Kantian and Neurosymbolic Artificial Intelligence: Applications and Insights in Computer Vision

Ivar A. Frisch
Student number: 6977839
Research Master's Artificial Intelligence
44 EC

Utrecht University
Department of Information and Computing Sciences
Supervisor:
Natasha Alechina
TNO Supervisor:
Dalia Aljawaheri
Second Supervisor:
Dominik Klein

1 December 2024

Abstract

Can Kant's philosophy of cognition be useful for computer vision? Kant explored how humans progress from subjective perception to objective understanding. Modern computer vision models, despite excelling at tasks like object detection, struggle with high-level visual reasoning, such as interpreting relationships between objects. Gary Marcus attributes this limitation to the reliance on deep learning, which fails to generalize beyond training data. This work extends Marcus' critique by arguing that pure deep learning-based vision models lack coherent object representations and cannot learn abstract, domain-general invariances from visual data. To address these limitations, I propose integrating Kantian principles into vision models, leading to neuro symbolic vision models. Specifically, I draw on Kant's notion of intelligence as a faculty of combination, enabling the transition from perception to understanding, due to its coherent object representations. I argue this concept can guide the development of systems capable of improved visual reasoning. However, neuro symbolic AI for vision is a young research field and a big challenge here is how to combine discrete and continuous representations. As such, to test the possible usefulness of these models, I conduct an experiment which focuses on data representation challenges. For this, I will use the neuro symbolic Apperception Engine, due to its close proximity to Kantian philosophy and its proven success on rudimentary vision tasks. My findings reveal that grid-based data representations are unsuitable for these models due to high information loss during the conversion from continuous to discrete data and the excessive complexity introduced by grid-size-dependent background knowledge. This suggests the need for alternative data representations in neuro symbolic vision models. Ones which better support relational reasoning from visual scenes. As such, this research contributes to the emerging field of neuro symbolic AI, offering insights into designing systems that bridge perceptual and conceptual understanding.

Keywords: Computer Vision, Visual Understanding, Kantian Intelligence, The Apperception Engine, Logic Program Complexity

Contents

Chapter 1

Introduction

The rise of deep learning in artificial intelligence (AI) research has greatly improved the performance of computer vision models on low-level tasks like object detection, object tracking, and object segmentation [79]. However, purely deep learning-based computer vision models still do not perform well in higher-level computer vision tasks, such as when out-of-distribution generalization or learning complex concepts about objects is necessary [51], [42], [70].

In order to combat these problems, researchers have moved to neuro symbolic AI [18], [69], [29]. Neuro symbolic AI aims to combine the advantages of logic-based AI with deep learning. In theory, this should enable deeper understanding of a given scene, like learning complex concepts on noisy data and out-of-distribution tasks through compositionality [55], [50]. However, neuro symbolic AI is a relatively new and underdeveloped field and, as such, it is challenging to implement. This is especially the case for computationally intensive data like videos. This raises the question as to how we can implement neuro symbolic AI in such a fashion that allows for models to gain a deeper understanding of visual scenes. More broadly, what does it mean for any intelligent system to have a deep understanding of scenes?

In fact, this is not a new question. Philosopher Immanuel Kant has written extensively about human cognition. Namely, in his *Critique of Pure Reason* [32], he described a cognitive architecture aimed at explaining how humans see, or make sense of the world. He proposed a hybrid architecture of concurrently interacting faculties, specifying how humans make sense of a sensory sequence. As such, it can be said that he provides a "blueprint for neuro symbolic AI" [22]

With that in mind, can Kant's theory of cognition be useful for solving computer vision tasks, specifically? Work has already been done on answering this question. Richard Evans developed the a working model of Kantian perception in 2022, known as the "Apperception Engine" [22], which has already been applied to sequences of small binary images. However, it remains an open question as to whether the Apperception Engine is a robust model for tackling higher-level computer vision tasks.

1.1 Contributions

To this end, I first show that neuro symbolic AI is most aligned with Kantian intelligence as a combination of faculties, and vice versa. This gives us strong theoretical backing as to why we ought to take Kantian understanding as a legitimate framework for implementing computer vision. I then test the Apperception Engine empirically by running a computer vision experiment which seeks to examine whether the model is a good candidate for higher-level computer vision tasks.

1.2 Thesis Structure

In order to get there, Chapter 2 acts as the theoretical background. First, I describe current paradigms in artificial intelligence generally. I then narrow our scope to the question of computer vision, wherein I give a technical explanation, and cover crucial difference between low-level and high-level tasks. Guided by this distinction, we are able to see how current AI paradigms fall short of deeper understanding of visual scenes, which leaves us with the open question as to how we are able to achieve this goal. Chapter 4 seeks to answer this question, wherein we discuss foundational ideas that drive the motivation behind this thesis, namely, of how Kant has already given us the blueprint as to how intelligent systems are able to move from visual awareness to understanding, and that neuro symbolic AI is the closest realization

of this abstract idea. Chapter 4 covers its implementation in the form of the Apperception Engine and discusses in depth how it works. Then, in Chapter 5, I present my experiment, wherein I test whether the Appercpetion Engine in its current implementation is indeed suitable for computer vision tasks. These findings are discussed in Chapter 6, after which I conclude in Chapter 7.

Chapter 2

Background

2.1 Paradigms in Artificial Intelligence

In this section, I will outline current paradigms to approaching the development of AI.

2.1.1 Symbolic AI

Symbolic AI, also known as Good Old-Fashioned Artificial Intelligence (GOFAI), is an approach to artificial intelligence that relies on explicitly defined rules, logic, and symbolic representations of knowledge. It operates under the assumption that human intelligence can be represented using symbols and manipulated using formal logic rules. Its main approaches include logic-based AI systems, expert systems and semantic networks. [7]

Logic Programming. Logic programming, a subset of symbolic AI, relies on clearly defined logical structures (better known as 'expert knowledge') to encode and process knowledge. This approach uses symbolic, localist representations where knowledge is explicitly encoded into (abstract) concepts. In localist representations, each concept or entity is represented by a single, dedicated symbol or predicate. For example, 'triangle(X)' represents the concept of a triangle, and 'red(X)' directly represents the property of being red. This one-to-one mapping between symbols and concepts makes the representations explicit and interpretable, but less flexible than distributed representations. [52]

The manipulation of these symbols through logical rules allows the system to infer new knowledge or make decisions based on a set of predefined rules. As such, logic-based AI is well suited for deductive reasoning and exhibits a form of transparent decision-making. However, they also face some drawbacks: due to the high complexity of the data, running time can be very high. Additionally, these systems can struggle with uncertainty and incomplete information, leading to brittleness in real-world applications where perfect knowledge is often unavailable. [7]

Syntax of Logic Programs. I follow Richard Evans in characterizing the basic concepts and standard notation from logic programming. I shall use `a`, `b`, `c`, ... for constants, `X`, `Y`, `Z`, ... for variables, `p`, `r`, `q`, ... for predicate symbols, and `f`, `g`, `h`, ... for function symbols. [22]
A **logic program** consists of rules and facts expressed in formal logic. Below, I will cover a list of basic concepts necessary to understand logic programs. For a more exhaustive list, see [13]).

- **Term**: A term that can be simple or complex. A simple term is a constant or variable. A complex term is of the form $f(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms.
- **Atom**: A logical statement of the form $p(t_1, \ldots, t_n)$, where p is a predicate symbol, and $t_1, \ldots, t_n$ are terms.
- **Function-free atom**: An atom where all terms are simple. If a is an atom, then $vars(a)$ denotes the variables in a. For example, $vars(p(X, f(X, Y), Z)) = \{X, Y, Z\}$.
- **Ground**: An atom or term that contains no variables. An atom a is ground if $vars(a) = \{\}$.
- **Literal**: An atom (positive literal) or its negation (negative literal, written as `not A`).

- **Clause** (or *Rule*): A clause with head and body literals of the form $h_1, \ldots, h_n$: - $b_1, \ldots, b_m$ where each h_i and b_j is a literal and the symbol ',' denotes conjunction. The symbols h_i are called the `head` of the clause. The symbols b_i are called the `body` of the clause.
- **Horn Clause**: A clause with at most one positive literal. For example, $p(X)$: - $q(X), r(X)$.
- **Definite Clause**: A clause with exactly one head literal of the form h : - $b_1, b_2, \ldots, b_n$. A definite clause states that the head is true if all of the body literals are true. For example, in the definite clause $p(X, Y)$: - $q(X), r(Y)$, $p(X, Y)$ is true if $q(X)$ is true and $r(Y)$ is true.
- **Clausal Theory**: A set of clauses.
- **Constraint**: A clause with no head of the form : - $b_1, b_2, \ldots, b_n$
- **Unit Clause**: A clause with no body. For unit clauses, we usually omit the : - symbol, e.g., $p(X, Y)$.
- **Fact**: A ground unit clause with no body, e.g., $p(a, b)$.
- **Substitution**: A function that maps variables to terms, represented by σ. When applying a substitution to variables, we write it as pairs of variable/term mappings. For instance, if $\sigma = \{X/a, Y/b\}$, then X is replaced by a and Y by b. To show that a substitution σ is applied to an atom a, we write $a\sigma$. For example, when applying $\sigma = \{X/a, Y/b\}$ to $p(X, Y)$, we get $p(X, Y)\sigma = p(a, b)$. Multiple substitutions can be combined, written as $\sigma\sigma'$. A special case is the empty substitution $\epsilon = \{\}$, which leaves any atom unchanged: $a\epsilon = \epsilon a = a$.

Semantics of Logic Programs. The semantics of logic programs is based on the concepts of the *Herbrand universe*, *Herbrand base*, and *interpretation* [22, p.23]:

- The **Herbrand universe** is the set of all ground terms that can be formed using the constants and function symbols in a program.
- The **Herbrand base** is the set of all ground atoms that can be formed using the predicate symbols and terms in the Herbrand universe.
- A **Herbrand interpretation** is a subset of the Herbrand base, representing the set of ground atoms that are true.

A logic program defines a set of possible interpretations. An interpretation is a *model* of a program if it satisfies all the rules in the program. A model satisfies a rule if, whenever all literals in the body of the rule are true in the interpretation, the head of the rule is also true. [22, p.23], [13]. I will now cover some prevalent paradigms for logic-based AI.

Prolog. Prolog [36], one of the earliest logic programming languages, provides a practical framework for symbolic computation and natural language processing through Horn clauses and SLD-resolution (an inference rule that sacrifices expressibility for efficiency).

Despite its logical foundations, Prolog is not purely declarative. The order of clauses and the presence of control structures like cuts means that understanding a Prolog program requires procedural knowledge—how the interpreter will execute the code—rather than just the logical meaning of the clauses. This makes program behavior highly dependent on clause ordering, affecting both execution efficiency and results. [37], [13].

Datalog. Datalog is an extension of Prolog which limits the complexity of logical expressions. As such, it offers a more restricted, yet more efficient, approach. Although it is less expressive than full logic programming, it guarantees termination and provides decidable inference. This makes it particularly suitable for database applications and certain types of analysis. Furthermore, unlike Prolog, Datalog is also fully declarative, meaning that clause ordering does not effect program execution. [13]

Datalog€ Datalog€ is defined by Richard Evans in [22] in order to express causal theories. Specifically, it is designed for modeling dynamics between states. As such, it extends Datalog with causal rules and constraints, such as frame inertia ("each atom remains true at the next time-step unless it is overridden by a new fact which is incompossible with it.") [22, p.32] and incompossibility ("Two facts are incompossible if there is a constraint that precludes them from both being true")[22, p.32]. It also extends the type of rules. Rules are either *arrows rules* or *causal rules*.

- **Arrow Rules**:
$$\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha_0$$
This states that if $\alpha_1, \ldots, \alpha_n$ all hold, then α_0 also holds at the same time-step.

- **Causal Rules**:
$$\alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \alpha_0$$
This states that if $\alpha_1, \ldots, \alpha_n$ all hold, then α_0 also holds at the *next* time-step.

Answer Set Programming Answer Set Programming (ASP) [27] extends the foundation of Prolog by introducing non-monotonic reasoning. As such, it allows systems to handle common-sense reasoning and complex constraint satisfaction problems more effectively. [40]

ASP extends logic programming by allowing *normal logic rules*, which include negation as failure (`not`). The semantics of ASP are based on *stable models* or *answer sets*, which represent consistent solutions to the program. [45, 53] A normal logic rule has the form:

```
a :- b1, ..., bn, not c1, ..., not cm.
```

This means `a` is true if all `b1, ..., bn` are true and none of `c1, ..., cm` are true. For instance:

```
p :- not q.

q :- not p.
```

This program has two stable models: one where `p` is true and `q` is false, and another where `q` is true and `p` is false. Stable models are computed by grounding the program (replacing variables with all possible constants) and iteratively deriving conclusions.

ASP also supports **choice rules** and **constraints**, which enhance its expressive power [72]. For example:

- **Choice Rule**:
```
{a, b} :- c.
```
Allows subsets of `{a, b}` to be true if `c` is true.

- **Constraint**:
```
:- a, b.
```
Prevents models where both `a` and `b` are true.

ASP is often used for combinatorial optimization, knowledge representation, and reasoning tasks, due to its ability to represent complex problems declaratively. However, it can also be used as meta-interpreter for other logic languages [15], as is done by Evans [22, p.27].

Complexity. The computational complexity of logic programming depends on the specific framework:

- **Datalog**: Polynomial-time data complexity but exponential-time (**EXPTIME**) program complexity due to the grounding process. [21]
- **Datalog€**: Polynomial-time data complexity (like Datalog), but potentially **PSPACE-hard** program complexity due to reasoning over temporal extensions and dynamic state transitions. [22, p.23-24]
- **ASP**: **NP-complete** for finding stable models; Σ_2^P-complete for optimization problems involving weak constraints or preferences. [17]

Inductive Logic Programming. Inductive Logic Programming (ILP) [13] generates interpretable hypotheses in the form of logical programs. As such, it bridges the gap between traditional logic programming and machine learning. Unlike traditional logic programming, which merely tests the satisfiability of data by defining pre-defined rules, ILP can learn logical rules and relations from data. Unlike conventional machine learning models, ILP learns relations and logical patterns from structured data (like graphs). This makes it highly suited for symbolic induction tasks.

	Statistical ML	ILP
Examples	Many	Few
Data	Tables	Logic programs
Hypotheses	Propositional/functions	First-/higher-order relations
Explainability	Difficult	Possible
Knowledge transfer	Difficult	Easy

Figure 2.1: Comparison between statistical machine learning and inductive logic programming. Taken from [13]

Essential Components. ILP systems contain four main components [13]:

1. **Learning Setting**: An ILP task consists of:
   - *Positive Examples*: Instances where the target relation holds true.
   - *Negative Examples*: Instances where the relation does not hold.
   - *Background Knowledge (BK)*: Contextual information, often relational or domain-specific, that assists in hypothesis generation. For example, BK can represent rules such as `add(A,B,C) :- C = A+B`.
2. **Representation Language**: Just like logic programming and first order logic, ILP uses discrete representations in order to describe hypotheses, examples, and BK. This allows ILP systems to encode and generalize relational structures like `has_angles(shape, 3)` or `has_sides(shape, 3)` into rules such as `isTriangle(X) :- has_angles(X, 3), has_sides(X, 3)`. Meaning that X is a triangle if it has three angles and three sides.
3. **Language Bias**: To ensure efficient learning, constraints are applied to restrict the hypothesis space. These biases can define syntactic forms (e.g., predicate arity) or semantic properties (e.g., logical consistency).
4. **Search Method**: ILP explores a discrete hypothesis space to find rules consistent with the examples and BK. A key feature of ILP is *predicate invention*, where new predicates are introduced to capture latent relationships. This specific future add to ILPs ability for abstraction and generalization.

As shown in Figure 2.1, ILP offers several advantages over traditional machine learning approaches. While traditional ML requires large amounts of data and learns statistical patterns, ILP can learn from just a few examples by using logical rules and background knowledge. The figure also illustrates how ILP's logical nature enables both relational reasoning and explainable outputs, in contrast to the often opaque statistical patterns of traditional ML. Furthermore, ILP's ability to integrate learned rules back into its knowledge base allows for continual learning and knowledge transfer. [13]

Finally, these models have shown tremendous improvement in performance on logical reasoning tasks, compositionality problems, and out-of-distribution generalization (e.g., [14], [13]). Furthermore, they have been applied to abstract visual scenes in order to provide abstract reasoning rules [24], [68], [69], [70], [67]. This is interesting for this thesis, because it shows the potential advantage of applying ILP systems to computer vision tasks.

2.1.2 Sub-Symbolic Learning

Sub-symbolic AI, exemplified by neural networks and deep learning, represents a different paradigm from symbolic AI. We can see this in both its operational mechanism and its data representations.

Essential Components. Neural networks consist of interconnected artificial neurons organized in layers. [28] Each connection carries a weight that determines the strength of signal transmission between neurons. As such, the network's knowledge is encoded in these weights. The weights, in turn, are continuously adjusted during training.

Working Mechanism. The primary operation for weight adjustment occurs through two key processes:

- **Forward Propagation**: Input signals propagate through the network, with each neuron computing weighted sums followed by non-linear activation functions:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \tag{2.1}$$

$$a^{(l)} = \sigma(z^{(l)}) \tag{2.2}$$

  where $z^{(l)}$ is the weighted input to layer l, $W^{(l)}$ is the weight matrix, $a^{(l-1)}$ is the activation from the previous layer, $b^{(l)}$ is the bias vector, σ is a non-linear activation function, and $a^{(l)}$ is the activation (output) of layer l.

- **Back propagation**: Errors calculated at the output layer propagate backwards through the network, adjusting weights to minimize prediction errors using gradient descent:

$$\delta^{(L)} = \nabla_a C \odot \sigma'(z^{(L)}) \text{ for output layer } L \tag{2.3}$$

$$\delta^{(l)} = ((W^{(l+1)})^T \delta^{(l+1)}) \odot \sigma'(z^{(l)}) \text{ for hidden layers} \tag{2.4}$$

$$\frac{\partial C}{\partial W^{(l)}} = \delta^{(l)} (a^{(l-1)})^T \tag{2.5}$$

  where $\nabla_a C$ is the gradient of the cost function C with respect to the activations a^L, σ' is the derivative of the activation function σ with respect to z^L, $\delta^{(l)}$ is the error at layer l, $\odot$ denotes element-wise multiplication, and $\frac{\partial C}{\partial W^{(l)}}$ gives the gradient for weight updates. [64]

Data Representation The effect of this mechanism is that neural networks, contrary to symbolic AI's localist representations, have distributed representations [30], [52]. As such, individual concepts are not mapped to specific units but are represented by many different neurons and their weights. Thus, each weight participates in representing multiple concepts. Information is continuous rather than discrete, because values are represented as real-valued weights rather than binary values.

Inference Form Due to these distributed representations, neural networks perform well at several forms of inference [5]. First, they can perform robust pattern recognition in noisy or ambiguous data. This is because their distributed information representation allows the system's performance to degrade only marginally when inputs are imperfect. Second, they are good at learning data-specific features [28]. Third, they can generalize well across similar inputs. The continuous nature of their representations allows for interpolation between learned examples. Finally, rather than following explicit rules, neural networks make predictions based on learned statistical patterns in the data. [64].

As such, this distributed nature of representation [5] makes neural networks particularly effective at handling uncertainty and noise in input data. However, this comes at the cost of reduced interpretability and data intensiveness compared to symbolic systems.

Binary Neural Networks A special case of neural networks are Binary Neural Networks (BNNs) [10]. They have shown increasing popularity because they are more efficient than standard neural networks. This is mainly due to their ability to represent continuous data discretely. [86]

Richard Evans, in [22, p. 30], follows [10], by constraining both weights and activations to binary values {0,1}. His activation function is determined by comparing the sum of XNOR operations between inputs and weights against a threshold:

$$\sum_{i=1}^{n} \mathbf{1}[x_i = w_i] \geq \left\lceil \frac{n}{2} \right\rceil \tag{2.6}$$

Here, x_i are binary inputs, w_i are binary weights, and $\mathbb{1}[\cdot]$ is the indicator function.

2.1.3 Neuro symbolic AI

Neuro symbolic AI represents the third wave of artificial intelligence [18], combining the strengths of both symbolic and sub-symbolic approaches. This combination aims to make use of the complementary abilities of each paradigm.

Essential Components. Neuro symbolic systems typically consist of a neural component for pattern recognition and handling noisy data, and a symbolic component for logical reasoning and knowledge representation. [18]

Modes of Integration. This integration between neural and symbolic components can occur through several modes of integration. Adhering to Kautz' taxonomy, we can identify the following types [34]:

Type 1 **Symbolic Neuro** represents a loose coupling with minimal integration. This is just deep learning. Symbolic inputs are converted to vectors, processed by neural networks, and converted back to symbols via softmax. The information representation is primarily distributed, with symbols being mapped to continuous vector spaces. Feature vectors are learned from data through, for example word embeddings, capturing semantic relationships. Symbol grounding occurs at input/output boundaries through vector encoding/decoding, but the internal representations remain distributed.

Type 2 **Symbolic[Neuro]** uses neural networks as subroutines within symbolic problem solvers. An example is AlphaGo which uses neural networks within a Monte-Carlo Tree Search algorithm [71]. This type maintains a clear separation between localist and distributed representations, due to its use of neural networks for pattern recognition while keeping symbolic representations for search and planning. Feature vectors are task-specific (e.g., board positions in AlphaGo). Symbol grounding is handled by the symbolic layer, with neural networks providing numerical evaluations.

Type 3 **Neuro — Symbolic** systems use neural networks to convert raw input into symbolic structures for symbolic reasoning. This approach often contains a double movement. On the one hand, the symbolic reasoning system can provide feedback signals to train the neural network. On the other hand, the neural networks can provide (probabilistic) predicates, which are used by the symbolic reasoning component to generate and test hypotheses. As such, approach can easily be combined with ILP. Examples of this are, PROPPER [29] and the neuro symbolic Apperception Engine [24]. Thus, this approach creates an explicit bridge between distributed and localist representations, where neural networks learn to map perceptual input to symbolic concepts. Here, feature vectors represent perceptual patterns that correspond to symbolic concepts. Symbol grounding is achieved through the learned mapping between perceptual features and symbolic concepts.

Type 4 **Neuro: Symbolic → Neuro** uses neural networks to perform symbolic reasoning. However, this is not due to an inherent generalization capability, but rather by training them on symbolic rules. For example, Lample and Charton's system learns to perform symbolic mathematics through training on mathematical expression pairs [39]. The representation is primarily distributed, but structured to capture symbolic rules. Feature vectors encode symbolic knowledge implicitly in the network's weights. Symbol grounding occurs through the training process where symbolic rules guide the formation of distributed representations.

Type 5 **Neuro_Symbolic** uses embedding techniques to transform symbolic representations into vector spaces, as seen in tensor product representations [73]. This approach creates distributed representations of symbolic knowledge through tensorization. Feature vectors represent logical predicates and relations in a continuous space and symbol grounding is achieved through the grounding of logical concepts onto tensors, with logical statements acting as constraints on the vector space. For example, [73] encodes the logical operations of compositionality into a function in the neural network. This means that the network learns the operation of composing new classes from distinct parts. Similarly, in [81] they add a relational bottleneck to an object-centric slot attention model, enabling it to learn relations on objects.

Type 6 **Neuro[Symbolic]** implements a fully integrated architecture where symbolic reasoning engines are embedded within neural engines. Based on Kahneman's "thinking fast and slow" model [**?**], this approach uses a neural system (System 1) that decides when to invoke symbolic reasoning (System 2). According to Kautz, this represents the most ambitious integration of distributed and localist representations, as it aims for combinatorial reasoning within a neural framework. Feature vectors must support both fast pattern recognition and systematic symbolic manipulation. Symbol grounding emerges from the interaction between the fast neural system and the deliberative symbolic system.

According to [18, p.16], these types can be roughly divided into two options. In option 1, symbols are translated into a neural network and symbolic reasoning is being performed within the network. Option 2, on the other hand, takes a more hybrid approach in which the neural network interacts with a symbolic system for reasoning. Under option 1, we could classify type 1, type 4 and type 5. Under option 2, we could classify type 2, type 3 and type 6. I believe that especially option 2 will get us closer to visual understanding, due to the fact that it is more likely to learn symbol manipulation rather than relying on learning structures which are build in from the start [50]. More on this in Chapter 4.

Interface Problem. However, option 2 also comes with a big problem: the interface problem [18, p.16]. In option 1, the symbolic representations are converted into network tensors, thus getting a continuous character. However, in option 2, we have two interacting modules consisting of two distinct types of representation. As such, this hybrid AI requires a bridge between localist and discrete representations versus continuous and distributed representations. [18, p.10], [50] The interface problem is about how we combine these representations in an efficient way.

Thus, the aim of this chapter was to provide the background knowledge containing necessary terms and AI paradigms in order to understand the rest of the thesis. Mainly, I showed that logic-based AI and sub-symbolic AI, also referred to as deep learning, differ from each other due to their different way of representing information; discrete, localist and relational, versus continuous, distributed and functional, respectively. This distinction also holds for more recent logic-based systems, such as ILP, which can learn logical theories from data.

Finally, I showed that neuro symbolic AI aims to form a unification of these two paradigms. The main challenge, here, is the interface problem. How to unify these distinct representations. Kautz' taxonomy proposes that this can be done through multiple levels of integration between the sub-symbolic component and the symbolic component. All in all, these different ways of integration can be subdivided in two options: option 1, where symbols are translated into a function within a neural network, and option 2, where the neural network and the symbolic module remain two separate, interacting components.

From here on out, I will refer to neuro symbolic ILP as neuro symbolic AI.

Chapter 3

Computer Vision and Visual Understanding

3.1 Computer Vision and Visual Understanding

Computer vision has long been an essential part of research in artificial intelligence. In general, the goal of computer vision is to understand the scene or features in images and videos of the real world, using methods like image processing and, more recently, pattern recognition. [79]. However, while contemporary computer vision models have achieved high success in annotation-based tasks like object detection and image classification, they continue to struggle with higher-level visual understanding tasks that require reasoning about relationships, causality, and context.

This limitation becomes particularly apparent in real-world applications such as in autonomous systems and robots when navigating their environment [57], [79], [84]. For instance, while a self-driving car might excel at detecting pedestrians and other vehicles, it may fail to understand complex scenarios that require reasoning about potential interactions or unusual circumstances outside its training distribution. As is evidenced by recent news articles covering the backlash of self-driving cars [63], [33], [6].

This gap between annotation capability and visual understanding raises questions about the nature of visual understanding and the current limitations of deep learning-based computer vision approaches. To what extent can these models form some kind of understanding about the data? This chapter will examine this. I will show that contemporary, deep learning-based models, although they excel at low-level tasks, still struggle with visual understanding. In order to get here, I will first give a short history of computer vision and the need for visual features and invariances. After this, I will distinguish more clearly between low-level and high-level tasks, as to lay the ground work for my definition of visual understanding. Next, I will show how some state of the art approaches still lack visual understanding due to their reliance on deep learning, by following Marcus' critique. As such, this chapter will show the need to move to a new theoretical framework for understanding in computer vision models, which will be the subject of the next chapter.

3.2 Foundations of Computer Vision

3.2.1 Image Processing

The field of computer vision has developed out of the field of image processing. Image processing focused on operations that transform images while preserving their essential information. This focus on manipulating and analyzing images includes techniques such as edge detection, filtering, and transformations.

As such, the field initially developed around the need to extract meaningful visual features from images [79]. Visual features are crucial elements that help to distinguish objects and patterns within images. These features need to be robust against various transformations and conditions. This capability for robustness is known as an invariance. As such, the ability to detect and describe these features according to stable, data-relevant invariances determines the success of any computer vision system.

The problem of visual feature invariance was a recurring challenge in image processing. It is difficult to recognize objects under changes in scale, rotation, lighting, and perspective [8]. Traditionally, these features were handcrafted, meaning they had to be manually constructed and determined. As the field

of image processing developed, techniques that were able to handle invariance in these features became more sophisticated, such as color histograms [75], the Canny edge detector [9], histograms of oriented gradients (HOG) [16], and Scale Invariant and Feature Transform (SIFT) [48]. In particular, HOG and SIFT provided good object features that were invariant to scale, rotation, and lighting. Crucially, however, these feature descriptors still depended on handcrafted engineering and thus were unable to learn data-specific features.

3.3 The Shift to Learning-Based Approaches

The onset of computer vision technology marked the transition from handcrafted features to learning-based approaches with neural networks. This paradigm shift aimed to address feature invariance by learning features directly from data. The rise of deep learning was a pivotal moment in computer vision research because it enabled breakthroughs in both accuracy and scalability.

3.3.1 Current Approaches in Computer Vision

In general, there are two main approaches to computer vision. Low-level features are essential aspects of an image or video, such as contours, edges, and colors. High-level features, on the other hand, contain more complicated details, such as causal properties and relational inferences. We can test for these types of features in low-level and high-level tasks, respectively. These tasks reflect the layered architecture of current vision systems, where earlier layers handle basic visual primitives, and deeper layers aim to model semantics.

Low-Level Tasks

Low-level tasks are annotation focused [84]. These tasks include for example object detection or segmentation. [79]. In object detection, the task is to determine "what objects are where" [90]. Specifically, in what patches of pixels of the input image do we find which objects? As such, the main challenge here is how to view all the different appearances of objects (different lighting, position, translation, scale, shadows, partially occluded, etc.) as part of the same object.

High-Level Tasks

On the other hand, high-level tasks focus on deeper understanding of the scene [84]. Specifically, this means that systems gain an understanding of more complex relationships like the dynamics, logicality and causality of objects in a scene. Examples of high-level tasks include visual spatial reasoning [46], visual compositionality [50], [42], and Bongard problems [3], [81].

3.3.2 Models Driving Computer Vision

As mentioned, deep learning marked a pivotal moment in computer vision. These models incorporate mechanisms which aim to achieve invariance. A classic example of an influential model in computer vision is the Convolutional Neural Network (CNN). A CNN is a neural network architecture designed to process grid-structured data like images by leveraging convolutional layers to extract spatial hierarchies of features [41], [38]. Due to the operation of convolution, it learns translational invariance [28]. This enables it to recognize features regardless of their spatial position in the input [12]. However, CNNs struggle with capturing long-range dependencies and relational reasoning [80], as they lack explicit mechanisms for modeling global context and interactions between objects [4].

On the other hand, more recent models like Vision Transformers (ViT) leverage self-attention mechanisms to process images as sequences of non-overlapping patches, allowing it to capture global contextual information across the entire image [20]. By treating images as sequences, ViTs learn position-agnostic feature relationships, enabling them to model complex interactions between distant parts of an image [78]. This approach offers significant flexibility in capturing global interactions and understanding the overall scene context. However, despite their strength in modeling global context, ViTs often lack the inherent inductive biases present in CNNs, such as locality and translation equivariance [20]. This absence of inductive biases means that ViTs require large amounts of data to learn effectively and can be less efficient for tasks that benefit from structured relational reasoning [59].

In a similar vein, Vision-Language Models (VLMs) combine visual and textual modalities to jointly reason about images and text, often using multimodal transformers. They learn cross-modal alignments,

such as associating visual features with corresponding textual descriptions. This allows them to achieve invariance across modalities. Recent examples of this are GPT-4v [1] and DALL-E 3 [62]. However, despite their large success across many tasks, they still perform poorly on multi-object visual reasoning tasks, such as relational scene understanding [42], [77], counting [87], [61] and visual analogies [55], [74].

Recently, slot attention [47] mechanisms have been shown great achievement in decomposing visual inputs into a set of latent slots that represent individual objects or entities in a scene. It learns object-centric representations that are invariant to object order and occlusions. [47] This object representation has made it a strong foundation model for compositional abilities, indicating their potential for visual reasoning. In [81] and [56], 'relational bottlenecks' were added, enabling the model to learn relationships between objects [81] and higher order relationships between objects [56].

However, slot attention models struggle to generalize beyond the training distribution, due to the required specification of the number of slots. [47] Furthermore, while slots offer interpretability in terms of grouping, understanding the semantics of individual slots or their learned embeddings can still be challenging.

Thus, contemporary computer vision models, despite their great success in performing annotation-based tasks, still perform sub-optimally at high-level tasks like visual understanding. This is especially the case for video data.

3.3.3 Toward Richer Representations: Scene Graphs

While the question of low-level features seems to be solved by current approaches in computer vision, work is still to be done on how to capture invariance in high-level tasks. Scene graphs act as a promising technique to help current models drive towards scene understanding as exemplified in high-level tasks and to move beyond raw pixel-based feature learning. Scene graphs provide a structured representation of an image by explicitly modeling objects and their relationships in a graph[89]. In this representation, nodes typically correspond to objects or entities in the image, while edges represent relationships or interactions between these objects. Each node and edge can carry semantic attributes. For example, nodes can contain object categories, attributes, and spatial information, while edges encode relationships such as spatial ("above", "inside"), functional ("holding", "wearing"), or compositional ("part of") relationships. As such, scene graphs can be seen as a method which helps machines to understand relationships between objects, enabling applications like image captioning, robotics, and reasoning. This may prove to be useful if the overarching goal is to move towards richer visual representations and therefore deeper understanding of visual scenes.

3.4 Critique of Sub-Symbolic Based Computer Vision

However, although techniques like scene graphs are a step in the right direction, they are often limited by the sub-symbolic systems that they rely on (e.g., deep learning based scene graph models to generate graph nodes). Why is this the case? Why do they often still perform poorly at high-level tasks? While some have argued that deep learning based vision models' poor performance on higher-level reasoning tasks is contingent on finding the right technique (e.g., through the likes of slot attention or scene graphs), this poor performance is arguably an inherent limitation of deep learning [81], [50], [49]. In other words, it could be that deep learning techniques on the whole lack the ability to gain a deeper understanding of visual scenes.

For example, according to Marcus, deep learning is based on statistical correlation of features [50], [49] and this fact means multiple things. First of all, in order to achieve statistical correlations required for computer vision tasks, deep learning models require very large amounts of labeled data to learn patterns. Because of these correlations, it cannot perform causal reasoning, only correlations. In a similar vein, the learned hypothesis of neural networks is not interpretable, because it is implicitly encoded in the weights.

Moreover, like statistical correlation, back propagation lies at the heart of sub-symbolic AI. Also like correlations, back propagation is purely mathematical and does not inherently build on the broader structure of the data – we cannot easily integrate background knowledge into the models. While effective for pattern recognition, back propagation embodies many of the shortcomings identified in Marcus' critique of deep learning. This is because back propagation optimizes network parameters by minimizing the loss on training examples, learning a mapping between inputs and outputs through gradient descent (see section 2.1.2). As such, this optimization process can only adjust weights based on observed training data, meaning that the model can only learn patterns and relationships present in its training distribution. Its focus on local error minimization over global understanding, its reliance on vast datasets, and its

inability to represent abstract symbolic knowledge all underscore the limitations of sub-symbolic systems in achieving deeper understanding of visual scenes.

Finally, deep learning models are unable to transfer knowledge to other domains. According to Marcus, understanding requires out of distribution generalization [50], applying learned patterns to data points beyond the training distribution. Similarly, Mitchell states that genuine understanding necessitates the ability to form abstract representations and reason analogically, enabling systems to generalize across diverse contexts [55]. In the words of Cropper, et al. we could call this ability "Knowledge Transfer" [13, p.6]. Thus, generalization is necessary for understanding. However, deep learning models cannot perform out of distribution generalization to other domains (this includes wholly different tasks, but also unobserved variations of the same task). Even at a basic level, a deep learning object detector can only detect 'cat' and 'dog' if it has been trained on a lot of images of cats and dogs. But if we subsequently ask it to classify a car, then it gets stuck [50], [49], [51]. Arguably, this is a major reason why vision models struggle with grasping deeper understanding of visual scenes.

We clearly see the relevance of these problems if we think back about the application of computer vision models in autonomous systems. Self-driving cars use computer vision in order to engage with their surroundings. However, these cars will unavoidably enter into new, unseen situations of which no labeled data was available. As such, they should be able to generalize their learned patterns to new situations [57], [84]. Furthermore, it would be desirable if the car could actually reason about relations between the objects it detects. For example, if the car detects multiple humans on the road we would like it to stop. Or if two object trajectories are on a collision course with the car, we would like it to avoid them. However, as we saw, in pure deep learning systems this is impossible. Thus, despite being able to effectively generate complex features of the data, the more recent models cannot fully bridge the gap between perception and reasoning.

3.4.1 Toward Hybrid Approaches

But, how do we actually bridge that gap? How do we understand visual scenes? This is not a novel question. Kant already aimed to answer this in his *Critique of Pure Reason* [32]. And, whether we can actually use this for computer vision is still an open question.

Due to the fact that most computer vision models still rely solely on neural networks, this critique can be said to extend to contemporary computer vision models as well. To that end, Marcus proposes a move to hybrid systems: neuro symbolic AI models.

However, is this all there is to this? What does this problem really boil down to?

My claim here is that we can interpret Marcus' critique through a Kantian lens. Using the definition of Kantian intelligence and coherent object representations, we can see that the underlying problem is that contemporary computer vision models do not have coherent object representations. The following chapter will discuss this idea in more detail.

Chapter 4

Kantian Intelligence

In his *Critique of Pure Reason* [32] philosopher Immanuel Kant aimed to show how we could integrate empiricism and rationalism into one system.

Whereas rationalists such as Descartes and Leibniz held the view that reason is the chief source of knowledge, asserting that reality has a rational structure that can be known through deduction, empiricists such as Locke, Hume posited that knowledge comes primarily from sensory experience, emphasizing observation and experimentation in understanding the world.

Kant, on the other hand, claimed that human intelligence consisted of applying abilities of logical processing (the faculty of the understanding) to sense data (the faculty of the imagination) in order to make sense of our experiences. In the words of Richard Evans, Kant's notion of intelligence has provided us with a cognitive architecture which helps us answer the question: "What, exactly, is involved in combining low-level perception with high-level conceptual thinking?"

We find this philosophical debate about the nature of intelligence echoed in contemporary AI research: "*...perhaps the most fundamental debate in artificial intelligence has been whether AI systems should be built on symbol manipulation — a set of processes common in logic, mathematics and computer science that treat thinking as if it were a kind of algebra — or on allegedly more brain-like systems called "neural networks."*" [50, p. 1].

Namely, it is being increasingly acknowledged that the strengths and weaknesses of neural networks and logic-based learning are complementary. In the previous section, I showed that neuro symbolic AI has emerged as a way to combine both of these approaches. At the same time, I showed that the weaknesses of deep learning neural networks also extend to the field of computer vision, which uses mostly deep learning-based models. This raises the question whether neuro symbolic AI can solve some of the problems of deep learning-based computer vision models.

However, although Neuro symbolic AI provides a promising approach, it is still met with a lot of criticism. This critique mostly comes from "Optimist" AI researchers who believe that deep learning alone can get us to a humanlike-level of understanding. [65] Their stance is mainly predicated on the argument that deep neural networks (DNNs) on their own can learn abstract, generalizable concepts. This critique, then, can also be extended to computer vision models. Namely, these models should be able to generate abstract, domain-general concepts from data alone.

In this section, contra to the claim of optimists, the claim will be defended that: Kant's philosophy of cognition can be beneficial to computer vision, because it: 1. helps us to redefine the underlying problem with deep learning-based computer vision, and 2. it points us towards a new architecture: neuro symbolic AI.

In order to arrive to this claim, I will show that: 1. Kant's philosophy of cognition provides us with a notion of 'intelligence as a faculty of combination'. Humans learn logical categories, by applying innate logical faculties in experience, to sense data. This process entails a combination of distinct types of representations: intuitions and concepts. Thus, 2. the faculty of combination provides us with "coherent object perceptions": the combination of distinct types of representations enables us to learn abstract (associative and implicative) invariances from data. Next, 3. neuro symbolic AI can also exhibit this "faculty of combination". The end-to-end connection of inductive logic programming with deep learning, through the combination discrete and continuous representations, can enable systems with the ability to learn logical concepts from data. As a result, 4. neuro symbolic ILP (as a faculty of combination) can contain "coherent object perceptions": the combination of distinct types of representations enables abstract (associative and implicative) invariances to be learned from data. After which I note that,

5. current computer vision models, often being purely deep learning-based, do not exhibit coherent object perceptions. They can not construct abstract, domain-general invariances. Finally, I will propose that, 6. neuro symbolic ILP can supply computer vision with 'coherent object perceptions' because it can learn abstract invariances due to its combined representations. After all this, it will be clear that Kant's philosophy of cognition can help the field of computer vision because it can help: 1. to redefine the problem with deep learning-based computer vision models, 2. point into the direction of a new architecture to solve this problem.

4.1 Kantian Intelligence as a Faculty of Combination

In his *Critique of Pure Reason*, Immanuel Kant aimed to address the capacities and limits of human reason, by reconciling the seemingly conflicting branches of knowledge represented by rationalism and empiricism [32]. I will now elucidate these concepts in order show Kant's notion of intelligence as faculty of combination, and that this faculty of combination leads to coherent object perceptions. Their explanation will not cover the complete breadth, depth and complexity of the distinction, but serves our purpose.

4.1.1 The Debate of Knowing: Rationalism and Empiricism

Rationalists such as Descartes and Leibniz held the view that reason is the chief source of knowledge and asserted that reality has a rational structure which can be known by performing logical deduction on innate categories. Their main reason for this is the "Superiority of Reason" thesis, which states that "the knowledge we gain in subject area S by intuition and deduction or have innately is superior to any knowledge gained by sense experience" [54].

Descartes provides an example of this with his account of wax in his book *Meditations on First Philosophy* [19, p. 22]. In *Meditations*, Descartes shows that wax is not wax because of its shape, color or texture. This is because of the fact that, were all of these qualities to change, then the substance would still be wax. On the contrary, Descartes states that wax is perceived "by the intellect alone" [19, p. 22].

Only when one understands the mathematical principles of the substance, such as its expansion under heat, figure, and motion, can the knowledge of the wax be clear and distinct. Only then can it be true knowledge. One should note here that, for Descartes, expansion under heat, figure and motion are mathematical, innate concepts, rather than empirically learned concepts. As such, it becomes clear here that, for Descartes, the wax can only be comprehended by the intellect a priori.

On the other hand, empiricists, such as Locke and Hume, posited that knowledge comes primarily from sensory experience. As such, they emphasized that humans' logical categories arise mainly from observation and experimentation rather than rational deduction. We could roughly summarize their view with the "Empiricism Thesis": "We have no source of knowledge in S or for the concepts we use in S other than experience" [54]. Thus, if we are to have any knowledge, it comes from experience. Notably, this thesis makes a less strong claim to knowledge. It does not claim that we can have any knowledge at all. It merely claims that if we can have knowledge about something, then this knowledge comes from experience. We find an example of this in Hume's rejection of understanding of causal relations.

According to Hume, our understanding of a causal relation between A and B cannot be grounded in pure sensory input, because this sensory input only provides us with the temporal succession of A and B. The sensory input does not provide us with a connection between A and B. There is no way of knowing whether this relation will still hold tomorrow.

Thus, Hume rejects that we can have true knowledge of a causal relation between objects because true knowledge should come from experience, however we can not get true knowledge from experience. As such, Hume's argument is empirical. If the real meaning of logical concepts can be attained, it can only be attained from experience.

4.1.2 Kantian Intelligence as Faculty of Combination

Contra both rationalists and empiricists, Kant argued that human cognition arises from the synthesis of intuitions (received from the faculty of sensibility) and the application of innate conceptual frameworks (given from the faculty of understanding) to these intuitions, with the faculty of the imagination mediating between these two faculties. Said differently, "thoughts without content are empty, intuitions without concepts are blind" [32, A51/B76]. Consequently, it is just as necessary to make the mind's concepts sensible — that is, to add an object to them in intuition — as to make our intuitions understandable by bringing them under concepts.

In the words of Hyeongjoo Kim, we can interpret Kant's position as that "the key to human intelligence is simply the conceptualization of sensible information" [35, p. 140]. As such, Kant's position can be seen primarily as a "faculty of combination" [35, p. 142]. Mental concepts need to be supplemented with empirical sense data, and vice versa, in order for humans to experience anything at all. However, how is this any different from the rationalist position of applying a priori mental concepts to empirical data in order to make sense of them?

Well, for Descartes, the logical concepts are innate, a priori. They are given to us independent of experience and we can utilize them through proper logical reasoning. However, according to Kant, following Evans' interpretation, what is innately given is not our concepts (or categories) themselves, but rather our faculties whose application to sense data produces those categories: "The pure unary concepts are not 'baked in' as primitive unary predicates in the language of thought. The only things that are baked in are the fundamental capacities (sensibility, imagination, power of judgment, and the capacity to judge). The categories themselves are acquired – derived from the pure relations in concreto when making sense of a particular sensory sequence" [23, p.74].

As such, for Kant, logical concepts are synthetic a priori. They are a priori true, but they can only be comprehended as such when we use them to understand our experience. For example, when the question is: *"how do we form the general idea of a triangle when we have only seen a series of particular examples?"*, Descartes might respond that we form a general idea of the triangle because the triangle is subject to mathematical truths which are universally true and a priori accessible.

For Kant, on the other hand, the concept of a triangle is grounded in our intuition of space, which allows us to construct the figure a priori in our minds. The truths about triangles, such as the sum of their interior angles, are synthetic a priori insofar that they extend our knowledge about reality, but do not derive from empirical examples. Thus, for Kant, it is only by first envisioning a particular triangle that we can come to understand the mathematical rules of any triangle as such. Thus, unlike Descartes, Kant emphasizes that knowledge depends on the synthesis of concepts and pure intuitions, rather than solely on logical deduction.

As such, we can frame Kant's answer in terms of the combination of sensory and conceptual representations. Only by actively applying innate logical functions to experience, can we experience reality in a conceptual manner. Thus, it is only by applying logical functions to raw sense data that we can learn synthetic a priori categories. [32, A140/B179]

4.1.3 Coherent Object Representations

But then, for Kant, how does this faculty of combination provides us with coherent object perceptions? And what do we mean by coherent object representations? Well, before, we saw that Kantian intelligence can be seen as a faculty of combination. Synthetic a priori categories are learned by applying innate logical faculties to experience and vice versa. These faculties (the faculty of sensibility and the faculty of the understanding) provide the mind with distinct representations of the same object: intuitions and concepts, respectively. [32, A19/B33]

As such, for Kant, producing coherent object perceptions becomes a process of synthesizing these different representations: unifying the intuitions by applying the concepts. "Synthesis is ... the action of putting different representations together with each other and comprehending their manifoldness in one cognition" [32, B103].

Here, an intuition is an immediate relation to an object. It presents an appearance directly to us [32, A109]. In Evans' words, an intuition is a representation of a particular object (e.g., this particular triangle) or a representation of a particular attribute of a particular object at a particular time (e.g., the red color of the particular triangle at this specific time under these specific lighting conditions). In logic terms, we could easily denote this immediate relation as `red(X)`, where `X` denotes the triangle.

The faculty of sensibility provides us with a manifold of intuitions [32, B68]. For example, the color of the triangle can change over time if the lighting conditions change. Each new perception of the triangle under new lighting conditions provides us with a different intuition.

Concepts, on the other hand, are a distinct type of representation. Whereas an intuition represents a particular object, a concept is a general representation that many intuitions may fall under [32, B377]. As such, concepts serve as rules for unifying and organizing the manifold of intuitions [32, A106] which sensibility leaves us with. The generality of concepts allows us to think about objects abstractly, beyond their immediate appearance. As such, they only relate to an object mediated through the intuitions.

For example, according to the Triangle Inequality Theorem, three line segments form a triangle if the sum of the lengths of any two segments is greater than the length of the third segment. This can be

represented by the following conditions: $a + b > c, a + c > b, b + c > a$. If all three conditions hold, then something counts as a triangle. This conception of a triangle is a concept which applies to all possible intuitions of triangles. It applies to triangle `X` when `red(X)` is an intuition, but also when `yellow(X)` is an intuition.

Thus, intuitions and concepts are different representations of the same object because they have a different relation to that object. Intuitions relate directly to an object. Concepts relate directly to intuitions, but indirectly to an object. As such, concepts' relation to objects is mediated by multiple intuitions of that object.

Importantly, this combination of these distinct types of representations, following from the combination of the faculties, enables abstract (associative and implicative) invariances to be learned from data. Remember that in section 3.1 we defined invariances as properties of objects which do not change despite manipulations made to the object. Following Reza Negarestani's interpretation of Kant, we could say that in the mere intuiting of objects, we are dealing with local variations and rudimentary invariant aspects of particular items [58, p.158]. For instance, different shapes, colors and sizes of triangle `X`.

However, "local variations (manifolds) of items (shape, [colour], etc.) supplied by the ... [faculty of sensibility] ... are not by themselves sufficient for object-construction since object individuation, classification, and recognition involve the construction of types of invariances which form a network of associations and implications" [58, p.158]. The general character of concepts, their status as logical functions, means that they provide us with (domain-)general and universal invariances of triangle `X`. Effectively, these invariances are higher order associations and implications about objects. They enable us to think about, what if I remove two sides of this triangle, would it still be a triangle? Importantly, "such invariances cannot be obtained without a complex interplay between the sensory intuitions ... and the pure concepts of understanding" [58, p.159]. This is because the intuition only provides us with empirical, visual features of the object, such as redness, brightness, position. The intuitionist representation of the object does not provide us with categorial features of it, such as negation (not-ness), disjunction (or-ness), universality (all-ness) or particularity (some-ness) [66, p.463].

As such, it can be stated that in order for the Kantian mind to construct coherent object perceptions, we need to combine the distinct representations of an object `X`, and this combination enables the mind to recognize intuitions of objects as coherent through the mediation of domain-general invariances.

By applying the concept of the triangle to the particular intuitions of triangles, the mind abstracts away from the particular, local, features (size, color, exact angles) of the particular triangle, in order to capture the general properties of triangles as such. They are domain-general, because this enables recognition of the triangle independent of the context it is in. These invariances enable us to recognize *any* triangle.

This will be highly relevant for my critique on deep learning-based computer vision models later on, but for now, let us proceed to show that Neuro symbolic AI (ILP) can be said to exhibit some form of Kantian intelligence, because it is also a faculty of combination; it provides us with abstract invariances for intuitions of objects by combining these intuitionist representations with conceptual representations.

4.2 Interpreting AI Paradigms Through Kant

Thus, I tried to characterize the debate between rationalism and empiricism as the debate between the question whether concepts are innately given, or empirically learned. Furthermore, I showed that Kant, with his notion of intelligence as a faculty of combination, aims to find a middle ground between the two by conceptualizing only our logical functions as innate, providing us with synthetic a priori concepts. As a consequence, we saw that coherent object representations are those representations which exhibit abstract and general invariances because they subsume intuitions under concepts.

This debate between empiricism and rationalism, and their consequences of object representations, can be said to extend to the current debate in AI research, between logic-based AI and deep learning: "the neural network is the intellectual ancestor of empiricism, just as logic-based learning is the intellectual ancestor of rationalism" [22, p. 19]. Likewise, Kant's unification of empiricism and rationalism can be said to point to a hybrid, neuro symbolic AI architecture. In, [22], Richard Evans already made this claim, however, he does not elaborate on why this actually is the case. As such, I will elaborate on this further.

In this section, I will provide evidence to show that a) logic-based AI, like rationalism, falls short of Kantian intelligence because it relies solely on innate concepts (pre-defined symbolic rules without meaningful connection to sensory experience), b) Deep learning, like empiricism, falls short of Kantian intelligence because it attempts to derive all knowledge from data alone, without innate logical capacities for combining and structuring that information, and c) Neuro symbolic AI approaches Kantian intelligence

more closely. It functions as a faculty of combination. As such, it can, in theory, learn abstract invariances from data leading to more coherent object representations.

4.2.1 Logic-Based AI as Rationalist AI

In section 2.1.1, we saw that logic-based AI is well-suited for deductive reasoning, due to its ability to check logical rules for satisfiability. We also saw that it operates on discrete, relational data like logic programs as opposed to noisy, continuous data, like an image or video.

Let us return to the triangle example that we saw earlier. We can construct a logic-based AI system in order to determine whether a discrete input is a triangle. The task of a logic-based system would be to determine if three line segments with lengths a, b, and c can form a triangle. Remember the Triangle Inequality Theorem from before. To reiterate: three line segments form a triangle if it satisfies the following conditions: $a + b > c, a + c > b, b + c > a$. Using logic programming, we could provide the conditions as knowledge base and the sizes of the three line segments as input. We could then check whether substitution of the variables makes these conditions true ('satisfiable', in logic terms) and, thus, if something satisfies as a triangle.

This discrete representation in logic-based AI parallels Kant's notion of concepts in an important way. Just as Kantian concepts are general rules that can apply to multiple instances (like the concept "triangle" applying to all possible triangles), discrete representations in logic are symbolic abstractions that capture general properties. For example, the logical predicate `isTriangle(X)` could represent the rules that can be applied to any input `X`, much like how Kant's concept of a triangle provides rules for recognizing any triangle independent of its context.

However, there's a crucial difference. While Kant's concepts are meant to be actively synthesized with intuitions through the faculty of combination, logical discrete representations remain purely symbolic and disconnected from sensory experience. They do not combine with continuous sensory data. As such, the discrete concepts are innate ideas; predefined rules that exist independently of experience.

We clearly see this in our example. It applies the a priori concept of a triangle to interpret sensory data and leaves no room for the concept of triangle to be formed by the data. The logic-based AI's a priori concept of a triangle is fixed and predefined. It lacks the capacity to dynamically learn the concept by applying the rules in experience and, thus, it does not synthesize the a priori concepts with empirical experience. Consequently, the concept of a triangle is not formed through application of the concept to sense data, as was the case for Kant.

What this effectively means is that logic-based AI's object representations are incoherent. The concepts, because they are not dynamically related to the object, are empty: brittle and static. Small changes or noise in the input data can make the model unable to find a satisfiable solution. Thus, unimportant changes made to the object do not keep the conceptual representation of the object consistent (the object can become unsatisfiable). As such, the concepts applied by logic-based AI, although abstract and general, are not invariant. Superficial changes in the object can easily change the conceptual representation of the object (from true to false and vice versa). As such, this form of rationalistic AI does not exhibit Kantian intelligence as a faculty of combination and, consequently, also does not exhibit coherent object representations.

4.2.2 Deep Learning and Neural Networks as Empiricist AI

Conversely, take the case of determining whether an image exhibits a triangle using a CNN. Here, we would train the CNN on a lot of variations of images where a triangle occurs. The images could contain weirdly shaped triangles, edges with different colors, different background, different lighting conditions, etc. Including as many of these variations would increase the generalizable capability of the model [41]. It is important to provide as much data as possible, because the CNN does not have any innate conception of what a triangle is. It does not contain an a priori representation of the Triangle Inequality Theorem.

Given this fact, it becomes clear how neural networks can be seen as empiricist AI. Earlier, we showed that empiricism learns a posteriori concepts to make sense of sense data and that the only knowledge we can have comes from experience (the "Empiricism Thesis"). As such, the concepts which empiricist AI learns are completely determined by experience or the input data.

The object representations of this empiricist approach to learning align with Kant's notion of intuitions. Just as Kantian intuitions provide immediate, particular representations of objects through sensory experience (like seeing this specific triangle under these specific conditions), neural networks learn to represent objects through their distributed, continuous representations of sensory data. The activation

patterns in a CNN's layers can be seen as analogous to intuitions. They directly represent particular features of the input, like edges, textures, and shapes, without any predetermined conceptual framework. For example, when processing an image of a triangle, the early layers capture immediate visual features (edges, corners) much like how our sensory intuitions first encounter particular aspects of objects.

This is exactly what happens in our example: the models build their understanding of the world purely through exposure to examples of images containing triangles, without any pre-programmed rules or concepts of what constitutes a triangle. Providing a completely different training data set would give the model a completely different representation of a triangle.

What this means for their **object representations** is that these are highly invariant to visual features. CNNs, for example, are invariant to changes in object position in the image, meaning that the triangle position in the image does not negatively effect the models ability to recognize it as one. [41], [26]

However, as such, similar to logic-based AI, neural networks also do not align with Kantian intelligence because they do not learn abstract concepts as a result of combining the faculties. They do not combine symbolic representations with distributed representations, nor do they have some form of innate 'spatio-temporal manifold'; an internal world model guiding their concept learning [51, p.145].

This is made even more apparent with the problem of 'spurious correlations' in neural networks which we already saw before [49]. According to Marcus, the decisions of neural networks are often based on nonsensical correlations. For example, a model could detect something as a triangle, just because there is a patch of grass in the left-upper corner. As such, these models seem far from a level of Kantian intelligence.

4.2.3 Neuro Symbolic AI as a Faculty of Combination

Thus, we saw that neither logic-based AI, nor neural networks approximate Kantian intelligence well, because neither one respects the condition of being a faculty of combination: learning synthetic a priori concepts by applying innate logical processes to make sense of experience. Because of this, they also do not exhibit coherent object representations: abstract, domain-general invariances.

I will now show that neuro symbolic AI (which we defined in section 2.1.3 as neuro symbolic ILP) approximates this intelligence more closely, because it can learn such synthetic a priori concepts from data by using innate logical processes. As a result, it exhibits more coherent object representations.

Neuro symbolic AI brings us closer to Kantian intelligence. In logic-based AI , we saw that the logical rules which are defined are often very task-specific rules [18]. For example, when trying to recognize a triangle in images, we have to define a priori what a triangle looks like and then train a neural network to recognize that in an image.

With neuro symbolic AI, on the other hand, we only have to define very general logical rules. For example, all objects are connected in space, and all objects, when not interacted with, persist through consecutive time steps [23]. These general logical rules guide the neural network learning in such a way that the logical representation of the triangle can be learned from raw visual data. Furthermore, the model can itself learn new, latent logical rules and predicates needed to make sense of a sensory sequence.

These learned logical rules and predicates are not purely relative to the data, as we saw with CNNs. On the contrary, the idea is that the learned logical concepts are generalizable beyond the training data to new, unseen data and that they can even help humans in learning higher-order concepts [22], [13]. As such, it could be stated that the learned logical concepts are a priori true, at least to some degree: they are also applicable to situations independent from the models' experience.

However, they are also synthetic, because they are learned by applying the innate logical functions of the model (the abstract rules about space) in order to make sense of the experience. As such, the learned logical concepts can be said to be synthetic a priori. This is in contrast to the a priori concepts applied in logic-based AI and to the data-relative concepts learned by deep learning models. In this way, we can see that this second form of Neurosymbolic AI, ILP, brings us closer to Kantian intelligence.

All in all, neuro symbolic AI models seem to bring AI research closer to exhibiting Kantian intelligence, due to the fact that both are characterized by a 'faculty of combination'. We saw that logic-based AI exhibits a form of rationalism because it defines a priori concepts and applies these to sense data, without the option to update the concept itself. Neural networks, on the other hand, aim to learn abstract concepts purely from data. As such, the concepts become highly subjected to the training data, which is highlighted by the problem of spurious correlations.

4.3 From Awareness to Understanding: Applying Kantian Intelligence to Computer Vision

In section 3.1, we saw that contemporary computer vision models, despite their great success in performing annotation based tasks, are still bad at high-level tasks like visual understanding. This is especially the case for video data. For example, CNNs only learn translational invariance, ViTs need large amount of data to be trained on, VLMs still struggle with higher-order tasks like scene understanding, counting and visual analogies, all of which are essential tasks for measuring the ability to generalize, and, thus, for visual understanding. Even OCRA (Object-Centric Relational Abstraction) [81], which imbued a slot attention model with a relational bottleneck, so that it can, arguably, learn relations and higher-order relations between objects and, thus, claims to be able to generalize to new domains, does not exhibit visual understanding in the Kantian sense. Their relational bottleneck computes pairwise relations and struggles to capture complex higher-order interactions that involve multiple objects simultaneously. Many complex relationships emerge not from direct pairwise relations but from the interaction of multiple pairwise relationships. For example, in a circular arrangement of objects, we could say that the overall structure is defined by the collective contribution of all pairwise distances (for example, object A is the same distance from object B as object C is from object D). This cannot be fully captured by independent comparisons. Detecting such patterns requires explicit higher-order relational operators or reasoning mechanisms capable of processing all objects collectively. Since OCRA lacks such mechanisms [81], it struggles to represent and reason about these relationships.

Fractured Perceptions: Interpreting Marcus' Critique Through a Kantian Lens. Computer vision models do not exhibit visual understanding because, through a Kantian lens, visual understanding requires coherent object representations. These are representations of objects which are both intuitive and both conceptual, they both have a direct relation to the object and an immediate relation to the intuitions of the object. These representations, as we saw for Kant, can only arise through the process of combining the faculties. Intuitions need to subsumed under concepts, represented to the mind *as* conceptual perceptions.

However, contemporary computer vision models, due to their dogmatic reliance on deep learning for concept learning, purely exhibit distributed representations of objects. They only exhibit intuitions; representations of objects based merely on their visual invariances. They can deal well with perturbations in object position, size, brightness or rotation, however they can not explain why. As such, they are unable to learn abstract domain-independent invariances.

4.4 Counterarguments

However, there are some clear points of critique we could make against this view. We will treat them, and their rebuttals, in turn.

1. Are we justified in viewing Kantian intelligence as human intelligence? In the field of cognitive science and philosophy of mind, there are countless views on intelligence. Why do we not just choose another view of intelligence? Although the Kantian view of intelligence is only one but many, it is a very influential one. It has been shown that Kant's theory of cognition has been very influential for contemporary paradigms in cognitive science, such as Predictive Processing, Functionalism and Enactivism [65].

These cognitive paradigms, in turn, are still very influential in AI development. As such, I believe Kantian intelligence provides a foundational view in intelligence, which has a lot of connection to current theories and practices, yet can still be critical enough, due to the fact that it is not a predominant paradigm itself.

2. Can we not jut encode logical abilities into neural networks? In their seminal paper, *Neurocompositional computing: From the Central Paradox of Cognition to a new generation of AI systems*, Smolensky et al. make the claim for neurocompositional computing, as opposed to neuro symbolic AI [73].

Basically, their claim is that we can encode logical functions, like compositionality, into neural network tensors, which would enable them to learn higher-level abstractions of concepts. Arguably, this would absolve the need for logical modules connected to neural networks in order to get to more human-like levels of understanding in AI, and thus for hybrid neuro symbolic AI in general. Their stance is predicated

on the idea that deep learning models have inductive biases enabling them to learn better. For example, in the case of CNNs, this is the bias towards textures [26].

Although this is a valid point of critique, it should be noted that this is, in fact, not any different than Descartes' rationalist stance. Although the model learns compositional concepts, these concepts are already predefined in their implementation in the DNN. Smolensky and colleagues define compositionality as the tensor product and integrate this in the layers of the DNN. However, as such, they still define the concept of compositionality a priori and they still subject sense data to this pre-defined concept. Furthermore, the structural operations of the DNN itself do not change in learning, only the value for the parameters. This is different from neuro symbolic AI, where the operations themselves can change (the concepts through which the intuitions are unified). So, in the Kantian sense, Neurocompositional computing does not bring us any closer to intelligence than Descartes' naive rationalism.

Chapter 5

The Apperception Engine

However, this was all still very theoretical. In practice, how can we envision a neuro symbolic AI model which can be applied to computer vision tasks? How can we envision a model which can learn abstract, domain-general invariances from visual intuitions, produced by the deep learning component, thereby unifying those intutions? The Neuro Symbolic Apperception Engine seems to be a potentially suitable model.

Specifically, The neuro symbolic Apperception Engine seems like an interesting model to apply to video tasks. This is mainly due to the fact that: 1. the model is specifically meant as a machine learning implementation of Kant's theory of cognition, 2. the model is meant as an unsupervised sequence model, aimed at making sense of sensory sequences and videos are themselves sequential data, because they are build up of sequences of frames, 3. Evans himself already applied the model very successfully to rudimentary sequential image tasks, like the sequential MNIST task and the Sokoban task. [24] As such, initially, the model would seem a good fit. [22] However, it has not yet been applied to actual video data or video understanding tasks. This raises the question as to what extent we could apply the model to this.

In its entirety, Evans' project can be said to consist of three contributions: 1. A formalization of Kant's aim in the *Critique of Pure Reason* to unify the intuitions: 'making sense' of a sensory input sequence, 2. A logic-based machine learning system "The Apperception Engine" aimed to implement the necessary conditions of this notion, 3. A hybrid, neuro symbolic architecture which connects the Apperception Engine to a Binary Neural Network (representing the perception module) in order to make sense of noisy data. [22], [24], [25].

5.1 A Computational Framework of Making Sense

> *Doomed to generalise... the only rules that the system is allowed to produce are general rules that quantify over all objects and all times. [24, p.31]*

In general, the Apperception Engine aims to 'make sense' of a temporal input sequence. A theory θ **makes sense** of a sensory sequence S: 1. if the trace of θ explains S, and 2. if the trace of θ satisfies the four conditions of unity.

A theory θ **explains** a sensory sequence S if its trace $\tau(\theta)$ **covers** S, written as $S \sqsubseteq \tau(\theta)$. This means that for a finite sequence $S = (S_1, ..., S_T)$, each element $S_i \subseteq A_i$ where $\tau(\theta) = (A_1, A_2, ...)$. As such, the trace can be said to provide a complete and determinate description of the infinite time-series, while S represents only a partial view. [25] Thus, the trace is the regeneration of the input sequence and serves to evaluate the potential theories generated by the system.

However, making sense of a sequence requires more than just coverage of the input sequence. The theory must also be unified. Unity is required at four distinct levels corresponding to the fundamental elements of a trace: objects, predicates, ground atoms (grouped into states), and the sequence of states itself. These conditions ensure that the theory isn't just a collection of disconnected elements but forms a coherent, integrated whole where each part relates meaningfully to the others.

Unifying intuitions means combining them using binary relations to form a connected graph, in such a way as to satisfy various unity conditions [23]. The four unity conditions are:

1. **Spatial unity:** objects are united in space by being connected via chains of relations
2. **Conceptual unity:** predicates are united by constraints

3. **Static unity:** atoms are united in a state by jointly satisfying constraints and static rules

4. **Temporal unity:** states are united in a sequence by causal rules

For their logical description, see [25, p.12-15]. According to Evans, the four unity conditions represent both a strong inductive bias and domain-independent prior knowledge. As such, the Apperception Engine, in theory, should be able to make sense of *any* sequence [22].

This highlights the difference between supervised ILP models, like POPPER [14] which require positive examples, negative examples and background knowledge, as shown in section 2.1.1. Contrary to this, the Apperception Engine is an unsupervised model. It takes as input a sensory sequence and background knowledge. As output it will provide us with causal dynamic rules. These are rules which define the steps needed for changes between the input states. They are in the form of Datalog∈ causal rules.

Thus, as output, the Apperception Engine produces a causal theory, defined in a first-order logic program, which aims to explain: 1. the initial conditions of-, 2. the evolution dynamics (how the states change over time) of- and 3. constraints of the sequence of observations.

5.2 The Algorithm

In section 2.1.1, I already touched on the fact that the search method in ILP systems consists of a discrete search in the hypothesis space. What this means is that finding a good theory becomes a search problem [22].[1] However, in this space, contrary to continuous vector spaces in neural networks, good solutions do not necessarily lay close together. See figure 5.1 for a visualization.

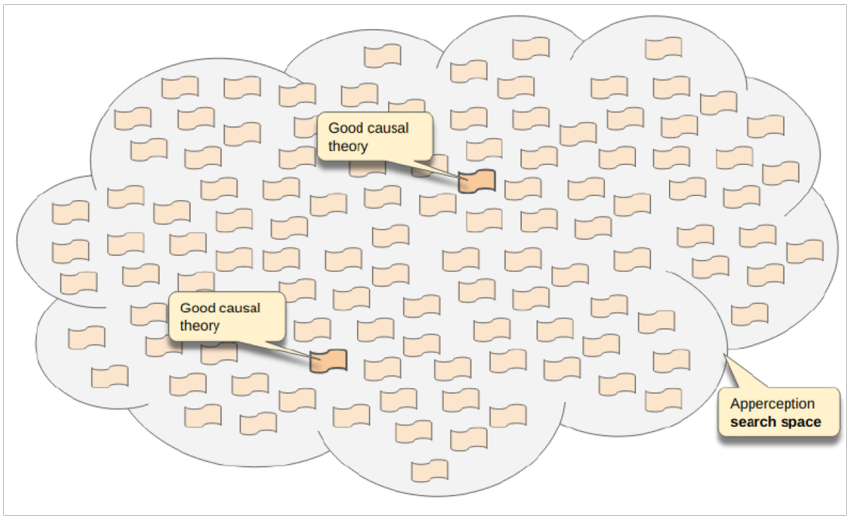


Figure 5.1: A visualization of the Apperception Engine search space. Image taken from [11].

Furthermore, the hypothesis space is infinitely large, because there can be infinitely many ways to represent the solution to a problem. We could for example add dummy predicates *ad infinitum* which would neither disprove the hypothesis, nor make it satisfiable. As a consequence, a brute force approach, trying the calculate all the possible hypotheses, will not work [22]. In, [11] the authors use multiple heuristics to cut down the complexity of the search space, at the risk of lower accuracy. I, however, follow Evans' Apperception Engine for scientific consistency.

As such, the search algorithm works as follows, see figure 5.2 for a visualization.

[1] Although, it might sound strange to view the Kantian process of unifying intuitions as a computational search space, this view has more proponents. For example, Reza Negarestani following Wilfrid Sellars states that "In this sense, transcendental logic, as that which supplies concepts with sensory intuition and applies classificatory concepts to intuitions, can be understood as a computational search space." [58, p.162]

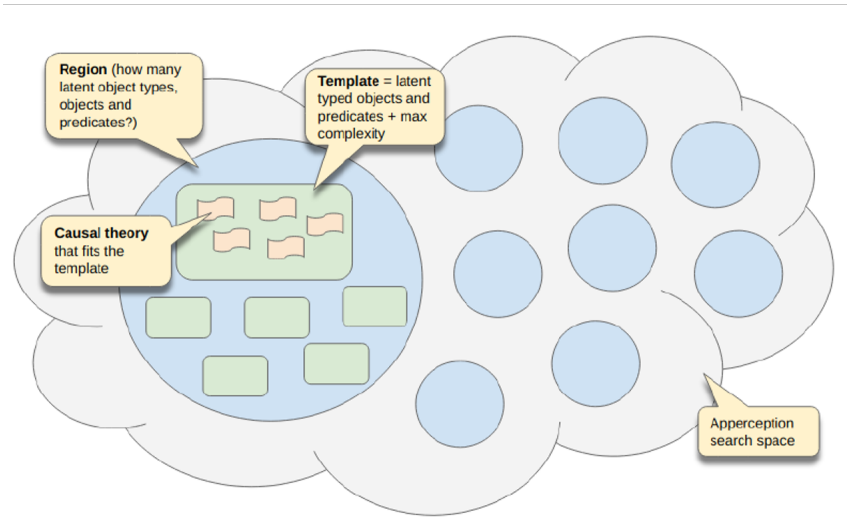


Figure 5.2: A visualization of the Apperception Engine search algorithm. Image taken from [11].

The Apperception Engine employs a search strategy which partitions the infinite search space into manageable regions. This partitioning is based on how "imaginative" a causal theory can be (how much predicate invention on the hypothesis is allowed). Specifically, it is based on the extent to which unobserved objects, properties, and relations can be added to the vocabulary.

The search process involves two main tasks. First, the engine iterates through templates, starting with a base template and progressively increasing template complexity. Each template defines its own vocabulary (extended from the base vocabulary) and sets limits on the number and size of static and causal rules, in accordance with the vocabulary's richness. Second, within each template, the engine searches for successful causal theories that are expressed in the template's vocabulary and do not exceed a maximum complexity threshold.

The algorithm follows a systematic approach: it begins with the simplest (base) template and iteratively increases template complexity until either a good solution is found or a time limit is reached (Evans set his time limit at 4 hours). This approach ensures that simpler solutions are considered before more complex ones.

While this partitioning strategy has the great advantage of ensuring that searches can terminate, it comes with a notable limitation: a satisfactory causal theory needed in order to explain the sequence might exist, but not be found within the allotted time if it lays in an unexplored partition of the search space. As such, the systems' search algorithm contains a trade-off between completeness and efficiency. [22]

5.3 Implementation Language and Complexity

The Apperception Engine uses a combination of Datalog, Datalog€, and ASP to model and reason about static and dynamic systems. Specifically, ASP is used as a meta-language to implement a Datalog interpreter. This enables the declarative encoding of the semantics of Datalog and its temporal extension, Datalog€ [22, p.22]. This design makes use of ASP's capabilities to define and enforce constraints, explore program spaces, and optimize solutions [22, p.27].

The integration of these languages has significant implications for expressivity, decidability, and computational complexity. Datalog supports static reasoning with polynomial-time (P) data complexity, but its program complexity grows exponentially ($EXPTIME$) due to grounding. Datalog€ extends Datalog with temporal reasoning. Although this enables dynamic modeling, it also increases computational costs, potentially reaching PSPACE-hard complexity. ASP supports optimization and dynamic reasoning via stable model semantics, where finding solutions is NP-complete, and optimization problems rise to Σ_2^P-complete. This point will be important for my discussion point about my complexity analysis.

5.4 The Apperception Engine in Action

As stated in the introduction of this chapter, Richard Evans has himself already applied a neuro symbolic variant of the Apperception Engine to rudimentary computer vision tasks (the sequential MNIST and Sokoban task). These tasks both consist of low-dimensional binary image sequences. The task of the Apperception Engine is to find a causal theory which can make sense of the sequence. What these tasks both have in common is that they test the causal theory by prediction. The tasks consist of asking the engine whether it can predict the next states.

I will not treat these into further detail here. The Sokoban task will be explained in more depth in Chapter 4. For the sequential MNIST task, see [24]. Evans has shown that the Apperception Engine can score well on these tasks with only one training sample (one sequence) [22], [24].

However, the main reason for treating these here is to show *how* Evans implemented the neuro symbolic architecture. As shown in, 2.1.3, a big problem in hybrid neuro symbolic architectures is how to deal with the interface problem. How do we combine the discrete, localist module with the continuous, distributed module? The way Evans solves this is by implementing a constrained version of a BNN as a single logic program in ASP. As such, "the state of the network can be represented by a set of atoms, and the dynamics of the network can be defined as a logic program" [22, p. 107]. Thus, Evans avoids the interface problem by making the neural network a binary logic program, meaning that the raw data itself will also be processed as a logic program. This raises the question whether the Apperception Engine *as is* is suitable for computer vision tasks on videos. The next chapter will explore this further.

Chapter 6

Experiment

6.1 Introduction

As we saw in the previous chapter, the neuro symbolic Apperception Engine has shown promising results in learning causal theories from sequential data using grid-based representations through predicting the next time step[22], [24], [25]. However, the data for these tasks was still very low dimensional, thus in order to see if the Apperception Engine could be applied to computer vision tasks, I wanted to test for higher dimensional data: grids representing object states in videos. In [82], [60] they test the capability of their unsupervised models for frame prediction in videos. Evans' achieved success with Sokoban-style grids suggests that similar grid-based approaches could be effective for understanding object dynamics in video data with the Apperception Engine. Due to the high complexity of the task, focus will be laid of the data complexity of the task [1] The CLEVRER dataset [85], with its focus on *in vitro* object interactions and causal relationships, provides a good testbed for this.

As such, this chapter will examine the feasibility of using grid-based representations for processing CLEVRER videos with the Apperception Engine. I investigate two critical aspects: the spatial accuracy of grid representations and their computational scalability.

6.1.1 Research Questions

As stated in the introduction, the main research question of the thesis is:

- *R.M*: How can Kant's philosophy of cognition be useful for computer vision?

In order to answer this question, an important sub question is:

- *R.2*: How can we test the Apperception Engine on computer vision tasks?

As we saw in 5, the Apperception Engine is specifically meant to make sense of a sensory sequence. As such, the Apperception Engine is a sequence model which explains temporal object dynamics. At the same time, [82] [60] test object dynamic modeling by asking the question whether the models can accurately predict the next n states of a video. As such, my operational research question becomes:

- *R.O*: Can the apperception engine accurately predict the next n state of object relations in a video?

Specifically, this means whether the Apperception Engine can successfully predict if objects in the next unobserved state are left or right from each other.

In order to answer this question, it is necessary to understand what the best data representation is for the Apperception Engine. This necessity comes from the fact that the Apperception Engine uses logic programming, which can be very computationally expensive and hard to apply to noisy data like videos. As such, a sub question necessary to answer the operational research question and the main question of this chapter, is:

- *R.O.1*: Is it feasible to use a Sokoban-style grid representation for CLEVRER videos, when they are provided as input to the Apperception Engine?

[1] Note that I *did* actually tried running the data on the Apperception Engine, but that it would never yield a solution.

6.1.2 Main Contribution

As such, the main contribution of this chapter will be that it shows that a Sokoban-inspired grid representation is not a good way to represent CLEVRER videos for the Apperception Engine. Although the grid representation works for Evans' simple data set, does not work well for for CLEVRER videos due to scaling limitations of the Apperception Engine. The main reasons for this are:

1. **Loss of spatial information:** In the conversion from CLEVRER videos to 2D grid we lose a lot of spatial information. First of all, we lose the z-axis. Secondly, the representation of the grids needs to large enough in order to correctly represent the CLEVRER object dynamics, however they also need to be small enough too handle for the Apperception Engine. Furthermore, it is hard to represent the top view correctly. Objects are often flipped alongside axes, without an immediate fix available.

2. **Line count analysis**: The line count analysis shows a quadratic increase in the number of lines needed to represent the background knowledge of the data: the grid. This means that we can not easily increase the grid size beyond Evans'. Evans uses grids of 4x4 and this already takes the Apperception Engine approximately 4 hours to solve. As such, increasing the grid size does not seem a viable option.

The rest of this chapter will treat both of these studies in turn, treating the used methodology for the analyses, their results and a discussion of the results.

6.2 Study: CLEVRER Transformation

6.2.1 CLEVRER Dataset

CLEVRER (Collision Events for Video Representation and Reasoning) [85] is a diagnostic video dataset designed to evaluate the ability of computational models to perform temporal and causal reasoning. Unlike other video reasoning benchmarks that focus on complex visual pattern recognition, CLEVRER emphasizes understanding the temporal and causal structures behind simple object interactions [85]. As such, it is widely used as an evaluation method for neuro symbolic vision models [69].

The dataset consists of 20,000 synthetic videos (10,000 training, 5,000 validation, 5,000 test) showing objects colliding on a flat surface, along with over 300,000 questions and answers. Each video is 5 seconds long and features objects with different shapes, materials, and colors interacting through physics-based collisions. The videos have a resolution of 480 x 320 pixels. See figure 6.1 for the distribution of videos dependent on the number of objects present in the video.

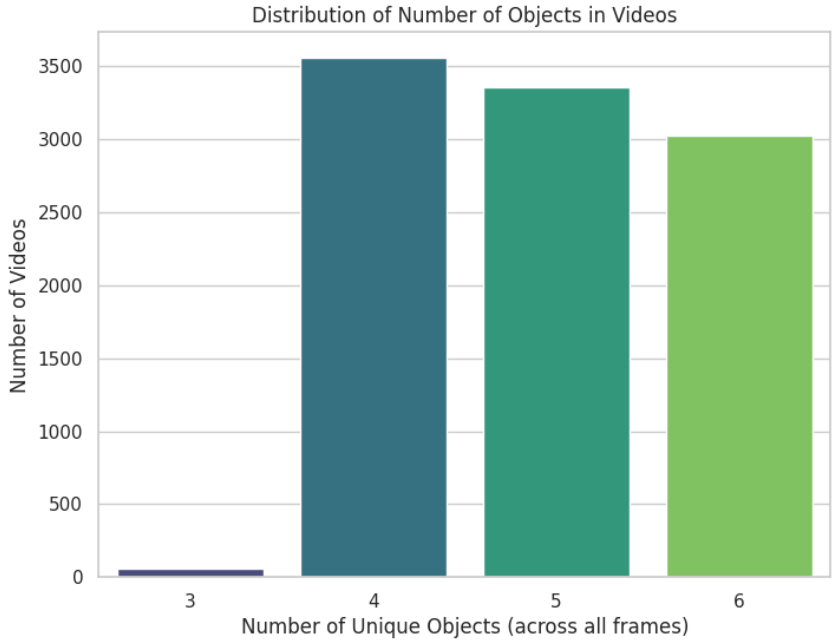


Figure 6.1: Distribution of videos in the CLEVRER dataset based on the number of objects in the videos.

CLEVRER aims to tests four types of reasoning: Descriptive: Understanding what happens in the video (e.g., "what color is the object?") Explanatory: Understanding cause and effect (e.g., "what's responsible for X event?") Predictive: Anticipating future events (e.g., "what will happen next?") Counterfactual: Reasoning about hypothetical scenarios (e.g., "what if X object wasn't there?") Each question comes with a functional program representing its underlying logic, and the dataset includes ground-truth motion traces and event histories. The controlled environment allows for systematic evaluation of AI models' reasoning capabilities while minimizing biases in visual recognition and language understanding.

When testing the visual causal reasoning capabilities of a model, it is more desirable to use a dataset like CLEVRER, which contains videos of *in vitro* object interactions, as opposed to a dataset like SUTD-TrafficQA [83], which contains real world videos of object interactions. In testing neuro symbolic based visual reasoning, faulty reasoning can occur because of two main reasons: 1. it can be the case that relevant objects are not detected properly, 2. assuming that all relevant objects are properly detected, the reason itself fails. [14], [70], [69] It is this second reason which we want to test. The CLEVRER dataset facilitates this better, because its objects only contain simple shapes and colors, whereas the objects in SUTD-TrafficQA are complex and harder to recognize for visual models. Consequently, using CLEVRER to test visual causal reasoning eliminates the possibility that faulty reasoning occurs due to the fact that the visual recognition component of the model is not able to detect objects properly.

6.2.2 Sokoban Task

I followed Richard Evans in using the Sokoban task as the domain. This is a puzzle game where the player controls a man who has to move around a two-dimensional grid world, pushing blocks onto designated target squares. I chose the Sokoban task because the format of this task came closest to the existing video understanding datasets, like CLEVRER.

Evans' Sokoban task consists of two variants. The discrete Sokoban task and the neuro symbolic Sokoban task. The discrete Sokoban task takes as input 1. a sequence of logical grid representations containing cells and objects, and 2. exogenous player actions. The sequence of grids serve as the data to be explained, whereas the exogenous actions help the system in coming to a solution, but do not need to be explained themselves. [22] On the other hand, the neuro symbolic Sokoban task takes as input 20x20 binary pixel images, one exogenous player action per time step, and uses held-out data in order to evaluate the binary neural network. See the figures 6.2 and 6.3 below for an example of each task.

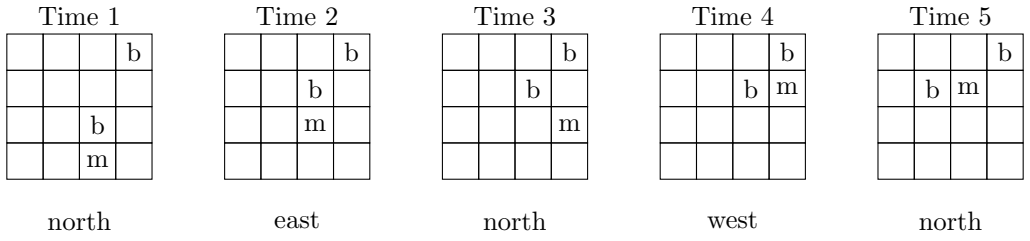


Figure 6.2: A Sokoban sequence showing five time steps with player moves. Each grid is 4x4 and shows the positions of the man (m) and blocks (b), with corresponding actions below. This particular sequence is based on Richard Evans' code.

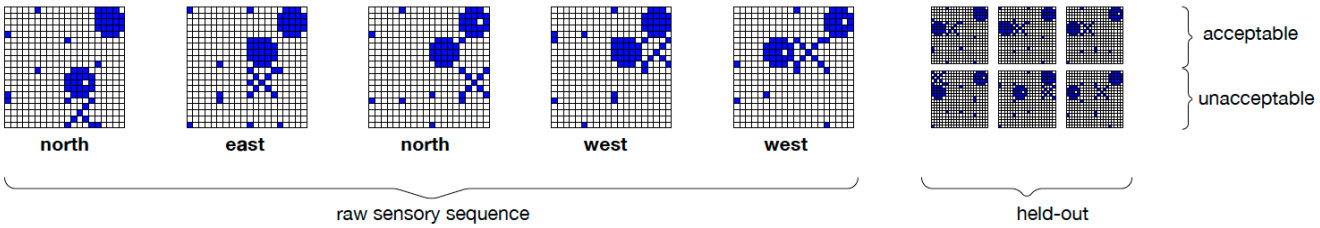


Figure 6.3: A Sokoban sequence with raw sensory input and held-out test cases. Figure reproduced from [22].

6.2.3 Converting CLEVRER to Grid Format

The conversion of CLEVRER videos into Sokoban grids proved a challenging exercise. On the hand, the grids need to be small enough in order for the Apperception Engine to find a solution in viable time (for more details, see section 6.3). On the other, if the grids are too small object dynamics can not be correctly represented. This section will elucidate this discussing a possible conversion implementation.

The conversion of CLEVRER videos into a grid format involved several key steps to transform the continuous 3D space of the videos into a discrete 10x10 grid representation suitable for Sokoban-style environments. For technical details such as Python version and packages used for grid conversion, see appendix B.

1 Keyframe Selection: First, I implemented a frame sampling system to extract key moments from the videos. This is a widely used technique in solving computer vision problems for videos [2]. The sampling function takes parameters for the total number of frames desired and ensures even distribution between a start and end frame. I created four different sampling densities (5, 10, 15, and 20 keyframes) to experiment with various temporal resolutions. See figure A.2, for an example with fifteen key frames.

2. Object Position Extraction: Next, I developed a system to extract object positions and properties from the CLEVRER annotation files. This involved loading the JSON annotation data, extracting the object properties (color, shape), collecting the object locations for each keyframe, transforming coordinates into a standardized format.

3. Coordinate System Transformation: The most crucial step was converting the CLEVRER coordinate system (which uses a 3D space with coordinates roughly between -6 and 6) into a 10x10 grid.

Importantly, I wanted a top view of the objects (similar to Evans' Sokoban grid). As such, I dropped the z coordinate. I implemented a conversion function that:

- Takes an object's (x, y) coordinates in continuous space
- Normalizes them based on the coordinate bounds:

$$\begin{aligned} x_{normalized} &= \frac{x - x_{min}}{x_{max} - x_{min}} \\ y_{normalized} &= \frac{y - y_{min}}{y_{max} - y_{min}} \end{aligned} \tag{6.1}$$

- Maps them to discrete grid cells:

$$\begin{aligned} col &= \lfloor x_{normalized} \cdot width \rfloor \\ row &= \lfloor y_{normalized} \cdot height \rfloor \end{aligned} \tag{6.2}$$

- Ensures all positions fall within grid boundaries:

$$\begin{aligned} col &= \max(0, \min(col, width - 1)) \\ row &= \max(0, \min(row, height - 1)) \end{aligned} \tag{6.3}$$

where:

- (x, y) are the input coordinates in continuous space
- (x_{min}, x_{max}) and (y_{min}, y_{max}) define the coordinate bounds
- $width$ and $height$ are the grid dimensions
- $\lfloor \cdot \rfloor$ denotes the floor function
- The max and min operations clamp the values to valid grid indices

Note that the y-coordinate calculation is inverted $(y_{max} - y)$ to match the conventional grid coordinate system where $(0, 0)$ is at the top-left rather than bottom-left. The bounds of the coordinate system were set to ((-6, 6), (-6, 6)) based on the observation of the coordinate ranges in the CLEVRER dataset. Hoping to ensure that all object positions could be properly mapped to the grid.

4. Grid Visualization: To verify the conversion accuracy, I printed the objects on a grid in the style of the Sokoban. It displays object IDs in their corresponding grid cells, shows empty cells with dots and includes object properties (color, shape) for reference. See figure A.3 in the appendix for an example. This example is based on the same video as A.2.

5. Final Output Format: As such, the final output format consisted of a sequence of grid states for each key frame labeled with object metadata (color, shape) associated with each grid position.

6.2.4 Results

As can be seen in A.3, this conversion process does not adequately transform the continuous motion in CLEVRER videos into a discrete grid-based representation. It does not preserve the spatial relationships between objects well in two different ways:

- First of all, the grid does not represent the distance between the objects well. In frame 27, for example, we see that the brown cube is much closer to the blue cylinder than the gray sphere is to the cylinder.
- Secondly, the grid does not represent well the relative position of objects to one another. Object 3 (gray sphere) appears left-below object 1 (blue cylinder), whereas it should appear left-above the blue cylinder. Similarly, object 2 (brown cube) appears directly above the blue cylinder, whereas it should appear left-of, slightly above the blue cylinder.

6.3 Complexity Analysis

This section presents an analysis of the computational complexity associated with the grid-based representations in the Apperception Engine. The analysis shows that grid representations are not a suitable data representation for CLEVRER videos as desired by the Apperception Engine, due to the fact tha we can not easily increase grid size. Grid representations, while suitable for 'simple' tasks like Sokoban, become computationally intractable for more complex scenarios like CLEVRER videos.

6.3.1 Context

To understand why grid representation don't scale well for CLEVRER videos, I analyzed the relationship between grid size and rule complexity. The analysis consists of two parts: line count analysis and cell adjacency analysis. Here, the cell adjacency is a subset of the total line count.

However, in order to properly test the increase in line count and cell adjacency I needed these values for larger grids. This is especially motivated by the fact that the frames of CLEVRER videos consist of 480×320 pixels, so even a 100x100 grid would provide an extremely coarse spatial resolution. As such, I set up a formula which can determine the line count based on the grid size. This allowed me to extrapolate the line count values for larger grids.

6.3.2 Dummy Data

Although Evans, in [22], states that he tests mainly for the neuro symbolic Sokoban task, his code also contains a discrete Sokoban task. For my purposes, the discrete Sokoban task serves as a preliminary test to the end-to-end differentiable neuro symbolic task. This is because it is easier to input the preprocessed data into the Apperception Engine *as is*, then it would be to build an end-to-end differentiable Apperception Engine. As such, I want to analyse the complexity of the discrete grid-representation, rather than the neuro symbolic one. I use Richard Evans' code for the discrete Sokoban task as basis for my complexity analysis [2].

My discrete Sokoban task takes as input a sequence of Sokoban grids, similar to figure 6.2. I wanted to test the data complexity for the simplest object interaction: interaction between two objects. This allows the test to focus more on the grid complexity, and thus on the usefulness of the data representation, rather than on the object interaction itself.

However, all the CLEVRER videos contain three or more objects. As such, I decided to create a dummy data set inspired by CLEVRER with only two objects in the grid sequence. As such, I have made the following adjustments to Richard Evans' discrete grids:

1. I changed the object denotations from "b" (block) and "m" (man) to integers, in order to denote different objects.
2. I removed the exogenous actions. This is because in our case, a Sokoban grid as representation for CLEVRER videos, there is no player taking actions.
3. Reduced the number of objects in the grid to two. This prevents unnecessary complexity from the number of objects at this stage.

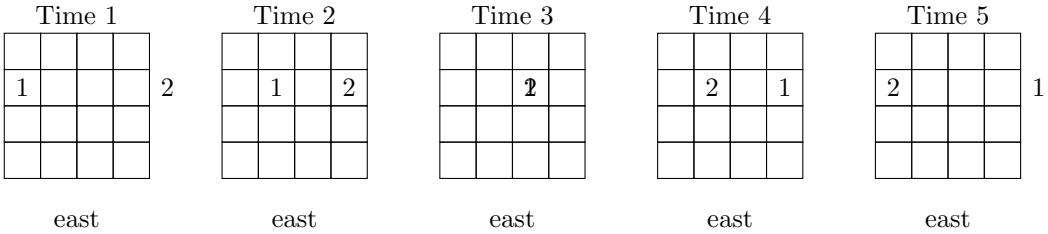


Figure 6.4: Sequence showing five time steps with two numbers moving in opposite directions. Number 1 moves from left to right while number 2 moves from right to left, crossing paths at Time 3.

This provided me with a grid of the form in figure 6.4. The figure only shows five time steps due to spatial limitations of the paper. However, in reality the depicted movement happened across fourteen

[2]Link to Evans' code: https://github.com/RichardEvans/apperception

time steps. This number was chosen because Richard Evans has shown that the Apperception Engine needs enough time steps to come up with a good enough causal theory (see appendix figure A.1) [22]

I manually comprised data for grid sizes ranging from 4×4 to 10×10. Following Evans' Sokoban task, the manually created files which all consisted of multiple sections:

The given sequence: Contains *senses* and *hidden* predicates documenting whether a time step is sensed by the Apperception Engine or not. It records the positions for each of the two objects (obj_x1 and obj_x2) across the total observed time steps. It uses the format $s2(c\_in\_1, obj\_x1, cell, timestep)$ and $s2(c\_in\_2, obj\_x2, cell, timestep)$. Where c_in_1 denotes the relation obj_x1 is in a cell and c_in_2 denotes the relation obj_x2 is in a cell. It spans from time step 1 to 14, with the final positions marked as hidden.

Elements (Cells and their object declarations): This section defines each of the objects (is_object) including: the two moving objects (obj_x1 and obj_x2) and all the grid cells ($obj\_cell\_1\_1$ through $obj\_cell\_4\_4$). Furthermore, it declares all the cells (is_cell) for the 4x4 grid.

Concepts: This section defines the concepts ($is\_concept$): in_1 and in_2. Each object in the grid is defined by its inherency relation in space [22], therefore there must be one in_x relation for any object x.

Time: This section declares the time range. For fourteen time steps, we have: $is\_time(0..14)$.

Logical Constraints: This section contains two uniqueness constraints per concept (at least one, at most one), so four in total. It also contains incompossibility rules preventing objects from being in multiple places. There is one such rule per concept.

Cell Adjacency: This section defines the cell adjacency relations between each cell in the grid. It defines relationships between cells using p_right for horizontal connections and p_below for vertical connections. There is no need for p_left and p_up, I can make those by negating the previous cell adjacency relations.

Walls: This section marks all the cells as not walls using $p\_is\_not\_wall$.

6.3.3 Line Count Analysis

Following Richard Evans' definition of the Sokoban grid, I assume that the grid is a square. Note, however, that this analysis also holds for non-square grids of size $w \times h$, as the total number of rules would still be quadratic in terms of the total area ($w \times h$). For simplicity of notation, I continue with the square grid analysis. I can then derive the components that contribute to the total number of rules in the grid-based representation.

Let us denote the grid size as $n \times n$, where n represents the number of cells per row and column. We can then derive the components that contribute to the total number of rules in the grid-based representation.

Elements: Each cell in the grid has one line to declare the cell as a cell and one line to declare the cell as an object. The number of cells in the grid is simply the product of the row and column counts. As such, the line count of cells can be calculated as follows:

$$\text{Line count for cells} = 2 \times n \times n = 2n^2 \tag{6.4}$$

Cell Adjacency Relations: The number lines representing cell adjacencies can be calculated as follows:

- Horizontal adjacencies: There are n cells in each row, and each cell (except the last one in the row) has an adjacency to the cell to its right. Therefore, the number of horizontal adjacencies is:

$$\text{Horizontal adjacencies} = n \times (n - 1) \tag{6.5}$$

- Vertical adjacencies: Similarly, there are n cells in each column, and each cell (except the last one in the column) has an adjacency to the cell below it. Therefore, the number of vertical adjacencies is:

$$\text{Vertical adjacencies} = n \times (n - 1) \tag{6.6}$$

- Total adjacencies: Summing the horizontal and vertical adjacencies, we get:

$$\begin{aligned} \text{Total adjacencies} &= \text{Horizontal adjacencies} + \text{Vertical adjacencies} \\ &= n \times (n-1) + n \times (n-1) \\ &= 2n \times (n-1) \end{aligned} \tag{6.7}$$

Wall/Non-Wall Declarations: For each cell in the grid, we need to declare whether it is a wall or not. Remembering that the grid is nxn, we get:

$$\text{Wall/Non-Wall Declarations} = n \times n = n^2 \tag{6.8}$$

Base Rules: In the file, the base rules consist of necessary to define the sensory sequence, constraints and compossibility rules. We can read this value from the file.

$$\text{Base Rules} = 124 \tag{6.9}$$

Combining these components, the total number of rules in the grid-based representation can be expressed as:

$$\text{Total Rules} = \text{Base Rules} + 2n^2 + 2n \times (n-1) + n^2 \tag{6.10}$$

Where:

- *Base Rules*: A constant number of rules, such as action definitions, incompossibility rules, and other static elements.
- $2n^2$: Element declarations for the $n \times n$ grid.
- $2n \times (n-1)$: Cell adjacency relations.
- n^2: Wall/Non-Wall declarations for the $n \times n$ grid.

We can prove that this formula is quadratic. Let:

$$\begin{aligned} f(n) \text{ is quadratic } &\iff \exists a, b, c \in \mathbb{R}, a \neq 0 : \\ &f(n) = an^2 + bn + c \end{aligned} \tag{6.11}$$

Proof. We can then show that the total number of rules is quadratic in grid size n:

$$\begin{aligned} \text{Total Rules} &= \text{Base Rules} + \text{Element Rules} + \text{Adjacency Rules} + \text{Wall Rules} \\ &= 124 + 2n^2 + 2n(n-1) + n^2 \\ &= 124 + 2n^2 + 2n^2 - 2n + n^2 \\ &= 124 + 5n^2 - 2n \\ &= 5n^2 - 2n + 124 \end{aligned} \tag{6.12}$$

Since this is formula is of the form $an^2 + bn + c$ with $a = 5 \neq 0$, $b = -2$, and $c = 124$ is constant, the total number of rules is quadratic in n. □

The following section will present the result of filling in the values in the formula for larger grid sizes. Importantly, "Line Count" is equal to "Total Rules". The lines in the data file which contain white spaces or comments (lines starting with "%") are not included in this number.

6.3.4 Results

Based on formula 6.12, I have extrapolated the line count to larger grid sizes. Table 6.1 shows the total line count for the grid sizes ($4x4$) to ($20x20$) and ($197x197$) to ($200x200$) and the figure 6.5 show the line count increase per grid size in a plot. To further dissect this growth, figure 6.6 shows the increase in cell adjacency dependent on the grid size. Cell adjacency complexity means the number of cell adjacency rules in the file.

Grid Size	Grid Area (w^2)	Line Count
4×4	16	191
5×5	25	234
6×6	36	287
7×7	49	350
8×8	64	423
9×9	81	506
10×10	100	599
11×11	121	702
12×12	144	815
13×13	169	938
14×14	196	1,071
15×15	225	1,214
16×16	256	1,367
17×17	289	1,530
18×18	324	1,703
19×19	361	1,886
20×20	400	2,079
⋮	⋮	⋮
197×197	38,809	193,770
198×198	39,204	195,743
199×199	39,601	197,726
200×200	40,000	199,719

Table 6.1: Line count and grid area for different grid sizes

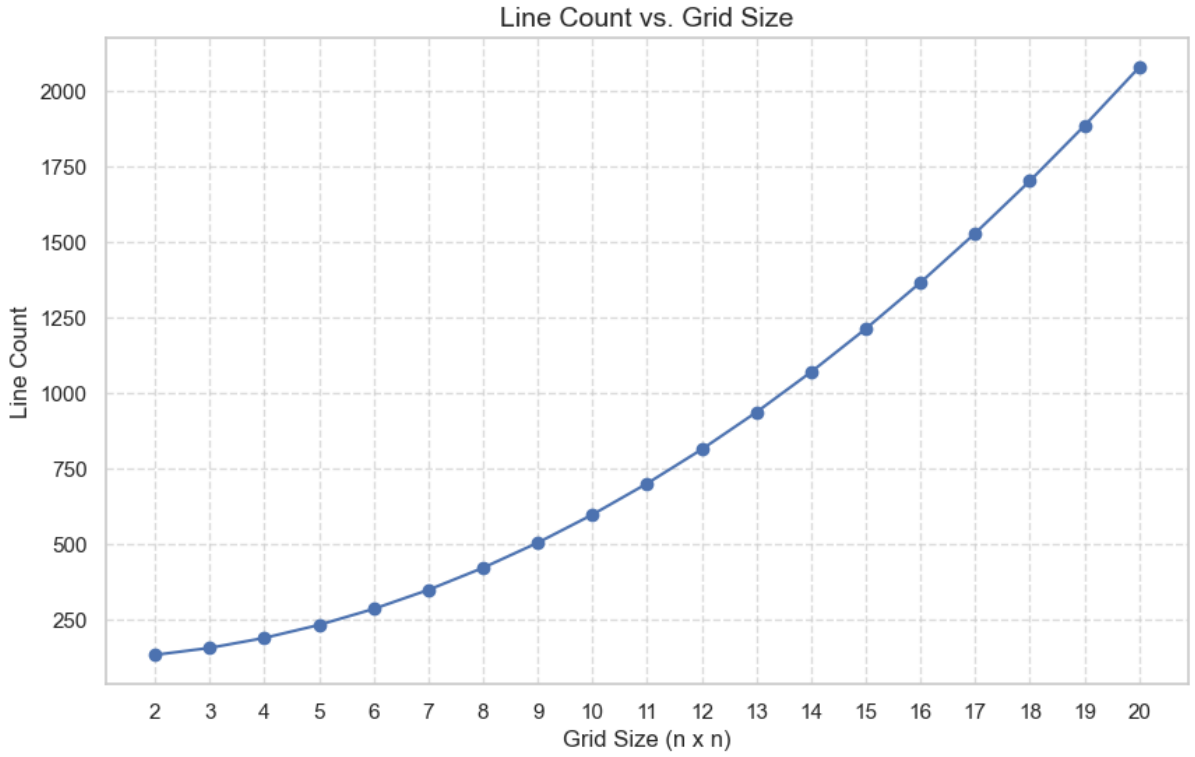


Figure 6.5: Figure shows the quadratic increase in the number of lines needed to define the cell adjacency of the grid (y) for an increase in grid dimension size (x).

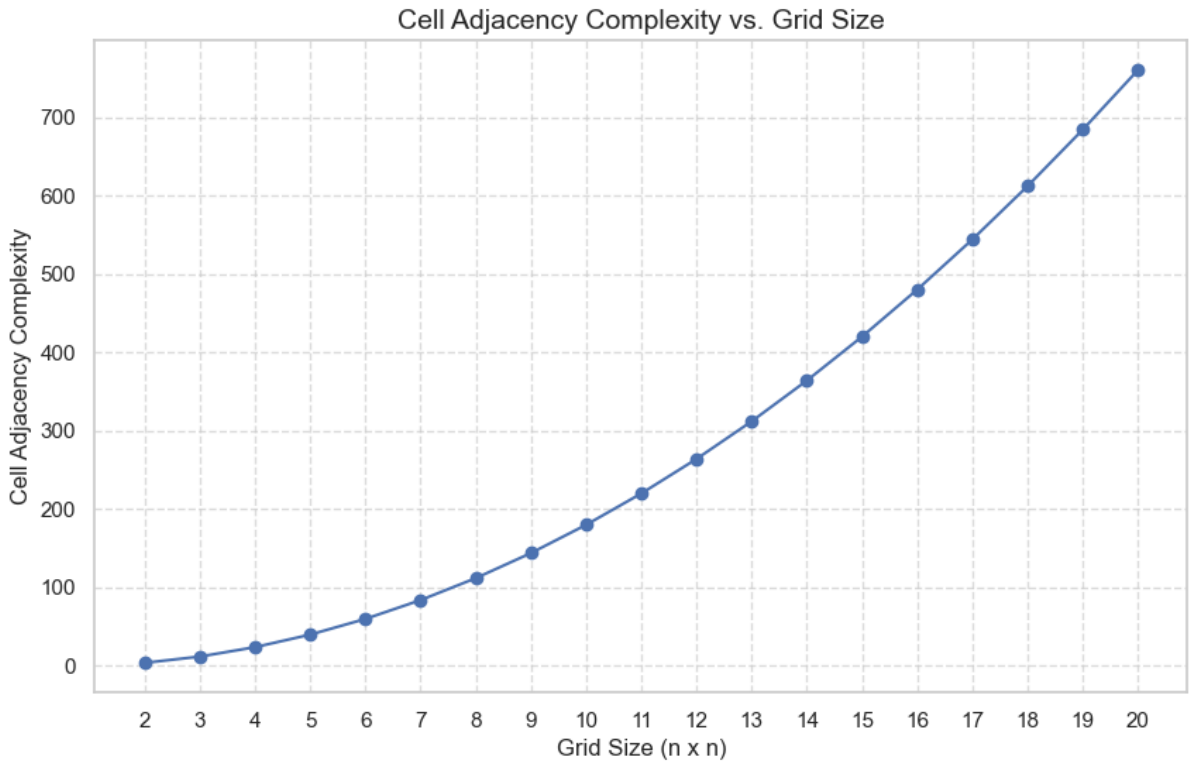


Figure 6.6: Figure shows the exponential increase in the number of lines needed to define the cell adjacency of the grid (y) for an increase in grid dimension size (x).

Based on this, I observe several key findings:

1. **Quadratic Line Count Growth**:
    - The results demonstrate a quadratic increase in line count as the grid size increases. This quadratic growth can primarily be attributed to the increasing number of connections between cells and to the growing complexity of the grid structure.
    - Specifically, for a grid size of N, the line count appears to grow approximately as $O(N^2)$. This growth pattern is inherent to grid structures because both dimensions (width and height) grow linearly with N, and their effects multiply.
2. **Quadratic Cell Adjacency Growth**:
    - The cell adjacency measurements also exhibit quadratic growth. This relationship can be explained by the fact that each new cell requires connections to its neighboring cells. This makes the number of possible cell-to-cell interactions increase as the grid expands.

Chapter 7

Discussion

7.1 Interpretation of Results

7.1.1 Grid Representation Scalability

The analysis of grid-based representations for video understanding shows several limitations and implications for scalability. My investigation demonstrates that the relationship between grid size and rule complexity follows a quadratic growth pattern ($O(N)$), as shown by the line count equation: $f(n) = 5n - 2n + 124$.

This quadratic growth has negative implications for computational scalability. While Evans' implementation processed a 4×4 grid in approximately 4 hours using parallel computing on a standard unix desktop, my complexity analysis shows that scaling to practical video resolutions is computationally intractable. For example, processing a CLEVRER video (480×320 resolution), which still is low resolution compared to real-world videos, would require approximately 613,724 rules. This quadratic relationship between rule count and processing time makes the approach impractical for real-world applications. Add to this the fact that, for all the dummy data used in my analysis, the grids contained only two objects. Adding more objects would add even more complexity in the data file.

This is not only the case for the grids as a representation which is not end-to-end (pre-processed grids are fed into the engine), but also when the grids are fed as input to the neuro symbolic Apperception Engine (likely even more so).

The fundamental challenge lies in the theoretical complexity of the underlying Answer Set Programming (ASP) approach. As shown by Richard Evans [22, p.64], the number of ground instances grows exponentially with the number of variables, while the ASP solving itself is NP-complete for finding solutions and Π_2^p-complete for finding optimal solutions. This theoretical barrier cannot be overcome merely through improved implementation or hardware capabilities.

Furthermore, the grid representation itself introduces significant limitations in representing spatial object relations in videos:

1. **Loss of Continuous Information**: The discretization of continuous spatial coordinates into grid cells results in a loss of precise object positions and movements.

2. **Resolution Dependencies**: The quality of representation becomes directly tied to grid resolution. This creates a trade-off between accuracy and computational feasibility.

3. **Spatial Relationship Distortion**: The grid structure does not capture complex spatial relationships between objects correctly.

These findings suggest that, while grid-based representations initially seem to offer an promising data representation to input spatial object states into the Apperception Engine, and thus to perform visual reasoning, in fact they are not suitable for processing real-world videos at practical resolutions with the Apperception Engine.

7.2 Limitations

7.2.1 One System or Two?

Richard Evans states that "we should use one system, because the higher-order logical rules should inform lower-level sub-symbolic learning" [23, p.93]

Currently, my test for the data complexity does not do this. I use the discrete apperception engine to test the data complexity. Thus, I assume that the input data representation is learned by a neural network which passes it into discrete concepts and then pass it to the Apperception Engine, in order to make sense of the sequence. Although this is not in line with Evans' claim, this serves merely as an initial, isolated test. In order to understand the better data representation for the end-to-end neuro symbolic Apperception Engine, we should first know the data complexity for the discrete Apperception Engine.

7.2.2 CLEVRER Transformation

Another possible limitation is that I based the transformation of the CLEVRER videos on the CLEVRER coordinate system, a 3D space with a (min,max) range of (-6,6). Another option would have been to convert object detection positions of the objects into a grid with manually specified grid bounds. This could particularly be an advantage because in real world computer vision applications object detections are often used. [29] Although it would be good to test this as well, getting correct object detections on the CLEVRER dataset provides a whole new level of complexity. To the best of my knowledge, currently there is no high end object detection model, like for example YOLOV8 [76], which has been trained or fine-tuned on the CLEVRER dataset. As such, this would need to be done first. Furthermore, these detections would be in terms of confidence percentages and object tracking often poses a challenge here. As such, in order to properly convert these object detections into positions on a grid, asides from training the model, the data labels should be corrected for each frame. This would prove a highly cumbersome task.

However, most computer vision models use object I based my transformation on the CLEVRER coordinate grid, why not object detection or something?

7.3 Future Work

7.3.1 Perception Module in ASP

Evans implements the perception module of the Apperception Engine, the faculty of sensibility, as a BNN in ASP [22, p.30]. He does this because of the same claim as mentioned in sub section 7.2.1. According to Evans, making sense of a sensory sequence requires a single end-to-end connected system. The conceptual symbolic realm should inform the decisions at the pre-conceptual sub-symbolic realm. The logical rules should inform the sub-symbolic learning process; the concepts must subsume the intuitions.

I agree with Evans' claim and reasoning that the separate components should be combined in one system end-to-end. However, I do not agree that the implementation of perception module as a BNN in ASP is the best way to do this. Especially not when we try to apply the neuro symbolic Apperception Engine to real- world computer vision problems, such as visual understanding from videos.

Evans implements the perception module as a BNN in ASP, because of the interface problem [22, p.30]. The problem is how to unify the distributed, continuous representation of back propagation with the localist, discrete representation of logic programming. This is a notoriously hard problem [50].

However, for real world computer vision videos, this implementation will likely be highly brittle and unscalable. It can not easily be applied to videos containing multiple objects, colors and high resolution. Secondly, it seems like an easy hack to subvert the problem. The BNN works only when the input data is made so simple, its dimensions are so heavily reduced, that it does not reflect real-world continuous data, like images and videos anymore.

This is a relevant point. The whole reason why Evans implements the Apperception Engine as an unsupervised system is because he states that *"the real world does not come with labels attached to sensory data"* [23, p.39] and that *"Kant's theory is intended to be a general theory of what is involved in achieving experience, so – if it actually works – it should apply to any sensory input"* [23, p.43]. Thus, the fact that the Apperception Engine has implemented perception module as a BNN should be reconsidered and future work should try to connect the discrete part of the Apperception Engine to a deep learning model for the low-level object recognition.

One possible way to implement this could be using the Scallop language [44], [88], [43], [31]. Scallop is a probabilistic logic programming language designed to bridge neural computation and symbolic reasoning. It enables the integration of noisy, probabilistic inputs (from neural networks) into symbolic reasoning processes. As such, it could serve as a wrapper with which we could connect a deep learning scene graph model, for instance, to the Apperception engine in order to learn causal rules over a sequence of CLEVRER scene graphs. For this, we would likely need to add background knowledge for graph reasoning to the Apperception Engine (such as, 'visit all nodes in the graph', 'all nodes are connected to each other', 'nodes do not have a spatial relation to themselves', etc.)

7.3.2 Alternative Data Representation: Scene Graph

The main significance of the line count plot and the cell adjacency count plot is the fact that these lines represent the background knowledge which necessary specifically for the data representation as a grid. The line count plot and cell adjacency plot represent the spatial arrangement of the cells in the grid.

As such, a large part of the computational overhang could be solved by using a different data representation. For example, another widely used data representation for video understanding are scene graphs [89].

Scene graphs provide a structured representation of visual scenes by encoding objects and their relationships as nodes and edges in a graph. Unlike the grid-based approach which discretizes space into fixed cells, scene graphs directly capture spatial and semantic relationships between objects [89].

A scene graph consists of nodes, which represent objects with their attributes, (position, size, color, shape) and edges, which encode relationships between objects (spatial, semantic, temporal). [89]

For example, a scene with "a blue cylinder left of a gray sphere" could be represented as two nodes (cylinder and sphere) connected by an edge labeled "left-of", with attributes encoding their exact positions and properties.

As such, scene graphs can offer multiple benefits compared to a grid-based approach:

- **Scalability**: The data complexity scales with the number of objects and relationships, rather than grid size.
- **Flexibility**: They can easily represent complex relationships between objects.
- **Efficiency**: They form a more compact representation than grids, because they only focus only on relevant objects and relationships.

7.3.3 Apperception PROPPER

POPPER is a recent ILP model that performs well in image classification with minimal examples and generates interpretable rules. [14] PROPPER, an extension of POPPER, introduces probabilistic reasoning into POPPER, making it suitable for neuro-symbolic tasks, for example determining whether the image depicts a car on a bridge. As such, it handles uncertainty in noisy inputs like visual data. [29] Specifically, an advantage of this model over the Apperception Engine is that it is much better documented and updated. An interesting way forward to visual understanding following Kantian principles would be Apperception PROPPER, integrating the Apperception Engines' conditions of unity into PROPPER. As such, one could add a temporal reasoning component and frame inertia, enabling Apperception PROPPER to track objects, infer causal rules between states, and handle dynamic scenes. This approach would be easier than trying to update the Apperception Engine, due to the simple fact that the Apperception is an old, undocumented model written in obscure coding methods.

7.3.4 Learning Background Knowledge from Data

We saw that the Apperception Engine still requires task-specific, handcrafted background knowledge, even though Evans makes it sound like it does not. For example, the Sokoban task needs the grid, and action exclusion rules. This is a common problem for ILP models [13] and makes the potential for their domain-generalizable ability an implausible ideality. At the same time, it is a well known fact that ILP models can invent predicates as part of the solving process. [13] In, [67] Shindo, et al. propose a method to invent high-level relational predicates from visual scenes.

It would be interesting to see whether this approach could be integrated with the Apperception Engine. Similarly to the previous sub section, the best way to implement this would likely be to add the Apperception Engines' unity conditions as domain-specific background knowledge in Shindo's model.

Chapter 8

Conclusion

All in all, this thesis aimed to shed light on the question how Kant's philosophy of cognition can be useful for the field of computer vision. Following in Evans' footprints, we saw that Kant's philosophy can be said to provide a theoretical framework and a practical blueprint for what machine vision models require in order to interpret sequential data as a *representation of an external world composed of objects*. Where traditional logic programming produces a priori relational object representations which are generalizable, yet brittle, and deep learning produces a posteriori distributed object representations which are ungeneralizable, yet susceptible to noise, neuro symbolic AI could potentially provide more coherent object representations. These are synthetic a priori representations which are learned by applying the innate logical faculties to experience and as such, are proven to be generalizable across contexts. These are objects which persist over time, possess attributes that change over time, and adhere to general laws. Crucially, these laws or concepts are *synthetic a priori*—they are not pre-defined but emerge from the system's intrinsic ability to unify perceptions.

This perspective demonstrates the potential advantages of Neuro-symbolic Inductive Logic Programming (ILP) models for computer vision. Kantian principles suggest that, in theory, computer vision models could perceive objects in videos as coherent wholes, rather than as mere aggregates of statistical correlations. This could be achievable when perceptions are unified under domain-general laws, enabling systems to move beyond pattern recognition to a more coherent understanding of dynamic environments in visual data.

However, while this does sound good in theory, there are practical matters that we must consider if we want to implement this architecture in an effective manner. We saw that the CLEVRER videos were hard to convert to grid representations and that these grid representations do not scale well. Thus, highlighting that a different way to represent objects and their relations in videos should be found. Nevertheless, with the right implementation, such as using a more suitable data representation and transferring the domain-general background knowledge to other ILP models we could be well on the way to realizing some form Kantian understanding in computer vision models.

Practically, this theoretical insight could be applied to a wide range of visual understanding tasks, such as visual question answering, where understanding relationships between objects and their transformations over time is critical. The architecture informed by these principles aims to bridge perceptual data and symbolic reasoning, creating models that reason about the world in a more human-like manner.

Appendix A

Supplementary Figures

A.1 Richard Evans Sokoban test

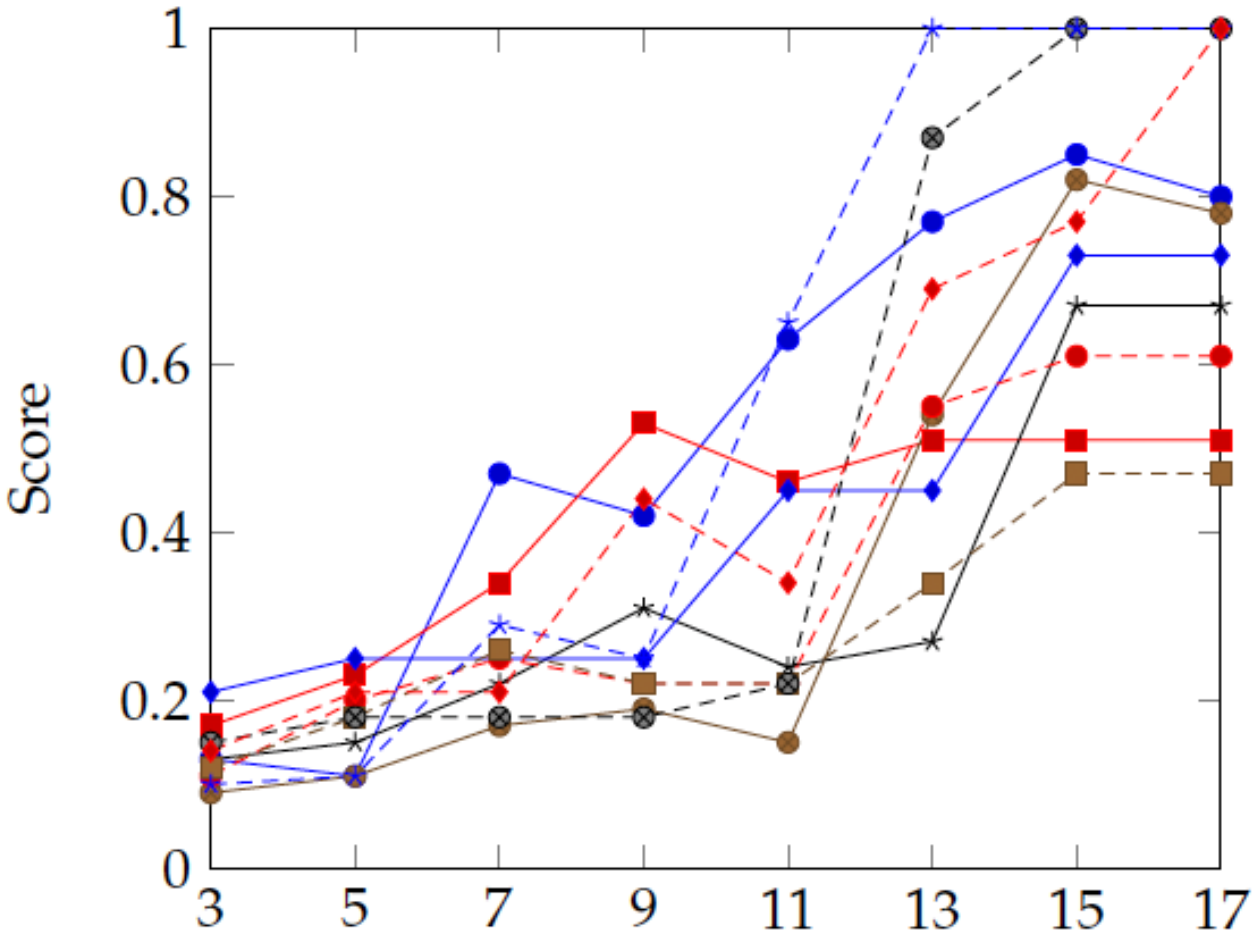
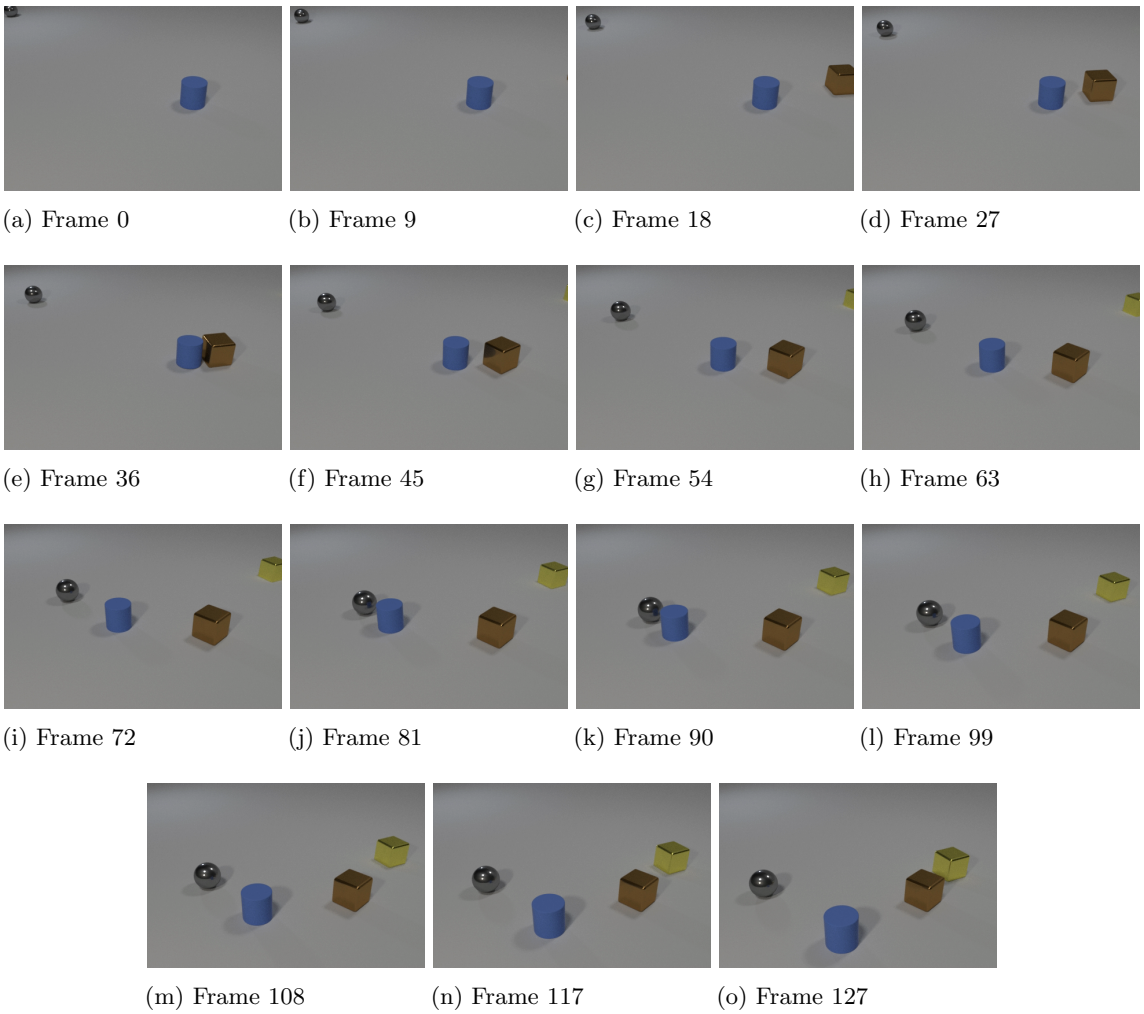


Figure A.1: Figure 5.13: The results for Sokoban on ten trajectories. The horizontal axis records the number of time-steps provided as input. The vertical axis records the mean percentage accuracy over the held-out time-steps. Figure reproduced from [22].

A.2 CLEVRER Frames Visualization

A.2.1 CLEVRER Keyframes



(a) Frame 0 (b) Frame 9 (c) Frame 18 (d) Frame 27

(e) Frame 36 (f) Frame 45 (g) Frame 54 (h) Frame 63

(i) Frame 72 (j) Frame 81 (k) Frame 90 (l) Frame 99

(m) Frame 108 (n) Frame 117 (o) Frame 127

Figure A.2: 15 key frames from CLEVRER video 10000 showing object movements.

A.2.2 CLEVRER Frames Transformation

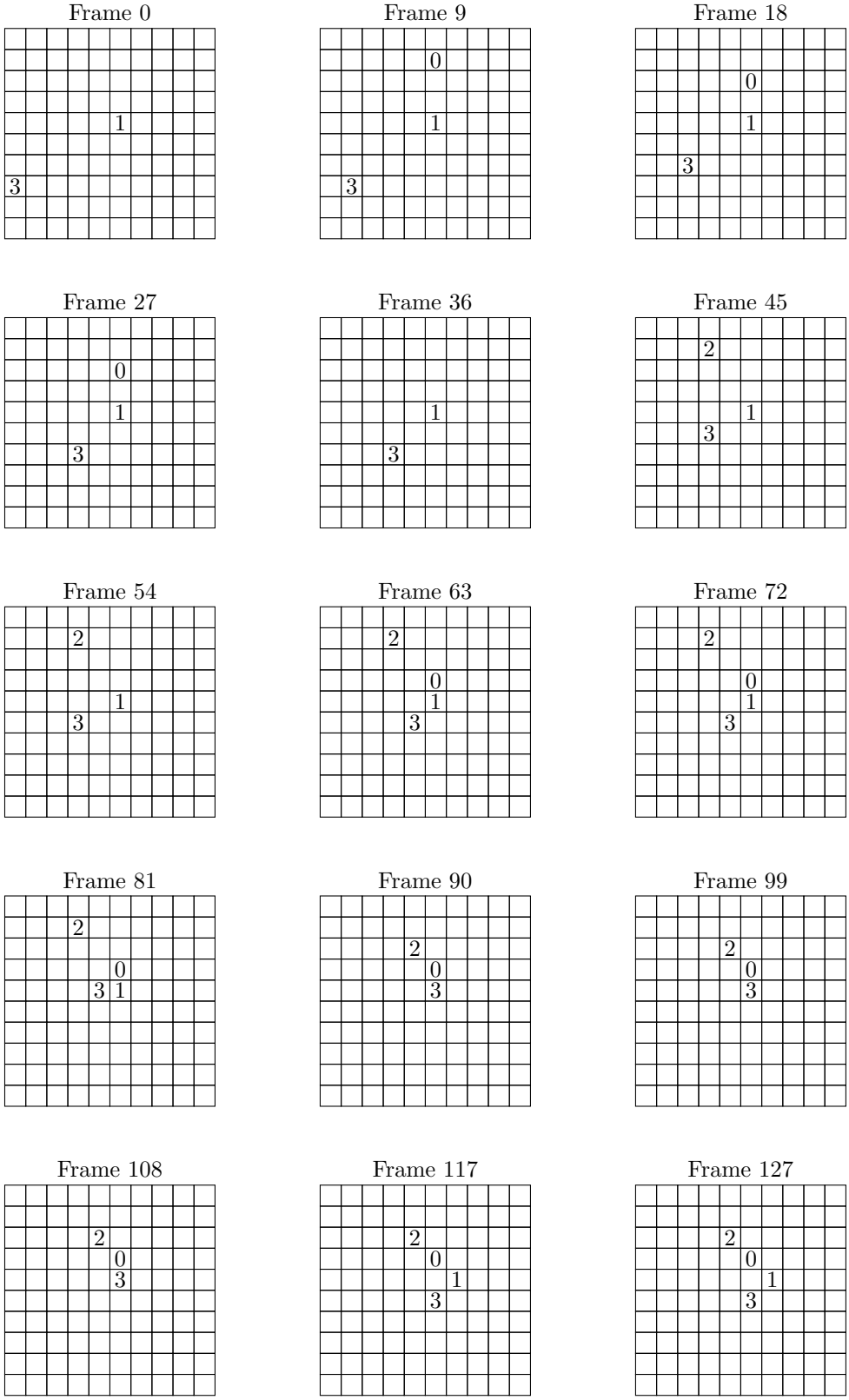


Figure A.3: Grid representation of CLEVRER video 1000. Grid size is 10x10. Objects: 0 (brown cube), 1 (blue cylinder), 2 (yellow cube), 3 (gray sphere). Disappearing numbers indicates that objects are occluded by other objects.

Appendix B

Technical Specifications

Package	Version
Python	3.12.7
aiohttp	3.10.10
beautifulsoup4	4.12.3
black	24.10.0
cloudpickle	3.1.0
contourpy	1.3.0
cryptography	43.0.3
flake8	7.1.1
matplotlib	3.9.2
numpy	2.1.2
pandas	2.2.3
pillow	11.0.0
requests	2.32.3
scikit-learn	1.5.2
scipy	1.14.1
seaborn	0.13.2
spyder	6.0.1

Table B.1: Manually Installed Python Packages and Versions

Bibliography

[1] Jack Achiam, Sam Adler, Sahil Agarwal, Luyu Ahmad, Izzeddin Akkaya, Felipe L. Aleman, David Almeida, Johannes Altenschmidt, Sam Altman, and et al. Anadkat, Sumeet. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Hongliang Bai, Lezi Wang, Gang Qin, Jiwei Zhang, Kun Tao, Xiaofu Chang, and Yuan Dong. Tv program segmentation using multi-modal information fusion. In *Proceedings of the 1st ACM international conference on multimedia retrieval*, pages 1–8, 2011.

[3] David G. T. Barrett, Felix Hill, Adam Santoro, Ari S. Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks, 2018.

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph neural networks. *arXiv preprint arXiv:1806.01261*, 2018.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[6] Blake Montgomery. California suspends driverless car company cruise after pedestrian injury. *The Guardian*, October 2023. Accessed: 2024-11-29.

[7] Margaret A. Boden. Gofai. In Keith Frankish and William M. Ramsey, editors, *The Cambridge Handbook of Artificial Intelligence*, chapter 4. Cambridge University Press, 2014.

[8] Gertjan J. Burghouts and Jan-Mark Geusebroek. Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113(1):48–62, 2009.

[9] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[10] Chih-Hong Cheng, Georg Nührenberg, Chung-Hao Huang, and Harald Ruess. Verification of binarized neural networks via inter-neuron factoring. In *Verified Software. Theories, Tools, and Experiments - 10th International Conference, VSTTE 2018*, volume 11294 of *Lecture Notes in Computer Science*, pages 279–290. Springer, 2018.

[11] Jean-François Cloutier. Finding causal theories quickly enough - building a responsive apperception engine. Technical report, Zenodo, 2023.

[12] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999, 2016.

[13] Andrew Cropper and Sebastijan Dumančić. Inductive logic programming at 30: a new introduction, 2022.

[14] Andrew Cropper and Rolf Morel. Learning programs by learning from failures, 2020.

[15] Andrew Cropper and Stephen H. Muggleton. Metagol system: ILP for learning higher-order logic programs. *Machine Learning*, 94(1):151–174, 2016.

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

[17] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[18] Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic ai: The 3rd wave. *Artificial Intelligence Review*, 2023. Published online: 15 March 2023.

[19] Rene Descartes. *Meditations on First Philosophy.* 1641.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021.

[21] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.

[22] Richard Evans. *Kant's Cognitive Architecture.* Phd thesis, Imperial College London, Department of Computing, March 2020.

[23] Richard Evans. *The Apperception Engine.* De Gruyter, 2022.

[24] Richard Evans, Matko Bošnjak, Lars Buesing, Kevin Ellis, David Pfau, Pushmeet Kohli, and Marek Sergot. Making sense of raw input. *Artificial Intelligence*, 299:103521, 2021.

[25] Richard Evans, José Hernández-Orallo, Johannes Welbl, Pushmeet Kohli, and Marek Sergot. Making sense of sensory input. *Artificial Intelligence*, 293:103438, 2021.

[26] Robert Geirhos, Patrick Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *International Conference on Learning Representations (ICLR)*, 2019.

[27] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. pages 1070–1080, 1988.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[29] Fieke Hillerström and Gertjan Burghouts. Neuro-symbolic reasoning with probabilistic and logical representations. *arXiv preprint arXiv:2408.11367*, 2024.

[30] Geoffrey E. Hinton and James A. Anderson. Parallel models of associative memory. 1984.

[31] Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25134–25145. Curran Associates, Inc., 2021.

[32] Immanuel Kant. *Critique of Pure Reason.* The Cambridge Edition of the Works of Immanuel Kant. Cambridge University Press, New York, NY, 1998.

[33] Kari Paul. San francisco robotaxis grind to a halt, blocking traffic, after major festival. *The Guardian*, August 2023. Accessed: 2024-11-29.

[34] Henry A. Kautz. The third ai summer: Aaai robert s. engelmore memorial lecture. *AI Magazine*, 43(1):105–125, 2022.

[35] Hyeongjoo Kim. *Tracing the Origins of Artificial Intelligence: A Kantian Response to McCarthy's Call for Philosophical Help.* De Gruyter, 2022.

[36] Robert A. Kowalski. The early years of logic programming. *Communications of the ACM*, 31(1):38–43, 1988.

[37] Robert A. Kowalski and Donald Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2(3/4):227–260, 1971.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[39] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

[40] Mark Law, Alessandra Russo, and Krysia Broda. The ILASP system for learning answer set programs. *Theory and Practice of Logic Programming*, 18(3-4):611–627, 2018.

[41] Yann Lecun and Y. Bengio. Convolutional networks for images, speech, and time-series. 01 1995.

[42] Martha Lewis, Nihal V. Nayak, Peilin Yu, Qinan Yu, Jack Merullo, Stephen H. Bach, and Ellie Pavlick. Does clip bind concepts? probing compositionality in large image models, 2023.

[43] Ziyang Li, Jiani Huang, Jason Liu, Felix Zhu, Eric Zhao, William Dodds, Neelay Velingker, Rajeev Alur, and Mayur Naik. Relational programming with foundation models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)*. Association for the Advancement of Artificial Intelligence, 2024.

[44] Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. *Proc. ACM Program. Lang.*, 7(PLDI), June 2023.

[45] Vladimir Lifschitz. What is answer set programming? pages 1594–1597, 2008.

[46] Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning, 2023.

[47] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *CoRR*, abs/2006.15055, 2020.

[48] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[49] Gary Marcus. Deep learning: A critical appraisal, 2018.

[50] Gary Marcus. Deep learning alone isn't getting us to human-like ai. https://www.noemamag.com/deep-learning-alone-isnt-getting-us-to-human-like-ai/, 2023. Accessed: 2024-03-19.

[51] Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of dall-e 2, 2022.

[52] Gary F. Marcus. Rethinking eliminative connectionism. *Cognitive Psychology*, 37(3):243–282, 1998.

[53] Victor W. Marek and Mirosław Truszczyński. Stable models and an alternative logic programming paradigm. *The Logic Programming Paradigm: A 25-Year Perspective*, pages 375–398, 1999.

[54] Peter Markie. Rationalism vs. empiricism, 2021. Accessed: 2024-04-10.

[55] M. Mitchell, A. B. Palmarini, and A. Moskvichev. Comparing humans, gpt-4, and gpt-4v on abstraction and reasoning tasks. *arXiv preprint arXiv:2311.09247*, 2023.

[56] Shanka Subhra Mondal, Jonathan D. Cohen, and Taylor W. Webb. Slot abstractors: Toward scalable abstract visual reasoning. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.

[57] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R. Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving, 2022.

[58] Reza Negarestani. *Intelligence and Spirit*. Urbanomic, Falmouth, UK, 2018.

[59] Namuk Park and Songkuk Kim. How do vision transformers work? *International Conference on Learning Representations*, 2022.

[60] Luis S. Piloto, Ari Weinstein, Peter Battaglia, and Matthew Botvinick. Intuitive physics learning in a deep-learning model inspired by developmental psychology. *Nature Human Behaviour*, 6(9):1257–1267, Sep 2022.

[61] P. Rahmanzadehgervi, L. Bolton, M. R. Taesiri, and A. T. Nguyen. Vision language models are blind. *arXiv preprint arXiv:2407.06581*, 2024.

[62] Aditya Ramesh, Prafulla Dhariwal, Alexander Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[63] Rose Horowitch. Waymo self-driving car kills dog in san francisco incident. *The Guardian*, June 2023. Accessed: 2024-11-29.

[64] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[65] Tobias Schlicht. Minds, brains, and deep learning: The development of cognitive science through the lens of kant's approach to cognition. In Hyeongjoo Kim and Dieter Schönecker, editors, *Kant and Artificial Intelligence*, pages 3–38. Walter de Gruyter, Berlin/Boston, 2022.

[66] Wilfrid Sellars. *In the Space of Reasons: Selected Essays of Wilfrid Sellars*. Harvard University Press, Cambridge, MA, 2007.

[67] Jingyuan Sha, Hikaru Shindo, Kristian Kersting, and Devendra Singh Dhami. Neural-symbolic predicate invention: Learning relational concepts from visual scenes. 2023.

[68] Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. Neuro-symbolic forward reasoning, 2021.

[69] Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. α ilp: thinking visual scenes as differentiable logic programs. *Machine Learning*, 112(5):1465–1497, 2023.

[70] Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. Learning differentiable logic programs for abstract visual reasoning, 2023.

[71] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[72] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.

[73] Paul Smolensky, R. Thomas McCoy, Roland Fernandez, Matthew Goldrick, and Jianfeng Gao. Neurocompositional computing: From the central paradox of cognition to a new generation of ai systems. 2022.

[74] Claire E. Stevenson, Alexandra Pafford, Han L. J. van der Maas, and Melanie Mitchell. Can large language models generalize analogy solving like people can? *arXiv preprint arXiv:2411.02348*, 2024.

[75] M.J. Swain and D.H. Ballard. Indexing via color histograms. In *[1990] Proceedings Third International Conference on Computer Vision*, pages 390–393, 1990.

[76] Ultralytics Team. Yolov8: The latest version of yolo for object detection and segmentation, 2023. Accessed: 2024-11-29.

[77] T. Thrush, R. Jiang, M. Bartolo, A. Singh, A. Williams, D. Kiela, and C. Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. *arXiv preprint arXiv:2212.10537*, 2022.

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[79] Markus Vincze, Sven Wachsmuth, and Gerhard Sagerer. *Perception and computer vision*, page 168–190. Cambridge University Press, 2014.

[80] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[81] Taylor W. Webb, Shanka Subhra Mondal, and Jonathan D. Cohen. Systematic visual reasoning through object-centric relational abstraction, 2023.

[82] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models, 2023.

[83] Li Xu, He Huang, and Jun Liu. Sutd-trafficqa: A question answering benchmark and an efficient network for video reasoning over traffic events, 2021.

[84] Jian-Ru Xue, Jian-Wu Fang, and Pu Zhang. A survey of scene understanding by event reasoning in autonomous driving. *Machine Intelligence Research*, 15(3):249–266, 2018.

[85] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: collision events for video representation and reasoning. *CoRR*, abs/1910.01442, 2019.

[86] Chunyu Yuan and Sos S. Agaian. A comprehensive review of Binary Neural Network. *Artificial Intelligence Review*, 2023. Originally posted on arXiv in Oct 2021.

[87] C. Zhang and S. Wang. Good at captioning, bad at counting: Benchmarking gpt-4v on earth observation data. *arXiv preprint arXiv:2401.17600*, 2024.

[88] Hanlin Zhang, Jiani Huang, Ziyang Li, Mayur Naik, and Eric Xing. Improved logical reasoning of language models via differentiable symbolic programming. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3062–3077, Toronto, Canada, July 2023. Association for Computational Linguistics.

[89] Guangming Zhu, Liang Zhang, Youliang Jiang, Yixuan Dang, Haoran Hou, Peiyi Shen, Mingtao Feng, Xia Zhao, Qiguang Miao, Syed Afaq Ali Shah, and Mohammed Bennamoun. Scene graph generation: A comprehensive survey. *arXiv preprint arXiv:2201.00443*, 2024. Accessed from user's document collection.

[90] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.