IMPERIAL

MEng Individual Project

Final Report

Imperial College London

Department of Computing

Object-Centric Explananda via Agent Negotiation

Author:
Benjamin Teoh

Supervisors:
Avinash Kori and
Prof. Francesca Toni

Second Marker:
Prof. Ben Glocker

June 13, 2025

Abstract

A key challenge in explainable AI, especially in vision, is aligning model reasoning with human-understandable concepts. In our project, we propose the Object-Centric Explananda via Agent Negotiation (OCEAN) framework, a classification framework that builds ante-hoc interpretability into the reasoning process, rather than applying it post-hoc. Our method uses Slot Attention to extract discrete object-centric features, and Consensus Game, a multi-agent collaborative game, to select the most class-relevant features. The game-theoretic approach encourages agents to agree on coherent, discriminative evidence, leading to faithful and interpretable predictions. We train our framework end-to-end and evaluate it against visual classifiers and established post-hoc explanations mechanisms, such as Grad-CAM and LIME, on two diagnostic multi-object datasets. Finally, we demonstrate results that highlight the potential of structured, object-centric representations in an ante-hoc explainability setting, contributing toward more trustworthy and transparent AI systems.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Avinash Kori and Francesca Toni, for their invaluable guidance, support, and insightful feedback throughout the course of this project. Their expertise in interpretable machine learning and encouragement during critical stages of development were essential to its success.

I also extend my thanks to the academic and technical staff at the Department of Computing, whose resources and support made this research possible.

Special thanks go to Su Rhou Wong and my friends for their helpful discussions, encouragement, and for participating in the user studies that evaluated the framework's interpretability. Finally, I am grateful to my family for their continuous support and understanding throughout this work.

Contents

Chapter 1

Introduction

Artificial intelligence has become cornerstone in modern innovation across a variety of domains, including healthcare [1], finance [2], and autonomous systems [3]. Yet its rapidly growing influence on the world presents a pressing challenge: understanding how complex "black box" models make decisions [4]. At the crux of explainable AI (XAI) lies the fundamental goal of clearing the opacity of machine learning models, making them more intelligible to human users. This need is particularly acute in vision-based applications, such as medical imaging and self-driving cars, where decisions have direct real-world consequences. Recent developments have produced methods like Grad-CAM [5] and LIME [6], which utilises visual elements like heatmaps to highlight regions most influential to a model's prediction, offering an intuitive way to visualise the decision-making process. Our focus on visual interpretability matters because it bridges the gap between complex model reasoning and human-understanding, fostering trust in high-stakes applications.

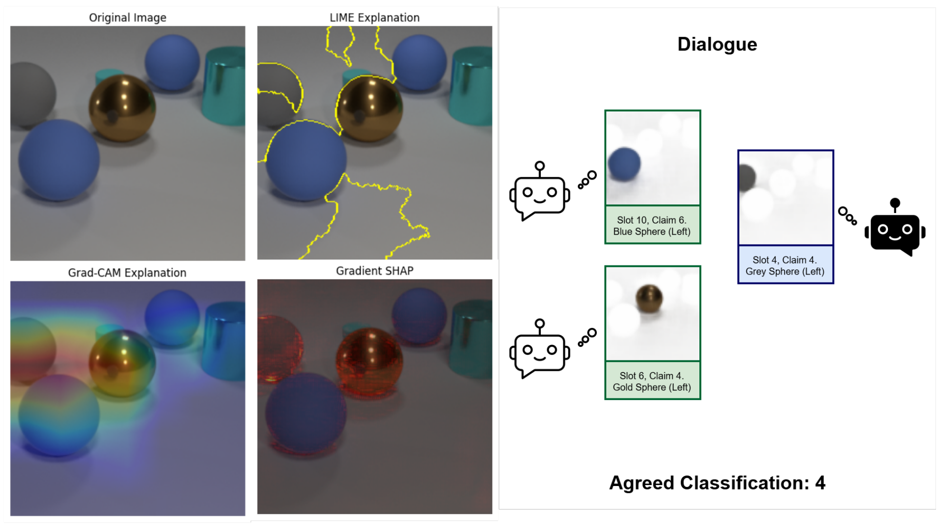

Original Image
LIME Explanation
Grad-CAM Explanation
Gradient SHAP
Dialogue
Slot 10, Claim 6. Blue Sphere (Left)
Slot 4, Claim 4. Grey Sphere (Left)
Slot 6, Claim 4. Gold Sphere (Left)
Agreed Classification: 4

Figure 1.1: A comparison of four different visual explainability mechanisms, including LIME, Grad-CAM, Gradient SHAP and an idealised dialogue from our Consensus Game framework.

In our project, we take a step towards more transparent, faithful and human-aligned reasoning in image classification. We introduce an ante-hoc, object-centric framework called OCEAN that integrates explainability directly into the classification process, by reasoning over object-centric features and exposing its internal decision-making through a multi-agent selection process. Central to this is our Consensus Game framework, a novel collaborative mechanism where agents iteratively select and share visual evidence to reach a shared classification. This design allows us to investigate whether structured interactions between agents can yield a more transparent and interpretable classification model. Our project builds on the prior work, Visual Debates [7], by grounding its agent-based argumentative design on object-centric representations and framing explainability as a transparent, structured dialogue between agents.

1.1 Problem Statement

Current approaches to XAI methods mostly rely on post-hoc [8] explanations that diagnose black-box models retrospectively rather than making its decision-making process inherently transparent. These methods often lack fidelity to the model's true inner workings, providing explanations that may be disconnected from the actual reasoning process. In contrast, ante-hoc [9] methods aim to build interpretability directly into the model itself, ensuring that explanations emerge naturally as part of predictions.

Prior work on Visual Debates [7] introduced a novel approach to enhance the interpretability of image classification models by leveraging a more structured reasoning mechanism, debate. Whilst the explanations generated from the Visual Debates framework are interpretable and faithful to the model's predictions, they are still fundamentally post-hoc, relying on surrogate models that interpret latent features. Research on The Consensus Game [10] offers an avenue for transforming the Visual Debates framework into an ante-hoc, consensus-driven process. Instead of relying on post-hoc estimations, we could reframe classification as a collaborative process, where explanations are the outcome of a dialogue held between agents reasoning over discrete evidence. This shift holds promise for enhancing the transparency and faithfulness of model reasoning, by incorporating a multi-agent reasoning mechanism with the classification task.

Additionally, the prior framework generates explanations using disentangled latent features obtained through quantisation [11], which often correspond to ambiguous regions within input images. The reliance on latent, non-intuitive features reduces alignment with human-understanding, as these features do not clearly map to recognisable objects or parts of the image. To improve this, a more object-centric approach is necessary, one that isolates and explicitly represents meaningful components within the image, enabling clearer and more interpretable reasoning. Slot Attention [12] is a promising extension to the Visual Debates framework, as it focuses on discrete object representations. This approach can enhance the interpretability of reasoning chains, particularly in complex scenes with multiple objects. While Slot Attention may not directly improve the faithfulness of the explanations, it offers a valuable means of aligning the model's reasoning with human-like understanding, where the concepts correspond to actual objects within the scene.

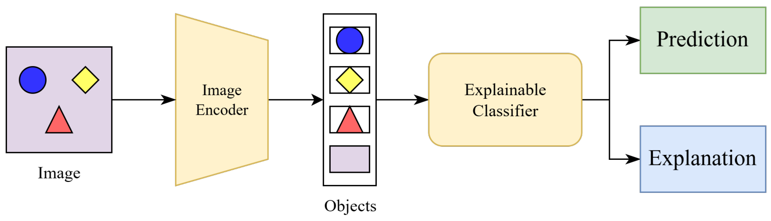


Figure 1.2: A simplified architecture diagram of an ante-hoc explainable framework for visual classification.

Addressing these challenges requires rethinking how explanations are generated, moving toward systems that integrate ante-hoc interpretability mechanisms with structured reasoning. Such systems hold the potential to bridge the gap between estimated reasoning and intrinsic model transparency, offering explanations that are more faithful and interpretable. This renewed approach could have broader applicability in other predictive tasks, offering a more transparent and human-aligned way to interpret model reasoning across a range of domains. With that, we look toward a framework that encapsulates a classification mechanism (Figure 1.2) that exposes its human-aligned reasoning process.

1.2 Main Contributions

This project takes inspiration from the prior work on Visual Debates, and introduces a novel framework for ante-hoc visual explanation through object-centric representations and collaborative reasoning. The key contributions are as follows:

- **Collaborative Multi-Agent Classification Game**: We designed agent interactions inspired by the Consensus Game, where two agents present arguments and iteratively converge to a shared prediction. This approach leverages the structured reasoning of Visual Debates while grounding it in object-centric representations in an ante-hoc manner. Through this, we investigate whether collaborative agent-based reasoning can achieve transparent classification and reasonable prediction performance against baselines. Our results suggest that this formulation not only exposes the reasoning chain very well but also serves as a strong foundation for future research in multi-agent explainability.
- **An End-to-End Learning Framework**: We develop a unified framework, OCEAN, that integrates Slot Attention with our novel Consensus Game module to perform object decomposition, explanation generation and classification jointly. Training it end-to-end optimises the learned object representations for the particular downstream classification task.
- **Evaluation Against State-of-the-Art Methods**: We assess the our method on synthetic diagnostic datasets and compare it against common classification methods such as ResNet [13]. In terms of explainability, we compare our generated explanations against the state-of-the-art mechanisms such as Grad-CAM [5] and LIME [6].

Together, these contributions demonstrate the impact of integrating object-centric learning with agent-based collaboration in building more interpretable, faithful, and human-aligned visual classifiers.

Chapter 2

Background

This chapter provides the background information on topics covered by our project by outlining key concepts and prior work. It begins with an overview of Reinforcement Learning (2.1) establishing the motivation for learning through interaction and reward in a game theoretic setting. This is followed by a review of the prior work, Visual Debates (2.2), highlighting some pitfalls our project aims to improve on. The chapter then introduces the inspiration behind key components of the project: object-centric learning (2.3) and the Consensus Game (2.4). Finally, we explore techniques in visual XAI (2.5) and summarise related work (2.6).

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a paradigm for solving problems or games through trial-and-error interactions, where an agent learns to maximise their cumulative rewards by exploring and exploiting its environment. This section provides a short summary of the required concepts for understanding the report. The use of RL motivates the use of decision-making strategies in our Consensus Game framework (3.1.2).

2.1.1 Strategies and Policies

A strategy in RL is represented as a *policy*, which maps states (or observations) to actions. Policy functions can be deterministic ($a = \pi(s)$) or stochastic ($a \sim \pi(a|s)$), depending on the nature of the task and learning approach.

2.1.2 Basic Reinforcement Learning Algorithms

Basic RL algorithms include:

- **Value-Based Methods:** Algorithms like Q-learning and SARSA focus on learning value functions $Q(s, a)$, which estimate the expected reward of taking an action a in a state s.
- **Policy-Based Methods:** These directly optimise the policy without explicitly estimating value functions. Examples include REINFORCE and Actor-Critic algorithms.
- **Model-Based Methods:** These methods learn a model of the environment and use it to plan or improve decision-making, e.g. Dynamic Programming and Monte Carlo Tree Search.

2.1.3 Function Approximation

To handle large or continuous state and action spaces, RL employs *function approximators*, such as neural networks. These are used to approximate value functions ($V(s)$ or $Q(s, a)$), policies ($\pi(a|s)$), or models of the environment (state transition and reward functions).

2.1.4 Policy Gradient Theorem

Differentiable policies are parameterised using neural networks, enabling the use of gradient-based optimisation. The policy gradient method leverages this property to compute updates directly from the policy's performance. The policy gradient theorem provides a foundation for optimizing such policies. It states:

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \right]$$

where $J(\theta)$ is the expected reward, π_θ is the policy, and $Q^{\pi_\theta}(s,a)$ is the action-value function. This gradient can be estimated using Monte Carlo or temporal difference methods.

2.2 Visual Debates

Visual Debates [7] is a novel XAI framework designed to provide insights into how deep learning classifiers make decisions via a multi-agent debate game. Kori et al. applies this framework in the context of image classification. Unlike traditional methods that rely on heatmaps or feature attributions, Visual Debates uses a structured, game-theoretic approach to uncover supporting and opposing features that are significant in the model's decision-making, inspired by other advocates of debate [14] in AI safety. In our project, its the techniques and explanation structure serve as inspiration for our project.

This is achieved by modeling explanations as a sequential zero-sum debate game between two fictional players. The players take turns presenting arguments based on discretised latent features of the classifier, with one supporting the classifier's decision and the other opposing it. These interactions aim to highlight the classifier's reasoning paths, including uncertainties, thereby offering more transparent and faithful explanations, as seen in Figure 2.1.

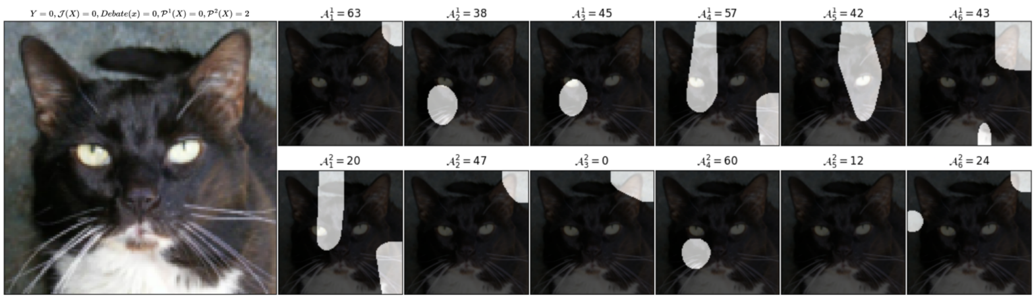


Figure 2.1: An example debate explanation from Visual Debates of an image of a cat from the AFHQ dataset, where players 1 and 2's claims are $\mathcal{C}^1$ = cat and $\mathcal{C}^2$ = wild. The first row of feature highlights show the argument chain of the affirmative player 1 for the classification of "cat", while the second row contains feature highlights of counter-arguments from player 2 in favour of "wild".

2.2.1 Pipeline Overview

Feature Extraction and Classification

The initial stage of the learning pipeline involves a pre-trained feature extractor f, and a pre-trained feature classifier g, which pair to produce image classifications. To do so, the features of an image following extraction are funnelled into the feature classifier, making the predictions. The purpose of the classifier is atypical of common predictive pipelines, as the pre-trained classifier is used to generate predictions y, which another classifier will attempt to learn. This means y will not be directly involved in the predictive outcomes of the pipeline but will be used to calculate the loss for quantised classification.

Feature Quantisation and Quantised Classification

The features extracted using f are mapped into a finite set of discrete vectors, called a "codebook" through a method called vector quantisation[11]. This effectively reduces the dimensionality of the data and enables more efficient processing. In the pipeline, vector quantisation discretises the continuous latent features, making them interpretable and more suitable for structured reasoning.

With interpretable features in the codebook, a quantised classifier, q, can learn the expected classification y, making predictions $\hat{y}$. Note that q is also the function mapping a set of quantised features to a classification. By optimising the quantised classifier to minimise the loss function of y and $\hat{y}$, it can fairly mimic the behaviour of the original pre-trained classifier despite having a modified feature set with lower dimensionality. This effectively makes the quantised classifier a surrogate model. However, the quantised classifier is much more interpretable than the initial one, making it a suitable candidate for post-hoc explanations.

Training

The framework is trained end-to-end, where the quantised classifier and the debate framework are trained simultaneously. The quantised classifier undergoes supervised training optimising for classification accuracy, while the debate framework is trained (2.2.2) through unsupervised reinforcement learning. The training process for the quantised classifier is typical of a CNN, except that the inputs are discretised features instead of continuous latent features.

2.2.2 Debate Framework

The debate framework models the explainability of the classifier as a structured, zero-sum game between two players – one supporting and one opposing the correctness of the prediction. These agents interact over a series of rounds, presenting arguments derived from the discrete quantised features from the codebook. The framework produces two final claims for both agents, representing the classification each agent is arguing for, which is an aggregation of the arguments put forward by either agent.

The authors define the framework formally with the tuple:

$$\Gamma = \langle \{\mathcal{Q}, \mathcal{Z}\}, \{\mathcal{P}^1, \mathcal{P}^2\}, \{\mathcal{A}^1, \mathcal{A}^2\}, \{\mathcal{C}^1, \mathcal{C}^2\}, \{\mathcal{U}^1, \mathcal{U}^2\} \rangle,$$

where $\mathcal{Q}$ represents a question relating to the classifier's prediction, $\mathcal{Z}$ represents the codebook, $\mathcal{P}^1, \mathcal{P}^2$ represents player one or two, $\mathcal{A}^1, \mathcal{A}^2$ represents the set of arguments, $\mathcal{C}^1, \mathcal{C}^2$ represents the claims made by each players, and $\mathcal{U}^1, \mathcal{U}^2$ are the utilities for each player.

Arguments and Claims

Each player will put forward arguments in turn. Strictly, *arguments* are put forward by $\mathcal{P}^1$, whereas *counterarguments* are put forward by $\mathcal{P}^2$. These arguments can be interpreted as the actions taken by the players during the debate game, and they are defined by the tuple:

$$\mathcal{A}^i_k = (z^i_k, c^i_k, s^i_k),$$

where A^i_k represents the argument of $\mathcal{P}^i, i \in \{1, 2\}$ at stage $k \in \{1, ..., n\}$, z^i_k represents a selected quantised feature, $c^i_k = \text{argmax } q(z; \text{do}(\{z^i_k\}))$ represents the argument claim, and s^i_k represents the argument strength. Here $q(z; \text{do}(\{z^i_k\}))$ denotes the class with the highest confidence score with z interventionally set to the feature z^i_k.

Specifically, *argument strength* is defined as:

$$s_k^1 = \begin{cases} 1 & \Delta \leq \tau, \\ -1 & \text{otherwise} \end{cases} \qquad s_k^2 = \begin{cases} 1 & \Delta > \tau, \\ -1 & \text{otherwise} \end{cases}$$

where $\Delta = |q(z)_y - q(z; do(\{z_k^1, z_k^2\}))_y|$, for $y = \mathcal{C}(x)$, and $\tau \in (0,1)$ is a threshold. Δ measures the effective contribution of an argument-counterargument pair towards the quantised classifier's decision.

The *claim* $\mathcal{C}^i$ of player $\mathcal{P}^i$ is some aggregation of both sets of latent features selected for arguments and counterarguments. In practice, the claims are derived from an accumulation of the full set of features in both arguments and counterarguments. The aggregation function is implemented using recurrent neural networks to aggregate features and a linear layer to map arguments to classes.

Utility and Rewards

There are two types of rewards during a debate game:

- **Argument rewards s_k^i:** This is received by the agent whenever they put out an argument and is exactly the argument strength.
- **Claim reward r_Γ^i:** This is obtained by comparing the prediction (Q) made with masked quantised features not present in the arguments and the claims made by both players. It is defined by:

$$r_\Gamma^i = \begin{cases} 1 & \text{if } Q = \mathcal{C}^i, Q \neq \mathcal{C}^{-i}; \\ -1 & \text{if } Q \neq \mathcal{C}^i, Q = \mathcal{C}^{-i}; \\ 0 & \text{if } Q \neq \mathcal{C}^i, Q \neq \mathcal{C}^{-i} \text{ or } Q = \mathcal{C}^i, Q = \mathcal{C}^{-i} \end{cases}$$

  where $Q = q(z, do(\mathcal{A}^1, \mathcal{A}^2))$, and $\mathcal{C}^{-i}$ is the claim made by the agent other than $\mathcal{C}^i$.

These are collated to define *utility* functions for each player agent:

$$\mathcal{U}^i = r_\Gamma^i + \sum_{k=1}^{k=n} s_k^i.$$

where $\mathcal{U}^i$ is the utility of player $\mathcal{P}^i$. The utilities satisfy the zero-sum nature of the game, where $\mathcal{U}^1 = -\mathcal{U}^2$, suggesting that the players should focus on different features.

Player Architecture

For players to learn strategies to maximise their utilities, the authors utilise reinforcement learning and training the players through self-play. The players are represented using sets of parameters θ^1 and θ^2, updated iteratively through policy gradient techniques.

To ensure the debate's fully observable nature, the *argument profile* of each player is defined as:

$$\alpha_k^1 : \bigcup_{t<k} \mathcal{A}_t^1 \cup \bigcup_{t<k} \mathcal{A}_t^2 \rightarrow \mathbb{P}_{\theta^1}(\mathcal{Z}), \quad \alpha_k^2 : \bigcup_{t \leq k} \mathcal{A}_t^1 \cup \bigcup_{t<k} \mathcal{A}_t^2 \rightarrow \mathbb{P}_{\theta^2}(\mathcal{Z}),$$

where $\mathbb{P}_{\theta^1}, \mathbb{P}_{\theta^2}$ are the probability distributions used to decide the quantised feature to use in the next argument.

With that, the *joint objective* of the debate game can be defined in terms of the players' utilities and log-likelihood of the policy distributions:

$$V(\mathcal{P}^1, \mathcal{P}^2) = \min_{\theta^1} \max_{\theta^2} \mathbb{E}\Big[\sum_{1 \le k \le n} \Big(\log \mathbb{P}_{\theta^1}(\mathcal{Z}_k) - \log \mathbb{P}_{\theta^2}(\mathcal{Z}_k) \Big) \, \mathcal{U}^2(\mathcal{S}^1_{\theta^1}, \mathcal{S}^2_{\theta^2}) \Big]$$

where $\mathcal{S}^i_{\theta^i} = \{z^k_1, ..., z^k_n\}$ represents the strategy of player $\mathcal{P}^i$, z^i_k is sampled from $\alpha^i_k(.)$, and $\mathbb{P}_{\theta^i}(\mathcal{Z}_{||})$ denotes the policy distribution at step k.

The definition above does not guarantee high-quality debates. Therefore, the authors introduce two properties, which can be enforced in practice to improve the quality of explanations:

- **Argument Entropy**: For some sampled feature z, $\mathcal{AH}(z) = -\mathbb{E} \log p$, where p is the probability of choosing z for an argument. The desired outcome is to minimise the number of features considered in the debate, making it more focused and manageable. Hence, the agents will attempt to minimise AH.
- **Argument Diversity**: $\mathcal{AD}(\mathcal{A}^1, \mathcal{A}^2) = \mathbb{E}((\tilde{\mathcal{A}}^1 - \mathcal{A}^2)^2) + \mathbb{E}((\mathcal{A}^1 - \tilde{\mathcal{A}}^2)^2) - \lambda \sum_{i \in \{1,2\}} \mathbb{E}((\mathcal{A}^i - \tilde{\mathcal{A}}^i)^2)$, where $\tilde{\mathcal{A}}^i = \frac{1}{n} \sum_{k=1}^{k=n} \mathcal{A}^i_k$ and λ represents a hyperparameter. The desired outcome here is to preserve the diversity between inter-player arguments but encourage coherence between intra-player arguments. Hence, the agents will attempt to maximise AD.

Since this is set up to be a finite-player, finite-strategy, fully observable, zero-sum sequential game, there exists at least one mixed strategy Nash equilibrium (NE) [15], represented as $(\mathcal{S}^i_{\theta^1}, \mathcal{S}^i_{\theta^2})$. At equilibrium, any argument or counterargument made by any player at any step k serves as the optimal response to the previous opposing argument.

Implementation

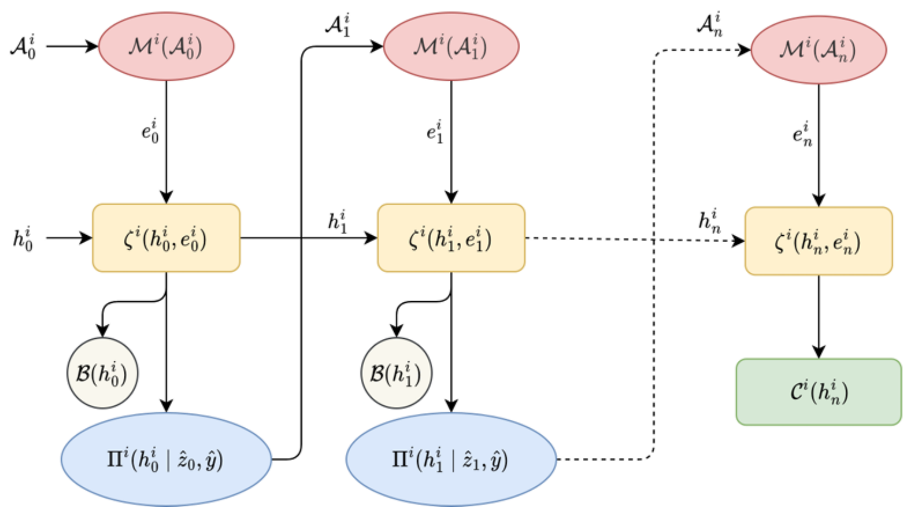


Figure 2.2: The Visual Debates architecture of Player $\mathcal{P}^i$. $\mathcal{M}^i$ is a modulator network and e^i_k is a modulated argument. ζ^i is a GRU backbone network, h^i_k is a hidden state vector, while $\mathcal{B}^i$, Π^i and $\mathcal{C}^i$ are the baseline, policy and claim networks, respectively. $\hat{z}_k$ is a masked state vector.

The player architecture (2.2) combines multiple neural networks to model argument generation and reasoning in a structured manner. Having made argument $\mathcal{A}^i_{k-1}$ at step $k-1$, the following steps are taken to produce argument $\mathcal{A}^i_k$. First, a modulator network processes $\mathcal{A}^i_{k-1}$ and maps them to a hidden state dimension for the next stage, producing a modulated argument e^i_{k-1}. Each player is also implemented using an unrolled gated recurrent unit (GRU) as the backbone network ζ^i, which processes the modulated argument, maintains a hidden state h^i_{k-1}, and updates its memory with the arguments. Within the player, there is a baseline network $\mathcal{B}^i$ for estimating the value of a particular argument using a recurrent model [16]. Then, the hidden state h^i_{k-1} from the GRU is shared with the policy network Π^i, which considers the hidden state conditioned on $\hat{z}_k - 1$ and the quantised classifier's prediction $\hat{y}$ to produce argument $\mathcal{A}^i_k$, where $\hat{z}_{k-1}$ is generated by masking the environment with respect to $\mathcal{A}^1_{k-1}$ and $\mathcal{A}^2_{k-1}$. Finally, there is an additional final step, where the hidden state of ζ^i is passed to a claim network $\mathcal{C}^i$ to generate the final claim. This allows arguments to be shared between players and become common knowledge for reasoning. This implementation informs most of the design for our agents in the Consensus Game (3.1.2).

2.3 Object-Centric Learning

Object-centric learning (OCL) is a paradigm in machine learning that focuses on decomposing a scene into discrete objects, each represented as an independent entity. This method aligns with human perception, as understanding individual objects and their relationships is essential for interpreting complex scenes. By isolating objects, OCL enhances the interpretability and adaptability of object discovery and set prediction tasks. For example, in autonomous vehicles, separating pedestrians, vehicles, and traffic lights provides actionable insights, while in explainable AI, it highlights specific objects of a scene that influence the model's decision. OCL is introduced to the project justify the need for structured, object-centric scene decomposition and understanding.

2.3.1 Key Concepts and Components

An object-centric learning pipeline iteratively deconstructs images into object-specific representations and reconstructs the scene from these components, optimizing to minimise reconstruction loss. See Figure 2.3.

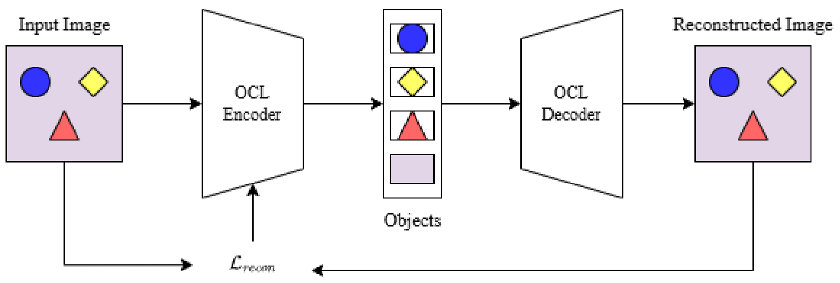


Figure 2.3: An Object-Centric Learning Pipeline.

- **Input**: Raw visual data, such as an image or video, containing multiple objects in a scene.
- **Outputs**: Reconstructed image, a mask to indicate which pixels in the input data correspond to each object, and latent representations of each object in the input data.
- **Object Decomposition**: Breaking down input data into object-level segments using attention mechanisms or clustering techniques. These object representations are then used for downstream tasks like classification.
- **Unsupervised Learning**: Many object-centric learning methods do not rely on labelled data but instead use inductive biases and self-supervised techniques to infer object representations from raw visual data.

There are many object-centric learning methods such as MONet[17], IODINE[18], SPACE[19] and Slot Attention[12], but the focus for our project is Slot Attention.

Algorithm 1 Slot Attention Algorithm[12]

```
1: Input: inputs $\in \mathbb{R}^{N\times D_{\texttt{inputs}}}$, slots $\sim \mathcal{N}(\mu,\ \text{diag}(\sigma)) \in \mathbb{R}^{K\times D_{\texttt{slots}}}$
2: Layer params: $k,\ q,\ v$: linear projections for attention; GRU; MLP; LayerNorm (x3)
3:   inputs = LayerNorm (inputs)
4:   for $t = 0 \dots T$
5:     slots_prev = slots
6:     slots = LayerNorm (slots)
7:     attn = Softmax ($\frac{1}{\sqrt{D}} k$(inputs) $\cdot\ q$(slots)$^T$, axis='slots')
8:     updates = WeightedMean (weights=attn + $\epsilon$, values=$v$(inputs))
9:     slots = GRU (state=slots_prev, inputs=updates)
10:    slots += MLP (LayerNorm (slots))
11:  return slots
```

2.3.2 Slot Attention

Slot attention is a powerful neural mechanism that bridges attention-based methods with object-centric learning. At its core, slot attention iteratively refines a set of fixed-sized latent vectors, called *slots*, which serve as object representations to be used in a downstream task. During the refinement process, the slots are iteratively updated to specialise in representing distinct objects, making the algorithm highly effective for object discovery tasks.

To obtain object-centric representations from unstructured visual data like images, slot attention relies on a perceptual backbone, such as a Convolutional Neural Network (CNN), to extract meaningful features for potential objects. The CNN, being used as an encoder, processes the input data into a grid-like representation of features through its convolutional and pooling layers, which apply filters over the training data and create feature maps that capture patterns across the image. Additionally, since Slot Attention is invariant to position, the backbone is augmented with positional embeddings to incorporate spatial information, making it more suitable for complex binary classification tasks. That is, each point on the feature grid is associated with a 4-dimensional vector, encoding the normalised distances to the borders of the feature map along four directions (top, bottom, left, right). This embedding makes object decomposition more accurate by providing more positional context.

Slot Attention is designed to transform the input feature vectors into some number of slots, each representing distinct objects. As depicted in Algorithm 1, the process begins with the feature grid and positional embeddings prepared by the CNN backbone and a set of randomly initialised slots (Line 1). These slots serve as learnable templates that will continuously be updated towards the ideal distinct object representations. To do so, the algorithm computes attention weights that determine how strongly each slot attends to specific parts of the feature grid (Line 7). These weights are normalised across the slots to enforce a competitive process, ensuring that each feature is primarily associated with one slot. Using those weights, each slot is updated using a weighted aggregation of the features it attends to (Lines 9, 10), incorporating the most relevant information. The attention computation and slots update is repeated over T routing iterations (Line 4), improving the object-centric decomposition with each step. Practically, the model learns through a reconstruction loss, which is the mean squared error between the input image and the reconstructed image.

The slots (2.4) generated by Slot Attention are highly versatile and can serve as input for a wide range of downstream tasks, including classification and generative modeling. In tandem with the downstream tasks, training is typically done end-to-end in a unified pipeline. This allows the learning process to optimise both the slot encoding and the task-specific model simultaneously, ensuring that the slots are not only good representations of objects but are also task-relevant.

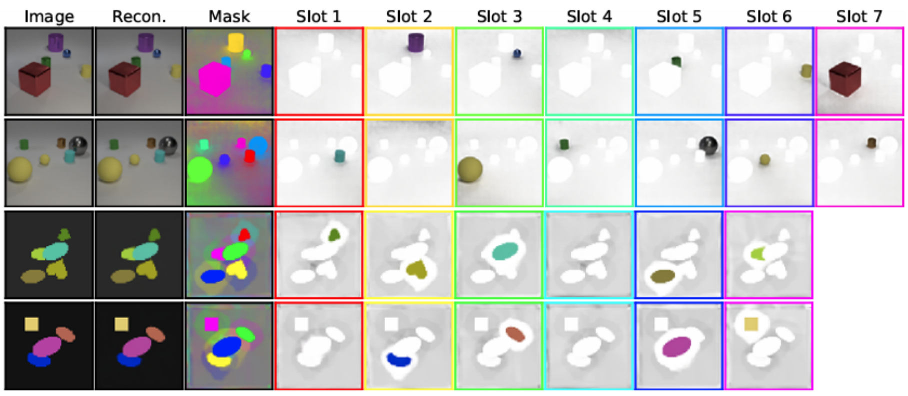


Figure 2.4: Example Slot Attention visualisation of per-slot reconstructions, image reconstruction and alpha masks for the CLEVR6 and Multi-dSprites datasets.

2.4 The Consensus Game

Although the primary focus of this project is not on language models, the methods employed in the Consensus Game framework, as described by Jacob et al. [10], provide a compelling foundation for agent interactions, which heavily influence agent and game design of our project. This section outlines the core components of the Consensus Game and its relevance to the project.

2.4.1 Language Model Consensus

There are two foundational methods for querying a language model: generatively and discriminatively. A generative approach typically involves s producing candidate answers to a question, while a discriminative approach usually aims to verify question-answer pairs for factual correctness. The two procedures may not consistently produce the same results on the same model due to some misalignment. The Consensus Game is a game-theoretic framework designed to address these inconsistencies by reconciling generative and discriminative methods.

The Consensus Game models the interaction between a *Generator* agent and a *Discriminator* agent as a sequential game. The Generator produces candidate outputs while the Discriminator evaluates their correctness. These interactions are formalised as a signalling game, where the goal is to reach a coherent agreement between the two agents. The key advantage of this approach is its ability to improve coherence and truthfulness in model outputs through iterative refinement and equilibrium search.

Consensus achieved in this model has the following key properties:

- **Coherence:** This ensures that the two procedures agree on which candidate answers are correct. In practice, this means that the outputs selected by the Generator align with the evaluations of the Discriminator, leading to consistent and stable predictions.
- **Reasonableness:** This ensures that predictions are not arbitrary but remain as close as possible to the original behaviours of the Generator and Discriminator. This property helps preserve the integrity of each agent's initial strategies while refining their outputs to reach a mutually agreed consensus.

These properties are formalised in the game-theoretic framework by leveraging regularised equilibrium concepts to define both coherence and reasonableness as objectives formally. By solving the game, the model would align its generative and discriminative query behaviour, achieving consensus. This is done using the novel Equilibrium-Ranking algorithm.

2.4.2 Equilibrium-Ranking

The equilibrium-ranking algorithm involves the Generator and Discriminator refining their strategies iteratively to maximise their payoffs. The reward structure encourages the Discriminator to correctly identify the correctness of a candidate answer generated by the Generator. This iterative process converges to a regularised Nash equilibrium, ensuring that the strategies of both agents are stable and reasonable.

The algorithm begins with an initialisation phase, where the Generator and Discriminator adopt initial policies based on pre-trained models. These policies are refined through no-regret learning, where each agent minimises their regret (the distance between the chosen action and the best action in retrospect) over multiple rounds of interaction. KL regularisation is applied to constrain the strategies, preventing over-optimisation and ensuring that agents do not deviate excessively from their initial behaviours.

At equilibrium, the Generator will propose outputs correctly validated by the Discriminator, ensuring mutual agreement. This alignment minimises inconsistencies in their individual behaviours, leading to stable predictions that reflect both agents' shared understanding of correctness. Practically, the framework can be applied to LLMs to ensure discriminative and generative queries align well when a user performs a query. For visual models, this concept can be translated to agents reasoning about object slots and reaching a consensus on their interpretations, enhancing the faithfulness and stability of the resulting explanations.

2.4.3 The Game Model

Output of a Model

The output of an AI model, such as a language model, can be described as a mapping of inputs x to outputs $y \in \mathcal{Y}$ according to the distribution $P_{LM}(y \mid x, v)$, where $v \in \{\text{correct, incorrect}\}$, indicating whether a correct or incorrect answer is desired. Let $\mathcal{V}$ be this set $\{\text{correct, incorrect}\}$.

Players

- **Generator(G):** Proposes candidate outputs to a generative query. The Generator proposes answers modelled after the distribution $P_{LM}(y \mid x, \text{correct})$.
- **Discriminator(D):** Evaluates the proposals based on truthfulness. The Discriminator assesses Generator outputs based on the distribution $P_{LM}(v \mid x, y)$ where $v \in \mathcal{V}$.

The strategies of each player are defined using policies π_G and π_D for the Generator and Discriminator respectively. $\pi_P^{(t)}$ is defined as the policy at time step t of some player P. The policies are normalised across v and y and are initialised using the base distribution P_{LM} representing the language model. This is important because it allows the extraction of a well-calibrated Generator and Discriminator from the start.

Action Space

The action space for the players can be described as follows:

- **Generator**: The generator selects a natural language string from a finite set of candidates $\mathcal{Y}$. These candidates could be generated using methods such as top-k sampling from the language model's distribution $P_{LM}(y|x, \text{correct})$. With that, $\pi_{\mathbf{G}}(y \mid x, v) \in [0, 1]$ represents the probability of generating y given query x and correctness parameter v.
- **Discriminator**: The discriminator's actions involve selecting a correctness label v based on the query-answer string provided by the generator. $\pi_{\mathbf{D}}(v \mid x, y) \in [0, 1]$ represents the probability that v is the right choice given the query-answer pair (x, y).

Rewards and Utilities

A naive approach to rewarding the players would be to give a payoff of **1** to both players if the Discriminator correctly evaluates the correctness of the Generator's proposal to a query. This is sufficient to achieve coherent outputs but the quality and truthfulness of the outputs are not guaranteed. Therefore, players are penalised when deviating from the intended behaviour of the model, which is achieved by introducing KL regularisation. Based on the definition of the rewards, each agent aims to maximise its utility based on its role and objectives while reducing deviation from their initial strategies.:

- **Generator:** Encourages outputs that align with the Discriminator's evaluations.
- **Discriminator:** Encourages accurate evaluations of the Generator's proposals.

The utilities of both agents are derived from the requirement to satisfy the desirable properties of coherence and reasonableness, as described in section 2.4.1. They are defined as follows:

$$u_{\mathbf{G}}(\pi_{\mathbf{G}}, \pi_{\mathbf{D}}) := -\lambda_{\mathbf{G}} \cdot \mathrm{D}_{\mathrm{KL}}[\pi_{\mathbf{G}}(\cdot \mid v), \pi_{\mathbf{G}}^{(1)}(\cdot \mid x, v)] + \frac{1}{2} \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} \pi_{\mathbf{G}}(y \mid x, v) \cdot \pi_{\mathbf{D}}(v \mid x, y),$$

$$u_{\mathbf{D}}(\pi_{\mathbf{G}}, \pi_{\mathbf{D}}) := -\lambda_{\mathbf{D}} \cdot \mathrm{D}_{\mathrm{KL}}[\pi_{\mathbf{D}}(\cdot \mid y), \pi_{\mathbf{D}}^{(1)}(\cdot \mid x, y)] + \frac{1}{2} \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} \pi_{\mathbf{G}}(y \mid x, v) \cdot \pi_{\mathbf{D}}(v \mid x, y),$$

where $\lambda_{\mathbf{G}}, \lambda_{\mathbf{D}}$ are the amounts of KL regularisation applied to each player, and D_{KL} is the KL divergence function.

To achieve reasonableness, the first term represents a regularisation term that the players aim to minimise to reduce the "distance" from the current policy to the initial policy. The second term promotes coherence by measuring the alignment between both players' policies, which the players aim to maximise.

Game Dynamics

The game progresses in rounds, with the Generator proposing outputs and the Discriminator providing assessments. Each round is completed, and both agents update their strategies based on the interactions of the round. The aim of each round in each round is to progress closer to the equilibrium point while minimising regret, derived from the piKL algorithm [20], and preventing large deviations from the initial policies $\pi_{\mathbf{G}}^{(1)}$ and $\pi_{\mathbf{D}}^{(1)}$. The game dynamics are modeled in the Equilibrium-Ranking algorithm described in section 2.4.2, which solves both the optimal strategies of the Generator and Discriminator players.

Outcomes at Equilibrium

Achieving the Nash equilibrium of the game provides an optimal pair of policies for both players. Based on the utilities of both players, the outcome would have the properties of consensus: coherence and reasonableness. With the optimal policies, one can make generative and discriminative queries to the respective agents and produce results that align more with their counterparts than initially possible.

2.5 Visual Explainability

Explainable Artificial Intelligence (XAI) seeks to make machine learning models more transparent by providing insights into their decision-making processes. In visual computing, XAI techniques bridge the gap between human interpretability and the complexity of deep learning models for tasks like image classification, segmentation, and object detection. This section explores the inherent tradeoffs in XAI, and the methods employed in the field, building up an understanding of the requirements and desirable traits in our explainable framework.

2.5.1 Post-hoc Methods

Without any need to modify the underlying architecture of pre-trained models, post-hoc methods are able to generate explanations by diagnosing the model's decision. These methods are widely used due to their flexibility, as they allow explainability to be applied to highly performant models without any alterations. Examples include Grad-CAM [5], which highlights regions in an image that contribute most to a model's prediction, and LIME [6], which approximates feature importance by perturbing input data. They often use techniques, such as feature importance attribution, saliency maps and heatmaps, to analyse the model's decision-making process. This makes post-hoc methods particularly flexible, as they can be applied to a wide variety of models while preserving performance. However, post-hoc methods often lack solutions for producing faithful explanations that accurately represent the model's internal reasoning, which may lead to misleading interpretations.

2.5.2 Ante-hoc Methods

Ante-hoc methods aim to embed explainability directly into a model's decision-making process, making the model inherently interpretable. These methods seek to avoid the pitfalls of post-hoc explanations by producing faithful explanations during the model's inference time and aligning with the model's reasoning. Thus, the explanations generated from the model stem from fully transparent predictions. This makes these models very suitable for high-stakes applications where trust, accountability, and transparency are essential. Designing such models requires a rethink of the structure of making predictions, often achieved by introducing architectural constraints, interpretable feature transforms, or modularising parts of the model. Examples include ProtoPNet [21], and Self-Explaining Neural Networks [22], which is a framework to design ante-hoc explainable neural networks. In spite of the benefits, the performance tradeoff is significant, and building such architectures requires more design effort and domain knowledge.

2.5.3 Explainability-Performance Tradeoff

The integration of XAI in machine learning models often involves a tradeoff between interpretability and predictive performance. Highly performant models, such as deep neural networks, are highly complex "black boxes", making their decision-making processes challenging to interpret for a user. Conversely, simpler models like decision trees or linear models provide greater transparency but may lack the performance necessary for complex visual tasks like image classification.

Despite its difficulty, achieving a balance between explainability and performance is crucial. Post-hoc methods attempt to explain complex models without altering their architecture, preserving performance but potentially compromising explanation fidelity. Conversely, ante-hoc methods aim to design models that are inherently interpretable, sometimes at the cost of predictive accuracy or scalability. This tradeoff underscores the importance of selecting appropriate XAI techniques based on applications' needs, such as interpretability in high-stakes domains or performance in resource-constrained settings.

2.6 Related Work

This section reviews key explainability methods relevant to our work, including both post-hoc techniques and inherently interpretable models. Grad-CAM, LIME, and Gradient SHAP represent widely adopted methods that provide saliency or attribution maps to highlight influential input regions. These methods inform our baselines and provide a direction for evaluating our framework's explanation quality.

2.6.1 Grad-CAM

Gradient-weighted Class Activation Mapping[5] is a widely used post-hoc explainability technique in deep learning image classification that visualises the parts of an input image that contribute most to a model's decision. This is achieved by computing the gradients of the predicted class with respect to the final convolutional layer's feature maps to obtain importance scores for each feature map. The gradients indicate the influence of each feature map on the prediction, enabling the model to produce a class-specific heatmap overlay for visualisation.

Grad-CAM is particularly useful in computer vision tasks such as image classification and object detection, as it provides interpretable insights into what the model "sees" when making predictions. This technique also aids in debugging model behaviour by helping developers identify potential biases or errors. This enables higher trust, accountability, and accountability on "black box" models, making it valuable for industries like healthcare and autonomous vehicles.

2.6.2 LIME

Local Interpretable Model-agnostic Explanations (LIME) [6] is a popular post-hoc explainability method that explains specified predictions by approximating local decision boundaries of any black-box model with an interpretable surrogate model, such as a sparse linear classifier. This is achieved by perturbing the input vector and observing how the model's probability values change in response. By fitting a simple model to the perturbations weighted by their similarity to the original instance, LIME identifies which input features are most influential for a specific prediction. It is particularly effective for interpreting vision models by identifying image regions that most influence a prediction. This makes LIME useful for debugging individual image classifications, though results may vary due to sampling noise and approximation instability.

2.6.3 Gradient SHAP

Gradient SHAP [23] combines ideas from Integrated Gradients [24] and SHAP (SHapley Additive exPlanations) [25] to attribute importance to input features. It estimates SHAP values by computing the expected gradients of the model's output with respect to inputs interpolated between a baseline vector and the true input, generating a more stable prediction of each feature's contribution to the model's prediction.

Unlike Grad-CAM, it focuses on the input level and can be used with any differentiable model, making it a more flexible and scalable alternative. For vision models, it produces pixel-level attribution maps indicating which regions most positively or negatively influenced the output. The SHAP framework provides some guarantees on the consistency and local accuracy of the attributions, which are crucial for trustworthy explanations.

Chapter 3

Object-Centric Explananda via Agent Negotiation

This chapter presents the Object-Centric Explananda via Agent Negotiation (OCEAN) framework, an ante-hoc interpretable vision model that combines object-centric learning with structured agent interactions to generate inherently faithful and interpretable reasoning chains. Here, we discuss the design (3.1) and implementation of the framework (3.2).

3.1 Design

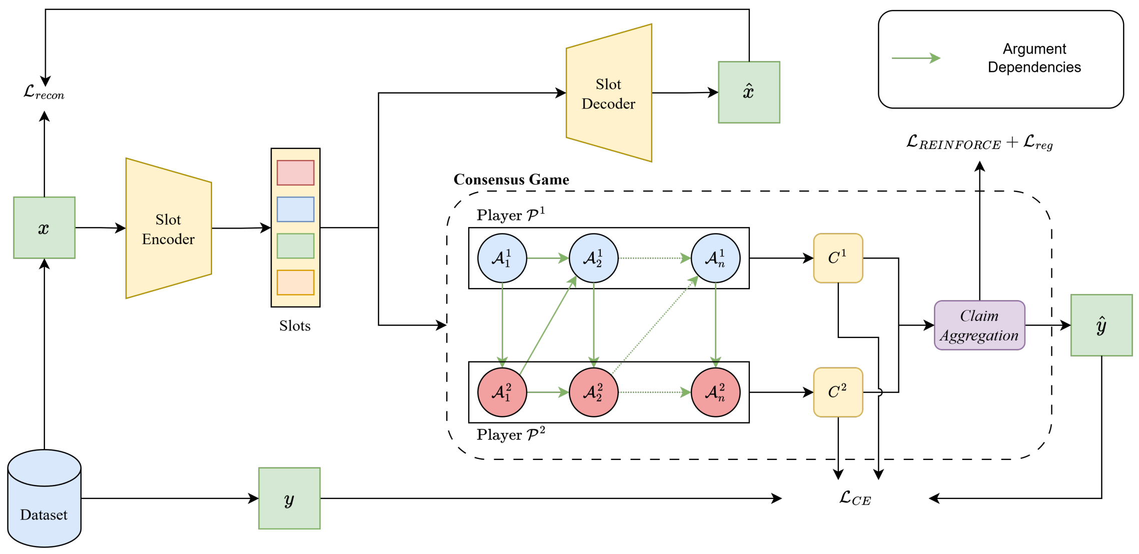


Figure 3.1: The End-to-End Learning Framework for OCEAN, showing the slot encoder-decoder architecture funneling into the downstream Consensus Game task. There, the player interactions inform the classification $\hat{y}$ and the generated explanation $\{\mathcal{A}^1, \mathcal{A}^2\}$. During training, we aim to minimise the reinforce, cross-entropy, reconstruction losses and an optional regularisation loss, which are $\mathcal{L}_{REINFORCE}$, $\mathcal{L}_{CE}$, $\mathcal{L}_{recon}$ and $\mathcal{L}_{reg}$ respectively.

3.1.1 Object-Centric Representations

A key challenge in explainable vision models lies in aligning internal representations with human-understandable concepts. Traditional feature extractions often lack semantic clarity, making it difficult to generate explanations grounded in recognisable parts of an image or object. Object-centric learning (OCL) addresses this by decomposing images into discrete, meaningful elements.

Slot Attention [12] was chosen primarily due to its lightweight design, effectiveness in unsupervised object discovery, and architectural compatibility with the downstream task. In the framework, the resulting object-centric representations, slots, serve both as reasoning units for agent arguments in the Consensus Game (section 3.1.2) and as the foundational components for classification.

3.1.2 Consensus Game

The Consensus Game module is a key component of our OCEAN framework, tailored to generate ante-hoc predictions for image classification. The key novelty lies in embedding transparent reasoning mechanisms into model's architecture, enabling inherently faithful explananda (explained predictions) to emerge as part of a structured dialogue between players during classification. By simulating cooperative reasoning through a game, we explore whether structured, multi-agent interactions can produce a more transparent and interpretable classifier. While the Consensus Game is compatible with arbitrary feature representations and any number of players, we present it in the context of object-centric slots extracted via Slot Attention used in a two-player game.

The Consensus Game framework consists of two players, $\mathcal{P}^1$ and $\mathcal{P}^2$, who take turns presenting a sequence of arguments, $\mathcal{A}^1 = \{\mathcal{A}_1^1, \mathcal{A}_2^1, ..., \mathcal{A}_n^1\}$ and $\mathcal{A}^2 = \{\mathcal{A}_1^2, \mathcal{A}_2^2, ..., \mathcal{A}_n^2\}$, before ultimately making their respective final claims, $\mathcal{C}^1$ and $\mathcal{C}^2$, regarding the classification of a shared set of slots $\mathcal{S} = \{s_0, s_1, ..., s_{k-1}\}$ that represent an image. Players collaboratively select the most salient slots as arguments to support their claims, which are then aggregated to produce the final classification. Finally, both players are equipped with a shared utility $\mathcal{U}$, representing the effectiveness of their choices of arguments. The framework Γ can be formally defined as the following:

$$\Gamma = \langle \mathcal{S}, \{\mathcal{P}^1, \mathcal{P}^2\}, \{\mathcal{A}^1, \mathcal{A}^2\}, \{\mathcal{C}^1, \mathcal{C}^2\}, \mathcal{U} \rangle.$$

Players

Each player $\mathcal{P}^1$ and $\mathcal{P}^2$ in the Consensus Game is instantiated as a reasoning agent over a shared slot pool $\mathcal{S}$. The players are symmetrical in capabilities, differing only in their order of interaction. During their turn, players present an argument $\mathcal{A}_k^i$. When it is not that player's turn, they take in information about the newly selected slot from the other player. These selections form the argument sequence $\mathcal{A}^1$ and $\mathcal{A}^2$, which culminate in final claims, $\mathcal{C}^1$ and $\mathcal{C}^2$, respectively. The claims are class predictions over a fixed set of labels $\mathcal{Y}$.

Player $\mathcal{P}^i$'s policy $\pi^i : \mathcal{H}_t \rightarrow a_t$ maps the current game history $\mathcal{H}_t$, comprising of all arguments put forward, at time step t to the next action a_t, which is an argument $\mathcal{A}_k^i$ at turn k of the game. Here, we emphasise the distinction between time step t and turn k. Multiple players can share the same turn number k, but the time step t always increases uniquely with each action.

Arguments

The sequence of arguments put forward in the game supports the final claims made by both players at the end. An argument $\mathcal{A}_k^i$ for player P^i at stage $k \in \{1, .., n\}$ composes of two elements: a slot selection s_k^i and a claim c_k^i, such as in the following:

$$\mathcal{A}_k^i = (s_k^i, c_k^i).$$

Claims

The claim c_k^i is a class prediction made by the player based on all the slots that have been selected by both players at turn k. The final claim $\mathcal{C}^i$, specifically, is the claim made by the player $\mathcal{P}^i$ at the end of the game. When both players have put forward the same claim in their most recent turn, those claims are deemed as the final claims and are used for classification. Only in the case where consensus could not be reached, no classification is put forward (abstention).

Game Dynamics

Players take turns making arguments based on the available slots. The game is governed by the following rules to ensure structured interaction:

- Each player may select only one slot during their turn. This selection can be any slot from the slot pool, and they may repeat their selections.
- The game terminates when all slots have been exhausted. Alternatively, the game ends early if both players arrive at the same claim (i.e., reach consensus) on their latest turn.
- A win condition is met only if the players' consensus matches the ground-truth label. Otherwise, they lose.

Rewards and Utilities

We employ a shared sparse reward function r_Γ, which only instantiates from the players' final claims. This reward structure encourages both agreement and correctness, rewarding agents only when consensus aligns with the ground truth, and penalizing both blind agreement and disagreement. It is as follows:

$$r_\Gamma = \begin{cases} 1 & \text{if } Y = \mathcal{C}^1 \text{ and } Y = \mathcal{C}^2, \\ 0 & \text{if } \mathcal{C}^1 = \mathcal{C}^2 \text{ but } Y \neq \mathcal{C}^1, \\ -1 & \text{otherwise} \end{cases}$$

The shared utility function U_Γ is then defined as the expected reward over the joint policy of both agents:

$$\mathcal{U}_\Gamma(\pi^1, \pi^2) = \mathbb{E}_{\mathcal{C}^1, \mathcal{C}^2 \sim (\pi^1, \pi^2)} \left[r_\Gamma(\mathcal{C}^1, \mathcal{C}^2, Y) \right],$$

where π^1 and π^2 denote the policies of players $\mathcal{P}^1$ and $\mathcal{P}^2$, respectively. During training, the agents are optimised jointly to maximise $\mathcal{U}_\Gamma$, promoting cooperative behaviour that balances consensus with predictive accuracy.

Equilibrium

In the Consensus Game, each player seeks to maximise their expected utility by learning a policy that governs their slot selection and final claim. Since both agents share a reward function r_Γ and are penalised for disagreement or incorrect consensus, their optimal strategy is cooperative.

Let π^i denote the policy of player $\mathcal{P}^i$. Each policy maps a game state history to a distribution over available actions. The goal of each player is to learn a policy π^{i*} that maximises their expected reward under the joint policy $\pi = (\pi^1, \pi^2)$:

$$\pi^{i*} = \arg\max_{\pi^i} \mathbb{E}_\pi[r_\Gamma]$$

The equilibrium of the game is reached when both players adopt policies that are mutually optimal, where neither can single-handedly improve their expected reward given the other's strategy. In this cooperative setting with a shared reward signal, the equilibrium aligns with the joint policy that consistently leads to correct consensus on the classification task.

3.1.3 Explananda

In our framework, predictions are derived from a structured reasoning process grounded in object-centric slots. Emerging with the predictions are a transparent reasoning trace. When paired with the resulting prediction, it constitutes an explanandum: an inherently explained classification.

The final prediction corresponds to the agreed-upon label at the conclusion of the game, assuming the agents reach consensus. If consensus is not reached, both players forfeit, and no prediction is emitted. In this case, the reasoning chain would provide insights into that occurrence. Explananda are generated ante-hoc via the structured reasoning chain of slot selections and claims. This design enables interpretable predictions grounded in distinct, semantically meaningful objects of the input image.

3.2 Implementation

The implementation consists of two primary modules: a Slot Attention-based feature extractor and a Consensus Game-based classifier. The entire framework was implemented using PyTorch in Python and the data flow remains best represented by the diagram above (3.1).

3.2.1 Slot Attention

We employ Slot Attention [12] to extract object-centric representations from input images. It forms the backbone of our framework by decomposing scenes into a fixed-size set of slots that serve as inputs to the Consensus Game. Slot Attention is trained as part of an encoder-decoder architecture that encourages object-centricity through a reconstruction loss.

The encoder is a shallow CNN that maps an input image of shape (B, C, H, W) to a dense feature map. Slot Attention then compresses these features into K slot vectors of dimension D, yielding an output of shape (B, K, D). The decoder reconstructs the image from these slots, producing both the full reconstruction and individual slot reconstructions with attention masks. These outputs would prove useful in the downstream reasoning chains.

3.2.2 Consensus Game

Following slot encoding, the Consensus Game module enables any number of agents to iteratively select slot information with the goal of jointly identifying the correct class label. This interaction is framed as a self-play reinforcement learning game, where each agent must communicate selectively to reach a consensus. The module can be seen in two complementary ways: as an ante-hoc visual explanation mechanism that reveals how decisions emerge through inter-agent reasoning, or as a novel explainable classification approach leveraging multi-agent reasoning.

Player Architecture

Each player in the Consensus Game operates as a neural agent composed of several cooperating components. The architecture is designed to process partial observations (slots) sequentially, maintain memory of prior selections, and make decisions that improve classification accuracy over time. The player is implemented as a composition of six main components: a recurrent state encoder, an index embedder, a modulator, a policy network, and a classifier. These components are trained jointly to optimise the performance objective of succinct informative selections and accurate classifications. First, to build intuition, we begin with a simplified overview of the player architecture. The diagram (3.2) illustrates the three core stages: (1) updating internal state, (2) making a selection and (3) making a classification. It abstracts away component-level details such as the dual RNN path and slot modulation.

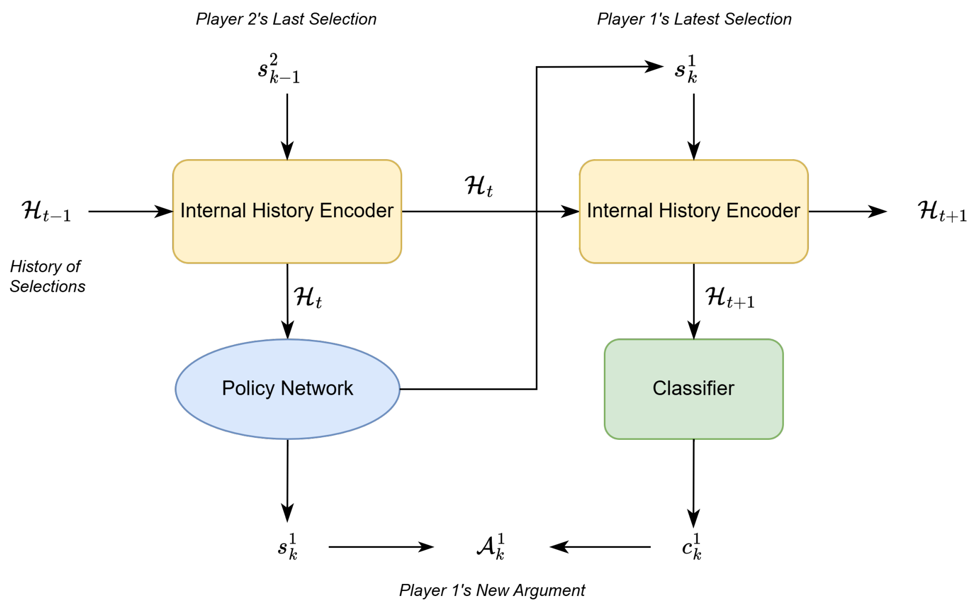


Figure 3.2: Simplified view of a player agent, namely player P^1, showing high-level modules and data flow during turn k and time step t, where $\mathcal{H}_t$ is some fixed-size encoding of the history of slot selections, s_k^i is a slot vector selected by player $\mathcal{P}^i$, c_k^i is a claim by player P^i, and $\mathcal{A}_k^i = \{s_k^i, c_k^i\}$.

Temporal Processing and Contextual Embedding. To reason over sequences of slot selections, each player maintains an internal memory of the selection history $\mathcal{H}_t$ using a Recurrent Neural Network (RNN), capturing the evolving state of the game from their perspective. At turn k, player $\mathcal{P}^i$ observes a selected slot with embedding s_k^j and a index embedding p_k^j (adapted from transformers [26]), where j denotes the player $\mathcal{P}^j$ who originally selected the slot, which could be either player. The index embedder is a learned embedding layer that maps slot indices to fixed-size vectors, enabling the model to distinguish between different slot indices.

These embeddings are summed and passed through the player-specific modulator network $\mathcal{M}^i$, a lightweight multi-layer perceptron (MLP) that reparameterises the input into a more informative representation:

$$e_k^j = \mathcal{M}^i(s_k^j + p_k^j)$$

This representation is then fed into the RNN $\mathcal{R}^i$ of player $\mathcal{P}^i$, which updates its hidden state h_t^i at time step t:

$$h_t^i = \mathcal{R}^i(e_k^j, h_{t-1}^i)$$

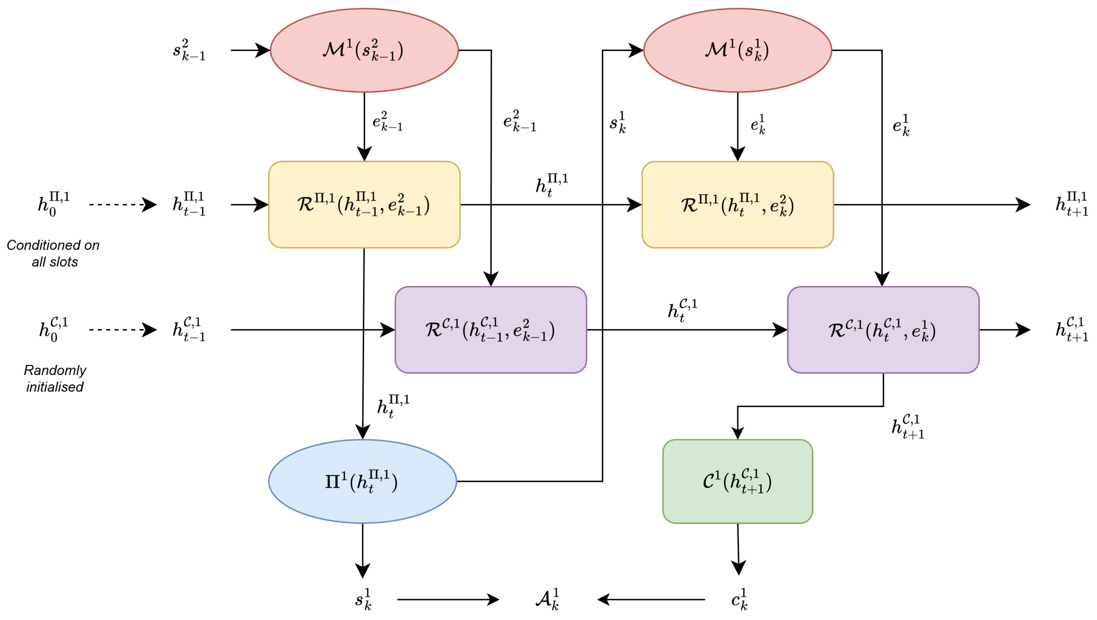
To enable the policy network to make informed decisions from the very beginning of the game, we initialise the RNN hidden state h_0^i with knowledge of the entire slot pool. This is done by sequentially processing all available slot embeddings through the RNN prior to the start of the game. However, this initialisation incorporates full information, which is incompatible with our intention with the classifier, where predictions should be based solely on cumulative evidence.

To resolve this, we maintain two separate RNN hidden states per player $\mathcal{P}^i$ throughout the game. The first, denoted $h_t^{\Pi,i}$, is used by the policy network and is initialised with all slots, providing a strong prior for action selection. The second, $h_t^{\mathcal{C},i}$, is used by the classifier and is initialised to zero, updating only as new slots are selected during gameplay. This separation ensures that the classifiers operate under partial information constraints, while the policy network benefits from full-slot context for strategic selections. The two RNNs may share architecture and parameters, but they operate over disjoint information flows:

$$
\begin{aligned}
h_t^{\Pi,i} &= \mathcal{R}^{\Pi,i}(e_k^j, h_{t-1}^{\Pi,i}) && \text{(Policy path)} \\
h_t^{\mathcal{C},i} &= \mathcal{R}^{\mathcal{C},i}(e_k^j, h_{t-1}^{\mathcal{C},i}) && \text{(Classifier path)}
\end{aligned}
$$

Action Selection and Value Estimation. The player's policy network is an MLP that produces a distribution over available actions at each time step. It receives the hidden state h_t as input and produces a logit vector π_t, which is passed through a softmax to define a stochastic policy:

$$\pi_t = \Pi^i(h_t^{\Pi,i})$$



s_{k-1}^2
$\mathcal{M}^1(s_{k-1}^2)$
e_{k-1}^2
e_{k-1}^2
s_k^1
$\mathcal{M}^1(s_k^1)$
e_k^1
e_k^1
$h_0^{\Pi,1}$
$h_{t-1}^{\Pi,1}$
$\mathcal{R}^{\Pi,1}(h_{t-1}^{\Pi,1}, e_{k-1}^2)$
$h_t^{\Pi,1}$
$\mathcal{R}^{\Pi,1}(h_t^{\Pi,1}, e_k^2)$
$h_{t+1}^{\Pi,1}$
Conditioned on all slots
$h_0^{\mathcal{C},1}$
$h_{t-1}^{\mathcal{C},1}$
$\mathcal{R}^{\mathcal{C},1}(h_{t-1}^{\mathcal{C},1}, e_{k-1}^2)$
$h_t^{\mathcal{C},1}$
$\mathcal{R}^{\mathcal{C},1}(h_t^{\mathcal{C},1}, e_k^1)$
$h_{t+1}^{\mathcal{C},1}$
Randomly initialised
$h_t^{\Pi,1}$
$h_{t+1}^{\mathcal{C},1}$
$\Pi^1(h_t^{\Pi,1})$
$\mathcal{C}^1(h_{t+1}^{\mathcal{C},1})$
s_k^1
$\mathcal{A}_k^1$
c_k^1

Figure 3.3: Partial architecture of player 1, showing component interactions and data flow at time step t and at turn k. We omit components such as the baseline network and the index encoder as they are implicit in the interactions.

Classifier as Feedback Mechanism. Central to the system is a classifier local to each player that integrates the selected slot features to predict the true class of the input instance. Each player maintains its own classifier instance, which receives the cumulative set of slot embeddings chosen up to the current time step. The classifier produces a distribution over class labels via a softmax layer, and the confidence assigned to the true class label is used to compute both sparse and dense reward signals, depending on the training regime. The classifier thus serves both as a target for individual inference and as a feedback channel to guide each player's learning.

Game Implementation

Here, we establish the distinction between the Consensus Game environment and the Consensus Game module. The environment functions as a turn-based game engine (Figure 3.4) that simulates the interactive setting in which agents operate. It maintains the state of the game, enforces constraints (3.1.2), tracks which agent's turn it is, and manages the progression of the game by applying each agent's actions. It is also responsible for emitting sparse rewards at the end of the episode based on whether agents reached an agreement and whether that agreement was correct.

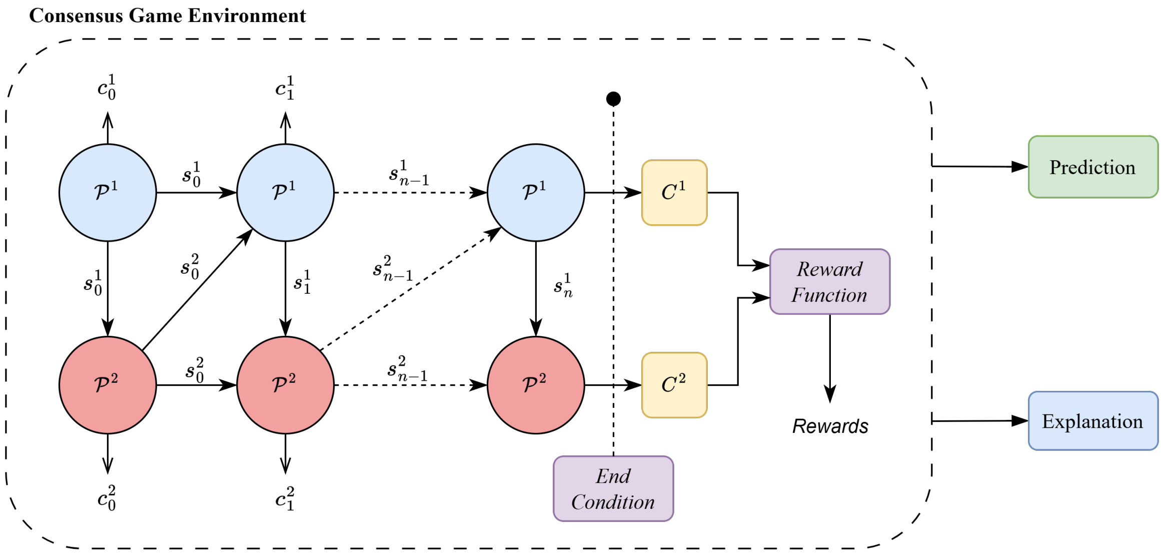


Figure 3.4: The Consensus Game Environment shows interactions between players $\mathcal{P}^1$ and $\mathcal{P}^2$. Each player $\mathcal{P}^i$ emits a selection s_k^i and a claim c_k^i to the environment and each other at turn k. Depending on their claims and confidence, they may satisfy the end condition, when the game ends and rewards are calculated. With that, the player would generate an explanation and a prediction, based on the selections and final claims C^1, C^2.

In contrast, the module is a neural controller responsible for learning and coordinating player behaviour. It manages the agents, executes the game loop, and computes losses to train both the policy and classification components. Each player is modelled as an independent agent, selecting slots and making claims over multiple turns. The module supports both policy-based reinforcement learning (via REINFORCE) and supervised learning (via cross-entropy), and includes regularisation mechanisms such as entropy bonuses and redundancy penalties to encourage effective and concise reasoning.

In the Consensus Game module, the players are coordinated to take turns selecting slots and making claims. To do so, turn management is handled by the environment, which maintains the current player index and enforces the order of actions, while the module queries the active player's policy network and classifier to produce probability distributions over possible slot selections and claims. Slot selections are stored in the environment and propagated to each player within the module to ensure well-informed decisions.

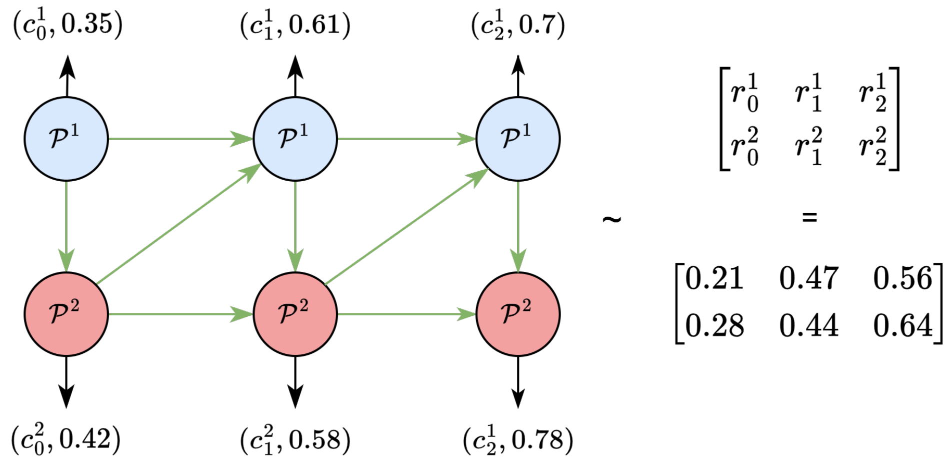
Rewards

In practice, we found that the initially designed sparse reward model, which only provided feedback at the end of the game, limited the agents' learning efficiency and led to poor strategies. To address this, we implemented a dense reward function that offers intermediate feedback at each turn, encouraging players to make moves that incrementally increase classification confidence. This leaves the adjustments for classification correctness to the cross-entropy loss.

At every turn k of the game of length K, each player outputs a probability distribution over class labels. Let p_k^i denote the probability assigned to the ground-truth class Y by player $\mathcal{P}^i$ at turn k. The reward for that turn is computed as the gain in confidence relative to a baseline value p_0, which can be initialised as zero or the uniform probability $|\mathcal{Y}|^{-1}$. The per-turn reward is thus:

$$r_k^i = p_k^i - p_0$$

This formulation incentivises agents to select informative slots that gradually build up confidence in the correct class, even before reaching the final prediction step. An example can be seen in Figure 3.5.



$(c_0^1, 0.35)$
$(c_1^1, 0.61)$
$(c_2^1, 0.7)$
$\mathcal{P}^1$
$\mathcal{P}^2$
$(c_0^2, 0.42)$
$(c_1^2, 0.58)$
$(c_2^1, 0.78)$
$\sim$
$\begin{bmatrix} r_0^1 & r_1^1 & r_2^1 \\ r_0^2 & r_1^2 & r_2^2 \end{bmatrix} = \begin{bmatrix} 0.21 & 0.47 & 0.56 \\ 0.28 & 0.44 & 0.64 \end{bmatrix}$

Figure 3.5: Example per-turn reward assignment for players $\mathcal{P}^1$ and $\mathcal{P}^2$. The green arrows show the argument dependencies, while the black arrows show the emission of the claim-confidence pairs (c_k^i, p_k^i) for player $\mathcal{P}^i$ at turn k. The rewards are calculated using $r_k^i = p_k^i - p_0$, where $p_0 = \frac{1}{7}$.

.

We also experimented with an alternative reward scheme based on relative changes in confidence between players, defined as:

$$r_k^i = p_k^i - p_{k-1}^{-i}$$

This approach also improved learning efficiency but introduced a risk of reward hacking, where agents could suppress early confidence to artificially inflate perceived improvement later. This undermined the integrity of the learning signal. As a result, we adopted the more stable and interpretable absolute-confidence-based dense reward formulation.

Training

The model is trained end-to-end through a combination of reinforcement learning and supervised learning objectives. Each epoch consists of multiple consensus games played over batches of input instances. For every batch, slot embeddings and class labels are extracted and passed to the game environment, which orchestrates the game dynamics such as the starting player, turn order, and game termination. To mitigate ordering bias, the starting player is alternated across batches. During training, the number of turns per game is fixed, simplifying reward attribution and ensuring consistent episode lengths.

At each time step, the active player selects a slot and optionally issues a classification claim. After each episode concludes, a reward signal is computed based on each players' classifiers' confidence in the correct label. This reward is used to train the players' policies via the REINFORCE algorithm. Concurrently, cross-entropy losses are computed between the classifier logits and the ground-truth labels, and updates both player's classifiers separately.

Inference

During inference, the Consensus Game operates very similarly to training mode depending on the end condition. Instead of running for a fixed number of turns, the game terminates once a certain condition is satisfied or until a maximum number of turns. This setup enables the agents to reach a dynamic consensus based on selected factors, rather than being constrained to a fixed interaction length. There are a few options for the end condition that we tested with, the results of which would be detailed further in chapter 5. This is a summary of the different end conditions for the consensus game:

- **Consensus**: The game ends once all players have made the same classification claim, which models convergence of opinions across players. This is extended further to also account for classifier confidence in their claim. This combines both agreement and sufficient certainty.
- **Repetition**: The game terminates when any player selects the same slot more than once. This provides a mechanism for a player to end the game early, showing that there was not a more informative slot to choose from than to repeat. This can be tuned to be stricter to apply across all selections between players.
- **Decreasing Confidence**: The game ends if a player's classification confidence decreased due to an additional selection. This allows early stopping when confidence degrades with more evidence.

A special case for the "Consensus" end condition emerged when analysing the reasoning chains and occasionally observing redundant selections in order to reach agreement. As an analogy, typical discussions between two people would have them providing their points and agreeing on a decision after some time. The key thing we mimicked for this end condition is to allow the players to agree without providing another argument, by adding a pre-selection claim step to check if the other player's arguments have "convinced" the current player.

End conditions play a very important role in the Consensus Game. Not only keeping reasoning chains concise, they align players' claims and provide a baseline to achieve before a trustworthy classification can be put forward.

Ablations and Improvements

During the development of the Consensus Game module, we encountered several challenges and unexpected behaviours that required careful analysis and iterative refinement. To address these, we conducted a series of ablation studies and experimented with different design choices across various components of the system. These investigations not only helped us identify the critical factors affecting performance but also guided improvements in model stability, learning efficiency, and overall robustness. Most of these ideas were detailed above. However, there were a few experiments that yielded mixed results.

The following points summarise the experiments that were not used in the final implementation, but did reinforce the design choices that were made and helped determine additional improvements:

- **Shared RNN:** An early experiment was using only a single shared instance of RNN to encode shared hidden states between both players. We maintained the dual RNN paths for the policy network and classifier due the difference in encoded information, but the paths both branched from the same RNN. Our intention was to unify the encoding process and the information that was available to both players, which would encourage the players to make decisions and classifications more in alignment with each other. However, we found that this had no measurable effect on the prediction accuracy, convergence speed, or the quality of slot selections. In fact, decoupling the RNNs offered greater architectural flexibility in the long run.

- **Different reward models:** As mentioned earlier (3.1.2), we initially used a sparse reward function. This approach failed to provide sufficient learning signal for players to develop effective slot selection strategies. Transitioning to a dense reward model improved performance significantly (3.2.2). However, with different inference-time end conditions, we also tailored the reward function to include additional rewards. For example, when repetition was used as the stopping criterion, we provided positive reinforcement for selecting repeated class-relevant slots, scaling rewards by a tunable weight, typically $w_{repeat} = 0.2$. Despite the gains, we observed that excessive reward shaping could inadvertently bias the learning process, leading to overfitting to highly specific behaviours or even suppressing exploration.

- **Regularisation Losses:** We designed three different types of regularisation losses, namely a diversity term, a selection uniqueness term, and a consensus term. While optional, these regularisation terms help provide a learning signal to encourage certain desirable behaviours in explaining the classification. The diversity term encourages the agents to select semantically different slots over time, reducing selections of similar slots, which should be impactful as Slot Attention may generate slots that appear identical to another. The selection uniqueness term could be used to either encourage or penalise repeated selections of the same slot, depending on the end condition. Finally, the consensus term rewards alignment of the two agent's final predictions, reinforcing collaborative behaviour, although not showing significant benefit towards convergence speed nor consensus rates. This suggests that the classifiers obtain sufficient signal from the cross-entropy loss already.

3.2.3 End-to-End Learning

The transition from a modular pipeline to an end-to-end (E2E) learning framework involved integrating the Slot Attention and Consensus Game modules into a unified training loop. To enable joint learning, we modified the architecture so that gradients from the Consensus Game could propagate backward through the Slot Attention module. This allowed the slot representations to be shaped directly by the downstream classification objective, aligning slot discovery with task relevance, as proposed by Locatello et al [12].

In practice, this introduced a strong learning signal from the cross-entropy loss associated with the classifiers' predictions. We found that this loss term significantly influenced the slot representations, often overpowering the original slot reconstruction loss due to a mismatch in their magnitudes. To address this, we introduced a weighting factor on the reconstruction loss to maintain object-centric quality in the slots while still benefiting from task-driven supervision. This adjustment enabled more informative and class-discriminative slot encodings to emerge during training. However, a noticeable side effect emerged, where ungrounded or empty slots began to carry sufficient information to support correct classifications. As a result, the agents did not have to pick actual object-centric slots, in order to successfully complete the game with high rewards. This undermined the interpretability of the generated explanations, as success in the game no longer guaranteed that the agents were grounding their arguments on objects in the input image. While constraints could, in theory, be applied to enforce object-centricity, they felt very forced.

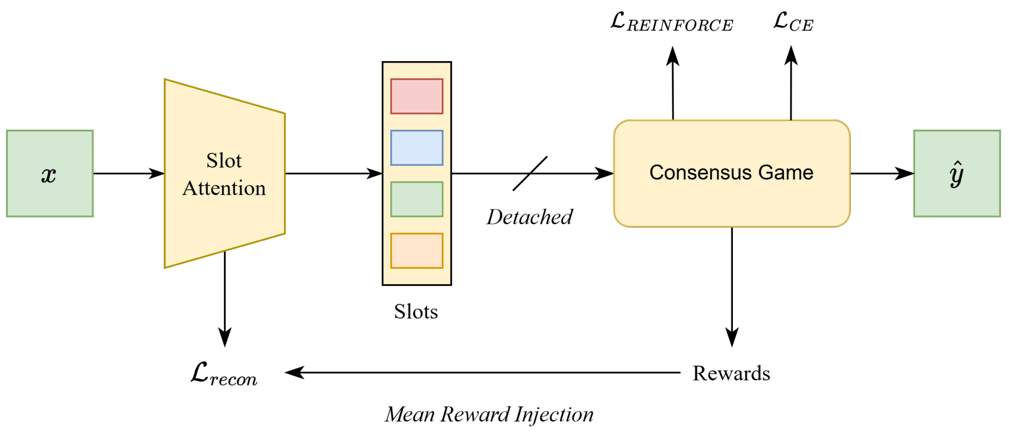


Figure 3.6: Reward injection into Slot Attention training. The Slot Attention module encodes the input image into slots (detached) used by Consensus Game, producing a reward signal, which is incorporated into the reconstruction loss $\mathcal{L}_{recon}$. This occurs in congruence with the backpropagation of the losses $\mathcal{L}_{CE}$ and $\mathcal{L}_{REINFORCE}$ within the Consensus Game module.

This was when we moved towards a learning paradigm that isolates the two modules better, while still providing an outlet for encouraging better slot encodings. The Expectation-Maximisation algorithm [27] inspires our training algorithm, where we have two maximisation steps aimed at maximising the reward outcome. In the first step (3.6), the Slot Attention module acts as the latent variable model, with the Consensus Game frozen. Slot Attention is updated to provide slot representations that better support downstream decision-making. The update comes in the form of an augmented reconstruction loss, where we subtract a scaled mean reward, thereby encouraging slot encodings to also correlate with high-reward outcomes. In the second step, the Consensus Game module is updated based on the current slots, with upstream Slot Attention module frozen, which is equivalent to the training protocol of the pipelined version of our architecture. The alternating updates decouple the learning dynamics of the two components and reduce interference.

Following EM training, the slot reconstructions look identical to the ones obtained from the original pipeline setup, suggesting that the added reward signal did not compromise the perceptual quality of the extracted slots. By framing the optimisation steps as an EM loop, we decouple the pressure from the combined loss from Consensus Game, while allowing some reward signal to flow back to the slot autoencoder. We also observed incremental improvements in accuracy and quality of explanations as a result of E2E learning, where its evaluation is detailed in chapter 5.

Chapter 4

Experimental Setup

In this chapter, we discuss the methods we employ to evaluate the quality of our framework. A good evaluation of our framework requires not only measuring classification accuracy, but also assessing how faithfully and clearly it traces it's decision-making. This motivates the need for metrics that reflect not just predictive success, but also interpretability, efficiency, and agent collaboration. Notably, to determine where our framework stands with the state-of-the-art visual explainability mechanisms, we conduct a user survey to gauge their opinions on the interpretability of the explanations of our framework and the baselines.

We begin by selecting suitable datasets (4.1) and baselines (4.2), followed by specifying model training hyperparameters for various versions of our framework (4.3). We also discuss the metrics to collect when evaluating our framework on both prediction and explanation quality (4.4).

4.1 Datasets

The success of this framework depends on datasets that support object-centric reasoning and interpretable explanations. Suitable datasets must feature multiple objects with compositional variability, where the images exhibit diverse combinations of object attributes such as colour, shape and position. Importantly, the images should be labelled based on logical rule-based conditions such as "a red square left of a green circle". These requirements ensures that the task has a well-defined reasoning component. Here, we discuss two datasets that would be used in the project.

4.1.1 CLEVR-Hans

CLEVR-Hans [28] extends from the original CLEVR [29] diagnostic visual reasoning dataset adapted for binary classification tasks. The dataset features images of 3D scenes with multiple coloured geometric objects on a plain grey background. Each image is associated with a label derived from a specific logical rule (e.g., "Large (gray) cube and large cylinder"), enabling study of concept learning and generalisation. The training and test splits are carefully constructed to assess whether a model can generalise to novel object combinations and avoid shortcut learning. For example, certain colour-shape combinations may not be seen during training but only appear in new configurations at test time for the test split (non-confounded). For example, "(gray)" in "Large (gray) cube and large cylinder" is an attribute of the large cube that is present in every training image for that class, but during test time, the gray attribute is optional. We will be testing with both splits (confounded and non-confounded) to analyse the generalisability of our model. Finally, the dataset's scenes with clearly distinguishable objects make the dataset particularly well-suited for mechanisms like Slot Attention.

4.1.2 Multi-dSprites

Multi-dSprites [30] builds on the single-object dataset dSprites [31], where each image consists of a white object set against a uniformly black background. This version of the dataset that we are using contains up to four coloured objects against a coloured background. The images varies by object positions, shape, colour, orientation, scale and number. The dataset includes are four object types: background, square, ellipse, and heart. The dataset did not include binary labels, as it is primarily intended for object discovery. To adapt it for our framework, we introduced five logical classes:

- **Class 0**: A red square and a heart.
- **Class 1**: Two hearts on the left side.
- **Class 2**: An ellipse and two squares.
- **Class 3**: A bright object infront of a dark background.
- **Class 4**: Three different shapes on the right side.

This addition introduces object-based logical reasoning into a previously unsupervised dataset. However, unlike CLEVR-Hans, we do not construct a non-confounded split, making it less suitable for evaluating generalisation, but still valuable for assessing structured reasoning and interpretability.

4.2 Baselines

We developed a few experimental baselines to evaluate our framework against in terms of prediction and explanation quality. The results of these baselines would be helpful in indicating the performance of our framework on both fronts.

4.2.1 Prediction Baselines

We employ two methods for our classification prediction baselines, both of which uses the standard feature extractor + multi-layer perceptron (MLP) framework.

CNN Feature-Based Classification

This baseline uses a standard convolutional neural network implementation, ResNet18 [13], trained directly on images to perform binary classification. The network processes the image through multiple convolutional layers to extract spatial features, which are then aggregated using global average pooling and passed through a fully connected layer for final prediction. This setup evaluates how well a conventional image classifier can perform without explicit object decomposition.

Slot-Based Classification

In this baseline, object-centric representations obtained from the Slot Attention module are used as input to a MLP for classification. Each slot corresponds to a distinct object-like region in the image, and the entire set of slots is flattened and processed jointly by the MLP. The MLP is trained E2E with the Slot Attention module, as employed in this project's framework. This approach tests whether decomposing a scene into discrete object representations can aid in learning more interpretable and potentially generalisable classifiers.

4.2.2 Explanation Baselines

Here, we explain how the explanation baselines were derived to generate explanations for the various state-of-the-art explanation mechanisms. These baseline explanations will be evaluated against the explanations generated from our framework, in a human survey specified in 4.4.1. We implemented the explanation mechanisms on top of the baseline models. As the mechanisms below are tailored to CNN-based classifiers, we will be discussing more on integrating the explanations mechanisms with the slot-based classifier. See section 2.6 for a background on these mechanisms.

Grad-CAM

Grad-CAM generates visual explanations by highlighting regions of the input image that contribute greatly to the model's decision. It uses the gradients of the target class flowing into the final convolutional layer to produce a coarse localisation map, which determines which features were important for the prediction. This method was first applied to the CNN baseline and adapated for the slot-based classifier by computing gradients with respect to individual slots. By doing so, we were able to compute an importance score associated with each slot, showing which slots were important in the classifier's decision.

For visualisation, Grad-CAM typically displays a coloured heatmap over the input image. With the version for the slot-based classifier, we observed that the highlighted regions often fitted over the objects nicely, and thus lacked clarity that the object was highlighted. The resulting explanations, while indicating slot-object contributions, were visually ambiguous and difficult to interpret. Thus, we utilise a different approach where the slots with lower importance scores are masked out instead. This method offers a unique opportunity for comparison with our framework's explanations, as both are object-centric. However, the key difference lies in presentation: this approach produces explanations within a single image, whereas our framework distributes them across multiple images in a dialogue-style.

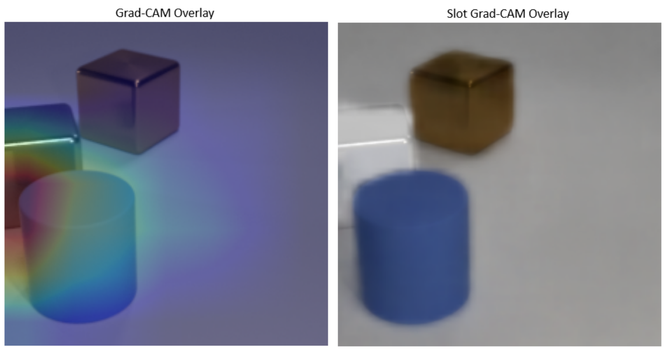


Figure 4.1: Grad-CAM Example for classification of a CLEVR-Hans image (class 0: Large Cube and Large Cylinder) for a CNN-based classifier (left) and a Slot-based classifier (right).

LIME

LIME, a model-agnostic mechanism, approximates the classifier locally with a simpler interpretable model by perturbing the input vector and observing changes in prediction. For our baselines, LIME highlights regions that contribute most to the classification by drawing around the boundaries of the regions. There were two approaches to applying this method to the slot-based classifier, where LIME perturbs either the raw image vector or the individual slot representations. We decided on the former, as this approach aligns better with LIME's design and does not produce a repeat of the importance score method we used for Grad-CAM.

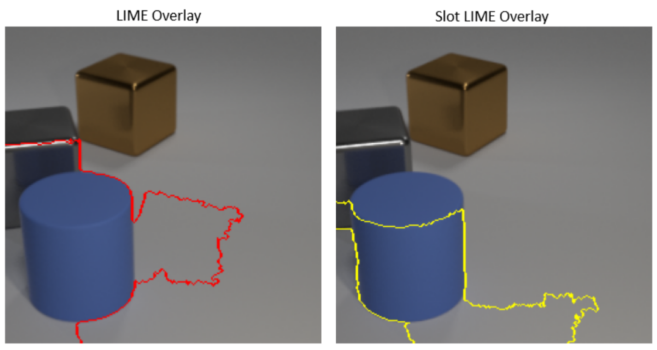


Figure 4.2: LIME Example for classification of a CLEVR-Hans image (class 0: Large Cube and Large Cylinder) for a CNN-based classifier (left) and a Slot-based classifier (right).

Gradient SHAP

Gradient SHAP is unified framework combining Integrated Gradients and SHAP (SHapley Additive exPlanations) values to estimate the contribution of each input feature. Effectively, it is similar to importance score assignment for each pixel or feature region in the image. We apply Gradient SHAP to both baselines, which produced similar results in the explanations.

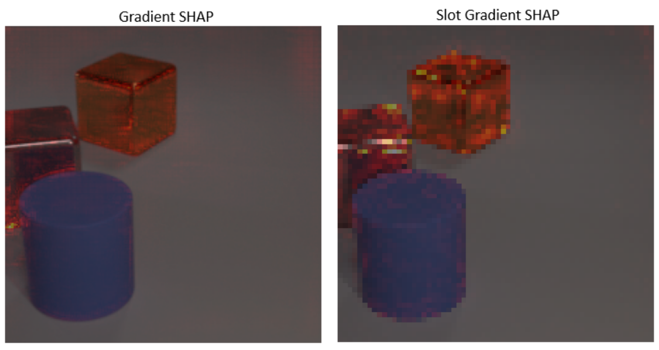


Figure 4.3: Gradient SHAP Example for classification of a CLEVR-Hans image (class 0: Large Cube and Large Cylinder) for a CNN-based classifier (left) and a Slot-based classifier (right).

4.2.3 Prior Work Comparison

A natural baseline for comparison would be the prior work, Visual Debates [7], which we draw inspiration from and also employs a multi-agent game for explanation generation. However, we chose not to inlude it as a direct experimental baseline due to several key difference in design assumptions, scope, and applicability.

First, the two frameworks are situated in fundamentally different domains. Our approach leverages object-centric representations via Slot Attention, which is well suited for synthetic multi-object datasets where discrete object-level reasoning is important. In contrast, Visual Debates operates on quantised latent features of CNNs, which are better aligned with high-resolution single-entity images such as those of the AFHQ dataset [32]. Second, while both methods uses multi-agent interactions for explanation generation, the underlying representations and modes of interaction differ significantly. Our framework aims to provide arguments in support of the classification, while Visual Debates provides arguments for and against the classification.

Given these differences, we find it more appropriate to compare our method against standard state-of-the-art explanation methods like Grad-CAM and LIME, which generalises well across varying types of datasets including multi-object datasets. Nonetheless, we consider Visual Debates as an important conceptual predecessor, where our framework was motivated by the lack of interpretability of entangled concepts in the explanations.

4.3 Model Training

When pre-training an initial Slot Attention model, we opted for an input resolution of 64×64, but we extended our experiment to include 128×128 as well. Then, we trained E2E three representative configurations of the framework for evaluation.

4.3.1 Slot Attention Specification

Two configurations were adopted based on image resolution: one for 64×64 inputs and another for 128×128 inputs. While the core architecture and algorithm remained consistent across resolutions, the higher resolution model employed wider convolutional layers to accommodate the increase in detail of the input images. Both models were pre-trained with a reconstruction objective on the two synthetic datasets to ensure the slots are encoded meaningfully before engaging in E2E training with the Consensus Game module. The table below summarises key hyperparameters used in each configuration.

Parameter	**Slot Attention** (64×64)	**Slot Attention** (128×128)
Input Resolution	64×64	128×128
Network Width	64	128
Slot Dimension	64	128
Routing Iterations	3	3
Epochs	500	1000
Learning Rate	2×10^{-4}	5×10^{-5}
Weight Decay	1e-6	1e-6

Table 4.1: Pre-training configuration for the Slot Attention model at 64×64 and 128×128 resolutions prior to training E2E with Consensus Game.

When we perform E2E training, we made some adjustments to the Slot Attention training specification to account for the augmented reconstruction loss, which is influenced by the reward signal from the Consensus Game module. In particular, we reduce the learning rate (1×10^{-5}) and weight decay (5×10^{-7}) to stabilise training and prevent the slot encoder from overfitting to the reward signal too quickly, allowing more gradual updates to the slot representations.

4.3.2 Consensus Game Specification

To evaluate the impact of different gameplay dynamics and learning settings, we trained three representative configurations of the Consensus Game module in tandem with a pre-trained configuration of the Slot Attention module. These configurations are denoted as Config A, B and C. They are summarised as follows:

- **Config A:** We utilise the 64×64 variant of the Slot Attention module for lower resolution slots in the Consensus Game, with the consensus-based end condition. The end condition here depends on the players arriving to the same claim while having a confidence that exceeds the threshold, 70% Conf(idence).
- **Config B:** We augment the end condition of the 64×64 variant to trigger on repeated selections by any player. To encourage this behaviour and end games early, we reward repeated good selections (scaled by 0.2), as described in section 3.2.2.

- **Config C:** We repeat Config A for the 128 × 128 variant of the Slot Attention module, with some modifications in the component widths.

Parameter	**Config A**	**Config B**	**Config C**
Input Resolution	64×64	64x64	128×128
Epochs	1500	1500	2000
Learning Rate	2e-5	2e-5	2e-5
Training Game Length	6	6	6
Reward Strategy	Conf	Conf + Repeat	Conf
End Condition	Consensus w/ 70% Conf	Repetition Across Players	Consensus w/ 70% Conf
RNN Input Size	128	128	256
RNN Hidden Size	128	128	256
Classifier Width	256	256	512
Policy Width	256	256	512

Table 4.2: Training configurations for three variants of the Consensus Game model.

4.3.3 Additional Experiments

We planned and executed various experiments throughout the duration of the project, where we explored different ablations, implementations and set ups. Here, we introduce some of the interesting experiments we had conducted. We use the CLEVR-Hans7 dataset as our representative testbed. Given that these experiments are designed to isolate specific architectural or training effects, a single dataset would provide sufficient evidence to validate observed behaviours. With that, we also fix the training and architectural hyperparameters across experiments wherever applicable, allowing us to attribute any changes directly to the experimental variable under investigation. The results can be found in section 5.5.

Pipelined and End-to-End Training

One key experiment planned from the start of the project involved compared pipelined training, where the Slot Attention module is pre-trained and frozen, with full end-to-end training of the entire framework. This comparison helps reveal how tight coupling between the slot encoder and the reasoning game affects interpretability and performance. The pipelined setup offers greater stability and preserves object-centricity in the slots, as the loss value used for training the Slot Attention module is based purely on the reconstruction loss. However, end-to-end training provides a reward signal to the Slot Attention module allowing for task-specific adaptation of the slots, potentially improving classification accuracy of our framework, albeit at a risk of introducing slot entanglements and reduced object fidelity. This experiment helps us understand how training dynamics influence both predictive performance and interpretability.

Partial Visibility of Slots Across Players

This experiment explores the effect of limiting the visibility of the slot pool between players. In the standard setup, each player is able to observe the entire set of slots. Here, we modify the environment so that each player can only observe and select from a disjoint subset of the slot pool. For example, in a two-player game with 10 slots, player 1 may observe slots 1 to 5, while player 2 may observe slots 6 to 10. This setup tests the information sharing ability of both agents, as they must collaborate to reach consensus and produce a coherent reasoning chain. By comparing behaviours and performance between the full visibility and partial visibility settings, we can better understand the role of shared visual information in our setup. Finally, to standardise the action space for both players, we look to split the slot pool evenly among the players. For CLEVR-Hans7, we fix the total number of slots to 10, such that both players have access to 5 slots each.

Different Number of Players

Here, we explore how varying the number of players affects predictive performance, consensus rates, collaborative reasoning, and explanation quality. While our framework is designed around two-player interactions, it can naturally extend to single-player or to configurations with three or more players. A single-player variant removes the need for consensus and serves as a useful baseline for evaluating the added value of collaboration in the reasoning process. On the other hand, introducing additional players increases the complexity of coordination and may affect consensus rates, redundancy in selections, or the coherence of the generated explanation.

Tuning Confidence Threshold

With this experiment, we only alter the confidence threshold for the inference end condition. We vary the threshold from 0% to 100% in 10% incrememts on the Config A setup. Increasing this threshold means that the agents are more confident in their claims before the game concludes, providing more accurate predictions due to the additional slots needed. This variable can be changed after training has concluded and thus this experiment would not be computationally costly. The results of which would provide users of our framework in choosing a suitable confidence threshold when employing the consensus-based end condition.

4.4 Evaluating Explanations

It is challenging to evaluate the explainability of any model objectively as there is no universally accepted heuristic for what constitutes a "good" explanation, unlike accuracy in measuring prediction quality. To address this, we adopt the strategy detailed below. First, we conduct a survey to capture subjective opinions of explanation quality, interpretability and usefulness. Then, to distinguish varying configurations of our framework, we build up a comprehensive set of metrics to gain additional insights into their explananda.

4.4.1 Survey Specification

Objective:

The purpose of the survey is to evaluate the interpretability and effectiveness of our OCEAN framework by comparing it against the state-of-the-art post-hoc explanation methods. The study aims to determine whether participants are able to infer the predicted class from an explanation alone, indicating how understandable and intuitive each explanation method is.

Target Audience:

The survey targets students and academic researchers, with some computer science/machine learning background. However, the task does not require deep technical knowledge and is also suitable for participants with general analytical skills.

Methodology:

Participants will be tasked with guessing the predicted class of the model based on the explanations alone. Explanations generated from most explanation mechanisms rely on the input image for highlighting regions of the image. Thus, the survey will also include six statements that the participants will rate using a Likert scale. They will score each statement from 1 (Strongly Disagree) to 5 (Strongly Agree):

- It is clear which parts of the image is the explanation.
- The explanation was easy to interpret without prior technical knowledge.
- The explanation would help me identify errors or biases in the model's prediction.
- The explanation only focused on relevant parts/objects of the image.
- The explanation helped me understand why the model predicted the class.
- I am satisfied with the quality of the explanation.

For each statement, we investigate these aspects respectfully: clarity, interpretability, error and bias identification, relevance, understandability, and satisfaction. This line of questioning would be repeated for five different explanation mechanisms and for two datasets. We use different images for each explanation to avoid familiarity of answers. The five mechanisms are: LIME, Grad-CAM, Grad-CAM on Slot Attention, Gradient SHAP and our OCEAN framework. Grad-CAM was the only explanation baseline built on top the slot-based classifier chosen for this survey. This is due to the explanations for LIME and Gradient SHAP being indistinguishable from their CNN-based counterpart.

Following collection of results, the correctness of the participants' answers and the average Likert scores would provide us an indication of the intuitiveness and understandability of different explanations.

4.4.2 Metrics

For our framework, we are able to quantitatively measure a number of metrics based on various aspects relating to: slot encodings, slot attention values, argument selection, game length, policy and classifier confidence, agent rewards, and player claims. Each metric is selected and designed in alignment with the key objectives we consider desirable in an explainable object-centric framework: correctness, interpretability, coherence, and informativeness.

Firstly, we find that correctness, consensus and confidence of player claims are central to evaluating the effectiveness of the Consensus Game. We assess how often both players' claims are aligned and correct with respect to the ground truth label. These metrics helps us compare different framework configurations and game dynamics in terms of correctness and reliability.

Next, game length can help measure selection efficiency and quality of arguments. High consensus and low game lengths would suggest the policies had learnt good strategies for slot selection. We can support that by observing the per-slot attributed attention values given by Slot Attention to determine the quantity of information in each slot. This quantity of information can be used as a heuristic for determining whether a slot corresponds to an object. In addition, uniqueness of slot selections would allow us to determine whether the model is able to balance between diversity and coherence in the agents' arguments.

Finally, with Slot Attention now trained end-to-end with the Consensus Game, it may diverge from its optimal weights due to external influence on its reconstruction loss. As a result, it is imperative to monitor the loss and the slot image reconstructions in maintaining the quality of slot decompositions, and thus maintaining interpretability.

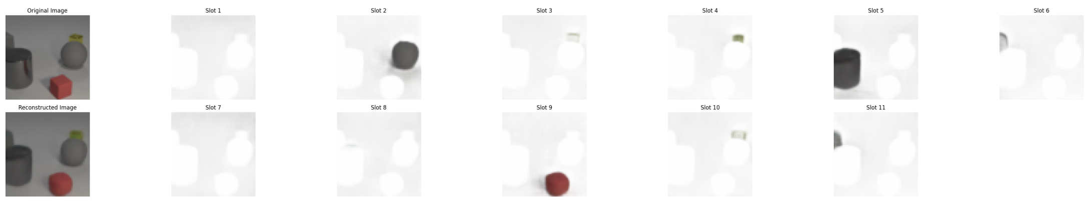
Chapter 5

Experimental Results

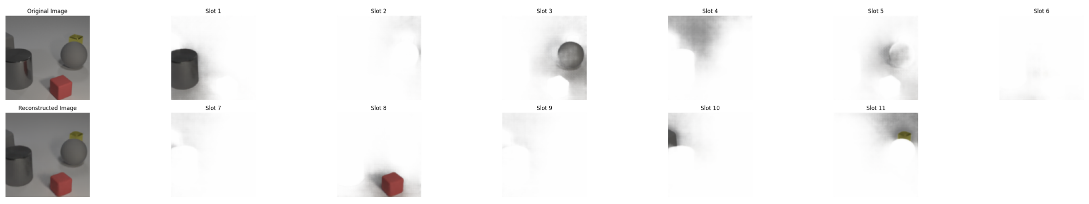
With the models evaluated, we present a comprehensive evaluation of our framework across multiple axes, including classification performance, interpretability, and explanation quality. We first touch on the Slot Attention performance (5.1). Next, we discuss the classification results obtained from our baselines and different configurations of our framework (5.2). We then analyse metrics from the Consensus Game (5.3) and the results from our survey (5.4). Finally, we discuss the results of various interesting experiments (5.5) conducted during the course of the project.

5.1 Slot Attention

As training the slot encoder is more computationally intensive and requires significantly longer convergence times, the optimal method to train our framework is to "warm up" the Slot Attention module. This can be achieved in two ways: train the slot attention module exclusively for some number of epochs or load in pre-trained weights. The examples shown below (5.1) were generated from a pre-trained slot encoder-decoder purely on its own reconstruction loss.



(a) Slot Attention outputs for a 64×64 image.



(b) Slot Attention outputs for a 128×128 image.

Figure 5.1: Breakdown of slot representations and reconstructions produced by Slot Attention at two different input resolutions for an image of the CLEVR-Hans dataset.

We observed good results (5.1) across the configurations (4.3.1). Although they have comparable reconstruction losses, the quality of the segmentation masks is much worse in the higher-resolution model. These masks were often unevenly filled and failed to cleanly isolate objects, resulting in less interpretable slot reconstructions, as seen in Figure 5.1. This discrepancy may stem from the increase spatial complexity in higher-resolution inputs, creating a harder optimisation problem with many possible (sub-optimal) local minimas. Further work and tuning has to be put in, in order to make the framework more scalable for higher-resolution images.

Datasets/Input Size→	**CLEVR-Hans**		**Multi-dSprites**
Training Stage ↓	64×64	128×128	64×64
Pre-Training ($\times 10^{-3}$)	1.847	1.046	6.712
Post-E2E ($\times 10^{-3}$)	1.264 ± 0.229	1.168 ± 0.252	6.099 ± 0.117

Table 5.1: Comparison of Slot Reconstruction Loss across input resolutions measured following pre-training of the Slot Attention module and an average value following E2E training across multiple runs.

5.2 Prediction Quality

A natural choice of evaluation of a classification framework is measure its prediction quality across various datasets and input resolutions. In this section, we assess the predictive performance of our proposed framework by comparing it against two standard baselines (4.2), a CNN-based classifier and a Slot-based classifier. We utilise three representative configurations of our own model, denoted as Config A, B, and C (4.3). We report results on three synthetic multi-object datasets, CLEVR-Hans3, CLEVR-Hans7, and Multi-dSprites. This evaluation will help quantify how well our framework performs classification while maintaining the structural constraints needed for interpretability and partial observability of the entire input. Additionally, we analyse a representative model's incorrect predictions to see in which aspects it struggles in.

5.2.1 Accuracy Metrics

To evaluate predictive performance, we compute standard classification metrics: accuracy, precision, recall, and F1-score, derived from three independently seeded runs. This evaluation is applied consistently for the baselines and our framework. Where relevant, we also report the results for multiple input resolutions.

Baseline results (5.2) were collected using a standard ResNet implementation trained on images of resolution 224×224, and a Slot-based classifier evaluated at resolutions of 64×64 and 128×128 across all datasets (except for Multi-dSprites on the 128×128 configuration). The baselines serve as performance anchors, particularly in scenarios where full image observablity and unconstrained inference are available.

Configs/Input Size →	**ResNet**	**Slot MLP**		**Config A**
Dataset ↓	**224**	**64**	**128**	**64**
CLEVR-Hans7 (CF)	$90.32 \pm 0.06\%$	$86.70 \pm 0.08\%$	$85.34 \pm 0.45\%$	$77.64 \pm 1.04\%$
CLEVR-Hans7 (NCF)	$83.54 \pm 0.09\%$	$86.66 \pm 0.07\%$	$86.39 \pm 0.19\%$	$70.15 \pm 1.23\%$
CLEVR-Hans3 (CF)	$93.11 \pm 0.25\%$	$86.62 \pm 0.38\%$	$87.90 \pm 0.69\%$	$78.50 \pm 0.48\%$
CLEVR-Hans3 (NCF)	$60.93 \pm 0.58\%$	$85.60 \pm 0.18\%$	$88.76 \pm 0.25\%$	$52.58 \pm 0.54\%$
Multi-dSprites	$83.20 \pm 0.41\%$	$92.72 \pm 0.14\%$	—	$79.76 \pm 0.46\%$

Table 5.2: Prediction accuracy across baselines and resolutions on various datasets. More detailed results for the baselines can be found in Appendix A.1.

In contrast, our framework operates under stricter constraints, where agents make predictions on partial observations through sequential slot selection, and are jointly optimised for both prediction and interpretability. Reflecting this trade-off, we report the results across the three representative configurations, Config A (5.3), B (5.4) and C (5.5), averaged across three seeded runs.

We observed that the configurations with the "Consensus" end condition outperformed Config B with its repetition-based end condition. The drop in performance can be attributed to the agents adopting repetition-based strategies that are less aligned with the goal of achieving consensus. Interestingly, Config B achieves comparable performance on CLEVR-Hans3 with the other configurations, likely due to the less complex combinations of objects that are needed for a conclusive classification.

Config →	**Config A**				
Dataset ↓	**Consensus**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CLEVR-Hans7 (CF)	94.63 ± 0.66%	77.64 ± 1.04%	77.84 ± 1.04%	77.55 ± 1.05%	77.49 ± 1.05%
CLEVR-Hans7 (NCF)	92.66 ± 0.47%	70.15 ± 1.23%	70.66 ± 1.25%	69.91 ± 1.19%	69.80 ± 1.22%
CLEVR-Hans3 (CF)	97.90 ± 0.11%	78.50 ± 0.48%	79.21 ± 0.85%	78.51 ± 0.48%	78.55 ± 0.48%
CLEVR-Hans3 (NCF)	97.10 ± 0.22%	52.58 ± 0.54%	51.11 ± 0.59%	52.25 ± 0.55%	50.03 ± 0.66%
Multi-dSprites	95.16 ± 0.37%	79.76 ± 0.46%	79.65 ± 0.34%	79.65 ± 0.47%	79.60 ± 0.44%

Table 5.3: Prediction metrics for a representative configuration of our framework, Config A, across the three datasets. We use two different splits of the CLEVR-Hans datasets, confounded (CF) and non-confounded (NCF).

Config →	**Config B**				
Dataset ↓	**Consensus**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CLEVR-Hans7 (CF)	87.43 ± 0.77%	65.56 ± 1.47%	65.46 ± 1.65%	65.36 ± 1.56%	65.27 ± 1.68%
CLEVR-Hans7 (NCF)	85.78 ± 0.63%	59.17 ± 0.27%	59.32 ± 0.24%	58.91 ± 0.22%	58.83 ± 0.16%
CLEVR-Hans3 (CF)	94.16 ± 0.48%	79.47 ± 1.20%	79.68 ± 1.23%	79.44 ± 1.21%	79.45 ± 1.21%
CLEVR-Hans3 (NCF)	92.84 ± 0.73%	52.73 ± 0.62%	49.86 ± 0.55%	52.27 ± 0.56%	49.41 ± 0.72%
Multi-dSprites	79.32 ± 0.61%	63.38 ± 1.03%	62.65 ± 0.42%	62.19 ± 1.10%	61.83 ± 1.42%

Table 5.4: Prediction metrics for a representative configuration of our framework, Config B, across the three datasets. We use two different splits of the CLEVR-Hans datasets, confounded (CF) and non-confounded (NCF).

Config →	**Config C**				
Dataset ↓	**Consensus**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CLEVR-Hans7 (CF)	94.29 ± 0.31%	76.80 ± 0.41%	76.81 ± 0.41%	76.74 ± 0.45%	76.69 ± 0.45%
CLEVR-Hans7 (NCF)	92.79 ± 0.21%	71.00 ± 0.80%	71.18 ± 0.72%	70.93 ± 0.80%	70.89 ± 0.76%
CLEVR-Hans3 (CF)	98.31 ± 0.25%	82.00 ± 1.34%	82.38 ± 1.17%	82.00 ± 1.34%	82.05 ± 1.32%
CLEVR-Hans3 (NCF)	96.33 ± 0.26%	55.27 ± 1.02%	53.12 ± 1.52%	54.62 ± 0.97%	50.72 ± 0.90%

Table 5.5: Prediction metrics for a representative configuration of our framework, Config C, across the three datasets. We use two different splits of the CLEVR-Hans datasets, confounded (CF) and non-confounded (NCF).

While these results underscore our framework's lag in pure classification accuracy, it is important to contextualise the numbers. Our model is not solely optimised for accuracy, but it is explicitly designed to support transparent reasoning chains aligned with human interpretability. However, our results for the non-confounded split for CLEVR-Hans7 show promise when compared with the metrics of the confounded split, displaying only a small drop in performance. While there is significant room for improvement in terms of robustness and generalisability, these findings suggest that the framework's structured reasoning is a suitable stepping stone for future improvements. In the next section, we look at the evaluation metrics relating to the interpretability of our framework in detail, in order to better inform our judgement.

5.2.2 Error Analysis

To understand the model's limitations, we examine the representative configurations for incorrect predictions and misalignments between players in the Consensus Game using confusion matrices. Specifically, we analyse Config C on CLEVR-Hans3 and CLEVR-Hans7, as this setup exhibited inconsistent results across the two datasets. For the diagrams, we denote abstaining predictions as -1.

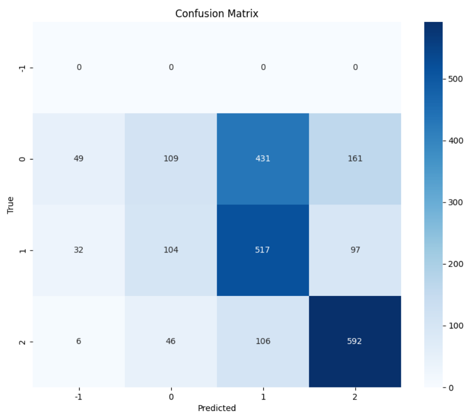


Figure 5.2: Confusion matrix for predictions made by Config C on the non-confounded split of CLEVR-Hans3.

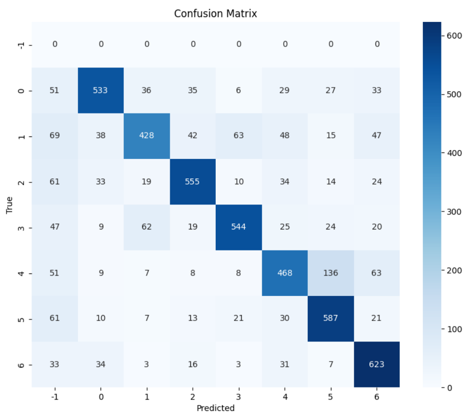


Figure 5.3: Confusion matrix for predictions made by Config C on the non-confounded split of CLEVR-Hans7.

Failure Mode and Overfitting

Based on the confusion matrix for CLEVR-Hans3, we can understand where the loss in predictive performance can be attributed to. Our model is predicting class 1 (Small metal cube and small (metal) sphere) very frequently for images labelled class 0 (Large (gray) cube and large cylinder). This confusion likely stems from overfitting, where the model picks up on the presence of common shapes, ignoring key distinguishing features such colour and size. In this case, shape features like cubes may be overweight, causing more nuanced features like size of material to be overlooked or conjuncted with the shape feature.

A contributing factor is the entangled slot representations, which group correlated features like size, colour and shape during training, making it hard for our framework to decouple those attributes when these correlations no longer hold. This entanglement makes it difficult for the players to reason over fine-grained details that are needed to perform well for the non-confounded test set. This behaviour reflects the model's tendency to overfit to prominent but irrelevant features rather than selecting holistically over the entire set of object in the scene.

A promising path to address this issue is to integrate neuro-symbolic techniques, such as the NeSy XIL framework [28], that explicitly factor object attributes into disentangled, symbolic concept representations. These symbolic concepts are then used in a concept learner to uncover the true class. Instead of having agents that coarsely reason over the slots, our framework could incorporate a concept extractor module or an equivalent that maps slots to symbolic rules. This shift would move our currently neural method into a more neuro-symbolic technique that still maintains its transparency and human-aligned reasoning chains. Arguments could have additional information that specify specific object attributes of the slots. This would ultimately improve both generalisation and transparency as the players would ground their arguments on both interpretable features and specific attributes.

Dataset Complexity

Notably, we do not observe the same level of confusion for CLEVR-Hans7. The classes for this dataset include all of the classes in CLEVR-Hans3 (classes 0, 1 and 6), but the model successfully distinguished between class 0 and 1, likely due to the difference in make up and difficulty of the datasets. One area of confusion, which was not seen in the confounded split, is the overlap between classes 4 and 5. This confusion is understandable as the labels share the "Three cylinders on the right" sub-rule. This behaviour suggests that the broader class distribution and more diverse visual scenes in CLEVR-Hans7 may play a role in helping to regularise training and reduce over-reliance on spurious features.

Reflections on Limitations

Despite our framework's interpretability, the predictive performance is ultimately hindered by the inherently transparent reasoning chain, where the slots selected and exposed to the user is exactly the slots used for classification. With a partial observation of the full image, the framework is bound to perform worse. For non-confounded cases, the performance is further inhibited by the reliance on entangled slot features that are very difficult to decouple with purely neural methods.

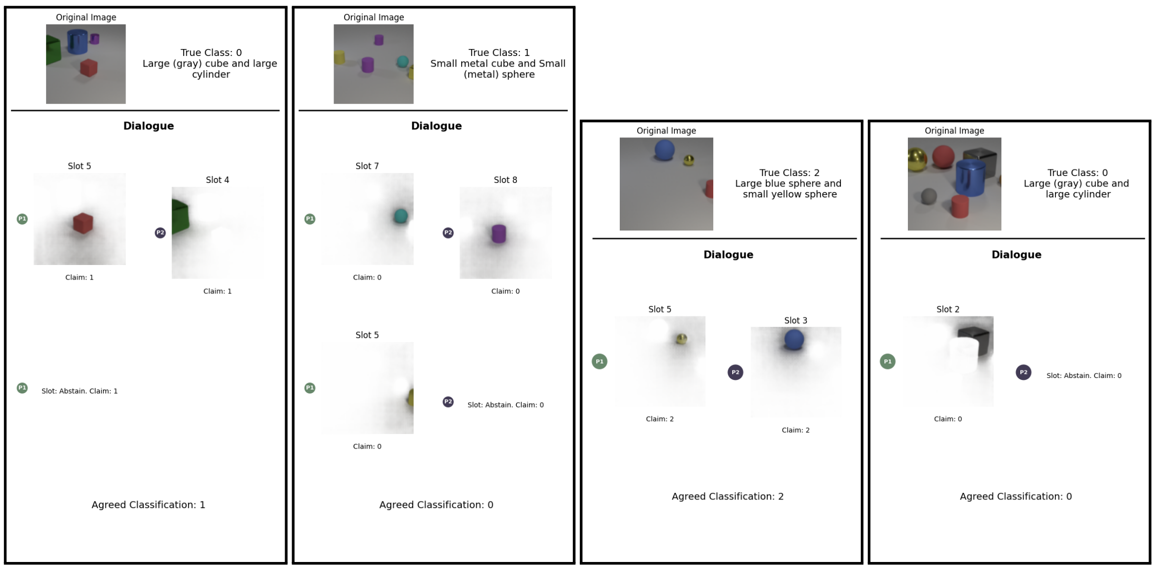
Although we focus our analysis on a single setup for clarity, similar patterns were observed across the other datasets and configurations, albeit with variation depending on dataset complexity and object diversity. Our analysis on the shortcomings of our framework provides a suitable direction for future work (6).

5.3 Reasoning and Consensus

Beyond prediction accuracy, our framework enables deeper analysis of the reasoning process through internal metrics that capture agent behaviours, slot content, and classifier outputs. These metrics offer insight into how efficiently and coherently the agents arrive at their predictions, and how informative their selections are in alignment with object-centric evidence.

5.3.1 Dialogue Examples

In this section, we show an example dialogue from the CLEVR-Hans3 dataset for each class label. To ensure that we do not show only handpicked results, we will fix the image id to 0, which is the first image for a given dataset split and class.



(a) Class 0, NCF (b) Class 1, NCF (c) Class 2, NCF (d) Class 0, CF

Figure 5.4: Dialogue visualisations for four sample classifications from Config C on CLEVR-Hans3, for both non-confounded (NCF) and confounded (CF) splits.

These non-confounded examples highlight the persistent confusion of our framework between classes 0 and 1 on the non-confounded split. For the dialogue of class 0, we can see the players immediately committed to class 1 when they spotted the small red cube, illustrating a shortcut bias and a class confusion. In contrast, for the example of class 1, although the shapes selected do not overlap with the objects for class 0, the players still misclassified the image. The dialogues also show a good example of the use of abstaining slot selections, when they already agree with the other player. More examples for the CLEVR-Hans7 and Multi-dSprites can be found in the appendix A.2.

Looking the confounded example, we can identify evidence of the grouped attribute overfitting where for the class 0 example, the players were very confident from just the single selection of the large gray cube, leading to shortcuts being taken as another object should be provided. This learned correlation would allow our model to perform well for the confounded case but very poorly for the non-confounded split.

5.3.2 Evaluation Metrics

As detailed above (4.4.2), we derived a comprehensive list of metrics that will enable us to quantitatively compare the quality of explanations from one of our framework's instances and another. Here, we present the results, discuss the significance of the results, and informally defining a heuristic for choosing the model that produces the best explanations.

Dataset/Configs →	**CLEVR-Hans7**			**CLEVR-Hans3**		
Metrics ↓	A	B	C	A	B	C
Consensus Rate	92.66%	85.78%	92.79%	97.10%	92.84%	96.33%
Accuracy	70.15%	59.17%	71.00%	52.58%	52.73%	55.27%
Avg. Game Length	3.41	2.71	3.41	2.46	2.98	2.36
Avg. Empty Slot %	4.89%	1.62%	1.50%	4.24%	2.76%	0.70%
Avg. Slot Uniqueness %	97.11%	98.47%	97.94%	99.41%	98.51%	99.51%

Table 5.6: Interpretability-related metrics for each model configuration across the CLEVR-Hans (non-confounded) datasets.

Dataset/Configs →	**Multi-dSprites**	
Metrics ↓	A	B
Consensus Rate	95.16%	79.32%
Accuracy	79.76%	63.38%
Avg. Game Length	2.86	1.99
Avg. Empty Slot %	21.61%	27.39%
Avg. Slot Uniqueness %	97.93%	99.29%

Table 5.7: Interpretability-related metrics for model configurations Config A and B for the Multi-dSprites dataset.

Across all datasets, we observe consistent trends across the measured interpretability metrics. Consensus rate and accuracy are the highest for Config C, suggesting more aligned and coherent reasoning between players. In terms of slot reconstruction loss, all configurations show comparable values, indicating that the E2E training regime was successful in maintaining the object-centricity (5.1). To ensure fair comparison of game length, we ensured to omit the final repeated selection in Config B, resulting in similar lengths across the configurations and datasets. For the other slot uniqueness and empty slot percentage, we observe consistent values across the board. However, there is an outlier in empty slot usage on Multi-dSprites, which is likely due to the lower number of objects relative to the number of slots.

From these observations, we find that agents that agree quickly and focus on informative, non-redundant slots is extremely desirable in the context of our framework. For an informal heuristic, we care about configurations that exhibit high consensus and accuracy, while maintaining low average game length, and high slot uniqueness and empty slot percentage. Config A and C best fit this profile, making them most effective for balancing predictive performance and explainability.

5.4 Survey Findings

Our survey results (5.8) reveal distinct differences in the interpretability and user satisfaction among the five explanation methods evaluated. The questions posed to the participants aim to qualitatively distinguish the various methods in terms of clarity, interpretability, bias/error detectability, relevancy and satisfaction.

Method	**Dataset**	**Acc (%)**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Avg Likert**
LIME	MdS CH7	92 100	4.4	4.0	4.2	3.6	4.0	3.8	4.0
Grad-CAM (Slot)	MdS CH7	85 81	3.2	3.4	3.2	3.4	3.2	3.1	3.2
OCEAN	MdS CH7	100 100	4.5	4.3	4.3	4.6	4.6	4.5	4.5
Gradient SHAP	MdS CH7	65 100	3.3	3.4	3.2	3.7	3.5	3.3	3.4
Grad-CAM	MdS CH7	73 96	4.1	4.0	3.7	3.7	3.9	3.7	3.8

Table 5.8: Survey Results Summary: Accuracy by datasets and shared interpretability scores per explanation method.

We do not place much weightage in the correctness of the prediction guesses in our analysis, as the ease of guessing the class, highly depends on the underlying input image. Some part of it has to do with the visibility of the input image. While our framework's exposed reasoning only contain slot images, all participants were able to correctly identify the corresponding predictions, suggesting that the participants understood the OCEAN dialogues

Each question in our survey which uses the Likert scale corresponds to a desirable trait of a good explanation mechanism (4.4.1). For each question, our framework leads in the average score, indicating that our framework provided good explanations for the input images. While the results are favourable, we should cautiously interpret their implications. In the context of synthetic, multi-object datasets, our framework performs well due to our object-centric encoder and sequential reasoning module, making our generations easier for users to interpret. As such, the generalisability of our framework to real-world image datasets should be investigated to determine whether the interpretability advantages of our framework persist for more complex or noisy images.

5.5 Experiments

Pipelined and End-to-End Training

Config/Datasets →	**Pipelined Config A**		
Metrics ↓	**CLEVR-Hans7**	**CLEVR-Hans3**	**Multi-dSprites**
Consensus Rate %	$90.68 \pm 3.46\%$	$82.48 \pm 2.69\%$	$90.10 \pm 1.17\%$
Accuracy %	$51.69 \pm 3.39\%$	$53.02 \pm 1.27\%$	$71.93 \pm 2.35\%$
Avg. Game Length	4.55 ± 0.12	3.15 ± 0.22	2.64 ± 0.08
Avg. Empty Slot %	$28.66 \pm 0.40\%$	$10.66 \pm 1.86\%$	$18.23 \pm 0.65\%$
Avg. Slot Uniqueness %	$72.38 \pm 0.45\%$	$97.42 \pm 2.15\%$	$95.81 \pm 1.26\%$

Table 5.9: Interpretability-related metrics for Config A on a pipeline training setup across the three datasets. The CLEVR-Hans datasets here are the non-confounded splits.

We observe that when our framework is trained in a pipelined fashion with the Slot Attention module frozen, the results are very poor. Hence, our decision to train our Consensus Game module in tandem with Slot Attention end-to-end is justified, exhibiting more frequent agreement between players, higher prediction accuracy and more diverse selections, while lowering game length and empty slot usage.

Partial Visibility of Slots Across Players

Configs → **Metrics ↓**	**PV**	**Config A**
Consensus Rate %	$74.63 \pm 1.81\%$	$92.66 \pm 0.47\%$
Accuracy %	$52.94 \pm 1.84\%$	$70.15 \pm 1.23\%$
Avg. Game Length	4.22 ± 0.13	3.41 ± 0.06
Avg. Empty Slot %	$24.32 \pm 0.93\%$	$4.89 \pm 0.54\%$
Avg. Slot Uniqueness %	$94.85 \pm 2.53\%$	$97.11 \pm 0.38\%$

Table 5.10: Interpretability-related metrics comparing the Partial Visibility experiment and the E2E training setup for Config A on the non-confounded CLEVR-Hans7 dataset split.

In this experiment, we set out to determine whether players could still converge to correct claims, even if they were given segmented information. We expected they would be able to do so, albeit with a higher average game length. However, we did not observe that, as the players tended to not agree on the same claim. This may be due to the additional noise and unfamiliarity from the other player's selections.

Different Number of Players

Num Players → **Metrics ↓**	**1**	**2**	**3**
Consensus Rate %	$100.00 \pm 0.00\%$	$92.66 \pm 0.47\%$	$98.43 \pm 0.28\%$
Accuracy %	$67.62 \pm 1.30\%$	$70.15 \pm 1.23\%$	$68.46 \pm 0.44\%$
Avg. Game Length	5.07 ± 0.11	3.41 ± 0.06	4.11 ± 0.05
Avg. Empty Slot %	$10.88 \pm 0.39\%$	$4.89 \pm 0.54\%$	$5.92 \pm 0.47\%$
Avg. Slot Uniqueness %	$80.58 \pm 0.10\%$	$97.11 \pm 0.38\%$	$85.19 \pm 0.12\%$

Table 5.11: Interpretability-related metrics comparing different number of player setups for Config A on the non-confounded CLEVR-Hans7 dataset split, training end-to-end.

Although our framework was built to enable reasoning with any number of players, our initial design was for a two-player game. That sentiment is reflected in our results (5.11), where the default configuration of two players performs slightly better than the single-player and three-player setups. One possible reason for this is that our architecture hyperparameters were tuned for the two-player setup, and were reused in this experiment. Results for individually tuned hyperparameters may show that the single-player system would outperform other configurations given that consensus and information propagation does not play a factor.

Tuning Confidence Threshold

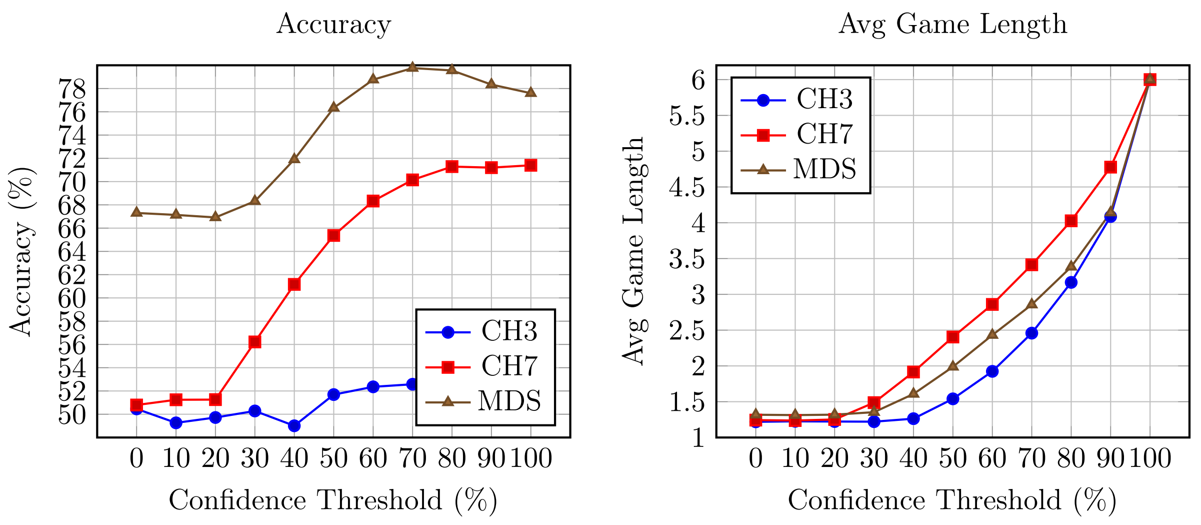


Figure 5.5: Accuracy and Average Game Length vs. Confidence Threshold for Config A across three datasets, with non-confounded splits for CLEVR-Hans.

Across the three datasets, the trends for both accuracy rate and average game length (number of slots) are consistent, showing the obvious correlation between quantity of information and prediction performance. To strike a balance between conciseness of explanations and accuracy of predictions, setting a confidence threshold between 50% to 80% appears reasonable.

Chapter 6

Future Work

While the current OCEAN framework demonstrates the potential of combining object-centric representations with interactive decision-making mechanisms like the Consensus Game, there are several promising directions for future research and development. Some of these, in part, help to overcome the limitations of the framework:

- The original implementation of Slot Attention performs sub-optimally for real-world and high-resolution images, leading to our framework lacking in scalability.
- Prediction accuracy of the Consensus Game module is dependent on the alignment of all players' claims. Thus, alternate game dynamics or varied player strategies that do not intentionally align players' claims may reduce prediction accuracy significantly.
- Our framework is prone to overfitting on confounded information, leading to reasoning relies on spurious correlations or incomplete arguments, limiting its reliability in out-of-distribution scenarios.

6.1 Alternative Encoders and Real-World Image Evaluation

Since our framework uses Slot Attention for object-centric learning, it faces limitations in scalability, particularly on high-resolution or real-world scenes. Future work could explore other Slot Attention variants or alternatively, other object-centric methods, such as MONet [17], IODINE [18] and SPACE [19]. One strong replacement is the more scalable DINOSAUR [33] model, which improves disentanglement and robustness for real-world datasets.

Building on this, applying the OCEAN framework to real-world datasets such as COCO [34] or AFHQ [32] could test its interpretability and reasoning in more complex settings. A suitable benchmark is the ProtoPNet [21] framework, as the goals on human-aligned reasoning closely aligns with the interpretability goals of our project. Evaluation would focus on classification accuracy and whether the agents maintain coherent, interpretable reasoning chains across more visually complex environments.

6.2 Role Differentiation and Flexible Game Dynamics

While all players currently serve the same role in the reasoning process, future work could introduce asymmetry between the players to encourage more diverse and meaningful interactions. Although introducing a more diverse set of actions may create richer dialogue, some measures would be needed in place to prevent fallback on the strategies observed in our framework. For instance, players could be assigned distinct roles such as proposing, verifying, or challenging claims. In a two-player setup, we could look towards the zero-sum nature of Visual Debates [7], or from role specialisation where players focus on specific object types or regions. Additionally, relaxing the requirement for strict consensus may allow agents to express uncertainty or partial agreement, creating richer game dynamics and enabling more robust decision-making in ambiguous or noisy scenes.

6.3 Mitigating Overfitting to Spurious Correlations

One limitation of our current approach is its vulnerability to overfitting on overly-specific, incorrect correlation, as shown by the results with the non-confounded split of CLEVR-Hans3. Specifically, in the instance of CLEVR-Hans3, the players were overfitting on the conjunctions of features which satisfy sub-rules of the class labels (e.g., large (gray) cube), even when only one attribute (e.g., large) was predictive in the non-confounded setting, hurting their generalisability to more diverse class rules. This may stem from entangled slot representations, where multiple visual factors are compressed into a single vector. To mitigate this, one promising approach is to factorise slot features using a neuro-symbolic concept learner, as seen in NeSy XIL [28], learning fine-grained object attributes per slot. A slightly different approach may be inspired from concept bottleneck models [35], with strong applicability to real-world images. Ultimately, addressing this issue is essential for extending the trustworthiness and applicability of our framework to more realistic settings as intended above.

Chapter 7

Conclusion

This project set out to explore whether collaborative, object-centric reasoning could enable more faithful and interpretable visual classification. Drawing from structured argumentation and "conversational" explanations in Visual Debates, we designed a transparent classification system, exposing its reasoning as a deliberation between agents. Throughout the project, several key findings emerged, along with practical and conceptual hurdles that shaped the final framework.

7.1 Challenges and Successes

Designing a Multi-Agent Classification Game

The key investigation was to explore whether cooperative agentic interactions could yield a more transparent and human-aligned visual classification process. While the initial idea of simulating classification through agent interactions was attractive, making it work in practice would require delicate reward shaping and control over agent behaviours. Our first implementation using sparse rewards and weak consensus constraints led to uninformative selections and unstable training. Looking back, we should have pivoted and experimented earlier with denser reward models and flexible, tunable termination conditions. These changes led to more coherent reasoning, better selections, and ultimately higher accuracy. A lesson learned here is to focus on the basic implementation, allowing for rapid iteration and experimentation with different approaches to building the framework. Due to that realisation, the resulting framework not only yielded competitive predictions, but also transparently showed the reasoning process behind them.

Building an End-to-End Learning Framework

Our first interpretation of the term "end-to-end" meant "all components learning asynchronously", leading to more of a pipeline where the Slot Attention and Consensus Game modules learned with their independent loss values. Only upon noticing the mistake and experimenting with a pure flow of gradients from the downstream Consensus Game task to the Slot Attention module, did we understood the effect of specialised gradient signals on a differentiable feature extractor. Further iterations of our E2E learning framework brought out our current EM-like implementation, where we observed significant improvements in slot information, agent selection strategies, and classification accuracy.

Evaluating the Framework

In building an explainable classification framework, we were aware of the evaluation challenges that were in our way. Fortunately, with experiments with a toy multi-agent Tic-Tac-Toe example, we had developed suitable architecture for both training and evaluation of multi-agent games.

Early on, we were able to track various kinds of metrics from our framework during training. However, designing appropriate metrics and benchmarks for interpretability was not straightforward. We supplemented standard classification benchmarks with interpretability-focused metrics, and conducted a user survey for assessing our framework against baselines. With the results obtained, we are confident in our findings and that we met our initial research goals.

7.2 Final Contributions

Through this work, we gained insight into the delicate tradeoffs between performance, transparency, and interpretability. Faithful explanations require an architecture where they can be derived naturally from the decision-making process. Our findings show that multi-agent interaction, when structured carefully, provides a promising scaffold for such reasoning.

We foresee exciting directions for future work in this domain. During our project, we observed the struggles of Slot Attention with scalability and generalisation, the complexity of building inherently explainable systems, and the difficulty of consistently and rigorously evaluating model explainability. One particular important challenge was the strong overfitting in confounded scenes in CLEVR-Hans3, highlighting the need for careful architectural design and better information disentanglement. Collectively these challenges point toward the importance of future research into more scalable object-centric models, principled design for interpretable systems, and robust evaluation frameworks for explainability.

Finally, we learned that explanations are not merely outputs to be generated after the fact, but processes that should be embedded into opaque systems to ensure transparency and trust. This shift in perspective is shared across the industry with broader implications for AI trustworthiness, particularly in high-stakes domains like medical diagnosis or autonomous decision-making. As a result, we are heading straight on to a world where understanding *why* a model makes a decision is just as critical as the decision itself.

Bibliography

1. Nasser M and Yusof UK. Deep learning based methods for breast cancer diagnosis: a systematic review and future direction. Diagnostics 2023; 13:161
2. Nguyen TT, Tahir H, Abdelrazek M, and Babar A. Deep learning methods for credit card fraud detection. arXiv preprint arXiv:2012.03754 2020
3. Badue C, Guidolini R, Carneiro RV, Azevedo P, Cardoso VB, Forechi A, Jesus L, Berriel R, Paixao TM, Mutz F, et al. Self-driving cars: A survey. Expert systems with applications 2021; 165:113816
4. Castelvecchi D. Can we open the black box of AI? Nature News 2016; 538:20
5. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, and Batra D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision.* 2017 :618–26
6. Ribeiro MT, Singh S, and Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016 :1135–44. DOI: 10.1145/2939672.2939778. Available from: https://doi.org/10.1145/2939672.2939778
7. Kori A, Glocker B, and Toni F. Explaining Image Classification with Visual Debates. 2023. arXiv: 2210.09015 [cs.CV]. Available from: https://arxiv.org/abs/2210.09015
8. Maveli N. Demystifying Post-hoc Explainability for ML models. 2021. Available from: https://spectra.mathpix.com/article/2021.09.00007/demystify-post-hoc-explainability
9. Sarkar A, Vijaykeerthy D, Sarkar A, and Balasubramanian VN. A Framework for Learning Ante-hoc Explainable Models via Concepts. 2021. arXiv: 2108.11761 [cs.LG]. Available from: https://arxiv.org/abs/2108.11761
10. Jacob AP, Shen Y, Farina G, and Andreas J. The Consensus Game: Language Model Generation via Equilibrium Search. 2023. arXiv: 2310.09139 [cs.GT]. Available from: https://arxiv.org/abs/2310.09139
11. Oord A van den, Vinyals O, and k kavukcuoglu koray. Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems.* Ed. by Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, and Garnett R. Vol. 30. Curran Associates, Inc., 2017. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf
12. Locatello F, Weissenborn D, Unterthiner T, Mahendran A, Heigold G, Uszkoreit J, Dosovitskiy A, and Kipf T. Object-centric learning with slot attention. Advances in neural information processing systems 2020; 33:11525–38
13. He K, Zhang X, Ren S, and Sun J. Deep Residual Learning for Image Recognition. 2015. arXiv: 1512.03385 [cs.CV]. Available from: https://arxiv.org/abs/1512.03385

14. Irving G, Christiano P, and Amodei D. AI safety via debate. 2018. arXiv: 1805.00899 [stat.ML]. Available from: https://arxiv.org/abs/1805.00899

15. Nash J. Non-Cooperative Games. ANNALS OF MATHEMATICS 1951; 54

16. Mnih V, Heess N, Graves A, and Kavukcuoglu K. Recurrent Models of Visual Attention. *Advances in Neural Information Processing Systems*. Ed. by Ghahramani Z, Welling M, Cortes C, Lawrence N, and Weinberger K. Vol. 27. Curran Associates, Inc., 2014. Available from: https://proceedings.neurips.cc/paper_files/paper/2014/file/3e456b31302cf8210edd4029292a40ad-Paper.pdf

17. Seong H and Shim DH. Self-supervised interpretable end-to-end learning via latent functional modularity. *Proceedings of the 41st International Conference on Machine Learning*. ICML'24. Vienna, Austria: JMLR.org, 2024

18. Greff K, Kaufman RL, Kabra R, Watters N, Burgess C, Zoran D, Matthey L, Botvinick M, and Lerchner A. Multi-object representation learning with iterative variational inference. *International conference on machine learning*. PMLR. 2019 :2424–33

19. Lin Z, Wu YF, Peri SV, Sun W, Singh G, Deng F, Jiang J, and Ahn S. SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition. 2020. arXiv: 2001.02407 [cs.LG]. Available from: https://arxiv.org/abs/2001.02407

20. Jacob AP, Wu DJ, Farina G, Lerer A, Hu H, Bakhtin A, Andreas J, and Brown N. Modeling Strong and Human-Like Gameplay with KL-Regularized Search. *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, and Sabato S. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022 :9695–728. Available from: https://proceedings.mlr.press/v162/jacob22a.html

21. Chen C, Li O, Tao D, Barnett A, Rudin C, and Su JK. This Looks Like That: Deep Learning for Interpretable Image Recognition. *Advances in Neural Information Processing Systems*. Ed. by Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, and Garnett R. Vol. 32. Curran Associates, Inc., 2019. Available from: https://proceedings.neurips.cc/paper_files/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf

22. Alvarez Melis D and Jaakkola T. Towards Robust Interpretability with Self-Explaining Neural Networks. *Advances in Neural Information Processing Systems*. Ed. by Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, and Garnett R. Vol. 31. Curran Associates, Inc., 2018. Available from: https://proceedings.neurips.cc/paper_files/paper/2018/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf

23. Captum. Captum: A unified and generic model interpretability library for PyTorch. https://github.com/pytorch/captum. Accessed: 2025-05-24. 2020

24. Sundararajan M, Taly A, and Yan Q. Axiomatic Attribution for Deep Networks. *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Precup D and Teh YW. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017 Jun :3319–28. Available from: https://proceedings.mlr.press/v70/sundararajan17a.html

25. Lundberg SM and Lee SI. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*. Ed. by Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, and Garnett R. Vol. 30. Curran Associates, Inc., 2017. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

26. Vaswani A. Attention is all you need. Advances in Neural Information Processing Systems 2017

27. Moon T. The expectation-maximization algorithm. IEEE Signal Processing Magazine 1996; 13:47–60. DOI: 10.1109/79.543975

28. Stammer W, Schramowski P, and Kersting K. Right for the Right Concept: Revising Neuro-Symbolic Concepts by Interacting with their Explanations. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2021 :3618–28. DOI: 10.1109/CVPR46437.2021.00362

29. Johnson J, Hariharan B, Maaten L van der, Fei-Fei L, Zitnick CL, and Girshick R. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017 :1988–97. DOI: 10.1109/CVPR.2017.215

30. Kabra R, Burgess C, Matthey L, Kaufman RL, Greff K, Reynolds M, and Lerchner A. Multi-Object Datasets. https://github.com/deepmind/multi-object-datasets/. 2019

31. Matthey L, Higgins I, Hassabis D, and Lerchner A. dSprites: Disentanglement testing Sprites dataset. https://github.com/deepmind/dsprites-dataset/. 2017

32. Choi Y, Uh Y, Yoo J, and Ha JW. StarGAN v2: Diverse Image Synthesis for Multiple Domains. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2020 :8185–94. DOI: 10.1109/CVPR42600.2020.00821

33. Seitzer M, Horn M, Zadaianchuk A, Zietlow D, Xiao T, Simon-Gabriel CJ, He T, Zhang Z, Schölkopf B, Brox T, and Locatello F. Bridging the Gap to Real-World Object-Centric Learning. 2023. arXiv: 2209.14860 [cs.CV]. Available from: https://arxiv.org/abs/2209.14860

34. Lin TY, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, and Dollár P. Microsoft COCO: Common Objects in Context. 2015. arXiv: 1405.0312 [cs.CV]. Available from: https://arxiv.org/abs/1405.0312

35. Koh PW, Nguyen T, Tang YS, Mussmann S, Pierson E, Kim B, and Liang P. Concept Bottleneck Models. *Proceedings of the 37th International Conference on Machine Learning.* Ed. by III HD and Singh A. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020 :5338–48. Available from: https://proceedings.mlr.press/v119/koh20a.html

Declarations

Use of Generative AI

I acknowledge the use of ChatGPT 4 (OpenAI, https://chat.openai.com/) to generate text outlines and LaTeX layouts for various sections and components in this report, including tables, figures and stylistic formatting. I confirm that no (semantic) content generated by Generative AI has been presented as my own work.

Ethical Considerations

This project involved use of synthetic datasets and did not collect or process any human, personal, or sensitive data. A small-scale survey with human participants was conducted to evaluate explanation quality, involving voluntary, anonymous participants. Ethical risks were assessed in line with the department's guidance, and appropriate measures were taken to ensure informed consent and data privacy.

Sustainability

To minimise environmental impact, the key effort was to conduct training and experimentation using resource-conscious practices. Model training was primarily conducted on shared GPU infrastructure, ensuring low log update frequency. The training and experimentation processes were designed in a way that any new additions or newly-tuned hyperparameters could be tested by resuming previously saved checkpoints, reducing redundant computation.

Availability of Data and Materials

All datasets used in the project are publicly available synthetic datasets, including CLEVR-Hans3, CLEVR-Hans7, and Multi-dSprites. For Multi-dSprites, we augmented the dataset with binary class labels using a Python script, which can be found on the repository. All source code and scripts are available in the following GitLab repository: https://gitlab.doc.ic.ac.uk/bt221/ocean. Instructions for training or using the models can be found in the README.

Appendix A

Additional Experimental Results

A.1 Detailed Baseline Prediction Results

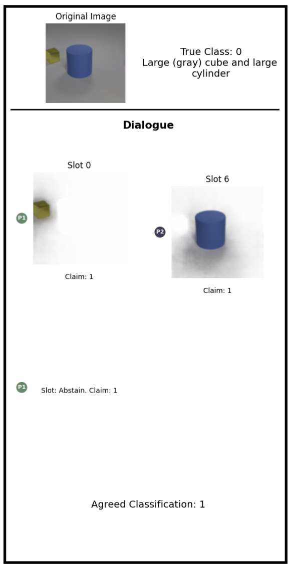
Configs →	**ResNet** (224 × 224)			
Dataset ↓	**Acc**	**Pre**	**Rec**	**F1**
CLEVR-Hans7 (NCF)	83.54±0.09%	83.65±0.29%	83.66±0.17%	83.65±0.08%
CLEVR-Hans3 (NCF)	60.93±0.58%	59.61±0.38%	60.93±0.58%	56.77±1.56%
Multi-dSprites	83.20±0.14%	82.94±0.45%	83.21±0.40%	83.00±0.40%

Table A.1: Detailed prediction metrics for the ResNet18 baseline.

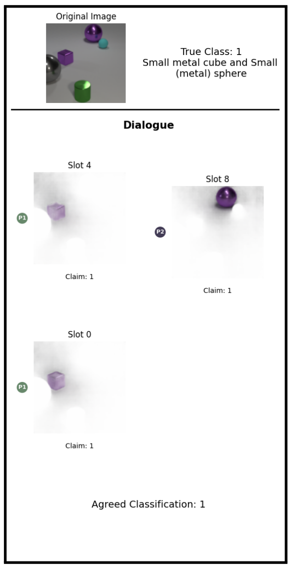
Configs →	**Slot MLP** (64 × 64)				**Slot MLP** (128 × 128)			
Dataset ↓	**Acc**	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**
CLEVR-Hans7 (NCF)	86.66± 0.07%	86.68± 0.07%	86.66± 0.07%	86.65± 0.06%	86.39± 0.19%	86.41± 0.19%	86.39± 0.19%	86.33± 0.25%
CLEVR-Hans3 (NCF)	85.60± 0.18%	85.59± 0.68%	85.94± 0.17%	85.57± 0.17%	88.76± 0.25%	88.85± 0.23%	88.76± 0.25%	88.74± 0.25%
Multi-dSprites	92.72± 0.14%	92.84± 0.26%	92.72± 0.14%	92.70± 0.26%	—	—	—	—

Table A.2: Detailed prediction metrics for the Slot MLP baseline.

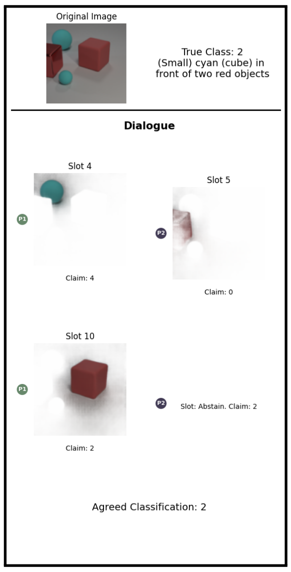
A.2 Example Generated Dialogues



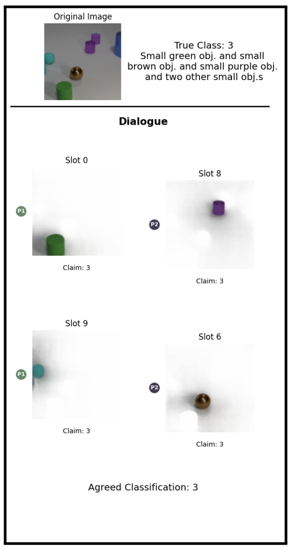
(a) Class 0



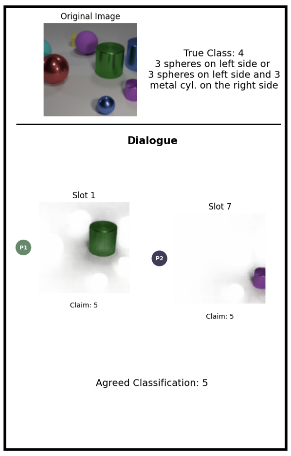
(b) Class 1



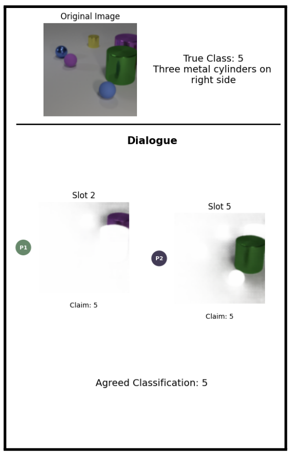
(c) Class 2



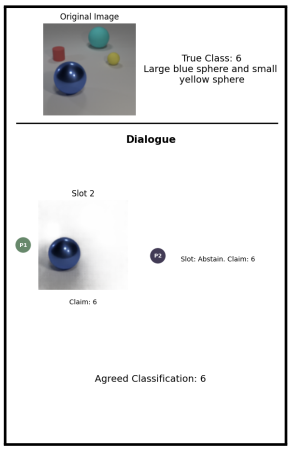
(d) Class 3



(e) Class 4

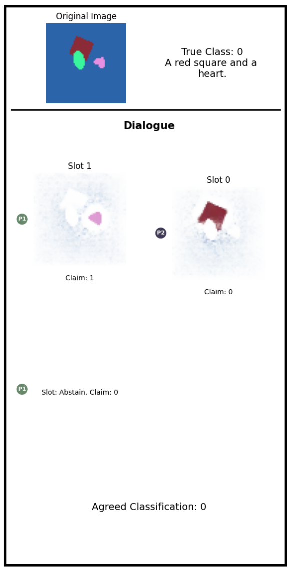


(f) Class 5

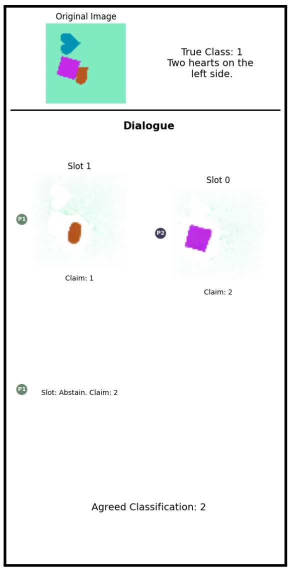


(g) Class 6

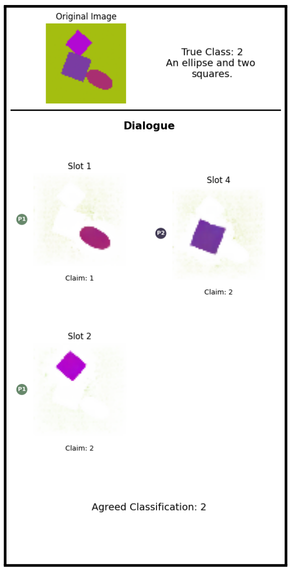
Figure A.1: Example dialogues generated from our framework for each class of the non-confounded split of the CLEVR-Hans7 dataset (image ID: 000000).



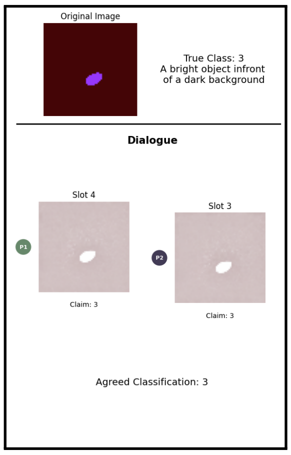
(a) Class 0



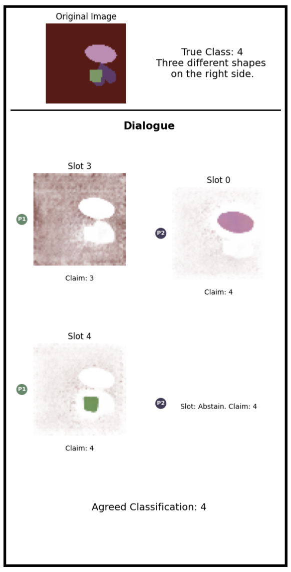
(b) Class 1



(c) Class 2



(d) Class 3



(e) Class 4

Figure A.2: Example dialogues generated from our framework for each class of the test split of the Multi-dSprites dataset (image ID: `0000`).

Appendix B

Explanation Quality Survey

This is an abstraction of the survey conducted using Microsoft Forms. The form can be accessed with this hyperlink, `https://forms.office.com/e/zW5FUjEtrP`. The following shows the various sections in the survey adapted for the report. As an example, we show the entire line of questioning for first explanation method, LIME. The same questions will repeat for the other explanation methods.

Some Background

In machine learning, explainable AI (XAI) techniques help humans understand why a model made a certain decision. In this survey, you'll be shown **explanations generated by different methods** for various images. **An explanation could be a region highlight, heatmap, or a selection of features of the image.**

Your task is to **guess which class (0,1,2...)** the explanation was meant to support, **without seeing the original prediction or the original image**. This helps us assess how understandable and intuitive each explanation method is.

For example, if an explanation highlights the shape of wings and a tail, you might infer that the model predicted "airplane."

You'll compare 5 different methods for two different images from different datasets. This survey should take no more than 5 minutes.

Datasets

The Multi-DSprites Dataset comprises of a coloured background, and up to 4 coloured "sprites" or shapes in the image. There are 5 different classes, but we will only be focusing on a few for this survey.

These are some examples of classes.

- Class 0: A red square and a heart.
- Class 1: Two hearts on the left side.
- Class 3: A bright coloured object in front of a dark background.

The image on the left satisfies class 0 with the red square and the green heart.

The CLEVR-Hans7 Dataset is a dataset that comprises of images that depict scenes with varying coloured and sized 3D shapes. The 7 represents 7 different classes, but we will only be focusing on a few for this survey.

These are some examples of classes.

- Class 0: Large cube and Large Cylinder
- Class 4: 3 Spheres on the left side OR 3 Spheres on the left side and 3 Cylinders on the right side
- Class 5: 3 Cylinders on the right side

The image on the right satisfies class 0, with the yellow cube and blue cylinder.

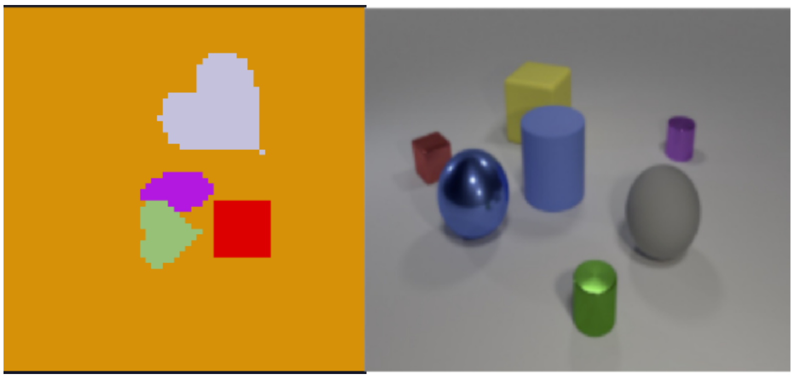


Figure B.1: Example images from the Multi-dSprites and CLEVR-Hans7 datasets.

B.1 Explanation Method 1

Your task is to guess which class (0,1,2...) the explanation was meant to support, without seeing the original prediction or original image. This helps us assess how understandable and intuitive each explanation method is.

B.1.1 Multi-dSprites Explanation

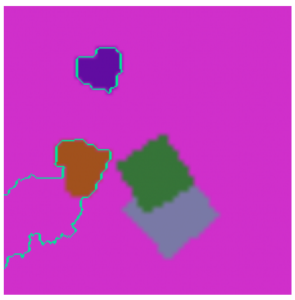


Figure B.2: A Multi-dSprites Explanation.

- ○ Class 0: A red square and a heart.
- ○ Class 1: Two hearts on the left side.
- ○ Class 3: A bright object infront of a dark background.

B.1.2 CLEVR-Hans Explanation

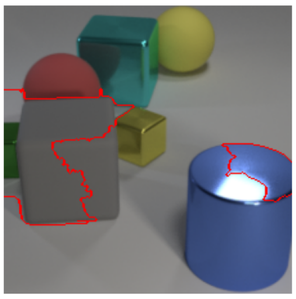


Figure B.3: A CLEVR-Hans Explanation.

- Class 0: Large cube and Large Cylinder
- Class 4: 3 Spheres on the left side OR 3 Spheres on the left side and 3 Cylinders on the right side
- Class 5: 3 Cylinders on the right side

B.2 Explanation Method 1 Quality

This question is to measure the quality of the explanation method you saw previously.

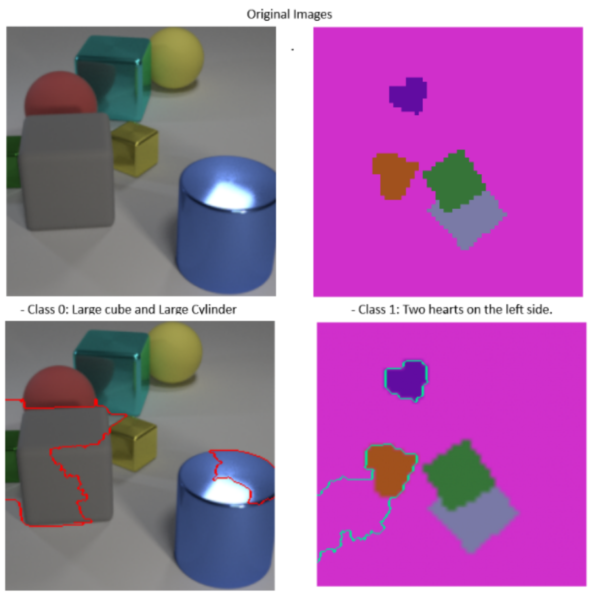

Original Images
- Class 0: Large cube and Large Cylinder
- Class 1: Two hearts on the left side.

Figure B.4: Summary image of the original images and their corresponding explanations for explanation method 1.

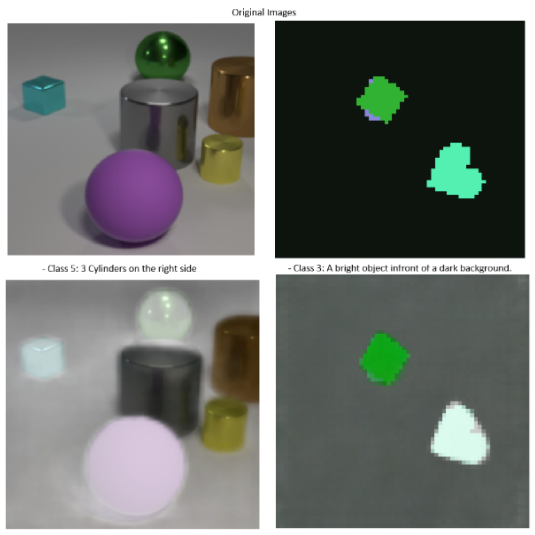
Based on your interpretation of the explanations, please provide an answer to the following statements about the explanations for the different images. (1) Strongly Disagree, (2) Disagree, (3) No preference, (4) Agree, and (5) Strongly Agree.

- It is clear which parts of the image is the explanation. ☐
- The explanation was easy to interpret without prior technical knowledge. ☐

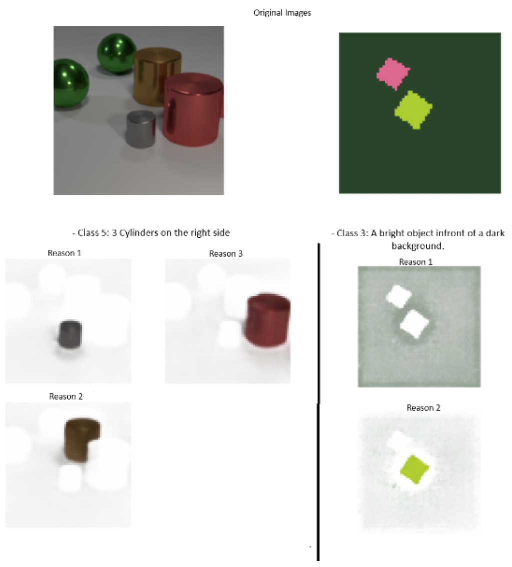
- The explanation would help me identify errors or biases in the model's prediction. ☐
- The explanation only focused on relevant parts/objects of the image. ☐
- The explanation helped me understand why the model predicted the class. ☐
- I am satisfied with the quality of the explanation. ☐

B.3 Other Sections

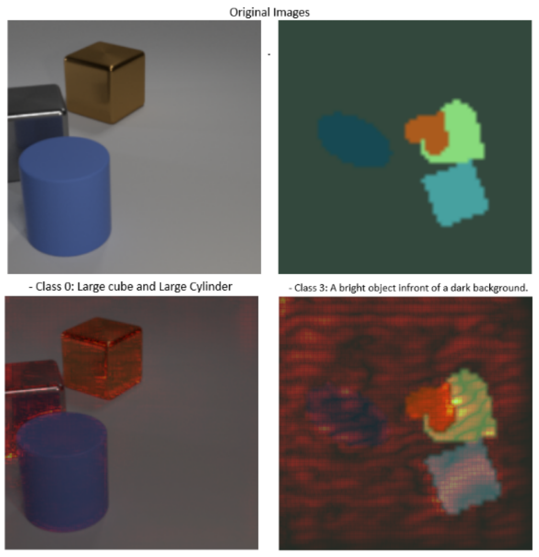
The survey continues for four other explanation methods as seen below.


Original Images
- Class 5: 3 Cylinders on the right side
- Class 3: A bright object infront of a dark background.

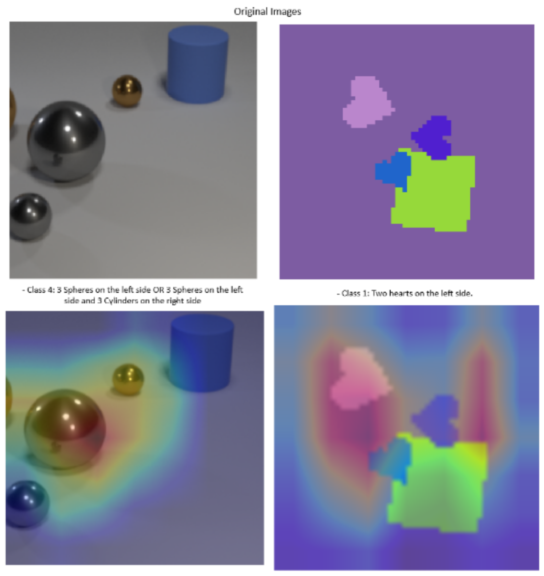
(a) Explanation Method 2: Grad-CAM for Slot Attention


Original Images
- Class 5: 3 Cylinders on the right side
Reason 1
Reason 3
Reason 2
- Class 3: A bright object infront of a dark background.
Reason 1
Reason 2

(b) Explanation Method 3: Object-Centric Visual Consensus


Original Images
- Class 0: Large cube and Large Cylinder
- Class 3: A bright object infront of a dark background.

(c) Explanation Method 4: Gradient SHAP


Original Images
- Class 4: 3 Spheres on the left side OR 3 Spheres on the left side and 3 Cylinders on the right side
- Class 1: Two hearts on the left side.

(d) Explanation Method 5: Grad-CAM

Figure B.5: Summary images of the original images and their corresponding explanations for explanation methods 1 to 5.