Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

HACR:
Hybrid Architecture for Concept Reasoning

Author:
Kexin Gu

Supervisor:
Prof. Alessandra Russo

Second Marker:
Dr. Krysia Broda

Abstract

The most basic way for humans to learn concepts is with our eyes. Simply by observing our surroundings, we can understand complex scenarios. Then, when we see something similar happen later, we can recognise the common concepts and reason about them with our new knowledge. Such a process requires high-level cognition, including visual perception, concept formation, and reasoning. In recent years, numerous machine learning approaches have focused on learning from complex or multi-modality data to model human-level cognition. In particular, there has been a rising wave of neural-symbolic models that incorporate symbolic representations into the architectures.

In this project, we develop a hybrid concept-learning pipeline called Hybrid Architecture for Concept Reasoning, or HACR for short. HACR is designed to learn from a challenging Video Q&A dataset, TVQA+ dataset. We construct human-interpretable symbolic representations by using pre-trained neural models to extract features from video frames and text. A state-of-the-art symbolic learning system, ILASP, uses these feature representations to search for general rules representing the concepts. Our evaluation shows that HACR is able to learn robust and general rules to represent complex real-life concepts. Furthermore, based on the learned rules, our pipeline can reason about concepts with high accuracy and transparency, providing the feature extraction is sufficiently accurate. This project paves the way for greater use of symbolic learning in neural-symbolic models in the future.

Acknowledgements

I would like to express my gratitude to Prof. Alessandra Russo for all the precious time, advice, insights, and help given to me throughout the project. It was a great pleasure to be under your guidance, and I am looking forward to continuing working with you in my PhD study.

I would like to thank Nuri Cingillioglu, Daniel Pace and Mark Law for all the inspiring suggestions and discussions during our meetings and emails. Your support has helped me tackle several technical issues and understand interesting experiment results throughout the project.

I would like to thank my personal tutor, Dalal Alrajeh, who has guided me and supported me throughout my study at Imperial College London. Your advice and help have encouraged me to go through all sorts of obstacles and helped me to decide the next step after this degree.

I would like to thank my fiancé, Matthew Baugh, for all your company, love and care when I am far away from home on my own. Your innovative ideas and suggestions on computer vision have helped to improve the visual feature extraction in this project.

I would like to thank all my friends and coursemates. Thank you for all the support and joy that made it an amazing four years of study. I wish everyone all the best with your next step after university.

Finally, I would like to dedicate this project to my parents, Tao Gu and Jun Zhao. You have been incredibly encouraging about my research passion and supported me throughout my study in London, even from a country that is thousands of miles away.

Contents

Chapter 1

Introduction

The process of concept learning for humans involves observing examples and categorising them based on their different attributes [1]. From just a few examples, humans can learn concepts and then apply them to recognise objects and events, while also providing justification and explanation. Although massive progress has been made in statistical machine learning in the past decades, these models still struggle with data hungriness and lack of interpretability.

This project develops a neural-symbolic architecture that learns visual concepts, specifically the concepts of 'holding' and 'entering scene', through video frames and question-answer pairs. We integrate symbolic inductive learning with neural networks to perform descriptive rule-based learning on spatial-temporal properties.

1.1 Motivations

Concept learning in complex environments is challenging for agents. They are required to have strong generalisation, which is a necessary step towards the next generation of robust AI [2]. Images and videos are both common and feature-rich mediums with concepts in various forms and complexity level. Visual Question Answering (VQA) and Video Question Answering (Video Q&A) are two benchmark tasks of complex learning that utilise these mediums and have been widely studied in recent years. Both tasks fuse multiple AI-complete tasks: natural language understanding, vision understanding, and knowledge reasoning [3]. However, Video Q&A is considerably more challenging than VQA due to the extra dimension of time (see Fig 1.1 as an example).

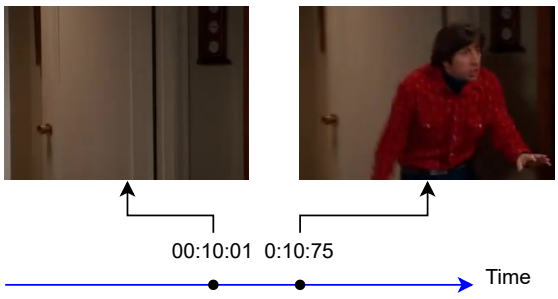


Figure 1.1: Concept of 'someone entering the scene'. Based on only one frame it's impossible to know if anyone enters the scene. (Images from The Big Bang Theory.)

Most approaches to these tasks are purely neural systems and incorporate various novel learning mechanism such as attention. Although they can achieve high levels of performance, it is extremely difficult to interpret both the internal representations and the reasoning process. In recent years, some newly-emerged research on hybrid systems has

shown exceptional results on improving transparency while maintaining high performance. The novel works include hybrid concept learners for VQA [4, 5], and the hybrid Video Q&A architectures [6, 7]. Compared to pure statistical methods, these systems encode concept representations as well as visual and textual features into functional programs, and then execute the programs to obtain an answer. Yet, concept learning is still performed with neural algorithms, which still lacks transparency. On the other hand, [8] has incorporated an efficient symbolic paradigm, Answer Set Programming [9], to replace synthetic programs. This system is also equipped with background knowledge to model physics and time, providing robust reasoning and high readability. However, the system does require human engineering to encode concepts such as physical laws.

We believe that the next step for the neural-symbolic research would be to utilise symbolic Inductive Learning systems to perform automated rule learning. Apart from symbolic learning, the pipeline would also use symbolic reasoning paradigm ASP for sound reasoning. Such neural-symbolic approaches would benefit from the accuracy of statistical learning, as well as the robustness and high transparency of symbolic learning and reasoning. These systems can model the high-level cognitive functions in full better than classic neural approaches [10]. Moreover, their ability to consolidate neural and symbolic learning would be a significant step towards artificial general intelligence (AGI) [11].

1.2 Objectives

This project aims to improve existing frameworks of neural-symbolic pipeline by using both symbolic learning and symbolic reasoning paradigm. Our approach explores the capability of such a hybrid pipeline in learning concepts through Video Q&A, primarily targeting TVQA+ dataset [12]. The following objectives are the key steps to achieve a concept-learning pipeline that can answer questions based on video clips.

Objective 1: Extract features and represent them in Answer Set Programming [9]. Our neural-symbolic approach should be able to handle unstructured data from the video frames and question-answer pairs and encode them in Answer Set Programming (ASP) syntax.

Objective 2: Learn general symbolic rules that help with question answering. During training time, our pipeline would encode the ground truth visual and textual features, and use them to create examples for an inductive learner. Our pipeline will make use of the state-of-the-art ASP programs learner ILASP[13], to learn rules that capture real-life concepts the best.

Objective 3: Develop an integrated pipeline for learning and reasoning, and evaluate it. At inference time, extracted knowledge from frames and texts should be combined with learnt rules from ILASP to form a reasoning model. The model's answer set(s) will include the predicted answer(s) for the question. Each learned concept will be a general first-order rule. Its performance will be evaluated by measuring how close the predicted answer is compared to the ground truth answer.

1.3 Challenges

The two main challenges introduced by using the TVQA+ dataset [12] for our hybrid concept learning task are:

- **Video Q&A on real-life videos with editing**: Video clips in the TVQA+ dataset are selected from the TV series *The Big Bang Theory*. Compared to the synthetic and angle-fixed videos in the CLEVRER dataset [6], the video clips in TVQA+ are full of real-life objects, events and concepts. Cinematic techniques are also applied

when producing these videos. Apart from finding the relevant objects and characters in a single frame, we also need to handle camera angle changes between frames.

- **Character identification**: All the questions refer to specific characters in the video clips, so we cannot effectively answer the questions without first identifying the people. Compared to the CLEVRER dataset, which only distinguishes physical objects such as sphere and cylinder, humans have more complex feature. Moreover, the appearance of humans when viewed from different camera angles varies much more than that of simple 3D objects.

1.4 Contributions

The main contribution of this project to neural-symbolic research is a hybrid concept-learning architectures on real-life video. Comparing to existing hybrid models such as [6, 7, 8] that are created for synthetic video and reason about only physics concepts, our project study human-involved concepts from real-life videos. To our knowledge, our hybrid architecture would be one of the first hybrid systems that experiment with the TVQA+ dataset.

Additionally, this project provides an integration model of neural and symbolic learning. Neural components extract features from unstructured data, and concepts are learned through an Inductive Learning system. In contrast to existing works, our pipeline utilises the symbolic paradigm ASP for both rule learning and reasoning, with minimum use of human-engineered concept encoding. Moreover, compared to pure statistical approaches, our symbolic learned concepts are easy to interpret, and symbolic reasoning offers more transparency in the question-answering process.

Chapter 2

Background

In this chapter, we briefly introduce the technical background of our project. We focus on the symbolic reasoning model *Answer Set Programming* [9], the inductive learning system *ILASP* [13], and the event-based time formalisation *Event Calculus* [14]. Apart from the symbolic basis of the project, we also introduce *Abrupt Shot Transition* and *Jaccard Similarity Coefficient* that play important roles in our approach.

2.1 ASP

Answer Set Programming (ASP) [9] is a logic programming language that efficiently solves problems in a declarative way. Given a problem and formalise it into a logic program, solving this logic program is equivalent to solving the original problem. The solutions to the logic program are called *answer sets*.

Normal rules are the basis of an ASP program. They are of the form $\texttt{h} :- \texttt{b}_1, \ ..., \texttt{b}_n, \texttt{not} \ \texttt{c}_1, \ ..., \texttt{c}_m$, where $\texttt{h}, \texttt{b}_i$ and $\texttt{c}_i$ are atoms, and `not` is negation as failure. We refer to `h` as the *head* of the rule and the conjunction on the right-hand side as the *body* of the rule. A rule without a body is a *fact* that always holds. A rule without a head is a *constraint* and rules out the candidate sets of atoms that satisfy the constraint's body. All ASP programs in this project are formed with normal rules, facts and constraints. Note that many other forms of rules are not mentioned here but are available as part of the ASP syntax.

A *Herbrand interpretation* is a set of ground atoms that are assigned to be true by the interpretation [15]. A Herbrand interpretation I is a *model* of an ASP program P if and only if every clause in the program can be satisfied by it. That is, for all $r \in P$, if I satisfies the body or r, then I must also include the head of r.

Given a grounded ASP program P and a set of ground atoms X, we can compute its *reduct*: $P^X := \{head(r) \leftarrow body^+(r) \mid r \in P, \ body^-(r) \cap X = \varnothing\}$. A Herbrand interpretation I is an answer set of program P if and only if it is the subset-minimal model of the reduct P^I. We use $AS(P)$ to denote the set of all answer sets of a program. In this project, we use the ASP solver clingo [16] to compute the answer sets of a program.

2.2 ILASP

Learning from Anser Set (LAS) is a framework of Inductive Logic Programming (ILP), which aims to learn a hypothesis from a set of examples encoded in ASP syntax that are called *partial interpretations* [13]. A partial interpretation has two sets of ground atoms: e^{inc} called *inclusion set*, and e^{exc} called *exclusion set*. An interpretation I is called to *extend* a partial interpretation $e = \langle e^{inc}, e^{exc} \rangle$ if and only if $e^{inc} \subseteq I$ and $e^{exc} \cap I = \varnothing$ [13].

A LAS task is formalised as a tuple $T = \langle B, S_M, \langle E^+, E^- \rangle \rangle$, where B is an ASP program representing the *background knowledge*, S_M is a set of ASP rules called the *hypothesis space*, and E^+ and E^- are sets of partial interpretations called *positive* and *negative examples* respectively [13]. A hypothesis $H \subseteq S_M$ is called to be an inductive solution of T if and only if it satisfies the following conditions [13]:

$$\forall e^+ \in E^+ . \exists A \in AS(B \cup H) \text{ such that } A \text{ extends } e^+$$
$$\forall e^- \in E^- . \nexists A \in AS(B \cup H) \text{ such that } A \text{ extends } e^-$$

ILASP [13] is a state-of-the-art ILP system following the LAS framework. Apart from learning from partial interpretations, ILASP also supports its variation, weighted context-dependent partial interpretations (WCDPIs) [17]. A WCDPI is defined as $e = \langle e_{id}, e_{pen}, e_{pi}, e_{ctx} \rangle$, where e_{id} is the example's identifier, e_{pen} is the penalty, e_{pi} is a partial interpetation, and e_{ctx} is an ASP program called *context*. e_{ctx} describe the 'settings' for the example, and for a WCDPI e to be accepted by a program P, there must be an answer set $A \in AS(P \cup e_{ctx})$ such that A extends e_{pi}. Similar to partial interpretations, WCDPI can be positive or negative. When learning from WCDPIs, ILASP searches for the optimal hypothesis $H^* \in S_M$ such that as many WCDPIs in E^+ are accepted by $B \cup H^*$ and as few WCDPIs in E^- are accepted by $B \cup H^*$. When a positive WCDPI is not accepted or a negative WCDPI is accepted, we pay the penalty e_{pi} of that example. The optimal hypothesis is the one with shortest length and least penalty.

2.3 Event Calculus

Event Calculus is an event-based formalisation of time using first-order predicates and negation as failure. In this project, we use the Simple Event Calculus[14] that was based on the original proposed Event Calculus in [18].

The basis of Event Calculus is fluents, which represent anything that might change over time. The basic predicates in Table 2.1 are used together with fluents and are able to describe a wide range of events and effects of actions that happen in a period of time.

Predicate	Meaning
$initiates(A, F, T)$	Action A at time T causes fluent F to start to hold
$terminates(A, F, T)$	Action A at time T stops fluent F to hold
$initially(F)$	Fluent F holds from time 0
$happens(A, T)$	Action A happens at time T
$holdsAt(F, T)$	Fluent F holds at time T
$clipped(F, T)$	Fluent F is clipped at time T

Table 2.1: Predicates used in Simple Event Calculus, based on predicates in [14].

The following axioms are required to relate the predicates:

$$holdsAt(F, 0) \leftarrow initially(F).$$
$$holdsAt(F, T+1) \leftarrow holdsAt(F, T), not\ clipped(F, T).$$
$$holdsAt(F, T+1) \leftarrow initates(A, F, T), not\ clipped(F, T).$$
$$clipped(F, T) \leftarrow terminates(A, F, T).$$

Event Calculus is a declarative and robust model for reasoning about time with a few simple predicates. In this project, we adjust the Simple Event Calculus for our use, and our task-dependent ASP implementation of it is detailed in Section 5.1.

2.4 Abrupt Shot Transition

A shot is a sequence of uninterrupted frames over a time period [19]. Shot transitions are commonly used as a cinematic technique in films and TV series. An *abrupt shot transition*, or abrupt transition for short, is a sudden change between two shots that happens in one frame [20]. For the TVQA+ dataset, most shot transitions used are abrupt transitions. In this dataset, abrupt transition mostly happens when the camera angle changes in the current scene, while it is also occasionally used to let the video move on to another scene.

It is vital to detect such an abrupt transition to reason about persistency. When someone in the camera suddenly 'disappears' in the next frame, if it is angle changes that causes the 'disappearance', then we know that the character is very likely still at the current scene even we cannot see them. Figure 2.1 provides an example of that abrupt transition occurs without affecting any persistency of the characters.

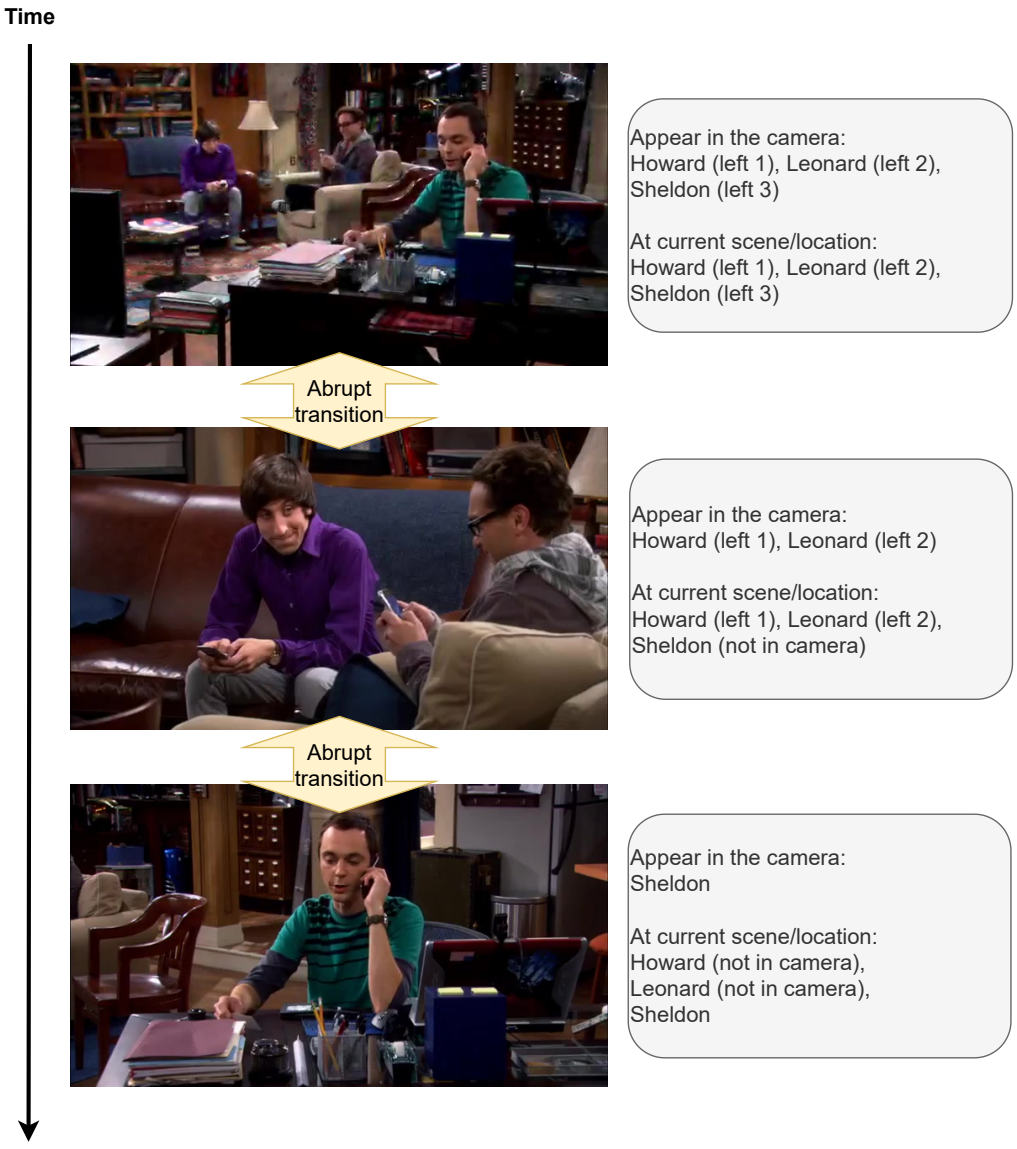


Figure 2.1: An example of abrupt transition that affects who is in the camera but does not affect the knowledge of who is in the current scene.

2.5 Jaccard Similarity Coefficient

Jaccard similarity coefficient is a measure of how similar two sets are. It is defined as the proportion of the intersection over the union:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

This score is commonly used for evaluating object detection models. For this project, we use it for evaluating the performance of the pipeline. The reason for choosing the Jaccard score and the detail of the evaluation are described in Chapter 6.

Chapter 3

HACR Overview

This chapter provides an overview of our pipeline, **H**ybrid **A**rchitecture for **C**oncept **L**earning (HACR in short), for concept learning from Video Q&A, and the dataset we use to solve the task. Due to the high complexity of real-life videos, our approach focuses on two specific concepts while still being a challenging task overall.

3.1 Targeted Concepts

HACR focuses on learning two visual concepts:

1. *Someone is **holding** something* ('holding'): The holding action in the dataset does not limit to just having an object in one's hand, but it can also be someone carrying a bag over their shoulder. The main idea of holding is that a person supports an object, and the object is not falling from them. This concept requires an understanding of spatial relations between a person and an object.

2. *Someone **enters** the scene* ('entering'): In general, the entering action involves location changing. Our simplified 'entering' concept focuses on knowing that someone who was not in the current scene beforehand appears next. The difference between the two types of entering is shown in Figure 3.1. Only recognising the existing character in the frame is insufficient to learn our simplified concept of 'entering the scene'. It requires more complex temporal reasoning that relate properties and changes between frames (see Figure 1.1 as an example).

3.2 Dataset

The dataset for HACR is based on the TVQA+ dataset [12], a subset of the TVQA dataset [21]. The videos clips from TVQA+ are collected from the TV series *The Big Bang Theory*, and the questions are human created and have various types of annotations. A sample in the dataset provides the QA pair, the ground truth answer, the ground truth bounding boxes for relevant objects and the ground truth relevant time span for answering the question. Within the scope of this project, we have created the following subsets of the TVQA+ dataset:

1. **'Holding' questions**: This is a subset of the TVQA+ questions, where the question for each sample would be in the form of 'What is *someone* holding ....'. We use dependency parsing to filter out the questions that do not match the pattern (The approach is detailed in Section 4.1 later). Our subset is slightly different in terms of the wording of the questions. The symbolic representation highly depends on the

(a) Leonard (in green jacket) and Dennis (in red cardigan) entering Sheldon's office.

(b) Sheldon (in green shirt) entering the scene where Leonard (in grey jacket) and Raj (in purple jacket) are right now.

Figure 3.1: (a) shows the case of people entering a location while staying in the scene whole time. A lot of the 'entering' questions from the TVQA+ dataset fall into this type of entering action. (b) is the situation where someone is not yet at the current scene and enter later, which would be the concept HACR focuses on.

correctness and grammatical precision of the text. Therefore, we manually fix the grammar mistakes and typos in the original texts to avoid unnecessary error and model-irrelevant poor performance.

2. **'Entering' questions**: The majority of 'entering' questions in the TVQA+ dataset focus more on location change as shown in Figure 3.1a, instead of the change of someone's appearance. Thus for learning specifically 'entering the scene', we have handcrafted 10 'Who enters the scene...' questions. The questions are based on some of the video clips in TVQA+ dataset. In the annotation, we include the relevant time span to answer the question, the appearance periods for the characters and who are already in the scene. We do not include any bounding boxes since the exact spatial location is non-essential for answering the question. We also annotate the abrupt transitions in the relevant time span. Figure 3.2 shows an example of the question and its annotations that we create.

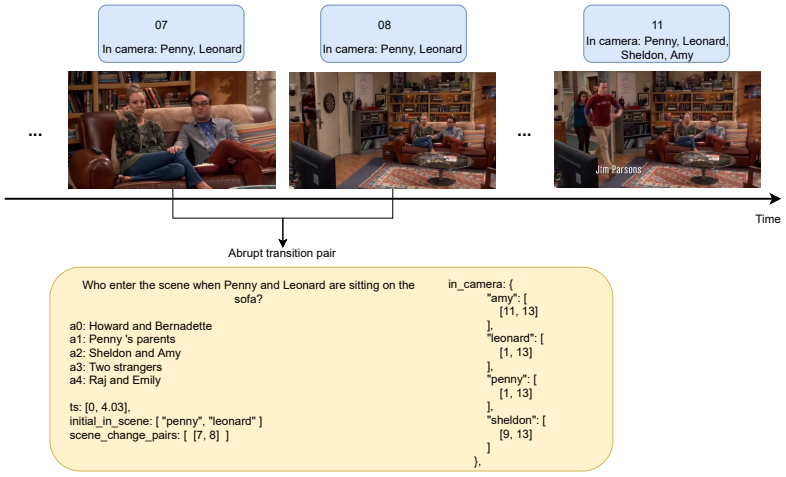


Figure 3.2: An example of question we create for learning 'entering the scene'.

3.3 Architecture Design

The neural-symbolic system introduced in this paper, namely the **H**ybrid **A**rchitecture for **C**oncept **R**easoning (HACR), can be represented as the pipeline shown in Fig 3.3. During training, we feed the question-answer pair with its ground truth annotations to the pipeline. During inference, we only give the question, multiple choices, video frames, and ground truth time span[1]to HACR. The pre-processing stage is shared during training and inference, yet the data flow through symbolic components differently.

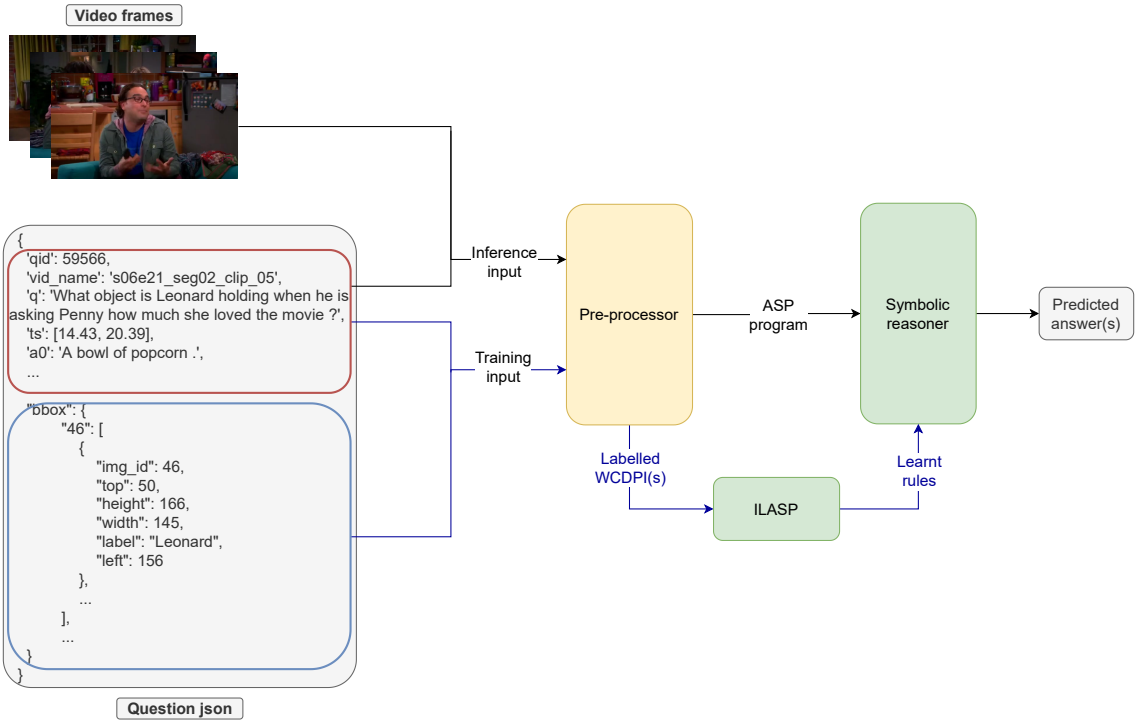


Figure 3.3: HACR architecture. Blue lines indicate data flow during training time, while black lines indicate data flow during inference time. Note that we do **not** feed the subtitles to the pipeline, as they are less relevant to our visual concept learning.

Pre-processing. The pre-processor utilises existing work on natural language processing and computer vision to extract features from the given question and its corresponding video frames. Extracted features vary from concepts to concepts. The pre-processing stage also encodes the features for the symbolic components. It translates the features into weighted context-dependent partial interpretations (WCDPIs) for training the symbolic learner ILASP, or into an ASP program to be directly executed by the symbolic reasoner to get an answer.

Training time - Symbolic learning with ILASP. HACR uses ILASP [13] for inductive learning. In order to reason about the effect of events and persistency over time, the background knowledge includes encoded general Event Calculus [14] rules for capturing the frames problem. We construct weighted context-dependent partial interpretations (WCDPIs) from ground truth annotations and let ILASP learn general rules representing the concepts from the examples.

[1]Since the purpose of HACR is to learn concepts, we consider that predicting relevant time span to answer the question is non-essential for the task. Our decision of giving the ground truth time span to our pipeline is a very different decision from existing neural architectures, as they are **not** provided with the ground truth during training or inference, but try to predict one themselves.

Inference time - Symbolic Reasoning with ASP. During inference, the reasoner combines the rules learned during training with encoded question-specific features into one ASP program. The fused program is then solved with clingo [16] to compute the answer set of the program. The answer sets include predicates of the concept the question asks for, and answers are extracted by matching a regular expression onto the predicates.

In Chapter 4 we describe the functions and design of the *Pre-Processor*. In Chapter 5 we introduce the two symbolic tasks of our pipeline, *Symbolic Learning* and *Symbolic Reasoning*.

Chapter 4

Pre-processor

This chapter presents the pre-processing component of our HACR architecture. We separate the feature extraction based on the input type: image and text. While the two processes handle the modalities differently, they come together in the last stage of translating features to symbolic representations. We first introduce the *Language Processing* that manipulates text and then present the *Frame Processing* that deals with images, with the description of the *Symbolic Representation Translation* component in the end.

4.1 Language Processing

TVQA+ dataset provides multiple types of text input, including questions, answers, ground truth annotations of labels, and subtitles. The subtitle is less relevant for our visual concept learning and thus not handled by our language processing. The rest is handled with our language processing with the following steps:

- **Dependency parsing**: Most questions share a very similar pattern of wording for the same concept. Instead of matching regular expressions on the questions, it is more robust to extract the action, subject, and object through the dependency tree of the questions, especially when the verbs are in past tense or other grammatical tenses.
- **Synonym, hypernym, and hyponym check**: For the 'holding' concept, the object we are interested in is usually a physical item. The label received through the object detection often differs from the wording used in the question or answer, although they refer to the same item. Checking whether two words are synonyms could cover some cases, while hypernym and hyponym are required for less trivial scenarios. For example, if the answer is a shirt and the object detector identifies a T-shirt, this could be a potential answer, as 'shirt' is a hypernym of 'T-shirt'.

4.1.1 Dependency Parsing

Dependency parsing aims to extract directed relationships between words [22]. Based on the roles of the words, they are categorised into head and dependents. Figure 4.1 shows an example of dependency parsing on a question. For our pipeline, we use the pre-trained `en_core_web_sm` model in the `spacy`[2] package to perform dependency parsing.

There would be only one root word that dominates the whole sentence, which usually is the action word. With dependency parsing, we can easily extract the subject and object associated with the action. These properties are used to filter the questions as well as

[2] https://spacy.io/usage/models

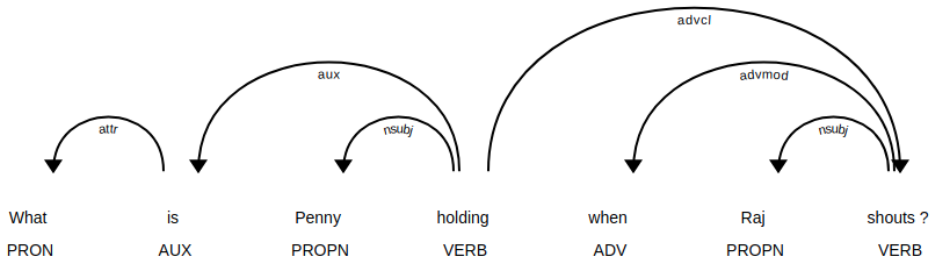


Figure 4.1: A visualisation of dependency parsing on a 'holding' question, with `displacy` from `spacy`.

constructing symbolic representation (detailed in Section 4.3). To see whether a question fits the concept learning task, we check if the question's root word is that action in the concept, and has a person as its subject dependent – i.e., a name with a 'nsubj' dependency tag. For example, in Figure 4.1, the root word is 'holding' and its noun subject dependent is Penny. We also check if the person is in the set of characters that we can recognise through our face detection module (detailed in Section 4.2.2). If not, we would ignore that question. Table 4.1 shows the result of our training set question filtering for 'holding'.

We construct the 'entering' training set ourselves: all questions follow the pattern of having 'enter' as the root action, with all people mentioned included in the pre-defined characters set. Therefore, further filtering is not required for the 'entering' learning task.

	Count	**% of all Qs**	**% of 'Holding' Qs**
All questions	23545	100%	–
'Holding' questions	908	3.86%	100%
+ parsed without error *	886	3.76%	97.6%
+ COCO answer obj. **	91	0.386%	10.0%

Table 4.1: 'Holding' questions number and percentage in TVQA+ training set.
*Parsed without error: There are a few edge cases that would not pass the check we mention above. For example, some questions involve multiple people as the action subjects. These questions are simply ignored without furthur processing.
**COCO answer obj.: These are the questions that their answers' objects are in the COCO dataset category. The reason to check whether an object is in the COCO category is due to our object detector, which is pre-trained on COCO dataset and can only detect objects in its category. More details of the object detector is detailed in Section 4.2.1. Whether an object is in the COCO category is verified by checking synonym, hypernym and hyponym.

Limitation

Accuracy when parsing specific names: In most cases, the `en_core_web_sm` model returns the correct dependency tree. However, we notice that the model often mislabels Raj as an 'adverbial modifier' (advmod), as shown in Figure 4.2. The name Raj is short for Rajesh, and the word Raj itself refers to'the former British rule of the Indian subcontinent' according to Merriam-Webster [23]. Under a perfect language model, we would only need to check if an action word's dependent is a noun or pronoun to decide if it can an action's subject or object. However, this inaccuracy in the pre-trained language model forces us to include 'advmod' as another possible tag for a word to be considered. While including the 'advmod' tag does not introduce any false positive for the current task, this does make the code less trivial and potentially could introduce errors. For example, if an 'advmod' word that is not 'Raj' is a dependent of an action word, then the word would be wrongly

treated as an action's subject or object.

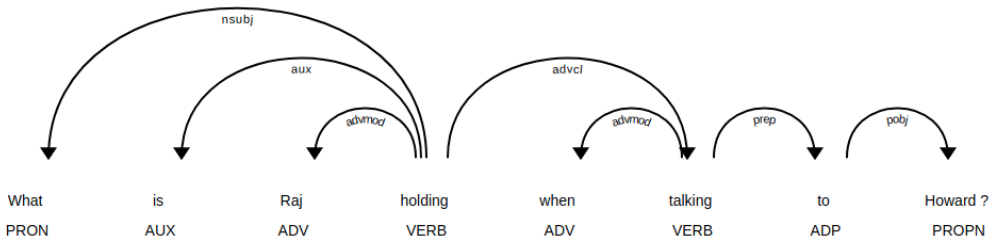


Figure 4.2: Dependency parsing for sentence including the name 'Raj'. While 'Raj' should be the action subject with the dependency tag 'nsubj', it gets a label of 'advmod' (adverbial modifier).

4.1.2 Synonym, Hypernym, and Hyponym Check

We use `wordnet`[3] from the `nltk` package to perform synonym, hypernym and hyponym checks. The `wordnet` interface provides a set of synonyms for each word, called `synset`. When checking if word A and word B are synonyms of each other, we check if word A is in word B's `synset` or vice versa. `wordnet` also provides the hypernym or hyponym of a `synset`. Thus, to check if word A is word B's hypernym or hyponym, we check if word A is a hypernym or hyponym of any word B's `synset`s.

Limitation

While the above method helps us to identify pairs of words that linguistically mean the same object in many cases, it cannot link two words with more complex relations. An example is 'cup' and 'mug', which is a type of cup itself. Although 'cup' and 'mug' are not linguistically linked via synonyms, hypernyms, or hyponyms under `wordnet`, because of the type-subtype relation, a human would pick the answer 'cup' if knowing there is a mug in the image and 'mug' is not an option. This lack of flexibility causes the pipeline to eventually miss answers even if the object is detected and reasoned as being held by someone.

4.2 Frame Processing

A single image provides stationary spatial attributes of objects, while consecutive frames provide temporal information. For general concept learning, the frame processing module should be capable of handling feature extraction for both cases. Focusing on 'holding' and 'entering', the frame processing stage for HACR is designed to achieve the following tasks:

- **Object detection**: We should be able to locate and label the objects (including human) in each frame. While non-human objects are not required for learning 'entering', detecting people in the frame is required for the later stage of identification.
- **Face detection and classification**: This is the stage of recognising a person in the frame. The questions that HACR would try to answer requires precise knowledge of the individual's identity, making it mandatory to identify the person.
- **Abrupt transition detection**: Getting a false positive of someone entering the scene is possible due to abrupt shot transition (see Figure 2.1). In order to avoid

[3] https://www.nltk.org/howto/wordnet.html

such problems, all abrupt transitions should be detected and passed to the symbolic representation translator.

A summarised pipeline is shown in Figure 4.3. In this section, we detail in order the design for each task mentioned above.

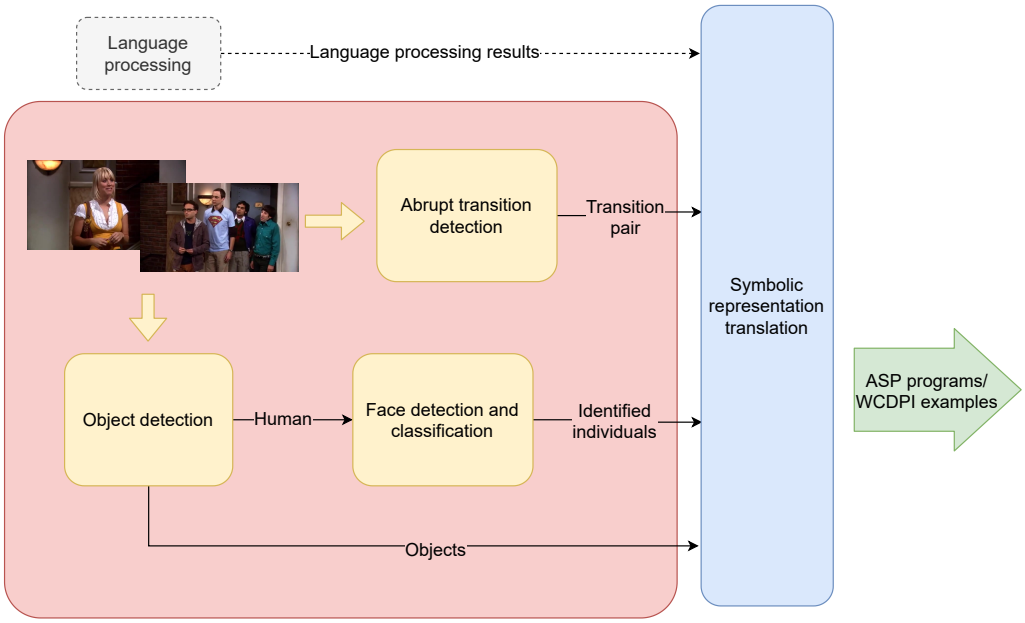


Figure 4.3: Frame processing overview. The symbolic representation translation is shared with the language processing pipeline. The components are described in the rest of this section.

4.2.1 Object Detection

One commonly-used two-stage object detection approach is Faster R-CNN [24]. We use `torchvision`'s pre-trained Faster R-CNN with ResNet-50-PFN backbone[4]. The model is already trained on the COCO train2017, and we use it for inference only without further tuning. The model takes in images with flexible size and gives a list of object predictions for each image. For each object, its prediction contains the coordinates of the bounding box's corners, the label of that object and the confidence score of the prediction. We filter out the objects that the model are not confident about (i.e. having a low confidence score, and in our case, the threshold is 0.7). Figure 4.4 shows the predictions from the model after filtering.

Figure 4.4: A demonstration of object detection with `torchvision`'s pre-trained Faster R-CNN with ResNet-50-PFN backbone.

[4] https://pytorch.org/vision/stable/models.html#faster-r-cnn

The kept objects are then categorised based on their type. The objects detected as humans are passed onto the next step of face detection and classification. Physical objects will be passed directly to the symbolic translator if we are learning the 'holding' concept. In the case of learning 'entering', they are dropped.

Limitation

Type of objects: As mentioned, the model is pre-trained on COCO train2017, limiting the objects that can be detected to objects in the COCO dataset[5]. Although some objects would be recognised as in the COCO train2017 class after applying synonym check, the majority are not recognised. This problem heavily affects the performance of the pipeline when reasoning about the 'holding' concept. If an object is not recognised, the symbolic representation would not be generated and pass to the symbolic reasoner. By solving the ASP program, the pipeline either gives the wrong answer or no answers at all. Table 4.1 shows the total number of questions that is with the type of 'What is someone holding...' and the questions that their answers can be recognised to be in COCO class through language processing. We can see that this problem drastically reduces the number of questions HACR can give answers to.

Detecting small and partially covered objects: The pre-trained torchvision Faster-RCNN struggles to detect small and partially covered objects in the frames. This problem may lead to missing the key objects relevant to the question. Figure 4.5 shows a scenario when the model fails to predict the small and partially covered remote control that is the answer to the question. Again, without the key objects being detected and parsed into ASP programs, the answer set will not include the answer even if we have a 100% accurate learnt concept rule. This is another factor that influences the performance of the pipeline.

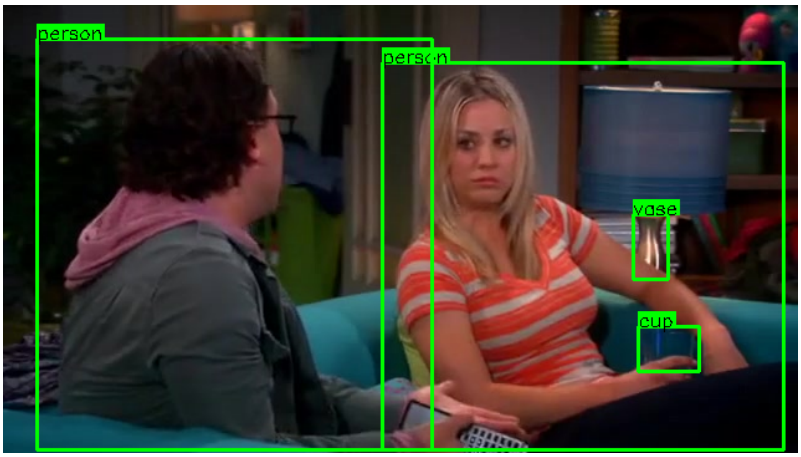


Figure 4.5: A example of failing to detect the small remote control that Leonard is holding, which is the answer to the question.

4.2.2 Face Detection and Classification

Both 'holding' and 'entering' concepts are actions from a human subject, making human detection and recognition essential for the pipeline to answer questions involving these concepts. As the nature of the TVQA+ dataset, the video frames are collected from *The Big Bang Theory* series, where people often change clothes between episodes or even within a short video clip. Thus simply training a model to distinguish people based on

[5]The classes of objects in COCO can be found here: https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

their whole bounding boxes (i.e. including their body and clothes) would not be sufficient. Facial features are generally unique between individuals, thus being an excellent fit to classify different characters appearing.

The pipeline for confirming who appears in the scene, visualised in Figure 4.6, consists of two processing stages while also relies on a collection task:

- **Process 1 - face detection**: From the object detector, we receive only the objects that are humans and extract the parts of the image based on the bounding boxes. The image fragments of humans are passed to a pre-trained Haar cascade model from OpenCV[6] to predict the face's bounding box from it. We encode the detected faces with a pre-trained facial feature encoder provided by the `face_recognition` package[7]. Before feeding it into the encoder, we extract the face from the image and resize it to match the encoder's input size of 150 by 150.

- **Collection - face samples collection**: With the cascade model and the encoding functionality, we collect all the faces of humans, encoding and their ground truth names based on the labels in the TVQA+ dataset. We limit our face identification process to recognise a particular set of 14 people. These are the main characters in *The Big Bang Theory* and are frequently involved in the questions and answers. The set of the characters can be found in Appendix A. There are in total 67305 face samples collected, and Figure 4.7 shows a small portion of them.

- **Process 2 - face classification**: Using the collected face samples, we fit a k-nearest neighbour (KNN) classifier[8] from the face encodings, with the standard Euclidean distance ($\sqrt{\sum_i (\boldsymbol{x_i} - \boldsymbol{y_i})^2}$) for the distance metric. When giving a prediction, the classifier decides based on the closest five training examples. This predicted name would replace the general label of 'person' of the object, which finishes the identification process.

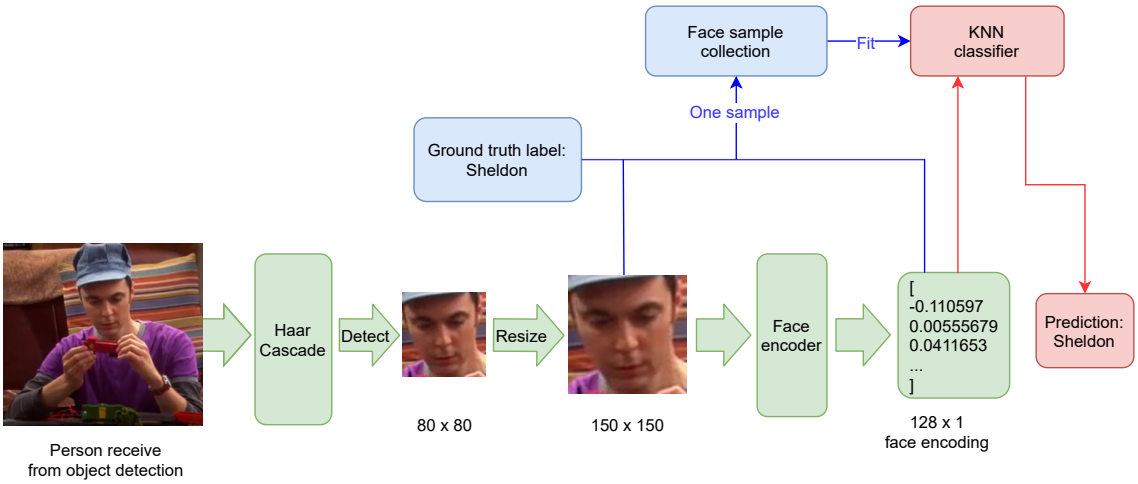


Figure 4.6: The pipeline for identifying a person in the frame. Green data flow and components are used both during training and inference. Blue lines represent the flow of collecting faces and labels and training the KNN classifier. Red lines are the classification process during inference.

[6] https://docs.opencv.org/4.5.1/d1/de5/classcv_1_1CascadeClassifier.html
[7] https://github.com/ageitgey/face_recognition
[8] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Limitation

Appearance of unknown characters: Since we only collected the commonly appeared characters, when a person out of the set appears, the face classifier would never predict their actual name. Instead, it would try to label them as one of the 14 names it knows. Moreover, we do not increase our collection of faces after the KNN classifier is trained, so the classifier cannot learn new faces. This issue means that, eventually, the appearance of unknown humans would be passed down as noise for the symbolic reasoning during inference and could potentially affect the final answers produced.

Accuracy of Haar cascade model: As shown in Figure 4.7, a few non-face examples are included in the collection. The Haar cascade model sometimes sees multiple horizontal lines with pixel intensity changes as humans' eyes, the bottom line of the nose or the mouth. It could be hard to reduce the number of false detection due to the relatively low resolution of people in the frames and that Haar features are sensitive to edges.

Figure 4.7: Face collected from all the video clips in TVQA+ with the Haar cascade model with the ground truth person name. There are a few anomalies being collected too.

4.2.3 Abrupt Transition Detection

This process aims to detect an abrupt shot transition between two consecutive frames to avoid false positives of recognising characters' sudden appearance as entering due to shot transition. An abrupt shot transition causes a significant change between two consecutive frames. Thus, in order to identify a possible abrupt transition, we first need to measure the dissimilarity of a frame pair. There are multiple ways we experiment to quantify the difference between two consecutive frames:

1. **Sum of absolute difference (SAD)**: The most straightforward way to measure the dissimilarity is to compare two frames pixel by pixel and sum the absolute values of the differences across all pixel pairs. The computation is as follows:

$$\sum_{c\in\{r,g,b\}} \sum_{i} |x_{c,i} - \widehat{x_{c,i}}|$$

2. **Whole-image histogram difference (Hist-All)**: This method computes the histogram difference instead of pixel difference. A histogram for a specific channel is constructed by counting the number of pixels with the same colour intensity value and put them into a bin. For an RGB image, there are three histograms, each with 256 bins. We compute the three histograms on the whole image for the pair of consecutive frames, and this results in three pairs of consecutive histograms. For a pair of histograms, we compare bin by bin for the absolute difference in count and sum over all the bins for all three pairs. This approach is mathematically formulated as:

$$\sum_{c\in\{r,g,b\}} \sum_{i=0}^{256} |\text{bin}_{c,i} - \widehat{\text{bin}_{c,i}}|$$

3. **Regional histogram difference (Hist-Reg)**: This algorithm is similar to the previous histogram difference, but we first segment the image into eight different regions and compute the histogram for each region. This results in 24 pairs of histograms, and we compute the difference for each pair and sum over the absolute values. The formula is as follows:

$$\sum_{s=1}^{8} \sum_{c\in\{r,g,b\}} \sum_{i=0}^{256} |\text{bin}_{s,c,i} - \widehat{\text{bin}_{s,c,i}}|$$

We collect the scores of differences between consecutive scenes with all three methods. Table 4.2 shows for each method the number of videos we use and the total number of frame pairs that we collect and compute a score for.

	SAD	Hist-All	Hist-Reg
# video clips	4199	1000	1000
# pairs of consecutive frames	4364926	1853275	1853275

Table 4.2: Counts statistic for all three frame difference scoring method. All video clips are used for collecting SAD score, while only 1,000 are used for histograms methods due to heavier computation in fixed execution time.

Since all the scores have no labels of whether it is an abrupt transition or not, we fit k-means clustering and Gaussian mixture models on the data to perform unsupervised learning. Both models have two clusters or components, representing the abrupt transition case and the smooth camera movement case. We assume that all scores should share the

same label as score 0 (i.e. no change at all) if it is not an abrupt transition, and this assumption is also used in the evaluation and inference.

We notice that the unsupervised model sometimes gives consecutive labels representing abrupt transition (i.e. different label from the score 0's label). It is almost impossible to have such cases of continuous abrupt shot changes in a TV show. Therefore, We apply non-maximum suppression (NMS) to avoid such issues. For a dissimilarity score labelled differently as score 0 (i.e. indicating a prediction of abrupt transition), we compare it with its two immediate neighbour values and suppress it to value 0 if it is not the largest value in the triplet. After NMS, the scores of all the frame pairs would be given to the model again to generate the final predicted labels.

The performance evaluations of all models with and without applying NMS are detailed in Section 6.2.3. We choose the k-means clustering model trained on the Hist-All scores with NMS applied for our HACR pipeline. During inference, we compute the Hist-All score for each consecutive frame pair in the relevant time span and get a label from the model. Furthermore, based on these initial scores and predicated labels, we apply NMS and generate the final labels for all pairs. If it gets the same cluster label as score 0, it would be predicted as not an abrupt transition; otherwise, it would be an abrupt transition.

Limitation

Computation of Hist-All and Hist-Reg scores: Hist-All and Hist-Reg both need to iterate through all pixels first to construct three histograms and then iterate through the bins, while SAD only needs to iterate through the pixels. This extra step of computation makes a big difference in computation time when the input of frame pairs are in tens of thousands. Fewer examples are collected in the same amount of time, as shown in Table 4.2. Moreover, the performance of the unsupervised learning models might be further improved if given more data.

4.2.4 Unused Component: Human Pose Estimation

Apart from the processes mentioned above, we also experiment with human pose estimation in each frame. Human pose estimation is the task of locating human joints in frames. Through localising joints, we hope to identify characters' hand positions and legs movements that could help with learning the concept of holding and entering.

We only try to estimate the pose of the identified characters in each frame, i.e. people with their names known either through face classification during inference or ground truth bounding box during training. The estimation is done with a ResNet-50 model from [25][9], pre-trained on the MPII dataset [26]. The model takes in a square input, and thus we extract the person from the image based on their bounding box and align them in the centre of a square background. The reason to align it is that the model assumes that the main character is in the centre of the image. The square is then resized to 256 by 256 to match the dimension of the model input. The final output contains 16 joints of the body and their confidence score, and the relevant pairs are connected for each body part.

The performance of the model is overall unreliable due to how the characters appear on camera in the dataset. As shown in Figure 4.8, while predictions are accurate when the character's entire body is shown in the frame, the result is inadequate for our purpose when only half of the body is in the frame. The model seems to attempt to fit the whole body onto the partial body appearing in the frame, giving misleading predictions. Because of its unstable performance and the fact that the key characters' bodies are not entirely in the camera for many clips, we have to discard this component from the HACR pipeline.

[9]The implementation of the paper that we use is at https://github.com/microsoft/human-pose-estimation.pytorch

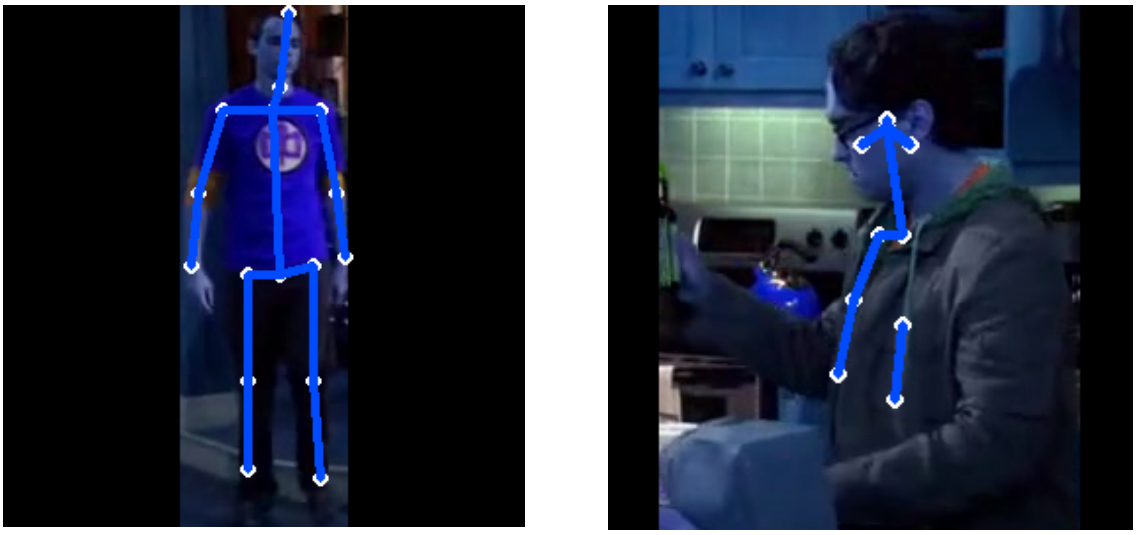


Figure 4.8: Human pose estimation on TVQA+ data. The blue sticks represent the limbs and body chunk, while the circles represent the joints.
On the left, Sheldon's whole body is shown and the prediction is accurate. However, on the right, only half of Leonard's body is shown and the prediction is wrong and confusing.

4.3 Symbolic Representation Translation

Receiving the results from language processing and frame processing, we translate them into symbolic representations to be used by the later task of learning or reasoning. In this section, we define our chosen representation language. We divide the predicates into three categories based on whether they relate to the concepts of 'holding' and 'entering': general predicates used by both concepts, predicates related to 'holding', and predicates related to 'entering'.

General predicates

- $time(T), T \in \mathbb{N}$: This predicate defines all the time values in a ground truth time span. It defines the timestamp which the learner/reasoner should focus upon.
- $person(P), P \in$ pre-defined set: From face detection and classification, we can identify a person and be specific to their name. This term defines an identified person in a video clip. The pre-defined set is the same as the set we used to collect all the faces (see Appendix A for the characters' names).

Predicates related to 'holding'

- $current\_time(T), T \in \mathbb{N}$: To learn if someone is holding something, we observe the video frame by frame. This predicate specifies the exact timestamp and frame we are looking at.
- $object(O)$: This term defines the items picked up by the object detector. Although people would be detected via object detection, they are not represented as objects but as instances of the $person(P)$ predicate.
- $bbox(T, L, Top\_left\_x, Top\_left\_y, Width, Height)$, where $T, Top\_left\_x, Top\_left\_y, Width, Height \in \mathbb{N}$: This defines the bounding box's properties for each physical item or person at certain time T. The bounding box's label L should also exist in either $object(L)$ or $person(L)$.

- $bbox\_intersec(T, L1, L2, P), T, P \in \mathbb{N}$: Based on the bounding boxes between two objects (including humans), this predicate defines the proportion of the overlap between two boxes. For bounding box 1 of label L1, the percentage P of its intersection with box 2 of label L2 is calculated as:

$$P = int(\frac{\text{area of intersection}}{\text{area of bounding box 1}} * 100)$$

  (The calculation for the intersection between two boxes is trivial and hence put in Appendix A.) The order of L1 and L2 is crucial for the meaning of the predicate: the percentage is calculated with respect to the first box, thus putting the first box's label before the second one.

- $holding(P, O)$: This predicate represents the concept of holding and is essential for constructing positive WCDPI examples for learning and ASP programs for reasoning. Through incorporating Event Calculus, we can describe that at time T, person P is holding some object O in the form of $holdsAt(holding(P, O), T)$, where 'holding' is considered now to be a time-dependent fluent.

Below is an example of symbolic representation translation for answering a 'holding' question during inference. For each frame, we identify the people and objects in it and provides frame with frame processing. We encode all their bounding boxes and intersection proportion in the ASP program. We assign an identifier to each object after their class name (`1_1` in the example). If there are multiple objects with the same class in the frame, we can still distinguish between instances based on the identifier.

Figure 4.9: Frame 94 for 'holding' question with qid 13712

```
time(94).
time(95).
bbox(94, amy, 378, 45, 171, 310).
bbox(94, penny, 130, 34, 158, 321).
bbox(94, cup1_1, 235, 240, 34, 47).
bbox(94, book1_1, 528, 314, 35, 45).
bbox(94, bowl1_1, 0, 240, 39, 27).
bbox_intersec(94, amy, book1_1, 1).
bbox_intersec(94, book1_1, amy, 58).
bbox_intersec(94, penny, cup1_1, 3).
bbox_intersec(94, cup1_1, penny, 100).
person(amy).
person(penny).
object(book1_1).
object(cup1_1).
```

Predicates related to 'entering'

- $abrupt\_transition(T, T+1), T \in \mathbb{N}$: This term encodes any detected abrupt transition between two consecutive frames. Although T theoretically can take any natural number, as we are only reasoning in a specific time span, both T and $T+1$ should fall into that time interval.

- $in\_camera(P)$: This represents the event of someone currently being in the camera. With Event Calculus, we can specify at which time T this property holds in the format of $holdsAt(in\_camera(P), T)$. The P in this term should be a detected person, i.e. $person(P)$ should hold.

- $at\_curr\_location(P)$: Different from being in the camera, a person can be at the current location without being included in the scene due to video editing and camera

changing (see Figure 2.1 as an example). This term describes someone being at the current place, assuming that the location has not changed. This assumption is automatically satisfied in our 'entering the scene' concept's training set.

Below is an example of symbolic representation translation for answering 'entering' question during inference. We identify the people in each frame with frame processing to construct $in\_camera(P)$ at each timestamp. From the question sub-clause, we extract the people at the current location from the beginning of the time span, Raj and Leonard. Since the symbolic learning and reasoning tasks incorporate Event Calculus to capture time (detailed in Chapter 5), the predicates are combined with our Event Calculus implementation for that task before given to the symbolic components.


```
time(71..101).
abrupt_transition(74, 75).
abrupt_transition(95, 96).
holdsAt(in_camera(leonard), 72).
holdsAt(in_camera(leonard), 74).
holdsAt(in_camera(raj), 75..95).
holdsAt(in_camera(sheldon), 96..101).
holdsAt(in_camera(stuart), 100).
holdsAt(in_camera(leonard), 101).
person(leonard).
person(raj).
person(sheldon).
person(stuart).
holdsAt(at_curr_location(raj), 71).
holdsAt(at_curr_location(leonard), 71).
```

Figure 4.10: Frame sequence of 'entering' question with qid 25625601

Chapter 5

Symbolic Learning and Reasoning

This chapter discusses the two symbolic tasks of our HACR pipeline: learning at training time, and reasoning at inference time. All spatial and temporal features either from ground truths or from the video frames, and the textual features from the question-answer pair, should have already been extracted and translated to symbolic representation from the pre-processing stage. The symbolic components focus on utilising the knowledge to learn general rules and construct ASP programs.

5.1 HACR Symbolic Learning Task

We aim to learn a symbolic definition of our event-based concept using the inductive learning system ILASP[13]. Our learning task is context-dependent with noisy examples that consist of three parts:

- Background knowledge B
- Hypotheses space S_M defined with mode bias
- A positive set of weighted context-dependent partial interpretations (WCDPIs) E^+

Our learning task does not focus on learning constraints and thus we do not construct negative WCDPIs.

In the following section, we discuss in order each part's construction. At the end of this section, we show the hypotheses obtained by ILASP for both concepts. These rules are later on used in the reasoning task during inference.

Background Knowledge

The time representation Event Calculus is independent of context and thus is included in the background knowledge. For our use case, we encode Event Calculus in ASP syntax. These rules are the concept-independent axioms and are used for learning both concepts. The actual implementations are yet slightly different between the two concepts.

The encoding for 'entering' is relatively straight forward, where we formalise 'entering' as action, and we encode people's appearances as events with two fluents: $fluent(in\_camera(P))$ and $fluent(at\_curr\_location(P))$. We also encode the relevant Event Calculus axioms introduced in 2.3:

$$\begin{aligned} holdsAt(F, T+1) :- \\ & time(T), time(T+1), fluent(F), \\ & initiates(A, F, T), not\ clipped(F, T). \end{aligned} \tag{5.1}$$

$$holdsAt(F, T+1) :- \\ time(T), time(T+1), fluent(F), \\ holdsAt(F, T), not\ clipped(F, T). \tag{5.2}$$

$$clipped(F, T) :- time(T), fluent(F), terminates(A, F, T). \tag{5.3}$$

More tweaks are applied when constructing the background for 'holding'. As the action of holding lasts over time, we formalise it as a fluent instead of continuous action. Thus we have $fluent(holding(P, O))$ in the background knowledge. We omit the action that initiates the 'holding' event, and simplify $initiates(A, F, T)$ and $terminates(A, F, T)$ to $initiatedAt(F, T)$ and $terminatedAt(F, T)$, and also discard the use of $clipped()$ and use $terminatedAt()$ directly in the rules:

$$holdsAt(F, T+1) :- \\ time(T), time(T+1), fluent(F), \\ initiatedAt(F, T), not\ terminatedAt(F, T). \tag{5.4}$$

$$holdsAt(F, T+1) :- \\ time(T), time(T+1), fluent(F), \\ holdsAt(F, T), not\ terminatedAt(F, T). \tag{5.5}$$

Apart from encodings of Event Calculus, we specifically create a rule that describes the spatial relation between a person and an object that would be useful for knowing if someone is holding something. It uses the predicate $bbox\_intersec(T, O, P, Percentage)$ representing how much intersection is taking place with respect to the object:

$$happensAt(close(P, O, Threshold), Time) :- \\ person(P), object(O), bbox(Time, P, \_, \_, \_, \_), bbox(Time, O, \_, \_, \_, \_), \\ bbox\_intersec(Time, O, P, Intersection), holding\_threshold(Threshold), \\ Intersection >= Threshold. \tag{5.6}$$

Lastly, there are two constraints specifically added for 'holding'. They are used together with predicate $goal()$ included in the WCDPI's context, and force any $holdsAt()$ that is in $goal()$ to be included in the answer set.

$$:- current\_time(T-1), not\ holdsAt(F, T), goal(holdsAt(F, T)). \\ :- current\_time(T-1), holdsAt(F, T), not\ goal(holdsAt(F, T)). \tag{5.7}$$

The constraints impose strong requirements so that no $holdsAt(F, T)$ different from the goal is included in the answer set. They essentially enforce the learner to find an rule that can accurately exclude all fluents that should be false from the answer set.

Hypotheses Space

The search spaces of hypotheses differ for the two concepts, while both are defined with mode declarations.

For 'holding', the hypothesis should represent some spatial relationship between objects and humans that triggers 'holding' to happen. Therefore, the goal is to let ILASP find the optimal threshold in Rule 5.6, which decides if an item and a human happen to be close enough. The mode declarations for this task is then:

$$\#modeh(initiatedAt(holding(var(person), var(object)), var(time))). \tag{5.8}$$

$$\#modeb(1, holdsAt(holding(var(person), var(object)), var(time))). \tag{5.9}$$

$$\begin{aligned}\#modeb(1, happensAt(close(var(person), var(object),\\ const(holding\_threshold)), var(time))).\end{aligned} \tag{5.10}$$

$$\#constant(holding\_threshold, 1..100). \tag{5.11}$$

Body Declaration 5.10 with specified Constant 5.11 together describe a search of threshold from 1..100. We also include Body Declaration 5.9 and expect it to be used as a negated term to represent that the event of a person holding a certain item does not hold when we want to initiate it. Note that ILASP automatically consider negated terms for every mode body in the hypotheses space. Overall, we expect to see a hypothesis similar to:

$$\begin{aligned}&initiatedAt(holding(P, O), T) :-\\ &\quad happensAt(close(P, O, i), T), not\ holdsAt(holding(P, O), T).\\ &\text{where: } i \in \mathbb{N}, i \in [1, 100]\end{aligned} \tag{5.12}$$

Rule 5.12 describes that if a person is not holding an object at time T and yet they are close enough that their bounding boxes have a large portion of overlay, then we initiate the fluent 'holding' at time T (although that fluent still will not hold at time T due to Rule 5.4).

The entering concept requires more temporal reasoning. The action initiates the fluent of someone who was not in the scene being at the place now. And any changes in appearances on camera due to shot transition should not affect the existing characters' state. Thus we set the mode declarations as follows:

$$\begin{aligned}\#modeh(initiates(enter(var(person)),\\ at\_curr\_location(var(person)), var(time))).\end{aligned} \tag{5.13}$$

$$\#modeb(holdsAt(in\_camera(var(person)), var(time))). \tag{5.14}$$

$$\#modeb(holdsAt(at\_curr\_location(var(person)), var(time))). \tag{5.15}$$

$$\#modeb(abrupt\_transition(var(time), var(time))). \tag{5.16}$$

$$\#modeb(next\_time(var(time), var(time))). \tag{5.17}$$

We construct the rule $next\_time(T, T+1) :- time(T), time(T+1)$, so that the predicate $next\_time(T1, T2)$ can be used in 5.17. This is to represent the increment in time that currently could not be generated with an arithmetic '+' in mode declarations by ILASP version 4.

The expected rule learnt should be similar to the follows:

$$\begin{aligned}&initiates(enter(P), at\_curr\_location(P), T) :-\\ &\quad not\ holdsAt(in\_camera(P), T2), holdsAt(in\_camera(P), T),\\ &\quad not\ abrput\_transition(T2, T), next\_time(T2, T),\\ &\quad not\ holdsAt(at\_curr\_location(P), T).\end{aligned} \tag{5.18}$$

This expected rule 5.18 gives the following conditions for an entering action to happen:

- A person is not in the camera at time $T-1$ but shows up at time T;
- The change of frame $T-1$ to frame T is not an abrupt transition;
- The person is not at this scene/location at time T.

If all conditions are satisfied, then the action happens and we initiate the fluent of a person being at the current location.

Weighted Context-Dependent Partial Interpretations (WCDPIs)

The purpose of using weighted context-dependent partial interpretations is to tolerate noises in the examples by assigning a penalty to each of them. The creation of the WCDPI's identifier and its penalty is very similar for both concepts. We use the question id (qid), with the current timestamp if learning 'holding', as the example id. All examples' contexts are constructed from ground truth annotations, thus having equal penalties.

For the holding action, as the ground truth bounding boxes are grouped by timestamp, we create a WCDPI for each unique timestamp. The number of WCDPIs depends on the number of timestamps that has bounding boxes annotation, and this varies between questions. On average, we can generate 3 WCDPIs for one question-answering sample. The context consists of humans, objects, their bounding boxes and any intersections. These spatial features from the training data are directly converted into logical facts through the symbolic translation component. We also specify the current time and the goal holding action that answers the question. These facts are obtained after parsing the questions' and answers' text and are to be used together with Constraints 5.7. By parsing the text, we can also construct the inclusion and exclusion set of the WCDPIs. The person in the question would be in the inclusion set and holding the answer object, while any other characters in the same frame should fall into the exclusion set. The exclusion set might not have included all possible combinations of grounded *holding*() predicates that are not a goal and have to be false. However, Constraints 5.7 in our background knowledge enforce that all negative combinations must be excluded. Thus, with these constraints in the background knowledge, exclusion set not being complete would not affect the learning.

Below is an example of translating ground truth bounding boxes at time 13 (shown in Figure 5.1) to a WCDPI:

Figure 5.1: Question 67301's ground truth annotation at frame 13.

```
#pos(p_67301_13@10, {
    holdsAt(holding(penny, purse), 14)
```

```
}, {
    holdsAt(holding(sheldon, purse), 14)
}, {
    goal(holdsAt(holding(penny, purse), 14)).
    current_time(13).
    person(sheldon).
    person(penny).
    object(purse).
    bbox(13, penny, 120, 35, 89, 291).
    bbox(13, sheldon, 439, 22, 62, 223).
    bbox(13, purse, 126, 76, 63, 113).
    bbox_intersec(13, penny, purse, 28).
    bbox_intersec(13, purse, penny, 100).
}).
```

The construction of WCDPI examples for learning 'entering' is relatively simple, with one example per video clip. Apart from putting the relevant time span of the questions given by TVQA+ into the context, we also add which characters are in-camera and the timestamps for their appearance, who are already in the scene when the period starts, and the abrupt shot transitions in the interval. Entrances of people in the answer should be wrapped in an *initiates*() predicate and be put in the inclusion set. Any characters that are already at the current location at the start of the period would not enter the scene at any point unless they leave and re-enter in the time span. Such edge case are not selected for any clip used in the entering learning task. Thus, the exclusion set contains the *initiates*() predicate for every already-in-scene character at all timestamps. An example of WCDPI constructed from ground truth for 'entering' task is as follows:

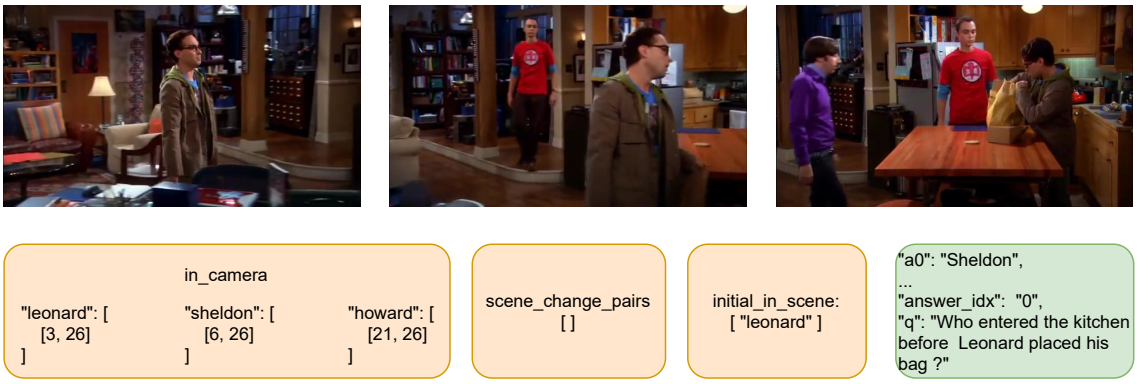


Figure 5.2: Frame sequence and ground truth annotations of question 25625604.

```
#pos(p_25625604_0@10, {
    initiates(enter(sheldon), at_curr_location(sheldon), 6)
}, {
    initiates(enter(leonard), at_curr_location(leonard), 3..26)
}, {
    time(3..26).
    person(howard).
    holdsAt(in_camera(howard), 21..26).
    person(leonard).
    holdsAt(in_camera(leonard), 3..26).
```

```
    person(sheldon).
    holdsAt(in_camera(sheldon), 6..26).
    holdsAt(at_curr_location(leonard), 3).
}).
```

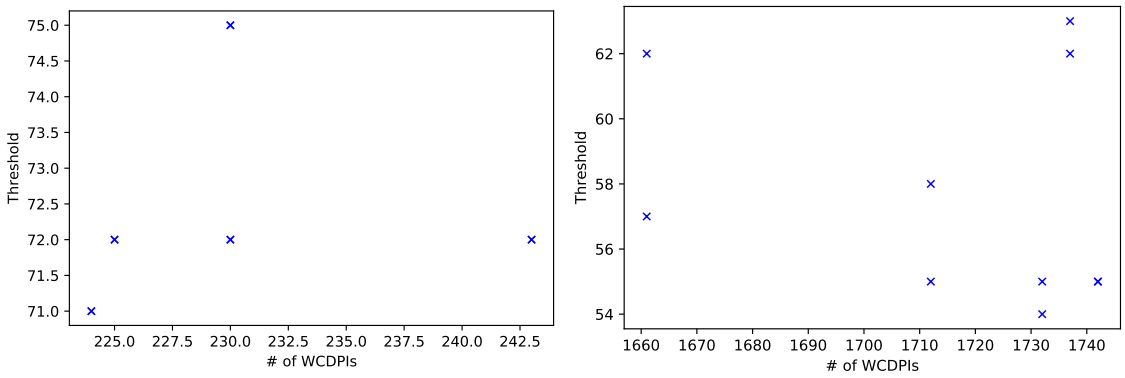
Learned Rules

With the background knowledge B and the set of positive WCDPIs E^+, ILASP searches through the hypotheses space S_M for the optimal hypothesis H that gives the least penalties when trying to cover as many $e \in E^+$ as possible.

Training with the 'holding' subset we create from TVQA+, ILASP learns the following hypothesis:

$$initiatedAt(holding(P, O), T) :- happensAt(close(P, O, 75), T). \tag{5.19}$$

The threshold varies with different number of WCDPIs provided, in a range of $[54, 75]$. From Figure 5.3 we can see that the threshold is lower when more examples are provided, and there are multiple choices for the same number of WCDPIs while resulting in the same penalties. But in general, a lower threshold is needed for accepting more WCDPIs.



(a) (b)

Figure 5.3: Threshold against number of WCDPI examples plots. (a): Threshold-against-WCDPIs-count plot for subset of 'holding' questions with COCO objects as answer, generated during 5-fold cross validation. (b): Threshold-against-WCDPIs-count plot for all 'holding' questions, generated during 5-fold cross validation.

Compare to our expectation 5.12, Hypothesis 5.19 omit *not* $holdsAt(holding(P, O), T)$. This is a more general rule than 5.12, but as Section 6.3 shows later on in the report, this general rule is sufficient for our task. The reason that ILASP learns a more general rule could be due to the fact that we construct the WCDPIs only on two consecutive timestamps. If using longer time interval in WCDPIs, ILASP should pinpoint the exact timestamp that initiates holding, as after the fluent is initiated, there is no need to initiate again. The hypothesis would not need to care about multiple initiations of 'holding' if there are only two timestamps to reason about, and could just omit *not* $holdsAt(holding(P, O), T)$. If two rules have the same penalties, the shorter rule would be optimal. Thus ILASP picks Hypothesis 5.19.

When training for the 'entering' concept, ILASP learns a consistent hypothesis regardless of the number of WCDPIs provided (8 when training for cross-validation on train set, 10 when training for evaluation on validation set):

$$initiates(enter(P), at\_curr\_location(P), T) :- \\ holdsAt(in\_camera(P), T), not\ holdsAt(at\_curr\_location(P), T). \tag{5.20}$$

In 5.18, we expect the use of $not\ abrupt\_transition(T1, T2)$ to prevent false positives of recognising entering action. However, ILASP decides it is sufficient to cover most WCDPIs without using it and gives a more general hypothesis 5.20. One explanation is that $not\ holdsAt(at\_curr\_location(P), T)$ subsumes $abrupt\_transition()$: once we know that someone is not at the current scene and appears at the current timestamp, knowing whether there is an abrupt transition becomes unnecessary. On the other hand, by looking at the video clips in detail, many 'entering' moments happen right at an abrupt shot transition in our training samples. If using our human-expected rule 5.18, the 'entering' action would never be satisfied and this would mean that a WCDP would not be covered with the hypothesis. While human perception under common sense is not completely accurate in this scenario, ILASP has made the optimal decision based on the nature of the video clips.

5.2 HACR Symbolic Reasoning Task

We construct and execute ASP programs during inference, and by parsing the answer sets, we obtain the predicates relevant to the question and extract the answer from it.

To create the ASP program, we inject the rules learnt with ILASP into the background knowledge to form the *base program*. Since some rules or constraints are used only for learning, they are discarded for simplification; for example, Constraints 5.7 are omitted. The final ASP program is built upon the base program with symbolic representations of spatial and temporal properties. These representations form the *context program*. The context program is constructed by extracting visual and textual features with frame processing and language processing respectively. Similar to the base program, non-essential predicates like *goal*() for the 'holding' learning task are not included in the context program. We provide some examples of the base and context programs for 'holding' and 'entering' in the following:

Base program for 'holding' concept:

```
fluent(holding(P, O)) :- person(P), object(O).
happensAt(close(P, O, Threshold), Time) :-
    person(P), object(O),
    bbox(Time, P,_,_,_,_),
    bbox(Time, O,_,_,_,_),
    bbox_intersec(Time, O, P, Intersection),
    holding_threshold(Threshold),
    Intersection >= Threshold.
holdsAt(F,T + 1) :-
    fluent(F), time(T), time(T + 1),
    initiatedAt(F,T), not terminatedAt(F, T).
holdsAt(F,T + 1) :-
    fluent(F), time(T), time(T + 1),
    holdsAt(F,T), not terminatedAt(F,T).
next_time(T, T+1) :- time(T).
holding_threshold(1..100).
initiatedAt(holding(V3,V2),V1) :- happensAt(close(V3,V2,72),V1).
```

Context program for 'holding' concept:

```
time(109).
time(110).
bbox(109, penny, 142, 31, 157, 328).
bbox(109, person_1, 334, 45, 166, 307).
bbox(109, cup1_1, 185, 191, 36, 46).
bbox(109, bowl1_1, 14, 241, 67, 25).
bbox(109, book1_1, 595, 309, 20, 50).
bbox_intersec(109, penny, cup1_1, 3).
bbox_intersec(109, cup1_1, penny, 100).
person(penny).
object(cup1_1).
```

Base program for 'entering' concept:

```
fluent(in_camera(P)) :- person(P).
fluent(at_curr_location(P)) :- person(P).
holdsAt(F, T + 1) :-
    time(T), time(T + 1), fluent(F),
    initiates(A, F, T), not clipped(F, T).
holdsAt(F, T + 1) :-
    time(T), time(T + 1), fluent(F),
    holdsAt(F, T), not clipped(F, T).
clipped(F, T) :-
    fluent(F), time(T),
    terminates(A, F, T).
next_time(T1, T1 + 1) :- time(T1), time(T1 + 1).
initiates(enter(V1),at_curr_location(V1),V2) :-
    holdsAt(in_camera(V1),V2); not holdsAt(at_curr_location(V1),V2).
```

Context program for 'entering' concept:

```
time(71..101).
abrupt_transition(74, 75).
abrupt_transition(95, 96).
holdsAt(in_camera(leonard), 72).
holdsAt(in_camera(leonard), 74).
holdsAt(in_camera(raj), 75..95).
holdsAt(in_camera(sheldon), 96..100).
holdsAt(in_camera(stuart), 100).
holdsAt(in_camera(sheldon), 101).
holdsAt(in_camera(leonard), 101).
person(leonard).
person(raj).
person(sheldon).
person(stuart).
holdsAt(at_curr_location(raj), 71).
holdsAt(at_curr_location(leonard), 71).
```

We combine the base and context program to get the full ASP program for answering a question. We solve this combined ASP program with clingo [16] to get the answer set. For each concept, we only focus on the primary predicates representing the concepts, which

are $holdsAt(holding(P,O),T)$ and $initiates(entering(P),at\_curr\_location(P),T)$. We match regular expressions onto them to get the answers to the questions: the object O in $holdsAt(holding(P,O),T)$ for 'What is *someone* holding...', and the person P in $initiates(entering(P),at\_curr\_location(P),T)$ for 'Who enters the scene...'.

Chapter 6

Evaluation

In this chapter, we detail the evaluation process of our HACR pipeline. We first list out our experiment settings, including the machine's spec, evaluation methods, and datasets. Then we present our evaluation results on individual components used in the approach, following with the overall pipeline.

6.1 Experiment Settings

All the experiments are run on a 64-bit Ubuntu 20.04.2 machine with 32GiB memory and a 6-core-12-thread Intel ® Core™ i7-8700 CPU @ 3.20GHz. The GPU used is NVIDIA GeForce GTX 1060 with 6GB memory.

There are two general principles for our evaluation process:

1. Use 5-fold cross-validation if evaluating on a training set: We randomly split the training data into 5 folds, 4 folds of data for actual training and the rest for evaluation, and repeat this 5 times with different fold for evaluation. The overall performance would be the average of all folds' result.

2. Train on all training data if evaluating on a validation set: If any data not in the training set is used for evaluation, we should train the pipeline on the whole training data and evaluate for the final result.

Below are the names of the data we collect for training/evaluating individual components or overall pipeline:

- **ABT-Test**: This test set is constructed with 10 randomly selected videos from the TVQA+ data, and we manually annotate all the abrupt shot transitions in each clip.

- **Train-Hold-ALL**: This training set is selected from the TVQA+ training set by checking the question's root action word to be 'hold' (detail in Section 4.1.1). We also remove the edge cases of questions that cause parsing error, resulting in 886 samples in total.

- **Train-Hold-OD**: This is a subset of Train-Hold-ALL with 91 samples. For each question, its answer object must be in the COCO category. The constraint is due to the object detector pre-trained on the COCO dataset (more detail in Section 4.2.1).

- **Val-Hold**: With the same selection process as Train-Hold-ALL, we collect 104 samples from the TVQA+ validation set. We do not further filter the questions with COCO objects in order to get a more general view of the pipeline's performance.

- **Train-Enter-HACR**: This is the hand-crafted training set with 10 questions for learning 'entering the scene', as mentioned in Section 3.2. As the format is slightly different from the TVQA+ data, this dataset cannot be fed into other existing neural pipelines directly.
- **Val-Enter-TVQA+**: We select 3 samples from the TVQA+ validation set. These questions are not exactly in the form of 'Who enters the scene...' but ask 'Who enters *location*...' and do not involve any location changes when a new character enter. We consider them very similar to our 'entering the scene' concept and use them to construct the validation set. We also annotate the abrupt transitions, which characters appear in each frame and are initially at the location.

Most existing neural approaches on the TVQA+ dataset are evaluated on the whole TVQA+ validation or test sets. Since our work only focuses on a proportion of the concepts, we can only assess our pipeline on subsets of the TVQA+ dataset or our own sets, making our result incompatible with neural architectures' evaluation results. Therefore, we run the pre-trained iPerceive Video Q&A model [27][10] on our subsets to ensure a fair comparison. While we tried to produce results for iPerceive Video Q&A with their best model configuration, we could not enable the common-sense reasoning component of the original work due to missing file from their GitHub repository. However, as shown in Table 7.4 in the later chapter, the common-sense reasoning boosts QA accuracy by only around 1%, and thus this missing component issue should not significantly worsen the performance of this neural approach.

6.2 Pre-processing

We evaluate the individual components in the frame processing with the ground truth annotations either from the TVQA+ dataset or us. Since language processing is used together with other processes, we do not evaluate it individually but together with the pipeline.

6.2.1 Object Detection

We use the analysis tool introduced in [28][11] to evaluate our pre-trained Faster R-CNN from `torchvision`. The tool provides computation of the common metrics for object detections: Average Precision (AP). This is defined as:

$$\text{AP} = \int_0^1 \rho(r)dr \quad \text{where } \rho(r) \text{ is the Precision-Recall curve} \tag{6.1}$$

The Precision-Recall curve usually has a zig-zag pattern and is smoothed down by interpolation to simplify the calculation. The interpolation is defined as:

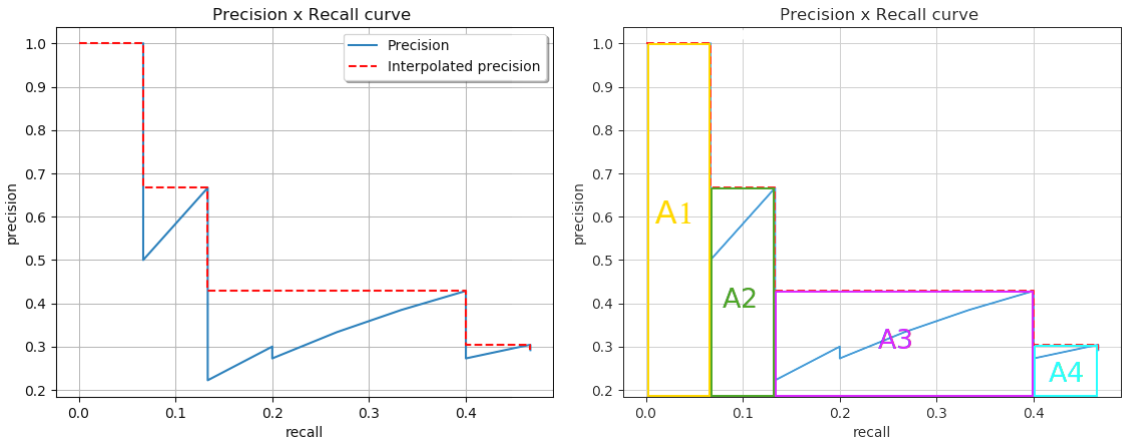
$$\rho_{\text{interpolation}}(r_{n+1}) = \max_{\tilde{r}:\tilde{r}\geq r_{n+1}} \rho(\tilde{r}) \quad \text{where } \rho(\tilde{r}) \text{ is the value of the precision value at recall } \tilde{r} \tag{6.2}$$

Based on the interpolation, AP is calculated by summing up the area underneath the smoothed curve:

$$\text{AP} = \sum (r_{n+1} - r_n)\rho_{\text{interpolation}(r_{n+1})} \tag{6.3}$$

[10]The implementation of their work is at https://github.com/amanchadha/iPerceive
[11]The implementation is at https://github.com/rafaelpadilla/Object-Detection-Metrics

Figure 6.1 from [28]'s GitHub repository provides visual demonstrations of the calculation of AP.



(a) (b)

Figure 6.1: Object-Detection-Metrics [28] official demonstration of AP calculation, cited from their GitHub repository[12]. (a): Precision-Recall curve with interpolation. (b): Precision-Recall curve with AP calculation.

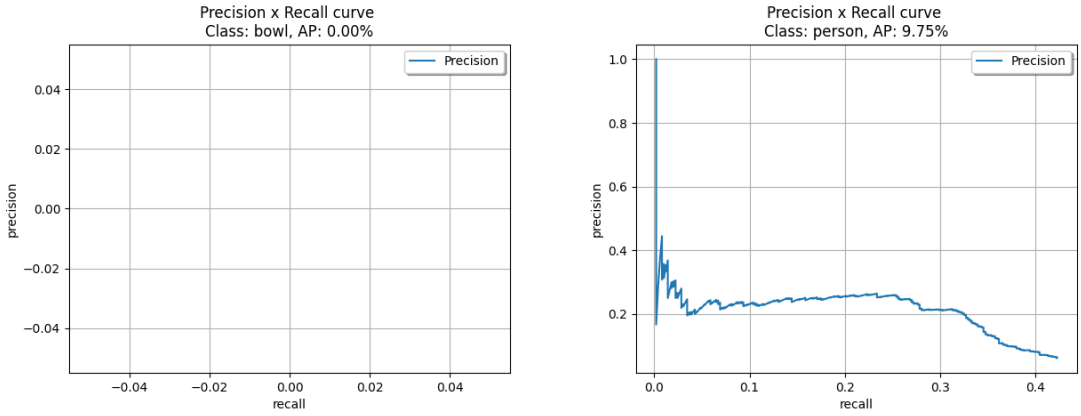
We randomly sample 100 samples from our Hold-ALL training set. For each video, the ground truth annotation provides the bounding boxes at a few timestamps, and we feed the frame of each time value to the pre-trained model. Object-Detection-Metrics compares the outputs and ground truth per frame per video and computes each class's AP score across all test instances. Table 6.1 shows the performance of the model on some of the unique classes and the overall performance. As we can observe, the detection performance is not impressive. The mAP score, which measures the average AP score across all classes, is significantly dragged down by the vast majority of classes with a 0 AP score.

Example classes	Basket	Bowl	Person
AP	0.00%	0.00%	9.75%
Total number of classes		93	
Number of classes with 0.00% AP		84	
HACR mAP		0.18%	
STAGE[12] mAP		25.22%	

Table 6.1: Evaluation for object detection using Object-Detection-Metrics [28]. There are in total 93 classes in the subset we tested and we list out 3 classes' AP scores and the overall mAP (mean Average Precision). We also compare our pre-trained object detector with the STAGE architecture proposed in [12] with attention mechanism. The significant difference shows that the pre-trained object detector is the main bottleneck of our approach and needs to be fine-tuned and more specific.

As mentioned in the Limitation part of Section 4.2.1, the model could only recognise objects that fall in the COCO category and struggles with finding small and partially covered items. The first class, 'Basket', is not included in the COCO dataset, making the model unable to detect it at all. Bowl is one of the objects included in the COCO dataset, yet the model still results in a 0 AP score. Figure 6.3 shows the case of the object detector failing to identify the yellow bowl held by Leonard and also covered by a kettle. This is

[12]GitHub: https://github.com/rafaelpadilla/Object-Detection-Metrics



(a) (b)

Figure 6.2: Precision-Recall curve for 'bowl' and 'person'.

(a) (b)

Figure 6.3: Model prediction and ground truth for a frame with a person and a bowl, with the left coming from model prediction and the right coming from ground truth annotation. There is a false negative for class 'bowl' (not detecting it) and also a false positive for class 'person' (detect something when there is not).

a false negative and also the only instance of 'bowl' in the 100 samples. The tool records every true positive and false positive to generate the precision-recall curve, and thus the false negative is not used for plotting the graph, resulting in a 0% AP score.

For the 'person' class, the ground truth from TVQA+ annotates characters by their names. To match our more general predictions, we change the ground truth labels to 'person' for every name included in the set of characters we define. This way, we evaluate the performance of object detector on humans without using face classification. Precision goes down when the number of false positives increases. We expect some precision drops since there are names in the ground truth labels that we could not recognise and translate to 'person'. However, as shown in Figure 6.2b, the precision of the 'person' class decreases significantly at 0 recall, indicating performance problem on the object detector's side apart from just mislabelling.

6.2.2 Face Detection and Classification

The TVQA+ dataset does not provide specific face localisation annotations, and we do not annotate the dataset further for evaluating the OpenCV Haar cascade model. However, we can still observe from Figure 4.7 that there are 3 instances of false localisation in the 40 examples, which provides some insight into the pre-trained model's performance. In most cases, the model gives the correct localisation while it could mistake horizontal clothes

wrinkles as human facial features. As our face collection relies on the Haar cascade model, these wrong localisations introduce noises in the collected data.

For the classification, we evaluate the KNN classifier with 5-fold cross-validation on our face samples collection. For each fold, we use 80% of the data to train the classifier and 20% for testing, and the predictions are compared to the ground truth labels. Table 6.2 presents the accuracy for each fold and the average, and Figure 6.4 is the confusion matrix of the the first fold in the cross-validation.

Fold Acc.	0.899	0.898	0.900	0.898	0.900
Train set size			53628		
Test set size			13407		
Avg. Acc.			0.899		

Table 6.2: 5-fold cross-validation result for KNN face classifier, accuracy to 3 s.f.

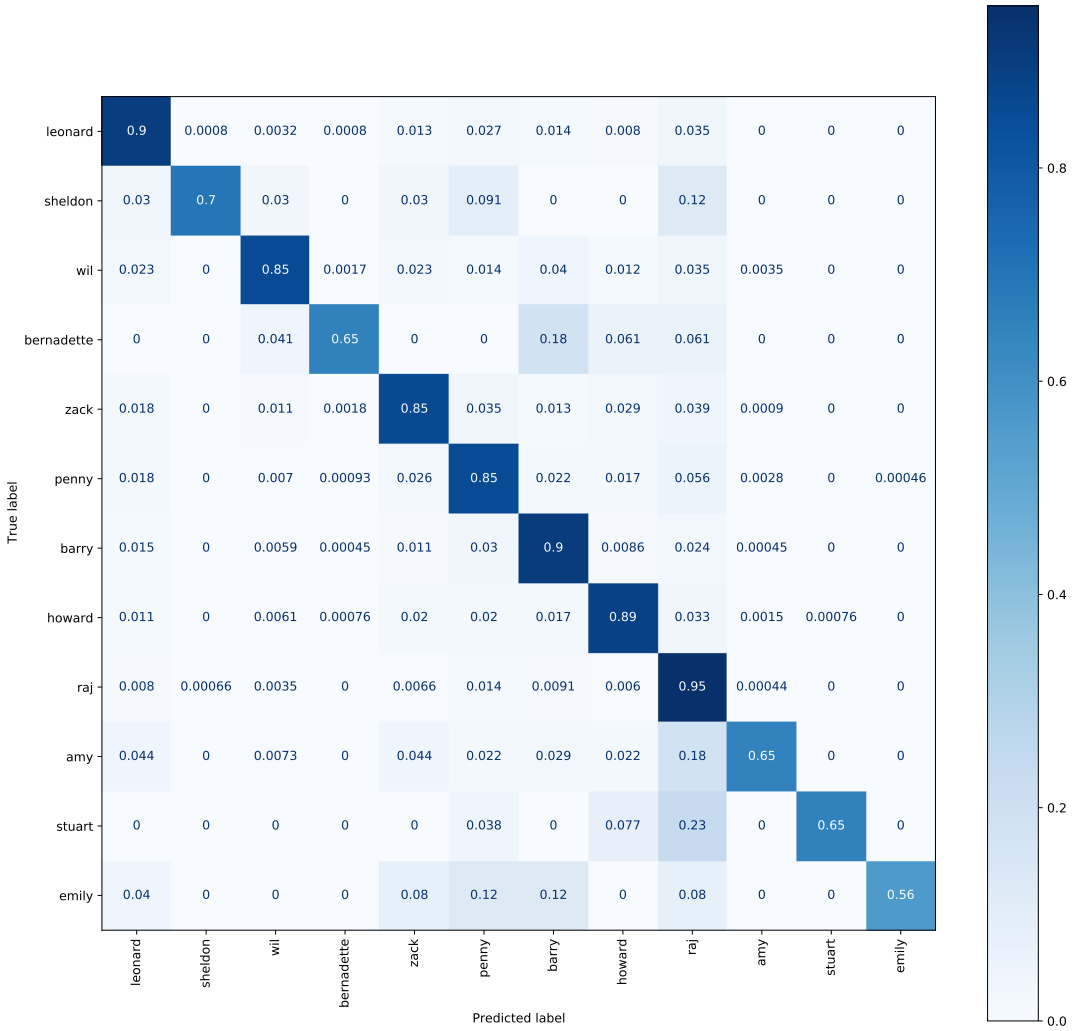


Figure 6.4: Confusion matrix for face classification over 12 classes of characters for the first fold in the 5-fold cross-validation.

The performance of the face classifier is overall stable and reliable. However, we can still observe poorer performance for certain characters. This issue could be caused by the imbalanced dataset, as the frequency of characters appearance varies in the dataset, making some people having more samples than others.

6.2.3 Abrupt Transition Detection

We use the ABT-Test subset we annotated to evaluate our abrupt transition detection process. We compute the dissimilarity score with one of the SAD, Hist-All and Hist-Reg methods for all the consecutive frame pairs in a given video clip.

Since there is no ground truth label associated with the score, we train two unsupervised models, the k-means cluster model and Gaussian mixture model, to fit the data. The number of training samples collected with each scoring approach is listed in Table 4.2, where more SAD training samples are available. Each fitted model gives a label to a dissimilarity score at test time. We expect the models to label a score with 0 value as a non-abrupt transition since this value indicates no transition. The label for score 0 would be picked as a negative class label, and a predicted label that shares the same label as score 0 is treated as a non-abrupt transition prediction. Otherwise, the frame pair is predicted as an abrupt transition. We also evaluate whether non-maximum suppression (NMS) affects the performance of the models.

We compute two metrics to measure the performance of the overall detection process:

- **Normalised Jaccard score**: We gather all the predicted pairs of abrupt transitions in a video clip and compute the Jaccard coefficient index score with respect to the ground truth pairs. We normalise the Jaccard score across all video clips.

- **Normalised binary classification accuracy**: The number of abrupt transitions in a given video clip is significantly smaller than that of non-abrupt transitions. Therefore, we balance the data so that the same number of pairs is picked for both abrupt and non-abrupt transitions. We select all the abrupt pairs from the ground truth for each video clip and randomly sample the same number of non-abrupt transition pairs. These together become the test sample pairs for a video clip, and we compute the average accuracy across all clips.

Binary classification accuracy provides an insight into how well the model would perform under a general setting where dissimilarity scores are randomly given. In contrast, the Jaccard score shows how similar the model's prediction of a whole video clip is compared to the ground truth.

	k-means cluster model	Gaussian mixture model
SAD	0.170	0.259
Hist-All	0.974	0.406
Hist-Reg	0.867	0.333

Table 6.3: Normalised Jaccard score of all abrupt transition detection models **without** applying non-maximum suppression, to 3 d.p.

	k-means cluster model	Gaussian mixture model
SAD	0.338	0.413
Hist-All	0.982	0.632
Hist-Reg	0.936	0.084

Table 6.4: Normalised Jaccard score of all abrupt transition detection models **with** non-maximum suppression applied, to 3 d.p.

	k-means cluster model	Gaussian mixture model
SAD	0.861	0.879
Hist-All	0.924	0.970
Hist-Reg	0.994	0.500

Table 6.5: Normalised binary classification accuracy across of all abrupt transition detection models **with** non-maximum suppression applied, to 3 d.p.

By comparing Table 6.3 and 6.4, we can see increases in normalised Jaccard score for most models, showing that applying NMS improves the performance. Table 6.4 and 6.5 shows the performance of models with NMS applied, measured in normalised Jaccard score and binary classification accuracy respectively. Generally, k-means clustering models outperform Gaussian mixture models, and using histogram-based scoring gives more reliable results than SAD, even with fewer training samples. We can observe that SAD models perform well when randomly giving a score but poorly as a whole when evaluated on a video clip. Hist-ALL k-means cluster and Hist-Reg k-means cluster have very similar and impressive results. We choose to use Hist-ALL k-means cluster with the higher normalised Jaccard score, indicating better predictions of the entire video clip.

6.3 Rule-learning

For both 'holding' and 'entering' concept learning, our training examples for ILASP are constructed from ground truth annotations. This approach removes the unnecessary noises that could prevent ILASP from learning a general and precise hypothesis that covers most real-life cases. The learned rule combines with the background knowledge to form the base program. Apart from the base, the context program representing the visual features from the frames is required to answer a question. To evaluate purely just how accurate the learned rule is, we minimise the inaccuracy in symbolic representations in the context program by using direct translations from ground truth annotations instead of constructing from scratch with our frame processing. We obtain the answer set by using clingo to solve the combined ASP program of the base program and context program with ground-truth features.

For the 'holding' concept, we parse the predicates to get all possible objects held by the character of the question. We try to match the objects with all the choices by checking synonyms, hyponyms or hypernyms. If an object matches one of the choices, we select that as our answer. Sometimes multiple objects are being matched, and thus the pipeline might select more than one answer. Similarly, for 'entering' concept learning, we parse the answer set to get all possible people who are concluded that have entered the scene. We match the people's names with all choices by simply comparing the exact names.

QA accuracy is the commonly-used metric to measure how well the model answers the multiple-choice question. Since we do not have confidence scores associated with the objects/people or a further heuristic to select a specific one as the final answer, we store all options selected by our symbolic reasoning in a list regardless of the number. This implementation choice means that we could have multiple predicted answers to a question, making QA accuracy incompatible with our model. Instead of using the accuracy, we adopt the Jaccard index score to measure the performance. For each question, the closer the Jaccard score is to 1, the closer our prediction is to the ground truth answer.

In Table 6.6, the column 'HACR With g.t. annot.' shows the general performance of HACR's rule learning ability. Our approach outperforms the iPerceive Video Q&A model with higher normalised Jaccard scores in all evaluation configuration for both concepts. This result shows the impressive generalisation and learning ability of the inductive learning

system. Moreover, the symbolic concepts representations are human-readable, showing higher interpretability compared to the neural model.

6.4 Overall Pipeline

In order to evaluate the overall pipeline, instead of translating ground truth annotations into the context program, we construct the context program based on features extracted through our frame processing. Like evaluation on rule learning, we keep all matched objects after synonym/hyponym/hypernym checking and use the Jaccard score as our metric. Column 'HACR with o.d./f.c.' in Table 6.6 shows the result of the overall architecture for all test configurations.

Concept	Test name	Eval method	# of Qs	iPerceive Video Q&A*	HACR	
					With o.d./f.c.	With g.t. annot.
Holding	Train-Hold-OD	5-fold	91	0.791	0.724	0.875
	Train-Hold-ALL	5-fold	886	0.729	0.161	0.741
	Val-Hold	Train on Hold-OD	104	0.539	0.143	0.793
Entering the scene	Train-Enter-HACR	5-fold	10	–	0.633	0.933
	Val-Enter-TVQA+	Train on Enter-HACR	3	0.667	0.333	1

Table 6.6: 'Holding' and 'Entering the scene' concept learning evaluation, performance measured in avg. Jaccard score.
iPerceive Video Q&A*: As we mentioned in Section 6.1, the model we run the experiments on is slightly different from the model reported in [27], since we could not enable the common-sense reasoning component. But the performance should be very close to the best model in [27].

For the task of 'holding', we can observe a significant drop in performance of the whole pipeline when moving from our Hold-OD dataset to Hold-ALL dataset. In contrast, the pipeline without any frame processing shows minor effects. The change in dataset introduces noises into the context program due to the use of frame processing. As discussed in Section 6.2.1, our object detector has very limited ability to provide reliable detection of various types of items and sometimes outputs false positives of human detection. Even when testing on Hold-OD where the answer objects fall into the COCO category, there is still a big gap of performance between the pipeline using object detector to get context programs and the pipeline using ground truth annotations. Similar observations are found for the 'entering' learning task: the pipeline with the object detector performs poorly compared to iPerceive Video Q&A and the pipeline that uses ground truth annotation in the context programs. These observations in both cases prove that the current object detector is the bottleneck for our pipeline, preventing the learned rule from being utilised fully.

Chapter 7

Related Work

In this chapter, we discuss existing models relevant to concept learning and Video Q&A. We first present pure statistical machine learning models focusing on the TVQA [21] and TVQA+ [12] datasets. As limited research is done on hybrid learning, we include architectures that aim for concept learning and Video Q&A problem, as well as some oracles in the field of neural-symbolic learning.

7.1 Neural Architectures for Video Q&A

Majority of the works on Video Q&A use neural architecture. While many models are developed, we only present the ones tested against TVQA dataset [21] or TVQA+ dataset [12] or both. Table 7.5 shows an overview and comparison of all models that we considered. We sperate these models into three main milestones, with the group of MSAN [29] and DHTCN [30] as a branch of the first milestone. We list out inspirational techniques introduced in each milestone. For more details of the models, we direct the avid readers to their original papers. We also discuss the neural approach's strengths and weaknesses in general at the end of this subsection.

Two-stream [21], PAMN [31], and Multi-Task [32]

These three models set the first milestone as TVQA [21] was first proposed, sharing some similar design features:

- **Feature fusion**. All three models implemented feature fusion involving different attention mechanism. Two-stream and multi-task models use context-query attention layer [33, 34], while PAMN uses dynamic modality fusion (explained more in the second point) based on dual memory embedding after progressive attention mechanism.
- **Modality alignment**. Both multi-task model and PAMN incorporate components specifically for modality alignment. Multi-task model uses a modality alignment network to match video with strongly supporting subtitle, with additional supervision. PAMN performs dynamic modality fusion to increase more relevant feature's contribution to the final output.
- **Temporal localisation**. To locate temporally relevant parts for question answering, multi-task model adds extra supervision via a temporal localisation network. PAMN achieves it during its progressive attention mechanism.

As the first few attempts with relatively simple architecture, these models perform reasonably worse in the comparison in Table 7.5. However, the design presented in them are highly influential and become fundamental building blocks for later works.

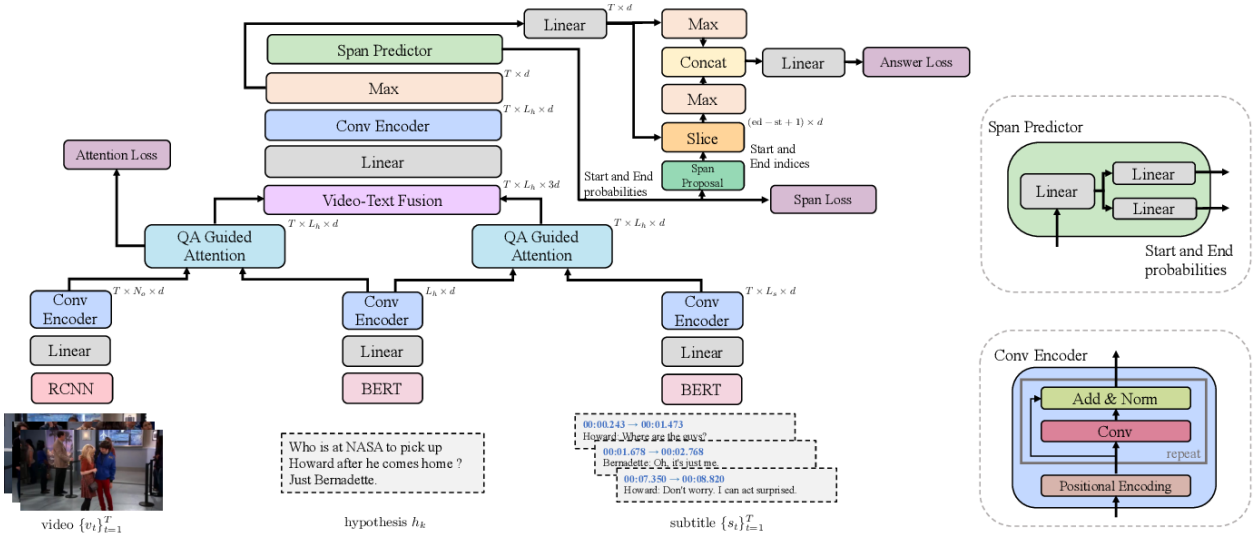


Figure 7.1: STAGE architecture, cited from [12]

STAGE [12]

TVQA+ dataset [12] is a subset of TVQA dataset [21] with moment localisation and object grounding annotation. STAGE became the second milestone with the addition annotations in TVQA+ and a more sophisticated design:

- **Convolutional encoder**. Consisting of positional encoding, CNN and layer normalisation, convolutional encoder acts as a recurrent network replacement. STAGE applies convolutional encoders to encode raw inputs as well as internal feature fusion.
- **QA awareness**. For each question-answer combination from one video clip, its encoding is used to compute attention score with visual features and subtitle features respectively. Feature fusion is then applied to these QA-aware representations.
- **Span prediction and span proposal**. Span predictor computes the probability of each fused input being the start and end of the time span. Dynamic programming [33] then makes several span proposals based on these probabilities.
- **Local and global features for question answering**. Generated by span proposals, local representations are combined with global representation to compute each answer's score.
- **Spatial and temporal supervision**. STAGE performs spatial-temporal supervision with the ground truth time span and object grounding boxes. In Table 7.3, we see that temporal supervision improves the model's overall performance, and spatial supervision brings a dramatic upgrade in object grounding precision.

	"what"	"who"	"where"	"why"	"how"
Question type percentage	60.52	10.24	9.68	9.55	9.05
QA Acc.	72.34	74.11	74.32	76.39	67.03

Table 7.1: TVQA+ val set results by question type [12].

With various enhancements on the original two-stream [21] architecture shown in Table 7.2, STAGE significantly outperforms its ancestor, providing a strong foundation for

future work. Furthermore, STAGE presents strong performance in "why" and "how" questions (Table 7.1), indicating some reasoning power in this architecture.

Model	QA Acc.	Grd. mAP	Temp. mIoU	ASA
STAGE (video)	52.75	26.28	10.9	2.76
STAGE (subtitles)	67.99	-	30.16	20.13
two-stream [21]	68.13	-	-	-
silkage*	72.14	-	30.68	20.99
Alchemistyui*	72.67	-	32.03	**22.94**
Anonymous_129*	73.80	-	-	-
lft3324581*	74.34	-	31.53	21.77
STAGE	**74.83**	**27.34**	**32.49**	22.23
Human	90.46	-	-	-

Table 7.2: TVQA+ test set results comparison. Models without labels and their metrics are cited from [12]. Models labelled with * and their metrics come from TVQA+ Codalab competition[13].
QA Acc. = QA performance accuracy; Grd. mAP = object grounding mean Average Precision; Temp. mIoU = span prediction temporal mean Intersection-oven-Union; ASA = Answer-Span joint accuracy

Model	QA Acc.	Grd. mAP	Temp. mIoU	ASA
baseline	65.79	2.74	-	-
+ CNN	67.25	3.16	-	-
+ Aligned fusion (backbone)	68.31	7.31	-	-
+ Temp. Sup.	71.40	10.86	30.77	20.09
+ Spat. Sup.	71.99	24.10	31.16	20.42
+ Local feature (STAGE)	72.56	25.22	31.67	20.78
STAGE with ground truth span	73.28	-	-	-

Table 7.3: Ablation analysis of STAGE on TVQA+ val set [12].

MSAN [29] and DHTCN [30]

These two models were proposed at a similar time as STAGE [12]. With less similarity with STAGE, MSAN and DHTCN were created based on architectures from the first milestone, but with advanced attention mechanism to improve overall performance.

- **Multi-head attention mechanism**. DHTCN adopts this mechanism from [35] to align different modalities. The aligned representation is then combined with bi-LSTM to form a component named AttLSTM. By applying it periodically in the pipeline, AttLSTM helps to generate fused representations in different scales.
- **Heterogeneous attention mechanism (HAM)**. Introduced together with MSAN, HAM combined three primary attentions (self-attention, context-to-query attention, and context-to-context attention) to seek modalities interactions. It acted as the core feature fusion unit in MSAN.

Although both models outperform STAGE, the improvement is marginal. Thus, we consider this group more an improved branch of the first milestone than its own milestone. However, the powerful attention mechanisms inspire the next group of models deeply.

[13] https://competitions.codalab.org/competitions/22705#results

hstar[14]**[36] and iPerceive [27]**

These third milestone models are highly influenced by the architecture of STAGE [12] and use dense caption as an additional modality to improve performance further:

- **Dense caption**. Dense caption encodes actions of objects, providing another way to correlate object and time. hstar and iPerceive each have a dense caption generator. hstar uses the pre-trained model from [37]. iPerceive's dense caption generator (iPerceive DVC) is built upon [38] with a common-sense reasoning model (more detailed in the third point).
- **Dual-level attention with multi-head self-attention**. Both architectures utilise this module. Dual-level attention generating process consists of two steps. QA-aware subtitle and visual features are first computed with word/object-level attention. These two attended representations are then aligned onto frames with frame-level attention. Dual-level attention is applied twice to generate frame-level attentions for video and dense caption. They are finally fused via multi-head self-attention.
- **Common-sense reasoning**. Based on causality reasoning in [39], iPerceive utilise common-sense generation in both dense caption generation and fusion with visual features. Table 7.4 shows that baseline iPerceive performs better with common-sense reasoning than with additional dense caption. It suggests that reasoning may be more critical in Video Q&A than additional modality.

As state-of-the-art in TVQA dataset, iPerceive utilises additional modality and a neural reasoning component, which provides a direction of incorporating reasoning in next-generation models.

Common-sense reasoning	iPerceive dense caption	QA Acc.
✗	✗	74.20
✗	✓	75.42
✓	✗	75.55
✓	✓	76.97

Table 7.4: Ablation analysis of iPerceive VideoQA on validation set[27].

In summary, neural approaches' power has shown to be growing and showing promising results for Video Q&A. The current trend consists of applying additional spatial-temporal supervision, and modality fusion with attention mechanism. Furthermore, iPerceive suggests that common sense reasoning is more likely in improving the overall performance.

However, all these models are not interpretable and suffer low transparency, especially decomposability [40]. Each component and internal representations lack instinctive explanations. For example, common-sense reasoner in iPerceive outputs vector representation of knowledge, making it hard for humans to interpret.

Another problem is that the reasoning power of neural modules does not seem to be robust enough. Due to poor interpretability, we cannot verify if the models are reasoning correctly. But evidence could be found in the ablation study of STAGE and iPerceive. The ASA metric in [12] is calculated as the probability $P(\text{Predicted span IoU} \geq 0.5 \mid \text{correct answer prediction})$. In Table 7.2 and 7.3, STAGE's ASA results are generally low, suggesting that the model might not be correctly reasoning based on temporal features. On the other hand, the improvement of common sense reasoning is marginal in

[14] Although hstar is the name of [36]'s submission name on CodaLab, the authors didn't give the model a name and we choose to use hstar to distinguish the model from the technique of dense caption.

iPerceive's ablation analysis (Table 7.4). While some may argue that the QA performance is a strong indication of reasoning, we still believe that symbolic learner with transparency and interpretability would be a more robust approach.

Model	Text Feat.	Video Feat.	Additional supervision	Attention	Val. set Acc.	test-public (w/o timestamp) Acc.
two-stream [21]	GloVe	vcpt, reg, img	-	context-to-query attention	65.85	66.46
PAMN [31]	GloVe	vcpt	-	progressive attention	66.38	66.77
multi-task [32]	GloVe	vcpt, img	temporal, modality alignment	C2Q attention	66.22	67.05
STAGE [12]	BERT	reg	temporal, spatial	hard attention	70.50	70.23
MSAN [29]	BERT	vcpt, acpt	-	Heterogeneous Attention Mechanism	70.79	71.13
DHTCN [30]	BERT	vcpt	-	multi-head attention	71.15	71.48
hstar [36]	Glove, RoBERTa	reg	temporal, spatial	dual-level, multi-head	74.20	74.09
iPerceive [27]	Glove, RoBERTa	reg	temporal, spatial, common sense reasoning	dual-level, multi-head	**76.97**	**75.15**

Table 7.5: Various models comparison on TVQA dataset. Follow the convention in [29], "img", "reg", "vcpt", "acpt" mean ImageNet feature, regional feature, visual concept feature and action concept feature respectively.

7.2 Neural-Symbolic Approaches

Neural-symbolic approaches have gained more attention in the last three years. We categorise current relevant works into two main categories: hybrid systems and integrated frameworks. We first introduce hybrid models targeting at either concept learning or Video Q&A. We then discuss existing neural-symbolic computation frameworks DeepProbLog [41] and NSL [42]. We direct the avid readers to the models' original papers for detailed implementations. Advantages and drawbacks for each category are discussed separately in its section.

7.2.1 Hybrid Systems

To our knowledge, there is no direct baseline for hybrid approach on TVQA and TVQA+. However, there are several inspirational hybrid systems in Visual Question Answering on

CLEVR dataset [43] and Video Question Answering on CLEVRER dataset [6]. All models provide some level of technical background towards a high-cognition hybrid model.

NS-VQA [44] and NS-CL [5]

CLEVR dataset [43] consists of synthetic images of simple objects with controlled bias and detailed annotations. The overall architecture of NS-VQA is relatively simple, with only three components: scene parser, question parser and program executor.

- **Scene parser**. A Mask R-CNN [45] first generates object segment proposals. A ResNet-34 [46] then extracts spacial properties based on the segment proposals and the original image.
- **Question parser**. The parser plays the role of program synthesis in the pipeline. It is implemented as an attention-based sequence to sequence model in an encoder-decoder style. The parser takes in the question text and output token sequence that shares the same representation as [43].
- **Program executor**. The program executor contains the logic operations for the questions. The tokens from question parser are translated into functional modules and executed sequentially on the spatial representations from scene parser. The last function module outputs the answer to the question.

Model	Type	Count	Exist	Compare Number	Compare Attribute	Query Attribute	Overall
Humans [43]	-	86.7	96.6	86.4	96.0	95.0	92.6
DDRprog* [47]	Neural	96.5	98.8	98.4	99.0	99.1	98.3
MAC* [48]	Neural	97.1	99.5	99.1	99.5	99.5	98.9
TbD+reg+hres* [49]	Neural	97.6	99.2	99.4	99.6	99.5	99.1
NS-VQA (270 prgrams) [44]	Hybrid	**99.7**	**99.9**	**99.9**	**99.8**	**99.8**	**99.8**

Table 7.6: CLEVR models comparison. * indicates that the model is trained on all program annotations.

As shown in Table 7.6, NS-VQA outperforms other neural models and even humans with almost perfect accuracy in all question types among existing models. The model also generalises well in other datasets. In [44], the model is tested against a Minecraft dataset, still showing overwhelming performance with a small number of annotated programs (accuracy of 87.3% at 500 programs).

Studying on the same dataset, NS-CL focuses on visual concept learning. With similar components in the pipeline, NS-CL learns connections between visual representations with concepts based on functional program execution. Visual representation learning is optimised via backpropagation from the program executor. While the semantic parsing is not differentiable, NS-CL uses REINFORCE [50] to optimise its performance.

Although the task of VQA is more straightforward than Video Q&A, these two hybrid models laid down solid foundations for models on CLEVRER dataset.

NS-DR [6], DCL [7], and HySTER [8]

These three models study another popular Video Q&A dataset CLEVRER [6]. Inspired by CLEVR [43], CLEVRER dataset consists of artificial videos of simple objects and collision

events. It tests out physical and causal reasoning abilities based on spatial and temporal features.

NS-DR is the pioneer model for Video Q&A and neural-symbolic learning, proposed together with CLEVRER dataset. Compared to neural models presented in Section 7.1, NS-DR has a much simpler overall architecture built upon NS-VQA [44], shown in Figure 7.2. Visual and textual features are extracted separately without further fusion. The question parser is very similar to the one in NS-VQA. The visual parser is changed to a Mask R-CNN [45] based parser with ResNet-50 FPN [51] backbone. The output from video parser is passed down to the neural dynamic predictor to learn underlying physical concepts. The predictor is implemented with Propagation Network (PropNet) [52], and its output encodes object states and relations throughout the video. A hand-crafted program executor takes in the functional programs from question parser and executes them on event traces from the neural dynamic predictor. NS-DR significantly improved the accuracy in answering explanatory, predictive and counterfactual performance compared to neural architectures before it, as shown in Table 7.7. NS-DR also provides some level of transparency as the answering process in the symbolic executor is human-interpretable.

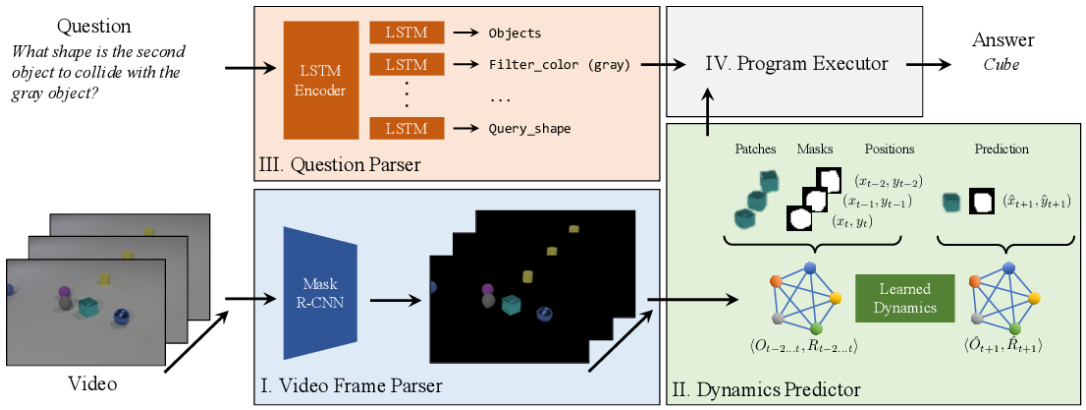


Figure 7.2: NS-DR architecture, cited from [6]

DCL adds more components on top of NS-DR's architecture to perform object tracking and trajectory refinement. Based on object detection, objects in consecutive frames are linked together to form trajectories. The dynamic predictor utilises the trajectory to optimise concept embeddings without labels on collision prediction. Furthermore, based on input video and trajectories, DCL extract three feature representations for object attributes, unary events and collision events, respectively. Finally, the symbolic executor runs the generated program from question parser on the features to predict an answer. From Table 7.7, we can see that trajectory prediction improves the overall DCL's performance, making it the state-of-the-art hybrid models. Moreover, DCL has shown to learn a new physical concept of 'falling' in real videos in [7]'s extension experiment.

The fundamental difference between HySTER from the previous two is its reasoning paradigm. Both NS-DR and DCL use PropNet [52] to learn underlying physics for collision events, whereas HySTER's reasoner uses symbolic rules implemented in Answer Set Programming (ASP) programs [9] and Event Calculus [14] for temporal reasoning. The video parser extracts object properties with spatial coordinates and encode them as facts. The predicate *on_camera* is used to encode spatial information of an object being in the frame at a certain time. Question parser translates questions into logic queries. The reasoner takes the scene representations and queries and combines them with encoded physics rules and task-specific event detection rules. The combined ASP program is solved by clingo [16] to get the final answer. Despite not being as accurate as DCL, HySTER has better interpretability compared to NS-DR and DCL. HySTER's collision rules learnt by the rea-

soner is encoded human-interpretable logic, whereas the PropNet's output is a collection of object states. Some may argue that it could be encoded with a direct graph where vertices represent objects and edges represent relations. However, what PropNet learns and how it learns is still hard to explain due to its neural architecture.

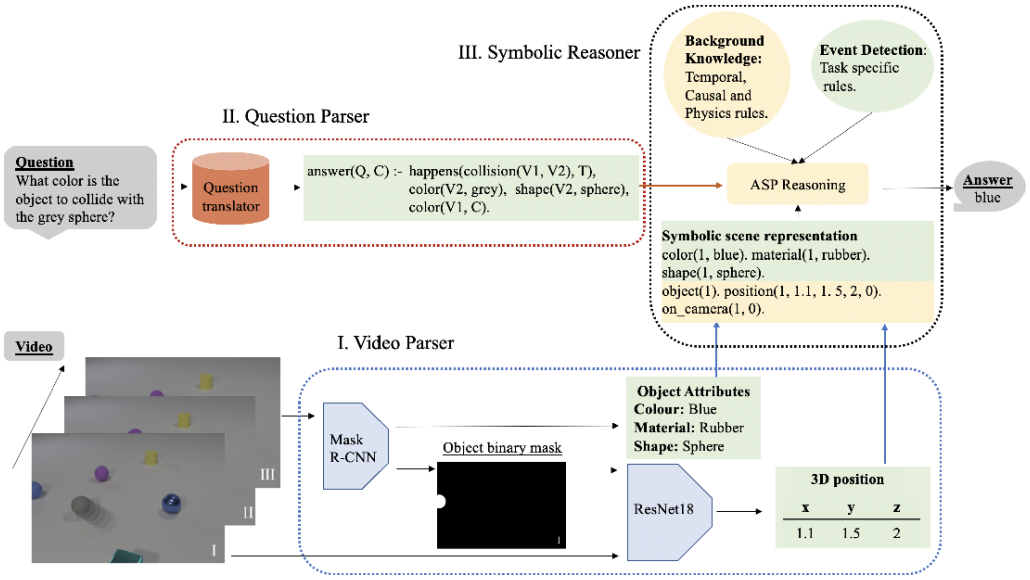


Figure 7.3: HySTER architecture, cited from [8]

Model	Symbolic component	Descriptive	Explanatory		Predictive		Counterfactual	
			per opt.	per ques.	per opt.	per ques.	per opt.	per ques.
STAGE [12]	-	72.0	63.3	23.7	70.3	48.9	53.9	4.1
MAC(V+) [48]	-	86.4	70.5	22.3	59.7	42.9	63.5	25.1
NS-DR [6]	Program executor	88.1	87.6	79.6	82.9	68.7	74.1	42.2
DCL-Oracle [7]	Program executor	91.4	89.8	82.0	90.6	82.1	80.7	46.9
HySTER-2(2D) [8]	ASP, Event Calculus	88.3	90.9	83.0	79.5	61.5	79.4	47.1
DeepMind Neural Model [53]	-	94.0	-	96.0	-	87.5	-	75.6

Table 7.7: Neural-symbolic models comparison on CLEVRER. The DeepMind model is used as baseline comparison.

Performance-wise, these three hybrid models show dominant performance over the classic neural Video Q&A models, until DeepMind's model with self-supervision, self-attention and soft (quasi)-discretisation came out recently [53]. Moreover, it requires much human-engineering to create the symbolic components, either the program executor or the background knowledge for ASP in HySTER. However, this does not suggest that neuro-symbolic architecture is not worth researching. ASP as an efficient model for collision events gives

us a direction of causal reasoning Video Q&A with hybrid systems. The system should utilise symbolic AI techniques such as Inductive Logic Programming (ILP) [54] that can learn while still maintains interpretability.

To summarise, the hybrid systems show decent causal reasoning and generalisation with better interpretability at the same time. However, the reasoning mechanism is not strong enough, and the system requires a large amount of human engineering. Both these aspects could be potentially improved by adopting symbolic learning paradigms like ILP.

7.2.2 Neural-Symbolic Frameworks

Neural-symbolic frameworks aim to provide a more general structure for different tasks. While various translation approaches could achieve neural-symbolic computation [10], lots of prior research is done on encoding logic into neural networks. Recent years, the focus has moved to combine neural and symbolic components in a hybrid way to enable learning and reasoning [55]. Both DeepProbLog [41] and NSL [42] fall into the category of hybrid framework.

- **DeepProbLog** [41]. DeepProbLog framework extents the ProbLog language with ground neural annotated disjunctions (nADs) to support neural networks. Its inference follows ProbLog's inference, with the special case of encountering neural predicates during grounding. In that situation, DeepProbLog performs a forward pass on the neural network to obtain the probabilities for the ground AD. One distinct property of DeepProbLog is its end-to-end differentiability. The loss of a DeepProbLog program w.r.t a query and its desired probability is calculated with the learning from entailment setting [56]. ProbLog program is transformed into an aProbLog program to compute the gradients and update parameters for probabilistic facts and ADs. Gradient descent in the neural network is done via traditional gradient optimisers. DeepProbLog combines symbolic, neural and probabilistic computation with an end-to-end manner. However, the logic rules are manually created then learned by the system itself. The framework might not scale well in Video Q&A, where the dynamics are exceptionally complicated to be fully covered with human-engineered rules.

- **NSL** [42]. In contrast, NSL uses scalable ILP system FastLAS [57] to learn rules from training data. NSL generates weighted context-dependent partial interpretation (WCDPI) examples based on features extracted by pre-trained neural component from unstructured data. The confidence of neural prediction is used as the penalty of a WCDPI example. NSL defines a penalty scoring function for FastLAS to maximise coverage and minimise hypothesis length at the same time. NSL also utilises FastLAS's ability to learn from noisy data and shown to be able to obtain robust rules in perturbed training data. Compared to DeepProbLog, although NSL is not end-to-end differentiable, it has outstanding scalability and data-efficiency due to the use of FastLAS, with little human-engineering. However, the neural component is required to be pre-trained and is not tunable during training. This property suggests that NSL might not be ideal for Video Q&A if we want to improve span proposal based on rules learned by the symbolic learner.

Although these frameworks might not be good fits for Video Q&A, it still addresses the robustness of logical reasoning as another reason for using neural-symbolic models.

Chapter 8

Conclusions and Future Works

We developed a neural-symbolic pipeline that learns and reasons about concepts through question answering on real-life videos. In this chapter, we first summarise our project's achievements and then discuss possible future works that could improve the system. Finally, we point out some ethical considerations of this project.

8.1 Achievements

In summary, our project has accomplished the followings:

1. **Multi-modality feature extraction**: Our frame processing extracts visual features from real-life videos such as people, objects and shot transitions. We obtain core events/actions related to the concept from text and recognise the relationship between words through language processing.

2. **Feature translation**: Features from both pre-processing and ground truth annotations can be translated into ASP programs. Translation of the ground truth is used to train and evaluate rule learning, while translation of the extracted features from video frames is used for reasoning during inference.

3. **Inductive rule learning**: We formalise the concept learning tasks into ILASP learning tasks. Apart from Event Calculus axioms, we provide little or no extra background knowledge to the system, and ILASP learns general hypotheses that capture the concepts. This learning process minimises human engineering in the symbolic component of the pipeline.

4. **Symbolic reasoning for question answering**: By combining symbolic feature representations, background knowledge and learned concepts at inference time, we extract the possible objects/characters from the answer set of the combined program. Finally, objects/characters that match with given choices are selected as part of the final answer to the question.

Although the overall HACR pipeline does not outperform existing neural approaches due to the object detection bottleneck, it has brought several exciting findings and improvements to the neural-symbolic research:

- **Automated symbolic rule learning**: Using the inductive learning system ILASP, we have learned accurate and general rules to represent concepts. Using these rules, HACR shows impressive performance if using ground truth features instead of pre-processing. This observation proves that ILASP offers powerful generalisation and high accuracy that can compete or even outperform statistical learning.

- **High interpretability and transparency**: Our internal representations and learned rules are all human-readable. Furthermore, our reasoning process is explainable. Compared to existing neural architectures, our approach has far more interpretability and transparency.
- **Ability to learn from real-life videos**: We show that the hybrid model is capable of handling Q&A on real-life videos, expanding on previous works that purely used synthetic videos. This further proves that neural-symbolic models have the ability of modelling human-like cognitive function in high-complexity environments.

8.2 Future Works

From the evaluation, we can see several issues with the current pipeline. Possible areas of improvement could be the followings:

- **Use a more fine-tuned object detector with more types of objects supported**: The pre-trained object detector has become the main bottleneck of our pipeline. Its poor performance has prevented accurate features from being extracted and passed to the symbolic reasoner. Moreover, it could only detect a small set of objects, meaning that it is impossible for our pipeline to answer a question if the answer is outside of that set. To overcome this obstacle, we would need a robust model, either fine-tuned from a pre-trained model or newly constructed, that can detect most classes of objects in the dataset. However, such a model could be extremely hard to develop and train due to the wide variety of objects in the videos.
- **Use a more accurate face detection**: The pre-trained Haar cascade model used to localise a person's face sometimes struggles to distinguish face features from horizontal clothes wrinkles. We could swap it for a pre-trained neural model to reduce the false positives and obtain more precise face samples for training our face classifier. We could potentially fine-tune the complete neural pipeline in the `face-recognition` package to handle detection, encoding and classification for our task.
- **Associate words that are connected by common sense**: Our language processing component uses WordNet to detect semantic relations between words by checking synonyms, hypernyms and hyponyms. This approach cannot associate words that are not semantically connected. For example, it fails to understand that mug is *a type of* cup. What we would need is a model that can incorporate common sense knowledge when relating words. The word knowledge graph, ConceptNet, provides 36 complex relations that go beyond semantics relation, such as *RelatedTo* and *FormOf* [58]. It successfully associates mug and cup by the relation 'mug *is related to* cup'. Using ConceptNet, the pipeline would be able to relate more words, thus avoiding the case of no answer being selected when the object detected and the choice both represent the same item but are not semantically connected.

Apart from improvements on individual components, there are a few general areas of future research based on our current approach:

- **Learning general 'entering' concept**: HACR currently focuses on a specific type of entering action: people who were previously not in the scene appear. A broader definition of 'entering' could be described as a person's location changes from one to another in a certain period. While the fluent and time changes could be handled by Event Calculus without further changes, we would require to more precise location representation rather than just 'current scene'. A model that can detect backgrounds

and distinguish two different environments would be needed to extract such spatial features.

- **Learning complex concepts beyond visual concepts**: Our approach is targeted towards learning visual concepts from video frames. Besides video clips, the TVQA+ dataset also offers each clip's subtitles that are not used in our current pipeline, as there are generally irrelevant. If incorporating subtitles, we would be able to learn other interesting real-life concepts such as 'speaking', which involves detecting a character's mouth movements and speech.

8.3 Ethical Considerations

This project utilises a subset of the TVQA+ dataset [12] for training and inference. The video frames in TVQA+ datasets are collected from the TV series *The Big Bang Theory*, created by Chuck Lorre, Bill Prady and Steven Molar and produced by Chuck Lorre Productions and Warner Bros. Television. The dataset is copyright under the TVQA team from the University of North Carolina at Chapel Hill (UNC). UNC provides access to the dataset under an agreement, which we have signed and strictly follow. The dataset is used by us only and for research purpose only. There are a few questions that we created during the project. Despite not being part of the original TVQA+ dataset questions, they are based on the frames in the TVQA+ dataset and share a similar format with the original questions.

As mentioned in Chapter 1, this project's result could be beneficial for stepping towards robust AI systems with sound reasoning and general knowledge [2]. Moreover, the human-readability of symbolic learner would increase the transparency of the hybrid pipeline. While improving specific aspects of AI safety, there remain potential harms caused by such hybrid architecture. For example, using biased training data, hybrid models might show bias towards specific objects, people or concepts. There exists research on traing robots on Video Q&A for visually impaired people [59]. If a hybrid Video Q&A model is applied to such robots and gives unsafe answers or judgement, it could put their owners' safety at risk. Researchers and users still need to consider AI ethics [60] when developing and applying neural-symbolic agents.

Bibliography

[1] Jerome S. Bruner, Jacqueline J. Goodnow, and George A. Austin. *A Study Of Thinking*, chapter 3. New York: Wiley, 1956.

[2] Gary Marcus. The next decade in ai: Four steps towards robust artificial intelligence. *ArXiv*, abs/2002.06177, 2020. URL https://arxiv.org/abs/2002.06177.

[3] Stuart C. Shapiro. Artificial intelligence. In *Encyclopedia of Artificial Intelligence*, pages 54–57. John Wiley & Sons, Inc., USA, 2nd edition, 1992. ISBN 0471503053.

[4] Chi Han, Jiayuan Mao, Chuang Gan, Josh Tenenbaum, and Jiajun Wu. Visual concept-metaconcept learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/98d8a23fd60826a2a474c5b4f5811707-Paper.pdf.

[5] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgMlhRctm.

[6] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkxYzANYDB.

[7] Zhenfang Chen, Jiayuan Mao, Jiajun Wu, Kwan-Yee Kenneth Wong, Joshua B. Tenenbaum, and Chuang Gan. Grounding physical concepts of objects and events through dynamic visual reasoning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=bhCDO_cEGCz.

[8] Theophile Sautory, Nuri Cingillioglu, and Alessandra Russo. Hyster: A hybrid spatio-temporal event reasoner. *CoRR*, abs/2101.06644, 2021. URL https://arxiv.org/abs/2101.06644.

[9] Vladimir Lifschitz. *Answer Set Programming*. Springer International Publishing, Cham, 2019. ISBN 978-3-030-24658-7. doi: 10.1007/978-3-030-24658-7_2. URL https://link.springer.com/book/10.1007/978-3-030-24658-7.

[10] Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017. URL http://arxiv.org/abs/1711.03902.

[11] Kai-Uwe Kühnberger Pascal Hitzler. Facets of artificial general intelligence. *Künstliche Intelligenz*, pages 58–59, 2009. ISSN 0933-1875. URL https://daselab.cs.ksu.edu/sites/default/files/publications/AGI-KI2009.pdf.

[12] Jie Lei, Licheng Yu, Tamara Berg, and Mohit Bansal. TVQA+: Spatio-temporal grounding for video question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8211–8225, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.730. URL https://www.aclweb.org/anthology/2020.acl-main.730.

[13] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs. In Eduardo Fermé and João Leite, editors, *Logics in Artificial Intelligence*, pages 311–325, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11558-0.

[14] Murray Shanahan. The event calculus explained. In *Artificial intelligence today*, pages 409–430. Springer, 1999.

[15] Leo Bachmair, Harald Ganzinger, David McAllester, and Christopher Lynch. Chapter 2 - resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, Handbook of Automated Reasoning, pages 19–99. North-Holland, Amsterdam, 2001. ISBN 978-0-444-50813-3.

[16] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014. URL https://arxiv.org/abs/1405.3694.

[17] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs from noisy examples. *CoRR*, abs/1808.08441, 2018. URL http://arxiv.org/abs/1808.08441.

[18] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, Mar 1986. ISSN 1882-7055. doi: 10.1007/BF03037383. URL https://doi.org/10.1007/BF03037383.

[19] Robert Sklar. *Film: An International History of the Medium*, page 526. Prentice Hall, 1993. ISBN 9780133280142. URL https://books.google.co.uk/books?id=zLkkAQAAMAAJ.

[20] J. Cabestany, I. Rojas, and G. Joya. *Advances in Computational Intelligence: 11th International Work-Conference on Artificial Neural Networks, IWANN 2011, Torremolinos-Málaga, Spain, June 8-10, 2011, Proceedings*, page 521. Advances in Computational Intelligence: 11th International Work-conference on Artificial Neural Networks, IWANN 2011, Torremolinos-Málaga, Spain, June 8-10, 2011. Springer, 2011. ISBN 9783642215001. URL https://books.google.co.uk/books?id=iEmt4qx7xVQC.

[21] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. TVQA: localized, compositional video question answering. *CoRR*, abs/1809.01696, 2018. URL http://arxiv.org/abs/1809.01696.

[22] NLP-progress. https://github.com/sebastianruder/NLP-progress/blob/master/english/dependency_parsing.md, 2021.

[23] Merriam-Webster. Raj. In *Merriam-Webster.com*. Merriam-Webster, Jun 2021. URL https://www.merriam-webster.com/dictionary/raj.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[25] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.

[26] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 3686–3693, USA, 2014. IEEE Computer Society. ISBN 9781479951185. doi: 10.1109/CVPR.2014.471. URL https://doi.org/10.1109/CVPR.2014.471.

[27] Aman Chadha, Gurneet Arora, and Navpreet Kaloty. iperceive: Applying common-sense reasoning to multi-modal dense video captioning and video question answering. *ArXiv*, abs/2011.07735, 2020. URL https://arxiv.org/abs/2011.07735.

[28] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021. ISSN 2079-9292. doi: 10.3390/electronics10030279. URL https://www.mdpi.com/2079-9292/10/3/279.

[29] Junyeong Kim, Minuk Ma, Trung X. Pham, Kyungsu Kim, and Chang D. Yoo. Modality shifting attention network for multi-modal video question answering. *CoRR*, abs/2007.02036, 2020. URL https://arxiv.org/abs/2007.02036.

[30] Fei Liu, Jing Liu, Xinxin Zhu, Richang Hong, and Hanqing Lu. *Dual Hierarchical Temporal Convolutional Network with QA-Aware Dynamic Normalization for Video Story Question Answering*, page 4253–4261. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450379885. URL https://doi.org/10.1145/3394171.3413649.

[31] Junyeong Kim, Minuk Ma, Kyungsu Kim, Sungjin Kim, and Chang D. Yoo. Progressive attention memory network for movie story question answering. *CoRR*, abs/1904.08607, 2019. URL http://arxiv.org/abs/1904.08607.

[32] Junyeong Kim, Minuk Ma, Kyungsu Kim, Sungjin Kim, and Chang D. Yoo. Gaining extra supervision via multi-task learning for multi-modal video question answering. *CoRR*, abs/1905.13540, 2019. URL http://arxiv.org/abs/1905.13540.

[33] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL http://arxiv.org/abs/1611.01603.

[34] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018. URL http://arxiv.org/abs/1804.09541.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[36] Hyounghun Kim, Zineng Tang, and Mohit Bansal. Dense-caption matching and frame-selection gating for temporal localization in videoqa. *ArXiv*, abs/2005.06409, 2020. URL https://arxiv.org/abs/2005.06409.

[37] Linjie Yang, Kevin D. Tang, Jianchao Yang, and Li-Jia Li. Dense captioning with joint inference and visual context. *CoRR*, abs/1611.06949, 2016. URL http://arxiv.org/abs/1611.06949.

[38] Vladimir Iashin and Esa Rahtu. Multi-modal dense video captioning. *CoRR*, abs/2003.07758, 2020. URL https://arxiv.org/abs/2003.07758.

[39] Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. Visual commonsense r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10760–10770, 2020.

[40] Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016. URL http://arxiv.org/abs/1606.03490.

[41] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *CoRR*, abs/1805.10872, 2018. URL http://arxiv.org/abs/1805.10872.

[42] Daniel Cunnington, Alessandra Russo, Mark Law, Jorge Lobo, and Lance Kaplan. NSL: hybrid interpretable learning from noisy raw data. *CoRR*, abs/2012.05023, 2020. URL https://arxiv.org/abs/2012.05023.

[43] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL http://arxiv.org/abs/1612.06890.

[44] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. *CoRR*, abs/1810.02338, 2018. URL http://arxiv.org/abs/1810.02338.

[45] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[47] Joseph Suarez, Justin Johnson, and Fei-Fei Li. Ddrprog: A CLEVR differentiable dynamic reasoning programmer. *CoRR*, abs/1803.11361, 2018. URL http://arxiv.org/abs/1803.11361.

[48] Drew A. Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. *CoRR*, abs/1803.03067, 2018. URL http://arxiv.org/abs/1803.03067.

[49] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. *CoRR*, abs/1803.05268, 2018. URL http://arxiv.org/abs/1803.05268.

[50] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[51] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. URL http://arxiv.org/abs/1612.03144.

[52] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[53] David Ding, Felix Hill, Adam Santoro, and Matt M. Botvinick. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *CoRR*, abs/2012.08508, 2020. URL https://arxiv.org/abs/2012.08508.

[54] Stephen Muggleton. Inductive logic programming. *New generation computing*, 8(4): 295–318, 1991.

[55] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *CoRR*, abs/1905.06088, 2019. URL http://arxiv.org/abs/1905.06088.

[56] L. De Raedt, K. Kersting, S. Natarajan, and D. Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Synthesis Lectures on Artifici. Morgan & Claypool, 2016. ISBN 9781627058414. URL https://books.google.co.uk/books?id=NYgHkAEACAAJ.

[57] Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. Fastlas: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2877–2885. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5678.

[58] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press, 2017.

[59] Haozheng Luo and Ruiyang Qin. Open-ended multi-modal relational reason for video question answering. *ArXiv*, abs/2012.00822, 2020. URL https://arxiv.org/abs/2012.00822.

[60] David Leslie. Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector, June 2019. URL https://doi.org/10.5281/zenodo.3240529.

Appendices

Appendix A

Pre-processing

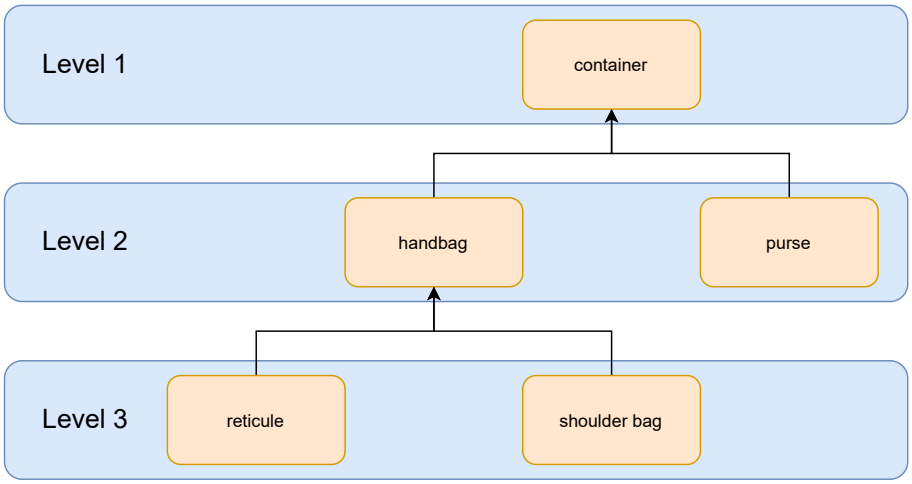


Figure A.1: An example of hypernym and hyponym. Level 2 words are Level 1 word's hyponyms; Level 1 word is Level 2 words' hypernym. Similarly, Level 3 words are Level 2 word's hyponyms; Level 2 word is Level 3 words' hypernym.

Character set

The set of characters we collect for face classification are:

Sheldon, Leonard, Howard, Raj, Penny, Bernadette, Amy, Stuart, Emily, Barry, Zack, and Wil.

Calculation for bounding box intersection

Assume bounding boxes b_1 and b_2. The top left corner of b_1 has a coordinate of $(x_{1,0}, y_{1,0})$ and the bottom right corner of it has a coordinate of $(x_{1,1}, y_{1,1})$. Similarly, the two coordinates for b_2 are $(x_{2,0}, y_{2,0})$ and $(x_{2,1}, y_{2,1})$. To calculate the area of the intersection, we first calculate the top left and bottom right coordinates of the intersection:

$$
\begin{aligned}
x_{i,0} &= \max(x_{1,0}, x_{2,0}) \\
y_{i,0} &= \max(y_{1,0}, y_{2,0}) \\
x_{i,1} &= \min(x_{1,1}, x_{2,1}) \\
y_{i,1} &= \min(y_{1,1}, y_{2,1})
\end{aligned}
$$

With the coordinates of the opposite corners, we can calculate the area of intersection as:

$$A_I = (x_{i,1} - x_{i,0}) \times (y_{i,1} - y_{i,0})$$

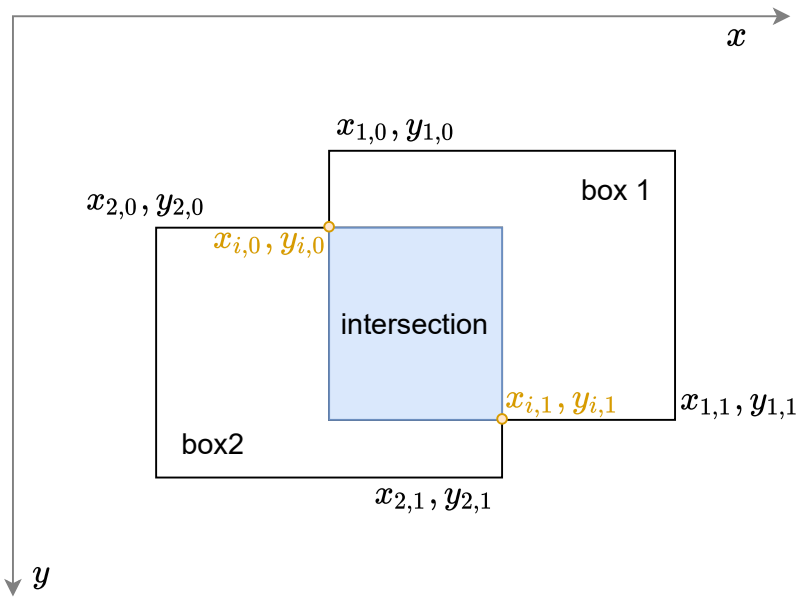


Figure A.2: An example of bounding box intersection calculation, where $(x_{i,0}, y_{i,0}) = (x_{1,0}, y_{2,0})$ and $(x_{i,1}, y_{i,1}) = (x_{2,1}, y_{1,1})$.