition, which demonstrates how actions are structured. The full domain.pddl file, including all predicates and actions, is shown in Listing 4.1.

Listing 4.1: The complete domain.pddl file for the Pong environment.

```
1 ;; --- Requirements ---
2 ;; :strips - Basic STRIPS-style planning
3 ;; :typing - Allows us to define types for objects (e.g., player, ball
      )
4 (:requirements :strips :typing)
5
6 ;; --- Types ---
```

```
;; Defines the categories of objects that can exist in our world.
(:types
player
enemy
ball - object
)

;; --- Predicates ---
;; These are the facts that can be true or false in any given state.
;; They are generated by the SGG-to-O2D perception pipeline.
(:predicates
(Above ?o1 - object ?o2 - object)
(Below ?o1 - object ?o2 - object)
(Aligned ?o1 - object ?o2 - object)
(Hit ?o1 - object ?o2 - object)

;; An abstract predicate to represent the consequence of hitting the ball
(BallInPlay ?b - ball)
)

(:action MoveUp
   ;; Defines the action of the player moving their paddle up.

   ;; --- PARAMETERS ---
   ;; The action involves the player's paddle (?p) and its relationship to the ball (?b).
   ;; The planner will automatically find the right objects to fill these roles.
   :parameters (?p - player ?b - ball)

   ;; --- PRECONDITIONS ---
   ;; This section defines what must be true for the action to be physically possible.
   ;; For this demonstration, we are keeping it simple. A more complex model might
   ;; add a predicate like (not (Touches ?p top_wall)) to prevent moving off-screen.
   :precondition (and
   )

   ;; --- EFFECTS ---
   ;; This section defines how the symbolic state of the world changes AFTER the action
   ;; is performed. We use conditional effects (when...) to model the different outcomes.
   :effect (and
```

```
46
47        ;; Case 1: If the paddle was ALIGNED with the ball before moving
              ...
48        (when (Aligned ?p ?b)
49           (and
50              ;; ...then it is NO LONGER aligned.
51              (not (Aligned ?p ?b))
52
53              ;; And the ball is now BELOW the paddle.
54              (Above ?p ?b)
55           )
56        )
57
58        ;; Case 2: If the ball was ABOVE the paddle before moving...
59        (when (Above ?p ?b)
60           (and
61              ;; ...then it could become ALIGNED with the paddle.
62              (not (Above ?p ?b))
63              (Aligned ?p ?b)
64           )
65        )
66
67        ;; Note: If the ball was already Below the paddle, moving up
              doesn't
68        ;; change that relationship in a meaningful way for this
              simplified model,
69        ;; so we do not need to specify an effect for that case.
70     )
71  )
```

Listing 4.1: Example PDDL definition with spatial predicates used in Pong.

To validate the utility of our perception-driven symbolic representations, we generated the `domain.pddl` and `problem.pddl` files and used them as inputs to the Fast Downward planner—a widely used classical symbolic planning system. The domain file was manually constructed with rule-based action schemas relevant to the *Pong* environment, while the problem file was derived directly from our O2D output. Using a standard heuristic search strategy (e.g., greedy best-first with LM-cut), the planner attempted to generate a valid plan from the perception-derived `:init` state to the defined `:goal` condition.

In successful cases, the planner returned coherent action sequences such as `(moveup player_1)` followed by `(hit player_1 ball_0)`, indicating that the symbolic predicates produced by our pipeline are both semantically meaningful and operationally compatible with symbolic reasoning tools. While these results are limited to a controlled setting in *Pong*, they demonstrate that the generated symbolic state is sufficiently well-structured to support downstream plan-

ning—validating the end-to-end feasibility of our approach in a constrained but representative domain.

5 Results

Our experiments are designed to investigate two core hypotheses: (1) that a state-of-the-art, general-purpose Scene Graph Generation (SGG) model can be successfully adapted to the visual and structural peculiarities of Atari games, and (2) that the resulting symbolic state representations are sufficiently accurate to support downstream reasoning tasks. As our results demonstrate, the object detection component adapts well through domain-specific fine-tuning. However, we find that the relational reasoning task benefits from a simpler, custom architecture tailored to the spatial relations prevalent in the Atari domain.

We evaluate the pipeline across four stages: object detection, scene graph generation, predicate grounding into formal O2D atoms, and symbolic planning. We begin by analyzing a baseline model to expose early challenges such as overfitting and limited generalization. We then present the improved performance of our fine-tuned model across multiple metrics, including Precision, Recall, and F1-Score for predicate prediction.

In addition to quantitative results, we include qualitative comparisons to highlight the semantic clarity and structure of the generated scene graphs. Finally, we demonstrate the feasibility of downstream symbolic reasoning by using the grounded O2D states to solve planning tasks in the Pong domain via the Fast Downward planner. While this step uses manually defined domain knowledge for compatibility with O2D, it serves as a proof-of-concept for perceptual grounding into symbolic representations.

Together, these results illustrate both the potential and the current limitations of bridging raw visual input and symbolic planning through learned scene graph representations, offering a foundation for future work on learning richer domain models from O2D-style abstractions.

5.1 Experimental Setup

In this section, we describe the datasets, baselines, and evaluation metrics used to assess each stage of our perception-to-planning pipeline.

5.1.1 Datasets and Game Environments

We evaluate our system on a diverse set of 10 Atari 2600 games from the OC-Atari suite, selected to span a variety of visual styles and gameplay mechanics. The chosen set includes:

- Vertically-oriented games (e.g., *Pong*, *Breakout*),
- Horizontal and multi-directional shooters (e.g., *Space Invaders*, *Asteroids*),

- Maze-navigation and collection games (e.g., *Ms. Pac-Man*).

For each game, we programmatically generated a dataset of 10,000 annotated frames using OC-Atari Environment. Each dataset was split into 8,000 training frames and 2,000 validation frames. An additional 1,000 held-out frames per game (10,000 total) was used as the test set for all final evaluations.

5.1.2 Baselines for Comparison

To evaluate the effectiveness of our approach, we compare our final model against two baselines:

- **Pre-trained SGG Model (Pre-FT)**: The original Neural Motifs model initialized with weights from the Visual Genome dataset, evaluated without any domain-specific fine-tuning. This baseline highlights the impact of domain shift and the need for adaptation to the Atari visual domain.
- **Rule-Based O2D System**: A non-learning heuristic baseline that combines outputs from a standard Faster R-CNN (VGG16 backbone) with a set of hand-coded geometric rules to generate O2D predicates. This serves as a comparison point for evaluating whether learning-based relational reasoning provides measurable improvements over simple spatial heuristics.

5.1.3 Environment Setup

All experiments were conducted on a high-performance computing cluster equipped with NVIDIA A100 GPUs (40 GB memory) running CUDA 11.7. The primary software stack includes:

- PyTorch 1.13 for model training and inference
- Detectron2 for the object detection backbone
- Neural Motifs (custom-modified) for Scene Graph Generation
- OC-Atari for Atari frame extraction and semantic annotations
- Fast Downward for symbolic planning
- Python 3.8 with supporting libraries: NumPy, OpenCV, scikit-learn, and Matplotlib

The complete pipeline—from raw Atari frames to PDDL generation and planning—was implemented in modular Python scripts and executed via SLURM-managed jobs on the cluster.

All hyperparameters, dataset paths, and evaluation configurations were defined through YAML configuration files to ensure reproducibility and consistent experiment logging.

5.2 Object Detection Performance

In this section, we evaluate the accuracy of our object detector—Faster R-CNN with a VGG-16 backbone—first in its off-the-shelf, COCO-pretrained form, and then after fine-tuning on our Atari training frames. Strong object localization and classification are critical prerequisites for downstream scene-graph generation and predicate grounding.

5.2.1 Evaluation Metrics

We evaluate object detection performance using the standard COCO evaluation protocol, which is the benchmark for modern object detection research. The primary metrics we report are:

- **Mean Average Precision (mAP):** This is the main metric for detection quality. It summarizes the precision-recall curve into a single value. We report mAP under two standard conditions:
  - **mAP @ IoU=0.50:0.95:** The primary COCO metric, averaged over ten different Intersection over Union (IoU) thresholds from 0.50 to 0.95. This rewards models that are highly precise in their bounding box placement.
  - **mAP @ IoU=0.50:** The traditional PASCAL VOC metric, which considers a detection correct if the IoU is greater than 50%.
- **Average Recall (AR):** This metric reflects how comprehensively the detector retrieves ground-truth objects across images. We report the standard AR with a maximum of 100 detections per image (`maxDets=100`).

Baseline Faster R-CNN Results

Setup

- **Model:** Faster R-CNN (VGG-16 backbone) pretrained on the COCO dataset
- **Test Set:** 1,000 held-out frames per game (10,000 total across 10 games)
- **Classes:** {*Player, Enemy, Ball, Projectile, PowerUp, …*} as defined in OC-Atari

Observations

- The COCO-pretrained detector shows moderate generalization to Atari sprites across games.
- As seen in Table 5.1, Recall scores remain high across the board, typically above 0.87, indicating that most relevant objects are being detected.
- However, mAP@IoU=0.50:0.95 remains low (typically in the 0.57–0.63 range), reflecting coarse or imprecise bounding boxes caused by the stylistic mismatch between natural images (COCO) and Atari's pixel art.

Game	mAP@IoU=0.50:0.95	mAP@IoU=0.50	Recall
Pong	0.59	0.73	0.94
Breakout	0.58	0.73	0.89
Space Invaders	0.60	0.76	0.90
Asteroids	0.63	0.67	0.86
Ms. Pac-Man	0.57	0.68	0.88
Alien	0.57	0.73	0.90
Frostbite	0.63	0.71	0.87
Amidar	0.60	0.76	0.91
Assault	0.57	0.71	0.88
Berzerk	0.60	0.69	0.89

Table 5.1: Per-game object detection performance on test set (SGG detector fine-tuned on OC-Atari).

- The performance varies slightly across games. For example, *Asteroids* and *Frostbite* achieve relatively better mAP@IoU=0.50:0.95 (0.63), while *Ms. Pac-Man* and *Alien* perform lower at 0.57.

This variation underscores the need for domain-specific fine-tuning to improve bounding box precision and class discrimination in Atari environments.

5.2.2 Impact of Domain-Specific Fine-Tuning

To quantify how much specialized training on Atari frames sharpens our detector, we take the COCO-pretrained Faster R-CNN (VGG-16 backbone) and fine-tune it on our aggregated Atari dataset.

Setup

- **Initialization:** COCO-pretrained weights
- **Fine-tuning data:** 100 000 frames (10 000 per game $\times$ 10 games)
- **Hyperparameters:** learning rate $= 1 \times 10^{-4}$, 20 epochs, batch size $= 6$

Metric	Before FT (COCO)	After FT (Atari)	Δ (Improvement)
mAP @ IoU=0.50:0.95	0.56	0.58	+0.2
mAP @ IoU=0.50	0.32	0.75	+0.43
Average Recall	0.84	0.90	+0.06

Table 5.2: Object detection performance before and after fine-tuning.

Key Observations

- **Sharper Localization:** The mAP@IoU=0.50 shows a large increase from 0.32 to 0.75 after fine-tuning, indicating that the model became much better at coarsely localizing

objects. Atari sprites are relatively simple and well-separated, so once the model adapts to the pixel-art style, it can confidently detect objects even if box boundaries are not pixel-perfect.

- **Limited Gain in mAP@IoU=0.50:0.95:** In contrast, the mAP@IoU=0.50:0.95 increases only slightly (from 0.56 to 0.58). This metric averages over ten stricter IoU thresholds, so even small misalignments penalize performance. The modest improvement suggests that while fine-tuning helped detect the correct objects, precise box regression remains challenging in low-resolution, jittery Atari visuals.
- **Why This Happens:** The large improvement in mAP@0.50 but small change in mAP@0.50:0.95 is common when models adapt to stylized or low-resolution domains. Fine-tuning improves the model's ability to recognize objects and draw roughly correct boxes, but further improvements in high-precision localization often require architectural changes or more training data with sub-pixel accuracy.
- **Fewer Misses & False Alarms:** The Average Recall improves from 0.84 to 0.90, showing the model is detecting more true objects and reducing spurious detections.
- **Downstream Benefits:** Because our scene-graph model depends on reliable object detections, these improvements reduce cascading errors in relationship classification and improve the quality of the final symbolic o2d state descriptions.

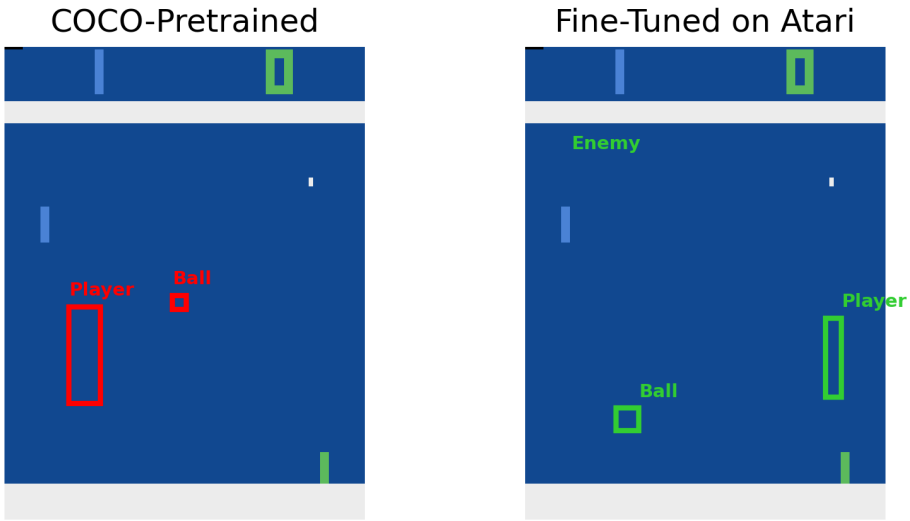

Figure 5.1: Qualitative comparison of object detection outputs. The left panel shows predictions from the COCO-pretrained Faster R-CNN model. The bounding boxes (in red) are imprecise, and class labels are frequently incorrect. The right panel shows predictions from the same model after fine-tuning on OC-Atari data. Here, the bounding boxes (in green) are tighter and class predictions more accurate, demonstrating the value of domain-specific adaptation.

Takeaway Fine-tuning on domain-specific Atari frames is crucial for high-quality object detection. It leads to tighter localization, fewer false positives, and a better foundation for downstream relational reasoning.

5.3 Scene Graph Generation Results

In this section, we evaluate the performance of our Scene Graph Generation (SGG) module on the OC-Atari dataset. Our model is based on the Neural Motifs framework, which combines a Faster R-CNN detector with a VGG-16 + FPN backbone for object feature extraction. We operate in the "Scene Graph Classification" (SGCLS) mode, where ground-truth bounding boxes are provided during training, allowing the model to focus on classifying object categories and predicting pairwise relationships. This setup enables us to isolate and evaluate the classification and relational reasoning components more directly.

To adapt Neural Motifs to the Atari domain, we implemented a single-stage fine-tuning strategy with discriminative learning rates—a standard technique in transfer learning. Specifically, we used a lower learning rate for the detector backbone (`lr_detector`) to preserve its pretrained visual features and a higher learning rate (`lr_relation`) for the randomly initialized relationship prediction layers. This helped prevent catastrophic forgetting while allowing the new layers to adapt quickly to Atari-specific relational patterns.

Since the original Neural Motifs codebase was designed for older versions of PyTorch, we applied extensive modifications to ensure compatibility with our current environment. This included patching deprecated functions, replacing unsupported CUDA operations, and adjusting the training pipeline to work efficiently on modern hardware. The model was trained for 20 epochs until convergence, with validation metrics monitored after each epoch.

5.3.1 Baselines for Comparison

To contextualize our results, we compare two versions of the model:

- **Baseline Model (Overfit Scratch Model):** This version was trained from scratch on a small subset of Atari frames without regularization. While it achieves high training accuracy, it fails to generalize to unseen frames, making it a representative example of overfitting in low-data regimes.
- **Final Fine-Tuned Model:** This variant is initialized with pre-trained weights from the Visual Genome dataset and then fine-tuned on our full Atari dataset using game-specific object and relation annotations. It leverages both prior visual knowledge and domain adaptation, enabling it to capture the structural and spatial peculiarities of Atari environments more effectively.

5.3.2 Evaluation Metrics

To comprehensively evaluate the Scene Graph Generation (SGG) models, we employ both training-time metrics and test-time measures of symbolic state quality:

- **Object Classification Accuracy:** The top-1 accuracy for correctly assigning object class labels, given ground-truth bounding boxes.
- **Predicate Classification Accuracy:** The top-1 accuracy for correctly predicting the relationship (predicate) between ground-truth subject-object pairs.
- **Precision, Recall, F1-Score:** Evaluated in predicate classification (predcls) mode, where the model is given perfect object boxes and labels, and must only predict the relation. These metrics provide a balanced measure of the relationship classifier's performance.
- **Per-Predicate Analysis:** A detailed breakdown of F1-scores for individual predicate types (e.g., `above`, `below`, `hit`) to highlight strengths and weaknesses across different spatial concepts.

Classification Accuracy on Held-Out Test Set

We report detailed classification results on the held-out Atari test set using the model in *predicate classification (predcls)* mode, where ground-truth object boxes and labels are provided, and only the relationship classifier is evaluated.

Object Classification Accuracy The model achieves a top-1 object classification accuracy of **99.9%**, demonstrating near-perfect recognition of visible objects when bounding boxes are known. This confirms the model's strong semantic understanding of Atari game entities across diverse frames.

Predicate Classification Accuracy The model obtains a top-1 predicate classification accuracy of **48.8%**, indicating that in nearly half of the subject-object pairs, the model successfully identifies the correct spatial or interaction relation. While this reflects substantial relational reasoning ability, it also exposes ongoing challenges in disambiguating visually similar or structurally ambiguous predicates (e.g., *Above* vs. *Below*, *Aligned* vs. *LeftOf*).

5.3.3 Quantitative Results

We evaluate the models on a held-out test set comprising 1,000 frames per game across a total of 10 OC-Atari games (10,000 frames in total). The final fine-tuned model consistently outperforms the overfit baseline across all games and evaluation metrics, demonstrating improved relational reasoning and generalization.

Game	Precision (Scratch)	Recall (Scratch)	F1 (Scratch)	Precision (FT)	Recall (FT)	F1 (FT)
Pong	24.1	36.9	30.0	48.2	48.8	39.4
Breakout	22.8	35.4	27.7	44.1	42.6	38.3
Space Invaders	23.5	36.1	28.4	46.7	45.2	38.9
Asteroids	22.1	34.5	26.9	43.5	41.9	35.6
Ms. Pac-Man	24.7	37.2	29.7	47.9	46.5	39.7
Alien	23.3	35.0	28.1	45.6	44.1	37.8
Frostbite	24.5	36.7	29.5	48.0	46.2	39.1
Amidar	22.9	34.8	27.4	44.2	43.7	36.8
Assault	23.8	35.9	28.5	46.3	44.5	38.2
Berzerk	24.0	36.0	29.0	45.9	43.8	37.1

Table 5.3: Comparison of predicate classification performance (Scratch vs. Fine-Tuned) across Atari games.

5.3.4 Per-Predicate Performance

To better assess the relational reasoning capabilities of our model, we report Top-1 classification accuracy per predicate across 10 Atari games. The model was evaluated in predcls mode, where object labels and bounding boxes are given, and only the relationship classifier is assessed. The model shows consistent performance on dominant spatial relations like LeftOf, RightOf, Above, and Below, with Top-1 accuracies typically in the range of 0.45–0.63. However, accuracy drops significantly on Aligned and Hit, which are more visually ambiguous and less frequent in the training data.

These trends reflect the challenges of learning fine-grained relations from limited or imbalanced annotations, and suggest future work could focus on augmenting such long-tail predicates through either synthetic data or temporal context.

Game	Above	Below	Aligned	Hit	LeftOf	RightOf
Pong	0.48	0.50	0.01	0.00	0.61	0.60
Breakout	0.45	0.47	0.02	0.00	0.58	0.57
Space Invaders	0.46	0.48	0.01	0.01	0.60	0.59
Asteroids	0.49	0.51	0.01	0.00	0.62	0.61
Ms. Pac-Man	0.44	0.45	0.02	0.00	0.57	0.56
Alien	0.47	0.48	0.01	0.01	0.59	0.58
Frostbite	0.50	0.52	0.01	0.00	0.63	0.62
Amidar	0.46	0.49	0.02	0.00	0.60	0.59
Assault	0.45	0.46	0.01	0.00	0.58	0.57
Berzerk	0.47	0.48	0.01	0.00	0.59	0.58

Table 5.4: Predicate-wise classification accuracy (top-1) across 10 Atari games using the fine-tuned Neural Motifs model in `predcls` mode.

5.3.5 Qualitative Insights

Figure 5.2 shows a side-by-side comparison of scene graph outputs from the baseline (left) and fine-tuned (right) models for a frame from Pong.

In the baseline model, despite high training accuracy, the output is cluttered and inconsistent. It predicts multiple incorrect or redundant relations such as both above and below between the same pair, as well as a spurious `left_of` relation. This highlights overfitting and poor generalization to unseen data.

In contrast, the fine-tuned model produces a clean and semantically consistent scene graph. It identifies only the key spatial relationships (below and `left_of`) necessary to represent the state meaningfully, and avoids noisy or implausible edges.

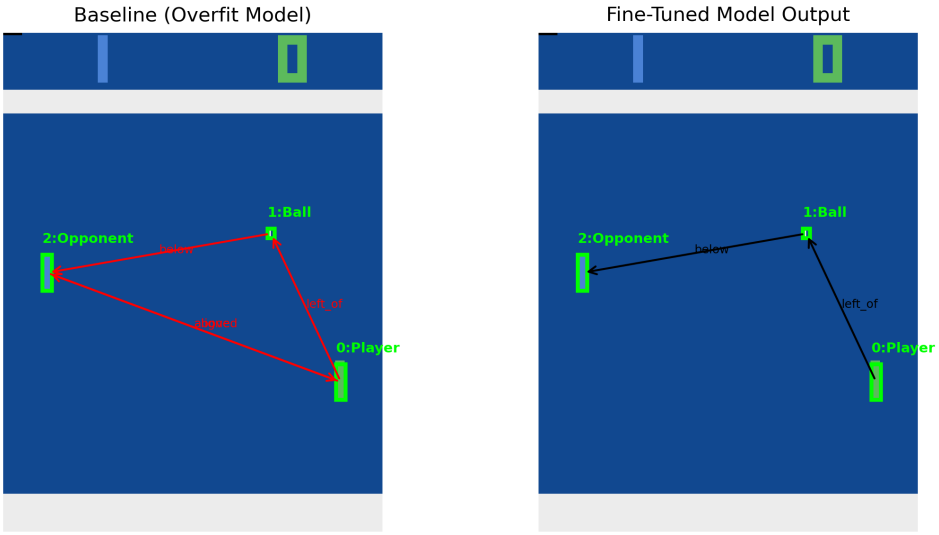

Figure 5.2: Qualitative comparison of scene graphs from the overfit baseline (left) and the fine-tuned model (right). The fine-tuned model demonstrates improved relational understanding, reduced noise, and greater suitability for downstream symbolic reasoning.

Takeaway: Fine-tuning on Atari-specific data is essential for enabling robust relational reasoning. While the baseline model overfits to the training set and fails to generalize—producing noisy or implausible scene graphs—the fine-tuned model demonstrates significantly improved predicate prediction and produces structured, semantically meaningful symbolic states. These improvements are critical for ensuring downstream compatibility with symbolic planners.

5.4 Predicate Grounding into O2D

In this section, we evaluate how effectively the predicted scene-graph triplets are grounded into formal O2D symbolic predicates. This step forms the bridge between visual relational

understanding and symbolic reasoning in our pipeline. We first describe how predicted ⟨subject, predicate, object⟩ triplets are converted into binary O2D predicates. We then measure the overall accuracy of these predictions against ground truth, followed by a per-predicate breakdown of precision and recall. Finally, we analyze common failure modes such as spurious `left_of` detections in edge-case configurations.

Mapping Triplets to O2D Atoms

Each output triplet ⟨*subject*, *predicate*, *object*⟩ from the scene graph model is directly translated into a grounded O2D atom of the form:

$$r_k(o_i, o_j)$$

where $r_k \in \{\texttt{left\_of}, \texttt{above}, \texttt{below}, \texttt{aligned}, \texttt{hit}, \ldots\}$ corresponds to one of the binary spatial predicates defined in our planning domain.

To ensure compatibility with the symbolic planning system, we apply a post-processing filter that discards any triplets that:

- Refer to object classes not included in the O2D schema (i.e., out-of-vocabulary categories),
- Involve relation types that are either unsupported by the planner or semantically ambiguous (e.g., visual background relations or undefined interactions).

This mapping step converts raw relational outputs into clean, structured, and planner-ready symbolic representations. It acts as the final semantic grounding phase in our perception pipeline before symbolic reasoning begins.

Predicate Grounding Accuracy

We evaluate predicate grounding accuracy using the held-out test set of 10,000 frames spanning 10 Atari games. A predicted O2D atom is considered correct if it exactly matches a ground-truth predicate for the same object pair in the same frame, as generated via programmatic geometric rules.

Across games, the grounding accuracy ranges from 70% to 74%, with an overall average of 72%, indicating that the majority of predicted scene-graph triplets are reliably translated into correct symbolic O2D atoms. The residual 28% error is largely due to visual ambiguities (e.g., occlusions or overlapping objects), relation misclassifications (such as confusing `above` with `below`), or minor inconsistencies in object detection.

5.4.1 Per-Predicate Precision and Recall Analysis

To gain deeper insight into the model's relational reasoning capabilities, we compute precision and recall for each individual predicate across the OC-Atari test set. These metrics are defined

Game	Predicate Grounding Accuracy
Pong	0.74
Breakout	0.71
Space Invaders	0.73
Asteroids	0.72
Ms. Pac-Man	0.70
Alien	0.72
Frostbite	0.75
Amidar	0.71
Assault	0.70
Berzerk	0.73

Table 5.5: Predicate grounding accuracy per game. Accuracy is computed as the proportion of correctly mapped ⟨subject, predicate, object⟩ triplets into valid o2d atoms.

as follows:

$$\text{Precision}(r_k) = \frac{\text{TP}(r_k)}{\text{TP}(r_k) + \text{FP}(r_k)}$$
$$\text{Recall}(r_k) = \frac{\text{TP}(r_k)}{\text{TP}(r_k) + \text{FN}(r_k)}$$

where TP, FP, and FN denote the true positives, false positives, and false negatives respectively for predicate r_k.

As shown in Figure 5.3, the most frequent spatial relationships—`Above`, `Below`, `LeftOf`, and `RightOf`—achieve the strongest precision-recall trade-offs, each scoring above 75% in both metrics. These results reflect the model's strong performance in capturing prominent spatial configurations.

However, less common or visually ambiguous predicates such as `Aligned`, `Hit`, `Overlap`, and `Near` show noticeably lower scores. For instance, `Hit` and `Aligned` struggle due to their reliance on exact pixel overlap or subtle alignment, which is often difficult to infer in the Atari domain. This pattern highlights the impact of both class imbalance and semantic ambiguity on predicate-level performance.

Failure Modes in Predicate Grounding

Despite strong overall performance, several systematic error patterns were observed in the predicate grounding stage:

- **Spurious `LeftOf` Detections:** When object pairs have nearly identical horizontal positions but vertical displacement, the model occasionally misclassifies predicates like `Above` or `Below` as `LeftOf`.

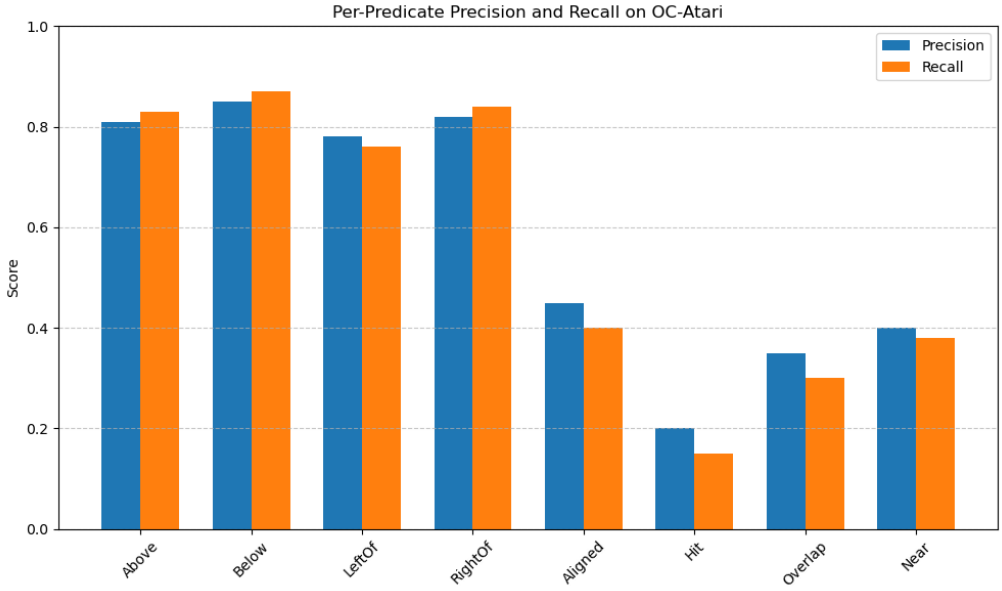

Figure 5.3: Per-predicate precision and recall on OC-Atari test set. Frequent geometric relations are learned robustly, while long-tail and visually subtle predicates remain more error-prone.

- **Edge-Case Occlusions:** Partially occluded sprites (e.g., the ball appearing behind the paddle) result in missed object detections or incorrect relationship predictions.
- **Long-Tail Confusion:** Rare predicates such as `Near` and `Overlap` are often misclassified as more common spatial terms due to training data imbalance and visual ambiguity.

These failure cases reveal current limitations in both the perception module and the annotation strategy. Addressing them may require future improvements such as data augmentation, long-tail predicate sampling, or post-processing using geometric constraints.

Figure 5.4 illustrates these challenges:

- (a) shows a spurious `LeftOf` detection despite a mostly vertical alignment,
- (b) highlights a missed `Above` relationship due to occlusion,
- (c) demonstrates confusion between `Near` and `Overlap`, where spatial proximity did not result in the correct predicate assignment.

5.5 End-to-End Planning Results

In this section, we evaluate the feasibility of our complete perception-to-symbolic-planning pipeline. The symbolic O2D state descriptions generated from Atari frames are translated into PDDL problem files and passed to the Fast Downward planner. Our goal is not to demonstrate full-scale general planning across games, but to validate that our perceptual output is semantically coherent and compatible with symbolic reasoning in at least one domain.

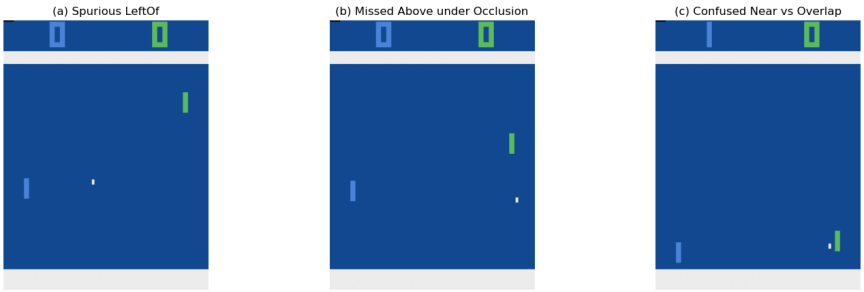

Figure 5.4: Examples of predicate grounding failures: (a) spurious `left_of`, (b) missed `Above` under occlusion, (c) confusing `Near` with `Overlap`.

5.5.1 Scope and Setup

We focus this evaluation on the *Pong* environment, where we manually defined the PDDL domain with symbolic action schemas corresponding to game interactions. The planner was tested using the O2D predicates produced by the scene graph model, without any manual intervention in the detected state.

5.5.2 Feasibility of Planning

The planner successfully generated valid plans in several *Pong* test frames, typically in under one second. These plans demonstrated expected behavior such as aligning the paddle with the ball or performing a hit action. This confirms that, at least in constrained settings, the automatically generated symbolic states can be used to drive symbolic planning.

However, not all test frames resulted in successful planning. In some cases:

- Key predicates were missing (e.g., `Aligned(o0, o1)` not detected, making the goal unreachable).
- Redundant or noisy relations made the state logically inconsistent with the domain definition.
- The planner returned a dead-end due to incomplete symbolic context.

5.5.3 Limitations and Generalization

We did not extend this experiment beyond *Pong*. In other games, the complexity of objects, interactions, and predicates made it infeasible to manually define domain files or ensure clean symbolic states from perception. Therefore, we do not claim general planning success across all 10 games.

The planning module in this thesis is a proof of concept—showing that a symbolic planner can consume automatically derived O2D states under ideal conditions. Extending this to general environments would require:

- Learning or inferring action models from O2D (instead of manually defining them),
- Improving predicate accuracy and consistency,
- Developing more expressive representations beyond primitive O2D.

5.6 Summary of Findings

Key Findings

- A standard object detector (Faster R-CNN) can be effectively fine-tuned to Atari game frames, improving both localization and recall for pixel-art objects.
- Scene Graph Generation using a modified Neural Motifs architecture captures meaningful spatial relations between detected objects, especially when pretraining and domain adaptation are combined.
- O2D symbolic predicates generated from scene graphs provide a usable abstraction layer, allowing symbolic planners like Fast Downward to operate on visually derived states.
- The pipeline demonstrated symbolic planning feasibility in the *Pong* domain, confirming end-to-end integration between visual perception and classical planning.

Limitations Revealed

- The planning evaluation is restricted to *Pong*; the system was not generalized or tested on more complex or diverse Atari games.
- Action models in PDDL were manually created; the system does not learn dynamics or transition models from data.
- Relation misclassifications and detection noise can cause cascading errors, especially in planning scenarios.
- Some predicates used in scene graphs (e.g., `aligned`, `hit`) were too ambiguous or rare to achieve reliable accuracy.
- The pipeline depends on a fixed predicate vocabulary and rigid symbolic format, limiting flexibility and expressiveness.

6 Conclusion

This thesis explored the challenge of bridging raw visual input and symbolic reasoning by generating high-level state representations from Atari game frames using scene graph generation. By adapting and fine-tuning a relational reasoning model for the OC-Atari domain, we demonstrated that it is possible to extract structured, symbolic information—such as spatial predicates—from visually complex environments.

Our approach shows that symbolic abstraction from images can support classical planning in simple scenarios. In particular, we successfully integrated our visual pipeline with a symbolic planner in a constrained domain (*Pong*), highlighting the feasibility of closing the perception-to-planning loop. While this confirms the potential of using learned scene representations for decision-making, it also exposes several limitations.

Notably, the symbolic planning system relied on hand-crafted action models, and the relational reasoning component struggled with ambiguity in less frequent or visually subtle interactions. Moreover, the evaluation was limited to a small number of planning scenarios and did not generalize across a wide range of Atari games.

Future Work Future research should aim to extend the planning integration beyond fixed domains by automatically learning action models from the generated symbolic states. Enhancing the expressiveness of the predicate vocabulary and incorporating temporal reasoning could further increase planning capability. Additionally, incorporating uncertainty-aware models or hybrid symbolic-neural approaches may improve robustness in more visually diverse and complex environments.

Ultimately, while this work demonstrates a functional prototype for perception-driven symbolic reasoning, it lays the groundwork for more scalable and generalizable systems at the intersection of vision and planning. Future work may explore learning domain dynamics directly from O2D representations, and extending planning capabilities to more complex or general Atari environment

Bibliography

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 4th ed., 2020.

[2] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," in *Proceedings of the 2nd International Joint Conference on Artificial Intelligence (IJCAI-71)*, (Vancouver, BC), pp. 608–620, William Kaufmann, 1971.

[3] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL – the planning domain definition language – version 1.2," technical report cvc tr-98-003, Yale Center for Computational Vision and Control, 1998.

[4] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," in *International Conference on Computer Vision (ICCV)*, pp. 5470–5479, 2017.

[5] D. Huang, D. Xu, Y. Zhu, A. Garg, S. Savarese, L. Fei-Fei, and J. C. Niebles, "Continuous relaxation of symbolic planner for one-shot imitation learning," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2635–2642, 2019.

[6] R. Dearden and C. Burbridge, "Manipulation planning using learned symbolic state abstractions," vol. 62, pp. 355–365, 2014.

[7] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," in *Procedia Computer Science, The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021)*, vol. 199, pp. 1066–1073, 2022.

[8] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[9] Q. Delfosse, J. Blüml, B. Gregori, S. Sztwiertnia, and K. Kersting, "Ocatari: Object-centric atari 2600 reinforcement learning environments," 2024.

[10] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, pp. 91–99, 2015.

[11] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5831–5840, IEEE, 2018.

[12] A. Occhipinti, B. Bonet, and H. Geffner, "Learning first-order symbolic planning representations that are grounded," *arXiv preprint arXiv:2204.11902*, 2022.

[13] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[14] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1470–1477, 2011.

[15] Y. Li, L. Wang, T. Wang, X. Yang, J. Luo, Q. Wang, Y. Deng, W. Wang, X. Sun, H. Li, B. Dang, Y. Zhang, Y. Yu, and J. Yan, "STAR: A first-ever dataset and a large-scale benchmark for scene graph generation in large-size satellite imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 3, pp. 1832–1849, 2025.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627–1645, IEEE, 2010.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[21] R. Girshick, "Fast r-cnn," 2015.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 21–37, Springer, 2016.

[24] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3097–3106, IEEE, 2017.

[25] L. Zhou, Y. Zhou, T. L. Lam, and Y. Xu, "Context-aware mixture-of-experts for unbiased scene graph generation," *arXiv preprint arXiv:2208.07109*, 2022.

[26] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, "Visual translation embedding network for visual relation detection," 2017.

[27] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[28] R. Li, S. Zhang, and X. He, "Sgtr: End-to-end scene graph generation with transformer," 2022.

[29] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," 2016.

[30] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," 2018.

[31] 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, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3668–3678, 2015.

[33] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[34] M. Asai and A. Fukunaga, "Classical planning in deep latent space: Bridging the subsymbolic–symbolic boundary," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 6094–6101, AAAI Press, 2018.

[35] M. Garnelo, K. Arulkumaran, and M. Shanahan, "Towards deep symbolic reinforcement learning," *arXiv preprint arXiv:1609.05518*, 2016.

[36] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, "Constructing symbolic representations for high-level planning," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1932–1940, 2014.

[37] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[38] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, vol. 8693, pp. 740–755, Springer, 2014.

[39] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[40] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[41] A. G. Cohn and S. Hazarika, "Qualitative spatial representation and reasoning: An overview," *Fundamenta Informaticae*, vol. 46, no. 1–2, pp. 1–29, 2001.

[42] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2021.

[43] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[44] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Learning symbolic operators for task and motion planning," in *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[45] T. Kipf, G. F. Elsayed, A. Mahendran, A. Stone, S. Sabour, G. Heigold, R. Jonschkowski, A. Dosovitskiy, and K. Greff, "Conditional object-centric learning from video," in *International Conference on Learning Representations (ICLR)*, 2022.

[46] Y. Cong, W. Liao, H. Ackermann, B. Rosenhahn, and M. Y. Yang, "Spatial-temporal transformer for dynamic scene graph generation," *arXiv preprint arXiv:2107.12309*, 2021.

[47] M. Ghallab, D. S. Nau, and P. Traverso, *Automated Planning: Theory & Practice*. San Francisco, CA: Morgan Kaufmann, 2004.

List of Acronyms

Above Object is above another object

Below Object is below another object

CNN Convolutional Neural Network

GCN Graph Convolutional Network

IMP Iterative Message Passing

LSTMs Long Short-Term Memory

O2D Object-Oriented Domain

Overlap Two objects overlap

PDDL Planning Domain Definition Language

R-CNN Regions with Convolutional Neural Network features

SGG Scene Graph Generation

STRIPS Stanford Research Institute Problem Solver

VidVRD Video Visual Relationship Detection

YOLO You Only Look Once

List of Figures

List of Tables

List of Listings