NORTHWESTERN UNIVERSITY

Visual Understanding using Analogical Learning over Qualitative Representations

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Kezhen Chen

EVANSTON, ILLINOIS

June 2022

Abstract

Visual understanding is an important area in artificial intelligence. Many researchers have proposed novel deep learning models for this, which have achieved impressive results. However, deep learning models of visual understanding do not model human-like cognition and are still far from human-level ability. Standard deep learning models do not model human-like structural, relational representations and thus lack understandability and data efficiency. On the other hand, symbolic methods are widely used to model human-like cognition. Symbolic qualitative representations can easily express structures and be interpreted. Also, these qualitative representations are adaptable to various input domains and can be generalized to new input domains. However, some researchers argue that these symbolic methods have much lower performance than state-of-the-art models. Thus, I aim to design AI approaches that model human cognition while having competitive results compared with state-of-the-art models.

In this thesis, I focus on improving the use of analogical learning, a symbolic machine learning approach, using novel qualitative representations on multiple visual understanding tasks. Firstly, I explore a visual task, sketched object recognition. To describe the geometric information of objects, I created two novel object-level encoding schemes, geon-based encoding [Biederman, 1987], and part-based encoding. Then, I extend the approach to real images. For real images, I create a hybrid architecture, the Hybrid Primal Sketch Processor (HPSP), which combines deep learning and qualitative representations to generate comprehensive and accurate descriptions of images. The HPSP is inspired by Marr's Primal Sketch [Marr, 1982], and extends it with more semantic information. We apply deep learning for low-level perception and analogical learning over qualitative representations for high-level perception. We aim to gain the advantages of the broad coverage on images from deep learning models and the high data

efficiency, clear understandability, and strong adaptation from analogical generalization. Specifically, we use the HPSP to generate qualitative representations via two types of novel encoding schemes, pair-level encoding and scene-level encoding. We utilize them on visual relationship detection and question answering tasks. Finally, I proposed a novel encoding scheme for temporal data and apply analogical learning over these novel representations to the task of human action recognition. The encoding strategies described in this thesis provide rich information for visual understanding. Experiments on these visual tasks illustrate our claims on analogical learning over qualitative representations.

Acknowledgment

First, I want to thank my Ph.D. advisor and mentor Ken Forbus. In the first one-one meeting with Ken, he told me, “We are working on real science.”. Indeed, in the last six years, Ken has been helpful and inspiring. I enjoy the conversations with him because he always has constructive research ideas and leaves a lot of freedom for students to explore. He showed me what a great researcher could be. I appreciate that I learned a lot from him.

I also appreciate being a Qualitative Reasoning Group (QRG) member. In this lab, I learn that Artificial Intelligence is not just deep learning, but many interesting ideas and methods are also there. We always pursue solving the most challenging tasks and try to model human cognition. This is more challenging and inspiring than achieving higher accuracy on available datasets. I believe that everyone in this lab is enthusiastic about research and enjoys exploring science.

I worked as an intern at Microsoft Research twice during my Ph.D. career. I appreciated all the help from my mentor, Qiuyuan Huang, and my collaborators, Paul Smolensky, Hamid Palangi, and Jianfeng Gao. At Microsoft Research, I learned about neural-symbolic research about combining deep learning models and symbolic approaches and hands-on experiences in implementing large-scale deep learning models. Without their kindly help, I could not publish two conference papers related to neural symbolic networks.

I would like to thank Ian Horswill, Bryan Pardo, and Han Liu for serving on my thesis committee. They shared many constructive suggestions on my thesis and the research. I also took their artificial intelligence classes related to machine learning and deep learning. These classes provided me fundamental with knowledge for my Ph.D. thesis and helped me prepare my research.

I want to thank the Office of Naval Research and Northwestern University for the funding and support of my research over these years.

I would also like to thank my collaborators Balaji Vasan Srinivasan, Niyati Chhaya, Madeline Usher, Irina Rabkina, and Matt McLure. As researchers from Adobe Research, Balaji and Niyati can always provide great suggestions from an industrial aspect. Maddy, Irina, and Matt are QRG members. We had a great time chatting with each other about research. I realize that collaboration is essential in research and life.

I would like to thank all the previous or current QRG members CJ McFate, Joe Blass, Chen Liang, Constantine Nakos, Max Crouse, Sam Hill, Will Hancock, Danilo Ribeiro, Chenghong Lin, Talor Olson, Mukundan Kuthalam, Wangcheng Xu, Roberto Salas Damian, and Walker Demel for interesting conversations in the past years. I was inspired and had a great time with you.

Most of all, this thesis is dedicated to my parents. They give me great love and support to my life and encourage me to pursue what I like. They never give me too much pressure and push me to be great. Instead, we are like my best friends and can chat about anything from life to science. They are the sunshine in my heart, so I never feel dark in my life.

Table of Contents

Contents

1 Introduction

It seems natural that humans can describe what they see. However, modeling human vision and cognition is one of the most challenging questions in Artificial Intelligence. In the past decades, many researchers have made efforts to build models for visual understanding. For example, inspired by neuroscience, Warren McCullough and Walter Pitts, two University of Chicago researchers, first proposed neural networks, which aimed to model connecting and passing signals in the human visual neural system [McCulloch and Pitts, 1943]. Stephen Palmer and David Marr used symbolic representations to model human cognition and perform visual reasoning [Marr, 1982]. However, these models are far away from a human-level vision system.

Recent progress on deep learning models had led to significant improvements. Many researchers have proposed novel deep learning models for various visual tasks. For example, CNN-based models such as ResNet [He, et al., 2016] and Transformer-based models such as Swin-Transformer [Liu, et al., 2021] have achieved impressive performance in image recognition. Faster-RCNN [Ren, et al., 2015] combines Convolutional Neural Network (CNN) backbones with region proposal networks to detect object locations by predicting object labels, bounding boxes, and corresponding confidence scores. Transformer models have also been applied to image and video understanding [Dosovitskiy et al., 2020, Neimark et al., 2021]. Following pretraining schemes, large-scaled pretrained models have been applied in the visual domain [e.g. Radford et al., 2021, Yuan et al., 2021, Chen, et al., 2022]. Pretraining provides better visual representations and performs more robust visual understanding. Researchers also explored generating graph structures, called *scene graphs*, to represent structural information for scenes in images [Dai, et al., 2017; Lu, et al., 2016; Peyre, et al., 2017; Xu, et al., 2017; Yin, et al., 2018; Yu, et al., 2017; Zhuang, et al., 2017; Zhang, et al., 2017; Zhang, et al., 2019a; Zhang, et al., 2019b; Zellers, et al., 2018]. Furthermore, some multi-model learning approaches combine visual models with other domains. For example, visual models are being combined with language models to solve tasks such as image captioning and visual question-answering [e.g. Mao et al., 2014; Vinyals et al., 2015; Devlin et al., 2015; Chen and

Zitnick, 2015; Donahue et al., 2015; Kiros et al., 2014a, b; Gao et al, 2019; Shum et al., 2018]. Indeed, deep learning models have achieved impressive results on multiple tasks. However, these models are performance-oriented and do not model human cognition. Firstly, off-the-shelf deep learning models such as Convolutional Neural Networks or Transformer models lack understandability. Standard deep learning models use dense vectors to represent knowledge or information. These dense vectors are hard to disentangle into understandable representations. Even with state-of-the-art models like Transformers, the learned self-attention maps are hard to comprehend as human-understandable patterns [Samek, Wiegand, and Muller, 2017; Buhmester, Munch and Arens, 2019]. On the other hand, vision psychologists have ample evidence that humans use structured, relational representations to describe the knowledge information in the world [Marr, 1982; Palmer, 1999], which provides strong understandability. Secondly, current deep learning models lack data efficiency. Standard deep learning models need much more data than human learning. For example, [Ciresan et al., 2011] uses data-augmentation to increase the training samples to learn a classification model with many epochs on the MNIST dataset. People, by contrast, often only need several training samples to learn a new concept. This is additional evidence that these systems do not model human-like cognition. Third, deep learning models have poor adaptation. These models are trained in end-to-end fashion on various tasks, and the sub-modules of these models are hard to adapt to other architectures or systems. Finally, these models usually lack reasoning ability. Even the recent large-scaled pretrained language model, GPT3 [Brown et al., 2020], fails in answering arithmetic questions.

Symbolic methods are widely used to model human-like cognition. For example, [Marr, 1982] introduced the Primal Sketch, which uses a series of structural representations to describe the visual components in the image such as edges. These components are grouped into larger semantic components with human-like semantic representations for object recognition and scene understanding. Discrete symbolic representations can easily encode structures and have strong understandability as they can be rendered into English. When using discrete symbolic representations in models, people can directly

explore them to understand what the models learn instead of using other process to analyze the model again. These representations can also be adapted to other systems or architectures without retraining. However, some encoding processes for generating these representations have a low tolerance for noisy data. As a result, many researchers argue that symbolic methods have lower performance on visual tasks than deep learning models.

If we carefully think about the human visual system, AI models exploit advantages from both sides. Can we find a promising research direction that models human cognition and achieves high performance on visual understanding tasks? The answer is yes. This thesis aims to introduce using analogical learning, a machine learning framework on symbolic structures, over qualitative representations generated by novel encoding schemes for visual domains. Our approach achieves competitive performance compared to deep learning models while having strong understandability and data efficiency. In this section, I introduce the importance of qualitative representations and analogy. Then, I describe the roadmap of this thesis. Finally, I will summarize the contributions of the thesis.

1.1 Qualitative Representations

Humans enjoy using structural, relational representations. Thus, they can easily describe what they see. Figure 1 shows a panda and a giraffe. Of course, everyone can describe how they look. For example,

Figure 1: A panda and a giraffe.

pandas are black and white in color, have two round ears, two small eyes, and look like bears. Giraffes have a very long neck and four strong legs. These descriptions implicitly construct understandable qualitative representations to encode the properties of pandas and giraffes.

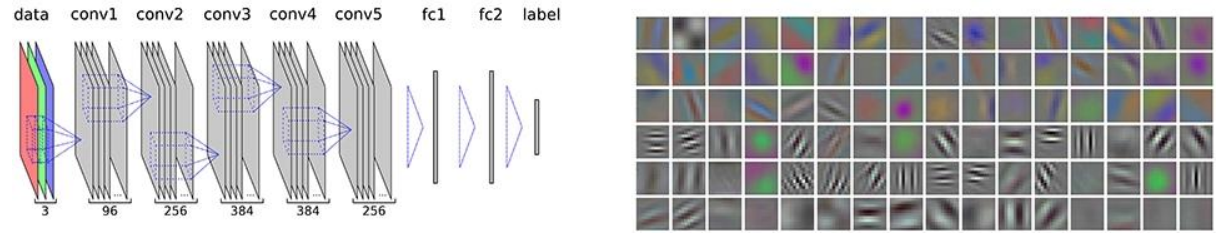


Figure 2: AlexNet and the visualization of the 96 filters on the first convolutional layer.

Qualitative representations are essential to model human cognition in Artificial Intelligence because they provide human-like understandability. However, deep learning models use dense vectors instead of qualitative representations to encode information such as visual features. Thus, they are like black-boxes, and people cannot understand what they learn. For example, Figure 2 shows the architecture of AlexNet, a popular CNN-based deep learning model, and the visualization of 96 filters on the first convolutional layer. These filters seem to detect oriented luminance edges at different frequencies, but people find it hard to describe them in a human-like way.

1.2 Analogy

Cognitive psychology provides evidence that analogy plays an essential role in human vision [e.g., Sagi et al., 2012; Anderson et al., 2018]. It has been demonstrated that computational approaches to analogical learning can match human performance on tasks that involve higher-order cognition [Kandaswamy et al., 2014; Lovett and Forbus, 2013]. Furthermore, analogical generalization has been used on multiple visual tasks such as data-efficient object recognition [Chen et al., 2019] and Raven's Progressive Matrices [Lovett and Forbus, 2017], performing at the 75th percentile on Ravens', which is better than most adult Americans. Section 2.1 introduces the analogy stack consisting of analogical matching, analogical retrieval, and analogical generalization.

1.3 Roadmap

In this thesis, I study visual understanding in multiple domains. Firstly, I focus on Sketched Object Recognition. Sketching is an essential tool for thinking and communicating. Supporting people who sketch, such as artists, designers, planners, or teachers is an important potential application for AI. Sketch recognition has been studied for decades [e.g. Negroponte, 1973; Hammond & Davis, 2005] but remains far from solved. Drawing styles are highly variable across people and adapting to idiosyncratic visual expressions requires data-efficient learning. Understandability also matters, so that users can see why a system got confused about something. On this task, I introduce two novel encoding schemes, geon-based encoding [Biederman, 1987] and part-based encoding, to construct object-level qualitative representations. These qualitative representations include the geometric features of these sketched objects, which can be adapted to any task that requires describing the objects' shape. By applying analogical learning over these qualitative representations, my approach achieves competitive results compared with deep learning models on sketched object recognition and has high data efficiency and strong understandability. I also apply analogical learning in a novel hierarchical approach. This hierarchical analogical learning improves the performance of traditional analogical learning while keeping the same data efficiency. Section 3 introduces the details of sketched object recognition task.

Then, I apply analogical learning over qualitative representations to natural images. My algorithms generate a qualitative representation based on the image and use it for visual understanding. However, real images have more noise than other types of visual structures such as sketched objects. Extracting semantic components requires object recognition on images, which is difficult for symbolic methods. Differentiable neural networks have a high tolerance to process noisy input and broad coverage of various data types. Therefore, I propose a hybrid visual-understanding system, the Hybrid Primal Sketch Processor (HPSP), that combines analogical learning over symbolic qualitative representations with deep learning models. HPSP is inspired by Marr's Primal Sketch scheme and extends it with more semantic information. HPSP applies deep learning for low-level perception and analogical learning over

qualitative representations for high-level cognition. The goal is to take advantage of the broad coverage, flexible adaptation, and high-performance from deep learning models and the high data efficiency, clear understandability, and reasoning ability from analogical generalization. Given an image, deep learning models and computer vision algorithms are used to recognize semantic entities including object categories, object bounding boxes, object boundary masks, and extract visual features such as edges, shapes, and colors. Then, a qualitative representation is constructed based on the recognized semantic entities and extracted properties. In section 4, I introduce the HPSP and use it on two visual understanding tasks, visual relationship detection and visual question answering. In the visual relationship detection task, AI models are required to detect semantic relations between pairs of detected objects, such as a person riding a horse, or a person wearing a dress. For each pair of objects, a qualitative representation should include their categorical information, pose information, and spatial relations between them. Thus, I develop an encoding scheme for object-pairs to include the information needed to detect semantic relationships. On visual question answering task, an AI model should answer questions based on the visual information from images. All questions are designed based on the objects and relations in the images. Therefore, a qualitative representation for this task should include the scene relationships in an image. I propose a scene-level encoding scheme on images to include comprehensive information for question answering in the CLEVR dataset. Experiments on these two tasks show that the HPSP can generate comprehensive qualitative representations for visual understanding tasks. Section 4 presents the architecture of HPSP, the encoding schemes, and implementation details.

Videos include temporal information in visual understanding. Thus, encoding temporal information between visual scenes is essential for visual understanding. When humans describe a sequence of video frames, they prefer to segment the video into several pieces and encode each piece as an event. For example, a cooking video might have two high-level events, food preparation and cooking. I developed an effective encoding scheme for temporal inputs based on this idea. A sequence of video frames or temporal inputs are segmented into pieces with uniform changes of qualitative relations or

spatial calculi. Thus, to represent the information in a temporal sequence, the system just needs to describe each segment's information and temporal changes between segments. Experiments show that analogical learning over such representations facilitates human action recognition. Section 5 describes the encoding scheme for temporal inputs and uses it on human action recognition tasks.

1.4 Contributions

This thesis has four significant contributions. (1) I present geon-based and part-based encoding schemes to represent geometric features of objects, which helps construct qualitative representations for various visual tasks. These representations are human-understandable. I also show how to extend analogical learning over these qualitative representations for the sketch understanding tasks. This approach provides understandability and data efficiency. (2) I propose a novel way to use analogical learning hierarchically to improve the computational efficiency and performance of the system. (3) I describe the HPSP architecture. To my knowledge, this is the first architecture that takes advantage of state-of-the-art deep learning models for low-level perception and analogical learning over qualitative representations to model human-like visual cognition. By combining these two types of AI techniques, the HPSP takes advantage of both approaches. (4) I introduce a sequential encoding scheme to represent continuous human actions. Analogical learning over encoded representations achieved competitive performance compared to machine learning models on human action recognition tasks, and our approach has better understandability.

2 Background

This section introduces the relevant background.

2.1 Analogical Learning and Reasoning

Analogical learning is a family of algorithms that be used for supervised and unsupervised learning tasks. Analogical learning has three important components: analogical matching, analogical retrieval, and analogical generalization. Analogical matching applies knowledge cases to new situations. Analogical

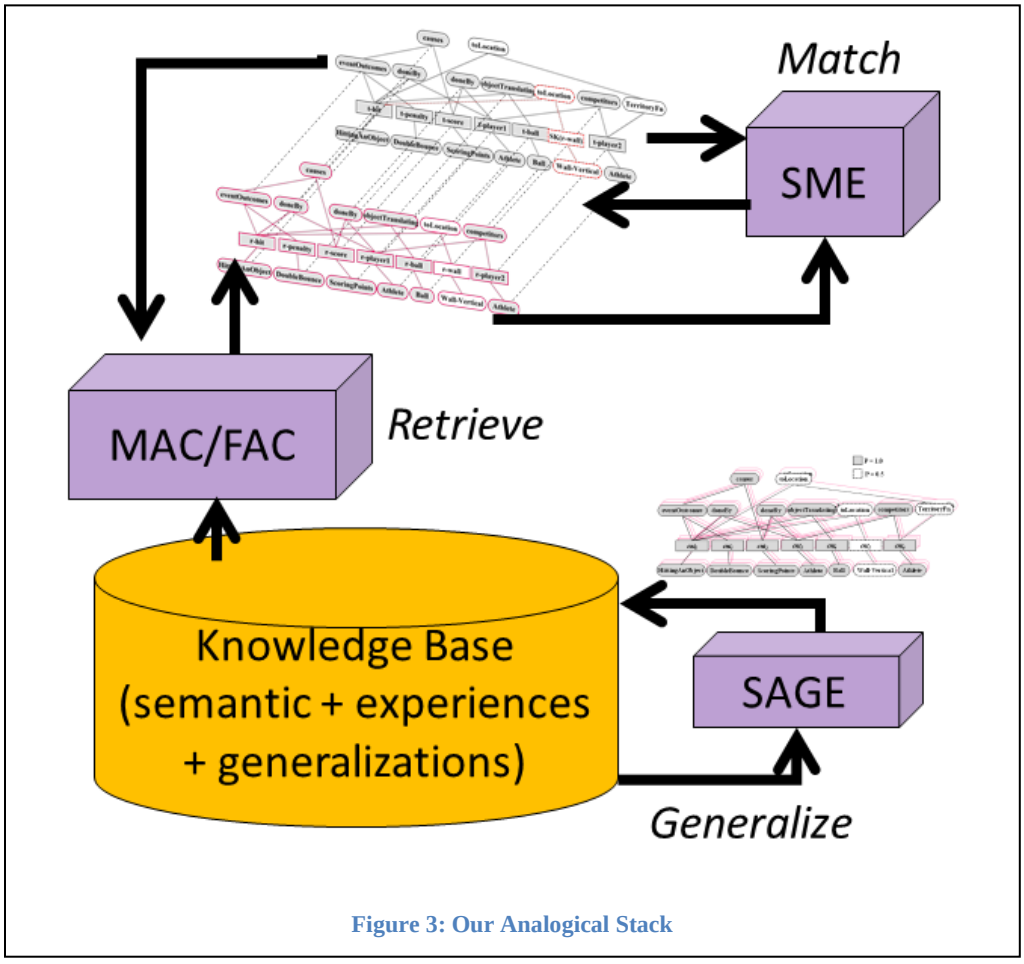


Figure 3: Our Analogical Stack

retrieval retrieves cases from a massive knowledge base. Analogical generalization constructs more transferrable knowledge for future use. In the analogy stack used in this thesis, all the systems use Structure-mapping Engine (SME) [Forbus et al., 2017] for analogical matching, which is based on Structure Mapping Theory [Gentner 1983] and MAC/FAC [Forbus, 1995] for analogical retrieval. The

Sequential Analogical Generalization Engine (SAGE) [Kandaswamy and Forbus, 2014] provides analogical generalization. Figure 3 shows an overview of these three models. Each module is introduced next.

Structure Mapping Theory (SMT) views analogy and similarity as the process of aligning two structured, relational representations. These representations can include object attributes as well as relationships between objects. Attributes can be perceptual, category information, or functional. Similarly, relationships can be perceptual, causal, functional, or evidential. The alignment process constructs a set of correspondences between the entities and statements in the two descriptions being compared. The matching process between two representations, base, and target, is governed by several constraints:

(1) One-to-one correspondence allows each element in the base to map to at most one element in the target and vice-versa (an injective mapping).

(2) Parallel connectivity ensures that if two relations correspond, their arguments will correspond.

(3) Tiered identicality only allows relations to correspond if they have identical predicates, or if aligning close predicates would support a larger relational match.

(4) The systematicity bias establishes a preference for larger, more interconnected aligned structures.

Based on these correspondences, candidate inferences consisting of information that can be projected from one description to another are proposed. Analogy constructs candidate inferences, but their evaluation is left to processes outside the matching process itself.

2.1.1 Structure-Mapping Engine

SME is a computational implementation of the Structure-Mapping Theory (SMT) [Gentner, 1983]. It matches two structured relational descriptions using a local-to-global process. Given two cases of structured, relational representations, called the base and the target, SME computes up to K (usually 3) mappings between them. A mapping includes a set of correspondences that align entities and statements in the base and the target, a similarity score indicating how similar the base and the target are, and candidate inferences, which are projections of unaligned structure from one case to the other, based on the correspondences. Here SME is used both as a similarity metric and to combine cases into generalizations. Figure 4 shows an example of two representations of sketched fishes, with each representation is visualized as a graph structure.[1] The left fish has an upper and lower fin, but the right fish does not have a

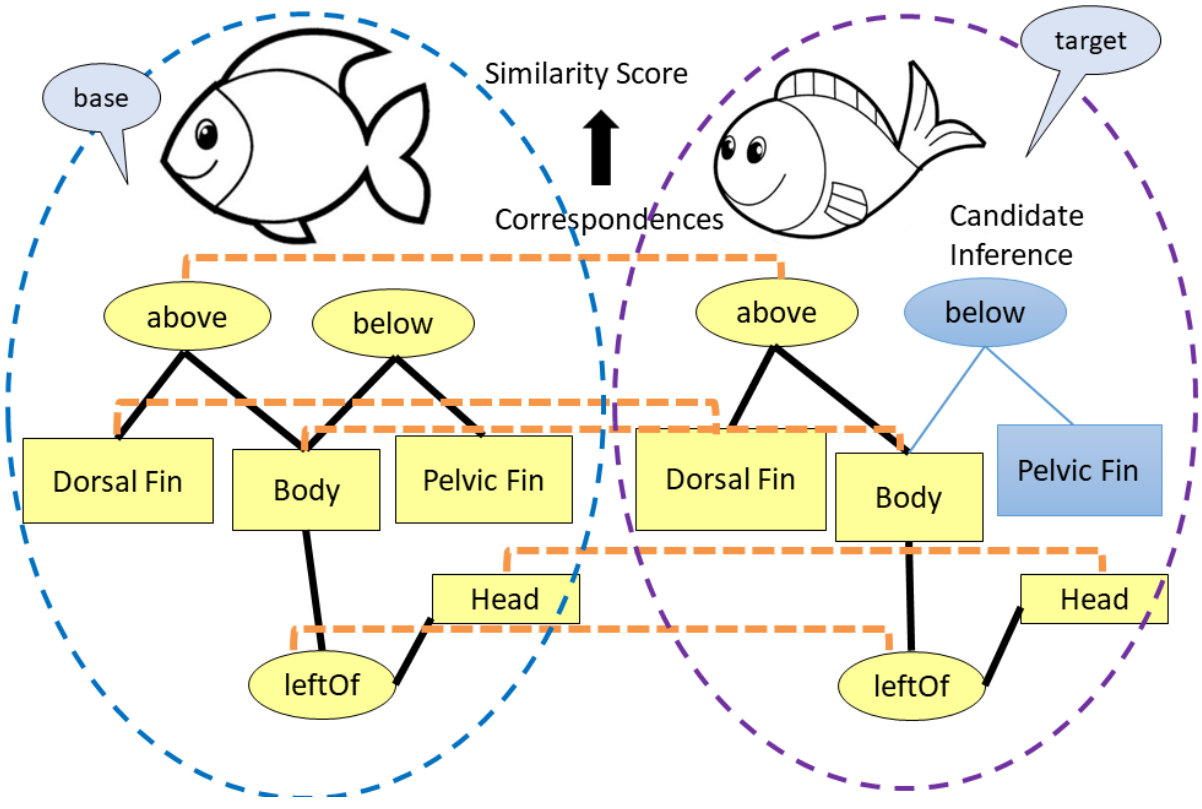


Figure 4: Example of a structure mapping between two sketched fishes.

[1] In Figure 4, I take a subset of the representations of each fish. Each part of the fish in a representation is named with an ID, such as Part-1, instead of a semantic name. I use English words for the parts for clarity.

lower fin. As shown in this figure, SME generates the correspondences between entities and relations. Also, based on the correspondences, SME generates a candidate inference about the right fish to infer that the right fish might have a pelvic fin. A similarity score is generated using the correspondences to evaluate how similar these two fishes are.

2.1.2 MAC/FAC

The MAC/FAC algorithm is a model of analogical retrieval. Figure 5 presents the two stages of this algorithm. Firstly, given a probe case and a case library, it retrieves up to K (generally 3) examples from the case library that is the closest match (i.e., have the highest similarity score) to the probe. Cases in the case library are structured relational representations. When a case is stored, a content vector representation is automatically computed and stored as well. Each dimension in a content vector represents a predicate, and its strength corresponds to the number of occurrences of it in that case. The dot product of two content vectors provides a rough estimate of what SME would compute for a similarity score for the corresponding structured representations. This score is used as a pre-filter. The MAC stage is a map/reduce operation, where dot products for a content vector of the probe are computed in parallel

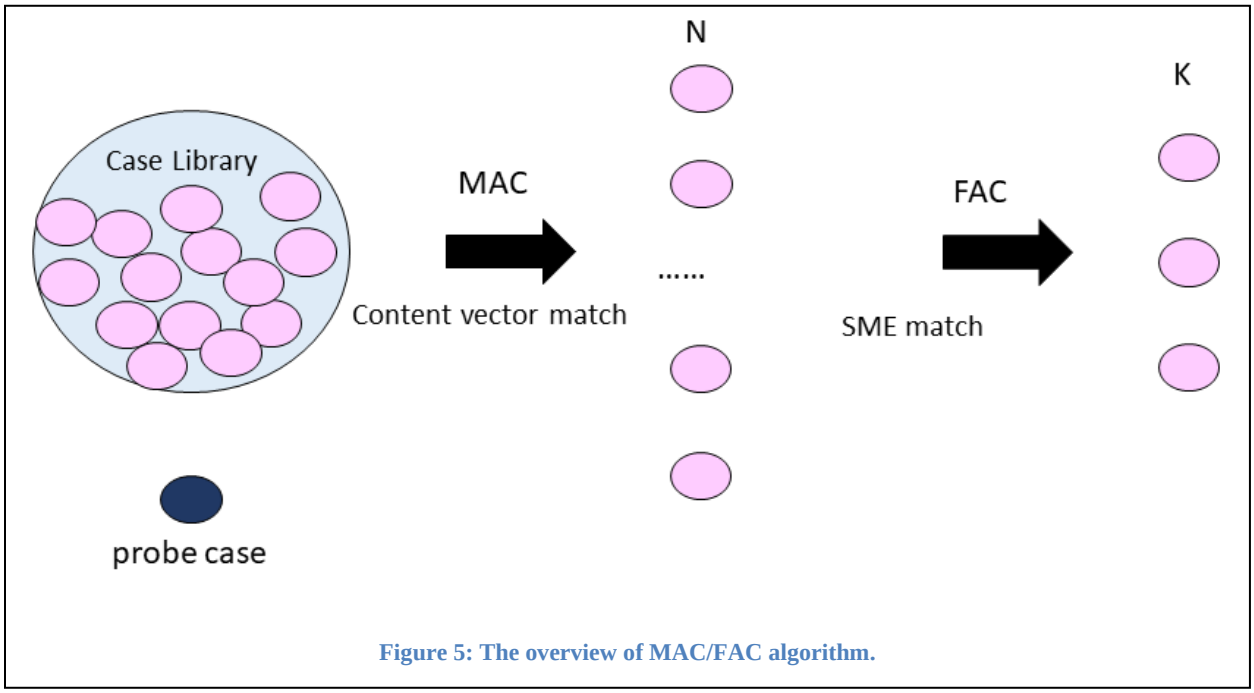


Figure 5: The overview of MAC/FAC algorithm.

with the vectors for all items in the case library, with the top N scoring cases passed on to the FAC stage as output. The FAC stage also is a map/reduce but using SME on the probe and the retrieved cases, keeping the best K cases. The MAC stage provides scalability since sparse vector dot products are cheap. The FAC stage provides the sensitivity to structure that human retrieval demonstrates, probably because structural similarity leads to useful conclusions.

2.1.3 Sequential Analogical Generalization Engine

SAGE is a model of analogical generalization which learns prototypes incrementally from symbolic representations of training examples. Under the supervised learning setting, each concept to be learned by analogy is represented by a *generalization pool*, which potentially holds both generalizations and outlying examples. The generalizations in a concept's generalization pool represent prototypes of the concept. Each generalization pool can have multiple generalizations and outliers. There are two basic operations: learning and classifying an example. Figure 6 shows how these two processes are performed in SAGE.

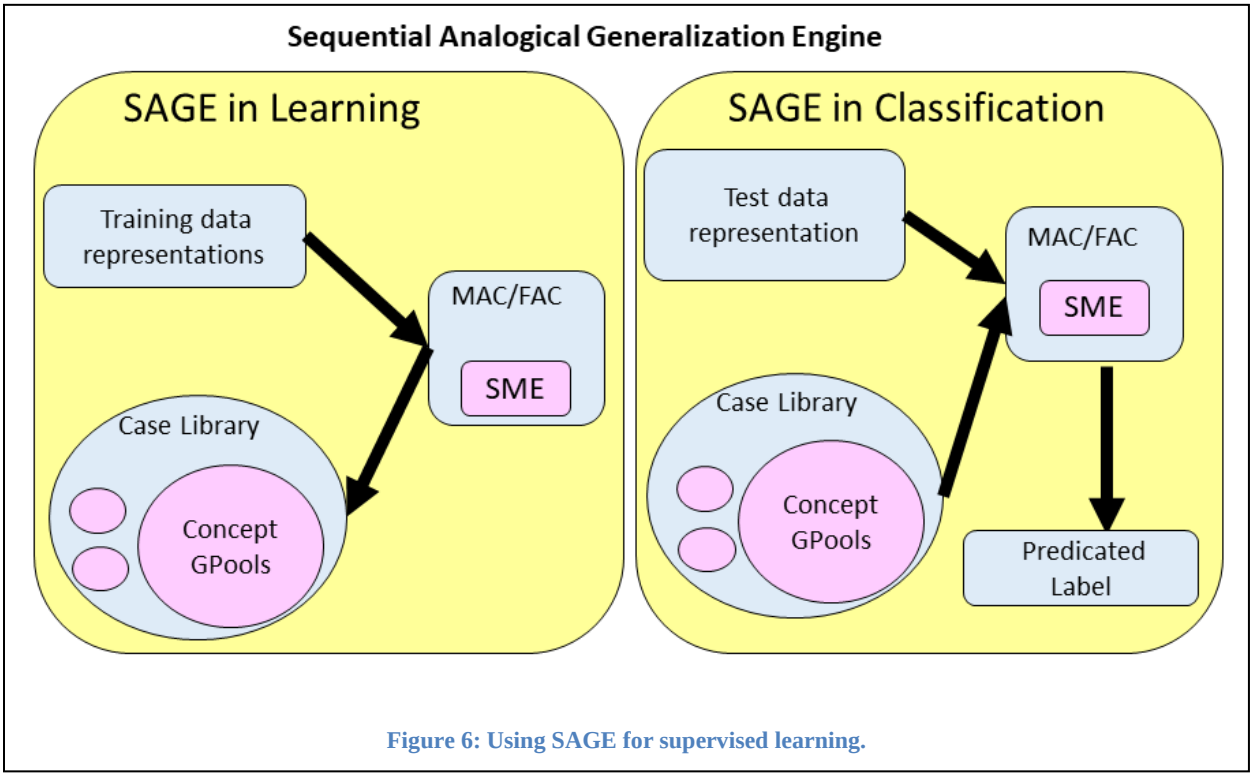


Figure 6: Using SAGE for supervised learning.

Here, I assume that the qualitative representation of a training example is labeled and added to a single SAGE generalization pool for that concept during learning. The example is merged with the most similar case retrieved from the pool via MAC/FAC. Suppose nothing is retrieved, or the similarity score associated with the top retrieval is below the assimilation threshold. In that case, the training example is added to the generalization pool as a new outlier. If the reminding is another example, then a new generalization is formed. This is done by replacing non-identical aligned entities with *skolems*, i.e. a new unique symbol, and taking the union of the statements involved. A probability is calculated for each statement—1.0 if it is aligned in the match, and 0.5 otherwise. A statement's probability reflects the frequency with which the examples assimilated into the generalization contained an expression that mapped to that statement.

If the reminding is a generalization, then that generalization is updated, by adding new statements, perhaps with new skolems for non-identical entity matches, and updating the probabilities for each statement. Statements whose probabilities get too low are eventually deleted, based on another threshold. Thus, over time a generalization pool can have a set of generalizations and outliers. Each generalization can be thought of as a component of a disjunctive model for the concept. In this sense SAGE is like k-means with outliers, except that there is no a priori determination of the number of clusters; the algorithm derives that from the data.

Classification is performed using MAC/FAC, where the testing data is encoded into qualitative representations. Given the representation of a testing sample to be classified, SAGE uses MAC/FAC on the union of generalization pools as the case library. The generalization pool from which the best reminding comes is used as the label for that example.

SageWM is the working-memory version of SAGE. It provides a model of analogical abstraction. SageWM creates new generalizations from a series of examples, by iterative application of SME. When given a series of examples, SageWM stores the first example. When the next example arrives, SageWM

compares it to the first one, using SME. If there is sufficient overlap (that is, if SME's score is above a pre-set assimilation threshold), the common structure is stored as a generalization. If the similarity to the abstraction is below threshold, the example will be stored separately. This process continues as new examples arrive. Thus, if new examples are sufficiently similar to the ongoing generalization, then the generalization will be updated to be somewhat more abstract. Note that SageWM does not have MAC stage. SageWM is used in Section 3.4 as part of a simulation of infant learning.

2.1.4 Related Work

Analogical learning has achieved impressive results on several domains and tasks. For example, analogical learning has been applied to many natural language problems such as word sense disambiguation [Barbella and Forbus, 2013], question answering [Crouse et al., 2018], multimodal dialog systems [Wilson et al., 2019], and link plausibility [Liang and Forbus, 2015]. Analogical learning has been also applied to several visual reasoning tasks. For example, [Kandaswamy et al., 2014] showed that the SME can model learning forced-choice tasks with visual stimuli. [Lovett and Forbus, 2013] showed that analogical learning over relational structures can solve mental rotation and paper folding tasks. [Lovett and Forbus, 2017] describes how analogical learning is used on Raven's Progressive Matrices, which performs better than most adult Americans. In this thesis, I describe how analogical learning can be applied to sketch understanding (Section 3) and sequential encoding (Section 5). Also, a hybrid system that combines analogical learning with deep learning models is introduced in Section 4. The system is adapted to two visual understanding tasks, visual relation detection and visual question answering.

2.2 NextKB

NextKB is an open-license knowledge base. This knowledge base incorporates and integrates the following contents:

- OpenCyc ontology from Cycorp
- The NuLex lexicon, a large-scale open license English lexicon.

- Material from FrameNet and VerbNet, integrated with NuLex and the OpenCyc ontology
- Visual and spatial representations our group developed in creating CogSketch [Forbus, et al., 2011], a software to model human visual cognition
- Representations to support qualitative reasoning, including QP theory and qualitative mechanisms.

My work uses NextKB as the open-domain knowledge base for visual encoding and to store the SAGE generalization pools for analogical learning.

2.3 CogSketch

CogSketch is an open-domain sketch understanding system that automatically computes various spatial relationships based on visual and conceptual information. CogSketch has been used for sketch worksheets in the classroom [Forbus et al., 2017, 2018], cognitive modeling of spatial problem solving [Lovett et al., 2009], tutoring in engineering design [Wetzel 2014], and in another AI research. CogSketch is capable of computing spatial properties (attributes and relations) at multiple representational levels on digital-ink sketches. Here, I first introduce how CogSketch processes visual inputs including both ink data and bitmaps. Then, I describe how CogSketch performs decomposition on visual inputs. Finally, I present the spatial relations that are used in my encoding scheme and how they are computed via CogSketch.

2.3.1 Visual input processing

CogSketch processes digital ink. Thus, it provides an interface for drawing digital ink. Every pen stroke produces a polyline, i.e. a series of points. When utilities in CogSketch are used to import images or bitmaps, it ultimately produces polylines. I use two different ways to convert bitmaps to vector data (digital ink). The first way is to convert bitmaps to files with SVG format, a vector format to store visual inputs. CogSketch has a built-in function to import SVG format as digital ink. Therefore, I use an image tracing program, Potrace [Krenski and Selinger], to convert an image bitmap to SVG format for CogSketch. The second way is using computer vision algorithms to generate digital ink from images. This

mainly focuses on sketched objects in grayscale images. Zhang-Suen's [1984] thinning algorithm is used to find the skeleton of the sketched object, then the ink is traced to automatically generate a set of polylines. These polylines are imported into CogSketch to reconstruct the visual information. To reduce noise and speed up encoding, each original image is resized so that its length is below 300 pixels. Then, the image is blurred and filtered to black and white using a threshold of 70. When we need to convert a real image to sketched inputs, Section 4 introduces a hybrid system that uses deep learning models to perform object detection and process the boundaries they produce accordingly.

2.3.2 CogSketch Representations

The CogSketch basic level representations concerns *glyphs*, which are visual elements which could be objects or scenes. Glyphs can be decomposed into edges and junctions, the most basic units used by CogSketch. To identify edges, ink is separated into segments at its discontinuities and junctions. At the edge level, the length, curvature, orientation, position, and topological relations (i.e., type of junctions, such as T-junction) are computed by CogSketch. Edges can be assembled into *edge-cycles* that form closed shapes, which provide larger units out of which representations of surfaces can be constructed. Edge-cycles have similar properties to edges and many properties of polygons, but, unlike polygons, can

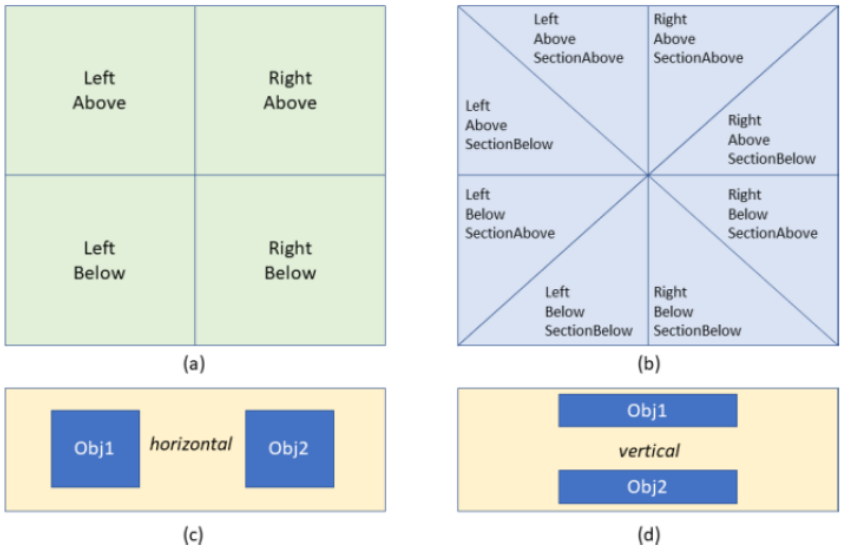


Figure 7: Positional relations between objects used in CogSketch.

also have curved edges. Shared edges between edge-cycles also provide important clues to visual structure. These representations are motivated by psychological studies of human visual processing and spatial cognition, when available, but due to the current state of knowledge in cognitive science, they are somewhat under-constrained.

2.3.3 Selected spatial relations

CogSketch can compute spatial features on visual elements (glyphs, edges, edge-cycles). For example, CogSketch can generate basic polygon properties on glyphs or edge-cycles including convexity, solidity, eccentricity, compactness, circular variance, elliptic variance, shape estimation. On edges, CogSketch can compute basic polyline properties including straightness, direction, and alignments. Between these visual elements, CogSketch can compute rich spatial relationships. Specifically, CogSketch is used to compute three types of relationships: *positional relations*, *topological relations*, and *size relations*. Figure 7 shows the positional relations used in this thesis. For example, I utilized four-way or eight-way relational positions to describe how objects are located with each other based on automatic testing. Also, some applications introduced in the later chapters describe whether two objects align horizontally or vertically. Topological relations, e.g. containing and overlapping, are represented by RCC8 relations [Randell et al.,

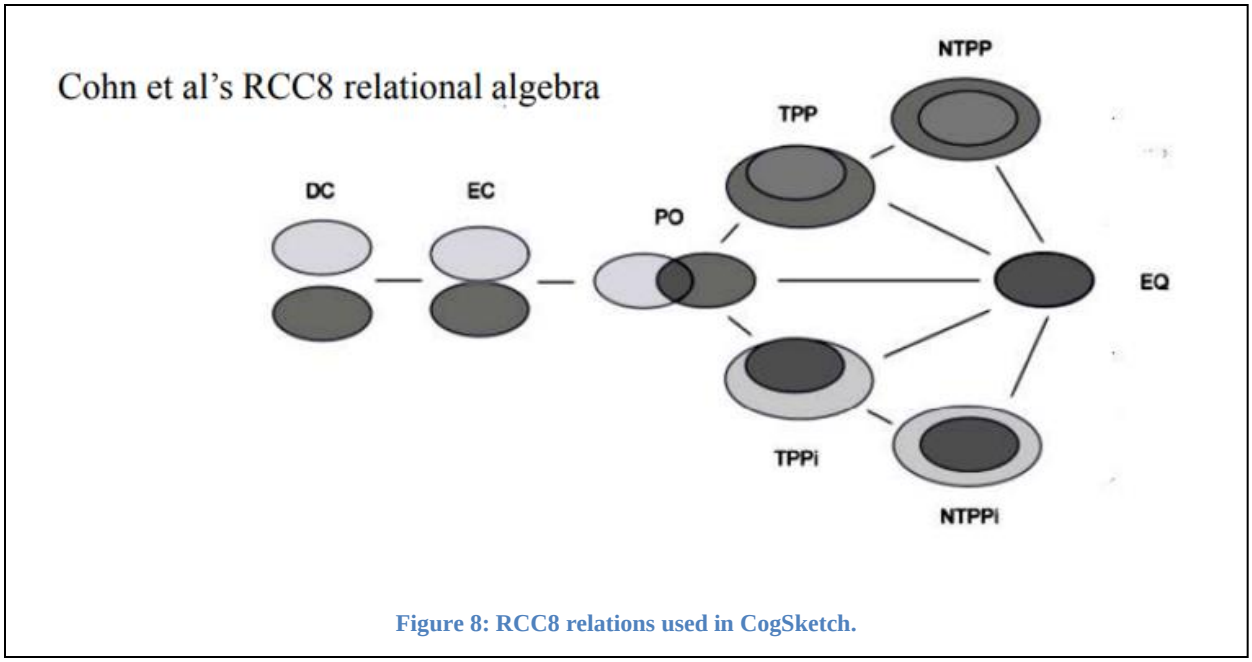


Figure 8: RCC8 relations used in CogSketch.

1992]. Figure 8 shows the eight different spatial relations in the RCC8 scheme. Size relations represent the relative sizes among a set of visual elements. To encode them, CogSketch uses the largest element as the reference object and computes the ratio between the area of each object and the reference object. The object is represented as **largeSize** if the ratio between the object and the reference object is 0.75-1.0, **middleSize** if the ratio is 0.5-0.75, **smallSize** if the ratio is 0.25-0.5, and **tinySize** if the ratio is 0-0.25. Our novel encoding schemes use these spatial relations for representing geometric features of objects and relations between object pairs, which is described in Section 3, 4, 5.

2.4 FIRE Reasoner

The FIRE reasoning engine [Forbus et al., 2010] provides multiple types of reasoning, e.g., reflexive reasoning and deliberative reasoning, that are needed for effective analogical reasoning and learning at scale. FIRE's KB infrastructure is implemented using a persistent object database, AllegroCache[2]. Collections, predicates, entities, and microtheories are all implemented as CLOS objects, with structural facts involving them. Entities are the individuals of concepts. Predicates indicate the relations between entities or concepts. Collections define concepts or groups of entities that share similar properties. Microtheories usually represent events or situations described by a number of facts and are used to represent cases in our analogy stack. FIRE uses a truth maintenance system for reasoning and tracking dependencies. The analogy stack operations and CogSketch visual operations can be accessed via predicates in FIRE, tightly integrating them with other forms of reasoning. In this thesis, FIRE provides the implementation of our analogy stack and helps perform reasoning in question/answering tasks (details in Section 4.3).

2.5 Deep Learning Models for Object Detection

Convolutional neural networks (CNN) are one of the standard types of deep learning models and are widely used for visual processing such as object classification. Object classification aims to recognize the

[2] http://www.franz.com/products/allegrocache/

most probable object category from an image. CNNs contain an input layer, output layer, and many internal layers between input and output layers. Each layer of a CNN is known as a feature map. The feature map of the input layer is a 3D matrix of pixel intensities for different color channels. The feature map of any internal layer is an induced multi-channel image, whose pixel can be viewed as a specific feature. Every neuron is related to a small portion of adjacent neurons from the previous layer, called its receptive field. Between each layer, the convolution operation convolutes a filter matrix (learned weights) with the values of a receptive field of neurons and takes a non-linear function to obtain final responses. Pooling operations, such as max pooling, average pooling, L2-pooling, and local contrast normalization, summarize the responses of a receptive field into one value to produce more robust feature descriptions. Researchers have developed many different CNN architectures with different connection approaches, additional techniques, and deeper layers, such as VGG [Simonyan and Zisserman, 2014], Inception [Szegedy, et al., 2014] and ResNet [He et al., 2015]. These architectures achieve state of the art on most object classification datasets such as MSCOCO [Chen, et al., 2015], or ImageNet [Deng, et al., 2009].

Researchers have built various networks for generic object detection with the off-the-shelf CNN architectures, a more complicated task than object classification. Instead of classifying the object category for a given image, generic object detection aims at locating and classifying multiple existing objects within a given image. Each object is labeled with a boundary (often a bounding box), class label, and confidence score. The frameworks of generic object detection methods can be categorized into two types. The first type uses two-stage detection, generating region proposals and then classifying each proposal into different object categories. Examples include R-CNN [Girshick, et al., 2014], SPP-net [He, et al., 2014], Faster R-CNN [Ren, et al., 2016], FPN [Lin, et al., 2016] and Mask R-CNN[He, et al., 2017]. The other type models object detection as a regression problem, generating categories and locations directly, e.g. YOLO [Redmon, et al., 2015], MultiBox [Liu, et al., 2015], AttentionNet [Yoo, et al., 2015], and DSOD[Shen, et al., 2017]. The Hybrid Primal Sketch presented in Chapter 4 uses Faster-RCNN for object detection. Faster-RCNN combines CNN architectures with an additional Region Proposal Network

(RPN). Faster-RCNN uses a CNN architecture to generate a convolutional feature map on images. RPN slides over the convolutional feature map using a fully connected square window with a fixed size. A low dimensional vector is obtained in each sliding window and fed into two sibling FC layers, box-classification, and box-regression, providing the bounding boxes and category classification results. Besides object bounding boxes, HPSP also detects object masks to estimate the pose information of objects in an image. Specifically, HPSP uses Mask-RCNN [He et al., 2017] for object segmentation. Mask-RCNN has a similar architecture to Faster-RCNN except that it has an additional layer to estimate the contour of the object on each detected object bounding box.

2.6 Recognition by Components Theory

The recognition by components theory was proposed by Irving Biederman in 1987 to explain object recognition. According to this theory, we are able to recognize objects by separating them into *geons* that represent the object's main component parts. In Section 3.1, inspired by this theory, I developed a geon-based encoding scheme to describe the geometric features of objects.

2.7 Datasets

This section introduces all the datasets used in this thesis.

2.7.1 MNIST

The MNIST dataset [LeCun et al. 1998] is a large dataset of handwritten digits, which is widely used to evaluate machine learning algorithms. It was created by "re-mixing" and "re-sizing" the samples from NIST's Special Database 3 and Special Database 1. It consists of 60,000 training images and 10,000

Figure 9: Sample images from MNIST dataset.

testing images of handwritten digits. Each image is a 20x20 pixel bitmap centered on a 28x28 pixel field. Figure 9 shows some data samples in this dataset. We tested our sketched object recognition algorithms on this dataset, as described in Section 3.

2.7.2 The Coloring Book Object Dataset

Our lab created the Coloring Book Objects dataset[3] (hereafter CBO) by collecting images from a collection of open-license coloring books. It contains 10 bitmap examples for each of 19 different

[3] The Coloring Book Objects dataset and CogSketch sketches can be found at http://www.qrg.northwestern.edu/Resources/cbo/index.html.

categories of animals and everyday objects. As a machine learning dataset, the size of this dataset is extremely small. This makes it a very hard task for statistical models, especially for deep learning models. Thus, this dataset is designed to test the data efficiency of AI models and evaluate whether a model can rapidly capture the patterns of these sketched objects with at most 10 examples per category.

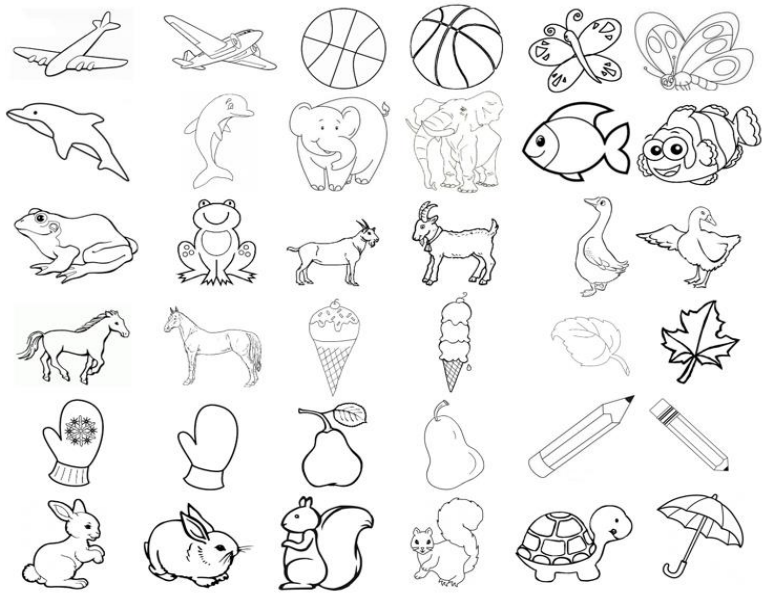


Figure 10: Sample images from CBO dataset.

Each image is a roughly 900x550 pixel field. The images in each category have a very high variety including style (e.g. realistic vs. cartoon) and view (e.g. profile vs. frontal). Figure 10 shows some examples from the CBO dataset. We chose objects and animals depicted in coloring books because they are designed to be recognizable by children, who have had little experience with the world. But as Figure 10 illustrates, they provide significant variability, nonetheless. CBO dataset is used in both Section 3.3 and Section 3.5 for two different experiments.

2.7.3 TU Berlin Dataset

The TU Berlin dataset [Eitz et al., 2012] is a hand-drawn sketched objects dataset. Amazon's Mechanical Turk was used to collect 20,000 sketches of 250 different categories (80 per category) from 1,350 unique participants. The categories include various animals, plants, household objects, vehicles, musical

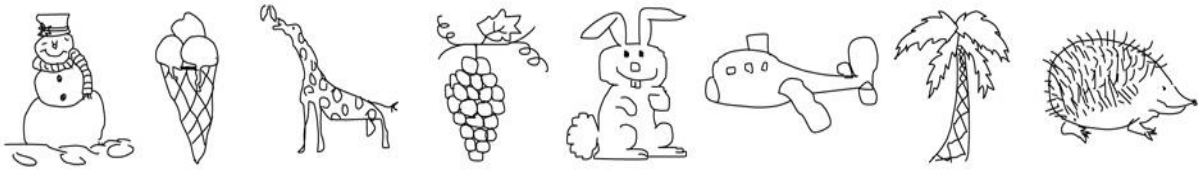


Figure 11: Sample images from TU Berlin dataset.

instruments, toys, body parts, clothing, food, weapons, tech gadgets, celestial bodies, buildings, and non-real-world categories such as angel, dragon, mermaid, and flying saucer. The median drawing time for each instance was 86 seconds and the median number of strokes was 13. Figure 11 shows some sample images from TU Berlin dataset. This dataset is used to evaluate our part-based approach to sketched object recognition.

2.7.4 Visual Relationship Detection Dataset

The visual relationship detection dataset [Lu et al., 2016] contains 5,000 images with 100 object categories and 70 predicates. The dataset contains 37,993 relationships with 6,672 relationship types and 24.25 predicates per object category. Some examples from the dataset area are displayed in Figure 12. The distribution of relationships in this dataset highlights the long tail of infrequent relationships. The training/test split is 4000/1000 images.

Figure 12: Sample images from the Visual relationship detection dataset.

2.7.5 CLEVR Dataset

The CLEVR dataset consists of images of synthetic objects of various shapes, colors, sizes, and materials and question/answer pairs about these images. The CLEVR universe contains three object shapes (cube, sphere, and cylinder) that come in two absolute sizes (small and large), two materials (shiny "metal" and matte "rubber"), and eight colors. Objects are spatially related via four relationships: "left", "right", "behind", and "in front". The questions require multi-step reasoning, and each question corresponds to a ground truth human-written program (Figure 13). There are 70k/15k images and ~700k/~150k questions in the training/validation sets. This dataset is used to evaluate whether our HPSP system can generate rich

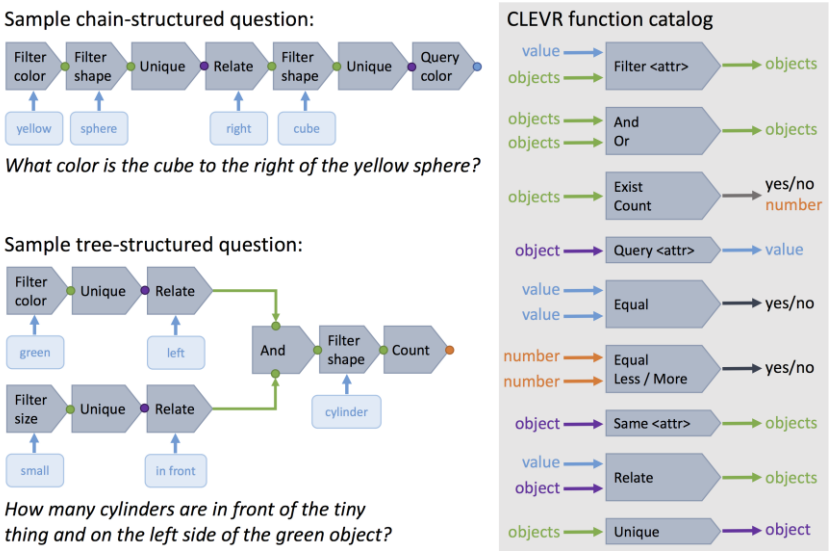
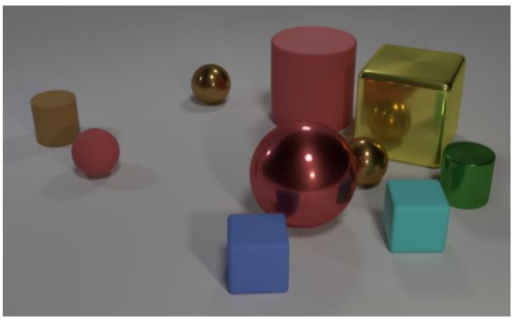


Figure 13: Two types of the CLEVR question programs and all functions used in these programs



Q: Are there an **equal number** of **large things** and **metal spheres**?
Q: **What size** is the **cylinder** **that is left of** the **brown metal** thing **that is left of** the **big sphere**?
Q: There is a **sphere** with the **same size as** the **metal cube**; is it **made of the same material as** the **small red sphere**?
Q: **How many** objects are **either** **small cylinders** or **red** things?

Figure 14: An image sample from the CLEVR dataset and its natural-language questions.

visual representations. In the visual question answering system using HPSP (described in section 4), I only focus on visual encoding from images. Thus, we directly utilize the programs as the questions and generate a reasoning query for each program. Figure 13 shows the types of question programs and all possible CLEVR functions used in programs. Figure 14 shows an image and corresponding natural language questions from the CLEVR dataset.

2.7.6 Human Action Recognition Dataset

Besides visual understanding, I also focus on constructing qualitative representations for temporal data for human action recognition. This task involves recognizing human actions from a set of categories given a sequence of human skeleton frames collected from a Kinect camera. Kinect sensors track 20 body points, so each human skeleton frame has the coordinates of these 20 points. I use three datasets for human action recognition: UTD-MHAD Dataset, Florence 3D Actions Dataset, and UTKinect-Action3D Dataset.

The University of Texas at Dallas (UTD) Multimodal Human Action Dataset (UTD-MHAD) was collected as part of research on human action recognition by Chen et al. (Chen et al. 2015). This dataset

contains eight different subjects (4 females and 4 males) performing twenty-seven actions in a controlled environment and each action repeats four times, collecting data from a Kinect V1 sensor. As our research only focuses on skeleton data, I only use skeleton files from skeleton videos. Furthermore, as Kinect only tracks 20 body points and some motions have very similar movements, such as “Clap” and “Arm curl”, I removed 6 similar actions and removed two actions with large noise in skeleton data. The other nineteen actions are tested in our experiments in Section 5.

The Florence 3D Actions dataset is collected at the University of Florence in 2012 using a Kinect camera. It includes 9 activities: wave, drink from a bottle, answer phone, clap, tight lace, sit down, stand up, read watch, bow. During acquisition, 10 subjects were asked to perform the above actions two or three times. This process resulted in a total of 215 activity samples.

The last human action recognition dataset is UTKinect-Action3D [Xia et al. 2012]. This Kinect dataset has 10 actions performed by 10 different subjects. Each person performs it twice for each action, so there are 200 behaviors in this dataset. The ten actions are: walk, sit down, stand up, pick up, carry, throw, push, pull, wave, and clap hands.

3 Sketched Object Recognition

I start by focusing on visual understanding in the sketch domain and explore how to represent object-level information for the sketched object recognition task. The problem of sketch recognition has been studied for decades, but it is far from solved. Drawing styles are highly variable across people and adapting to idiosyncratic visual expressions requires data-efficient learning. Understandability also matters, so that users can see why a system got confused about something. Analogical learning over qualitative representations provides high data efficiency, strong understandability, and competitive performance. But how can a system effectively represent object-level information in qualitative representations? In this Chapter, I introduce two novel object-level encoding schemes to describe geometric features of sketched objects. I first describe geon-based encoding, and how it is combined with analogical learning on sketched object recognition. Then, based on recent studies on infants' visual systems, I introduce a method to model aspects of infant visual processing and simulate experimental results. Finally, inspired by the simulation model, I develop a novel hierarchical part-based encoding scheme and use analogical learning in a new hierarchical way. By using hierarchical analogical learning and part-based encoding, this approach improves the performance of sketched object recognition using geon-based encoding.

3.1 Introduction

Sketching is an essential tool for thinking and communicating. Supporting people who sketch, such as artists, designers, planners, or teachers is an important potential application for AI. Sketch recognition has been studied for decades [e.g. Negroponte, 1973; Hammond & Davis, 2005] but remains far from solved. Deep learning approaches have recently achieved some impressive results [Li et al., 2013; Seddati et al., 2015; Lin, et al., 2019; Lin et al., 2020; Yu et al., 2016], but have several drawbacks. First, to achieve reasonable performance they require massive amounts of data, and many epochs, for training. A deep neural network with the lowest error rate [Ciresan et al., 2012] on MNIST dataset, for example, was trained on 6 slightly deformed versions of the MNIST dataset and validated using the original—a total of

420,000 training examples. Eleven-year-old children, however, require fewer than 150 examples to learn to identify a novel symbol [Gibson, 1963]. Adults require even fewer examples. Clearly, human learning is far more data-efficient than learning by deep neural networks. In supporting sketching in real-world situations, data-efficient learning is crucial. Sketching style varies widely across people, so adaptation should not require massive data for retraining. For example, when building interactive sketch education software to educate children, the system should be able to rapidly understand what children draw and capture their drawing styles. Moreover, the open-ended nature of creative work puts a high premium on *incremental* data-efficient learning. When a new concept is introduced during a sketching session, systems need to pick up on it quickly, and not require massive amounts of new training data and retraining from scratch, as today's standard deep learning systems require. Thus, learning with a small amount of data is important for sketch understanding. The second problem is that deep learning produces opaque models, making it difficult for users to understand them and their results. When people perform sketch understanding, even with different sketching styles, people can easily identify a sketched object and explain their reasons. Understandability is essential when building interactive sketch systems. Again, when we use sketching to teach children common concepts such as animals or objects, we want to explain what we draw and their properties, so children can match them in the real world.

I claim that analogical learning over qualitative representations has all the advantages that we need. We can construct a qualitative representation for each sketched object using well-designed encoding schemes. These qualitative representations comprehensively describe the geometric features of objects and their relationships and provide strong understandability. Also, analogical learning operates incrementally and has high data efficiency.

Here, I introduce two novel approaches to sketched object recognition, ***G****eon-based* ***A****nalogical* ***L****earning* (GAL) and ***P****art-based* ***H****ierarchical* ***A****nalogical* ***L****earning* (PHAL). Both approaches perform analogical learning over qualitative representations of sketched objects, but they use different encoding

schemes and apply analogical learning in different ways. GAL uses a geon-encoding scheme to generate a representation corresponding to each object. Given a sketched object, GAL segments the objects into uniform pieces by computing contour closures. GAL constructs a qualitative representation including the geometrical features of each segment and the spatial relations between segments. Then, analogical learning is used for supervised categorical learning to recognize objects. Experiments on MNIST and the Coloring Book Objects datasets show that this approach has high data efficiency and achieves competitive results compared to off-the-shelf deep learning models. Then, I propose a computational model to simulate recent studies on infants' visual understanding. Our computational model assumes that infants use an encoding scheme with a hierarchical structure that starts by representing objects using coarse information, such as color and contour shape, then moving to detailed information, such as textures. Inspired by this idea, our second approach, PHAL, uses a hierarchical encoding scheme. PHAL focuses more on the local properties of objects. Given a sketch, PHAL automatically segments the object into parts and constructs multi-level human-like qualitative representations. By performing analogical generalization at multiple levels of part representations hierarchically and using coarse-grained results to guide the reasoning at finer levels, I show that PHAL can also learn explainable models in a data-efficient manner. Importantly, PHAL achieves better performance on the Coloring Book Object Dataset compared with GAL and it is competitive with deep learning models on the TU Berlin dataset.

I begin by introducing the most relevant related work on sketch recognition. Then I present GAL, the simulation experiment of infants' visual encoding, and PHAL, including the encoding schemes, analogical learning processes, and experiments. Finally, this section is concluding with future work.

3.2 Related Work

Sketch recognition has been studied for decades, so making a detailed review is beyond the scope of this thesis. Consequently, I focus only on the most relevant related work. Eitz et al. [2012] created the TU Berlin sketch dataset, containing 80 sketches per category from 250 different categories. They

demonstrated that multiclass support vector machines using a bag-of-features sketch representation could achieve 56% accuracy. Prior work with analogical generalization on this dataset [McLure et al., 2015a] achieved similar levels of performance on a subset of this dataset by using automatically computed qualitative visual relations and by introducing an Ising model to handle textures over edge cycles. GAL and PHAL also use qualitative visual relations but encode segments or parts to provide a stable intermediate representation. Furthermore, PHAL also encodes the texture information by recognizing textures via analogical learning. McLure's Ising model for texture detection only works on connected edge-cycles, thereby missing many textures. By contrast, the new texture detection method in PHAL can detect textures that are connected or non-connected edges or edge-cycles.

Deep learning models have achieved impressive results on sketch recognition [e.g. Li et al., 2013; Seddati et al., 2015; Lin, et al., 2019; Lin et al., 2020; Yu et al., 2016]. They designed large-scale deep learning models and performed data augmentation or pretraining to achieve state-of-the-art performance. For example, Lin et al., [2020] designed Sketch-BERT, which used BERT as the backbone and pretrained it with many sketches. Then, they fine-tuned the model for downstream sketch recognition/retrieval tasks. However, this is the opposite of data-efficient learning, and with small training datasets, deep learning models fail to achieve reasonable performance. Also, as previously mentioned, deep learning models lack understandability. Both GAL and PHAL are highly data-efficient and understandable. This thesis demonstrates that they can achieve competitive results compared with deep learning models but with much less training data.

3.3 Geon-based Analogical Learning

3.3.1 Approach

Sketch understanding can start with digital ink or bitmaps. Here the learning system starts with bitmaps, to provide a closer comparison with vision-based approaches. Consequently, the first step is converting bitmaps into digital ink, and then using CogSketch to construct relational representations. The second

step is analogical learning using qualitative representations as cases. The work in this section was originally published in [Chen and Forbus, 2019]. Figure 15 (a) shows a sketch of a fish from the Coloring Book Object dataset as an example.

The process of converting a bitmap input into a qualitative representation has three stages: (1) bitmap preprocessing to reduce noise and create a sketch for CogSketch, (2) object segmentation to extract the edges and junctions that make up the sketch, and (3) spatial encoding to create geon-based relational representations. All stages are introduced below.

Bitmap Preprocessing

Given a sketched object bitmap, the system needs to construct the skeleton ink of the object in CogSketch. Therefore, I use the second approach described in section 2.3.1 to convert bitmaps to sketches, i.e., tracing the lines to reconstruct them in CogSketch. Firstly, the system applies Zhang-Suen's thinning algorithm [1984] on bitmaps. After thinning, all the edges of the sketched objects are 1-pixel width. This is a necessary step to get clean polylines for reconstruction in CogSketch. Then, the system uses a depth-first search algorithm to trace the edges and generate polylines. The system first stores all black pixels in a graph structure (white pixels are background). Each black pixel is a node and if two black pixels are neighbors with each other, an edge is added between them. Each node is ranked based on

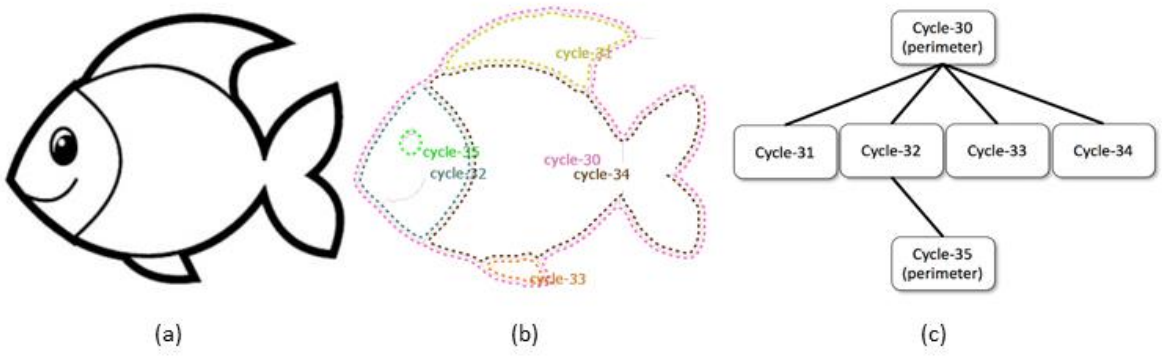


Figure 15: (a) is a sketched fish from Coloring book dataset. (b) is the edge-cycle decomposition of (a) from CogSketch. (c) is the corresponding decomposition tree.

the number of neighbors and the algorithm starts on the node with least neighbors. By using depth-first search, the system keeps searching neighbor nodes until the node does not have any unvisited neighbors. If a node has multiple neighbors, the node that has the smoothest angle with previous node is chosen. Therefore, the system can always get a smooth polyline. All visited pixels construct a polyline. This process is repeated on the graph until all pixels are visited. With these polylines, CogSketch reconstructs the sketched object as a glyph for later encoding.

Object Segmentation

Each digital-ink sketch imported into CogSketch generates a glyph, whose geometric features are described with a qualitative representation. Firstly, the glyph is decomposed into closed edge cycles and edges by CogSketch. Figure 15 (b) shows the edge-cycle decomposition of the sketched fish depicted in Figure 15 (a). Each edge-cycle is a closed contour of the whole glyph or an inside region. For example, in Figure 15 (b), the pink edge-cycle is the contour of the fish and yellow edge-cycle is the upper fin. After decomposition, the edges used in edge-cycles are removed. The edge cycles and edges are stored in a *decomposition tree*. From outside to inside of the object, each edge or closed edge-cycle is stored in a

Algorithm 1: Object Segmentation

Input: An edge-cycle C

Functions:
MedialAxisTransform (EC): compute the medial axis transformer for an edge-cycle EC. This function returns the medial axis set, A.
MA-PTR (P, Dir): return either the left or the right point in a pair of medial axis points on the edge-cycle with respect to a point P on a medial axis.
PreCondition (P1, P2): check whether the pair of points P1, P2 satisfy the conditions described in the last paragraph in Section 3.3.1 Object Segmentation.
CreateLine (P1, P2): create a line between two points P1, P2.

Output: a list of segmentation lines, L

```
1:  A = MedialAxisTransform (C)
2:  L = []
3:  For point Pi in each medial axis Ai in A do
4:      Pi1 = MA-PTR (Pi, Left)
5:      Pi2 = MA-PTR (Pi, Right)
6:      If Concave (Pi1) or Concave (Pi2) and PreCondition (Pi1, Pi2) do
7:          L.append( CreateLine (Pi1, Pi2) )
8:  return L
```

node of the decomposition tree from root to leaves, so the root node contains the contour edge-cycle of the whole sketched object. Figure 15 (c) shows the decomposition tree for the fish. Note that the inner edge-cycles correspond to the eye, fins, body, and head of the fish. This depiction of the representation does not include the edge of the mouth in the decomposition tree for simplicity.

To represent properties of edge-cycles, I draw on Biederman's [1987] recognition-by-components theory. Inspired by this theory, each edge-cycle is segmented into uniform geons, each described by several geometric attributes. Algorithm 1 describes this process. The first step is finding the medial-axis transform by computing the grassfire transform [Blum, 1967] on an edge-cycle (Alg. 1 Line 1). Each medial axis point has at least two closest points on the edge-cycle. CogSketch generates multiple pairs of closest points for each medial axis point. If the medial axis point only has two closest points with respect to the edge-cycle, it only has one pair. Otherwise, the system constructs multiple pairs of closest points. The pairs are then iterated over, to find *closures* of the edge cycle. A closure contains at least one concave point relative to the edge cycle. A line segment is added for each closure (Alg. 1 line 3-7). For example, Figure 16 (a) shows the medial-axis transform of the contour edge-cycle (the pink one) in Figure 15 (b). As shown in the figure, both points in the closure of the dorsal fin are concave so a segmentation line is added to segment the dorsal fin. There are five closures detected for the fish contour and the fish contour is segmented into six geons as shown in Figure 16 (b). Notice that each edge-cycle in the decomposition tree (Figure 15(c)) is segmented into several pieces using the same segmentation algorithm, because the GAL encoding described in next section is performed on each edge-cycle.

To reduce segmentation noise, there are several constraints on closure detection. These are: (1) the sum of the angles of two closure points should be less than 3.05 radians, (2) one of the angles of two closure points should be less than 2.85 radians, (3) the distance between the two points of each closure should be less than one-sixth the length of the perimeter of the edge cycle, (4) only one closure with the smallest angle sum is detected in a certain range (i.e. 1/20th the length of contour) and (5) all segments whose area is below a preset threshold and which do not connect more than one other segments are dropped. These parameters were all determined experimentally on pilot data. In Algorithm 1, these five constraints are checked in the *PreCondition* function.

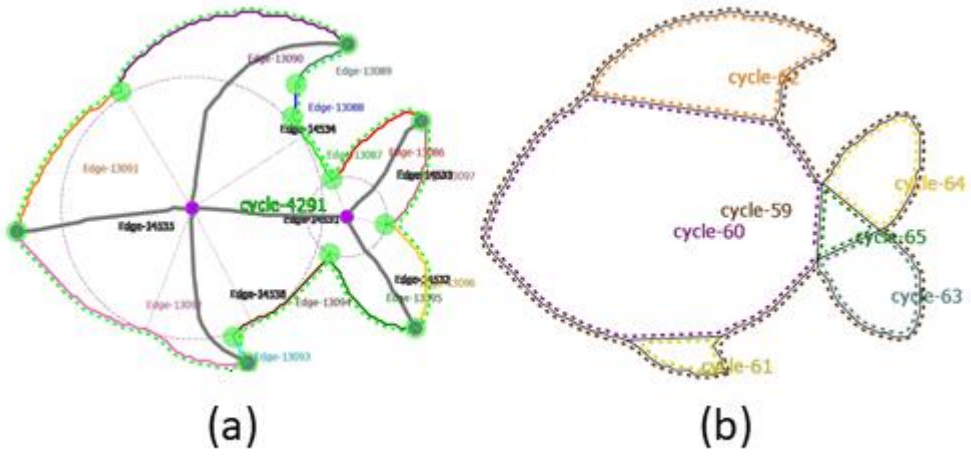


Figure 16: (a) is the medial-axis transform of the contour edge-cycle of Figure 15 (a). (b) is the segmentation result.

GAL Encoding

The GAL encoding algorithm is shown in Algorithm 2. After decomposition and segmentation are completed (Alg. 2 line 6), the geometric features of each edge-cycle (Alg. 2 line 6-13) and the spatial relations between edge-cycles are encoded (Alg. 2 line 14-21). Each edge-cycle is described as a combination of the attributes of its geons (Alg. 2 line 7-9), as well as the positional relations and connection relations between geons (Alg. 2 line 10-13).

Algorithm 2: GAL Encoding

Input: The decomposition tree T

Function:
EncodeSegment (S): encode the eight attributes in Table 1 for a segment S.
EncodePairEdgeCycles (C1, C2): encode connected and positional relations between a pair of edge-cycles C1, C2.
EncodeRCC8 (C1, C2): encode the RCC8 relation between a pair of edge-cycles C1 and C2.
Segmentation (C): object segmentation algorithm in Algorithm 1.

Output: A list of encoded statements E

```
1:    E = []
2:    For Li from 0 to Length (T) do
3:        C = T[Li]
4:        CP = T[Li-1] if Li > 0 else []
5:        For Ci in C do
6:            S = Segmentation (Ci)
7:            For Si in S do
8:                F = EncodeSegment (Si)
9:                E.append (F) for f in F
10:           For S1, S2 in S do
11:               If connected (S1, S2) do
12:                   PS = EncodePairEdgeCycles (S1, S2)
13:                   E.append (PS) for f in PS
14:       For C1, C2 in C do
15:           If connected (C1, C2) do
16:               PC = EncodePairEdgeCycles (C1, C2)
17:               E.append (PC) for f in PC
18:       For C1 in C, C2 in CP do
19:           If RCC8NTPP (C1, C2) do
20:               RC = EncodeRCC8 (C1, C2)
21:               E.append (RC) for f in RC
22:   return E
```

Attribute selection for geons and edge cycles poses a tricky trade-off: the more attributes described, the more details of the segment are available—and the more training examples are needed to learn useful generalizations in analogical learning (see below). To address this trade-off, I used greedy

search to select five attributes with the best discrimination out of eight possible attributes. The selection of the eight-attribute scheme is based on visual analysis of the datasets and inspired by Recognition-by-components theory [Biederman, 1987]. All attributes are converted to a qualitative value description (i.e.,

Attribute	Description
Eccentricity	The principal axis ratio computed from the covariance matrix of all polygon contour points.
Compactness	The ratio between the polygon area and estimated cycle area based on perimeter of polygon.
Circularity	The ratio between the standard deviation and mean of radial-distance between polygon contour points and polygon centroid.
Ellipsity	The ratio between the standard deviation and mean of d-primes between polygon contour points and polygon centroid.
Convexity	The ratio between the perimeter of polygon and the perimeter of its convex hull.
Solidity	The ratio between the area of polygon and the area of its convex hull.
Orientation	The orientation of the polygon main axis: vertical, horizontal, and angular.
Area Size	The relative area size of the polygon with respect to other polygons in one segmentation.

Table 1: Descriptions of the eight encoding attributes

low, medium and high) according to preset thresholds. Table 1 describes the details of the eight attributes. During encoding, the **isa** predicate in Cyc is used to express attributes, for example,

(isa <segment-geon> HighEccentricity)

(isa < segment-geon > LowCompactness)

(isa < segment-geon > MediumCircularity)

(isa < segment-geon > HighEllipsity)

(isa < segment-geon > MediumConvexity)

……

The connection relations between segments are described via the **segmentsConnectToViaEdge** predicate. The first argument of this predicate is the connecting edge of first edge cycle and the second argument is the connecting edge of second edge cycle. For example:

(segmentsConnectToViaEdge

(rightOfEdgeOfCycleFn < segment-geon-1>) (leftOfEdgeOfCycleFn<segment-geon-2>))

The positional relations **above** and **leftOf** are used to describe the relations between edge cycles of the same degree in the decomposition tree. RCC8 relations [Randell et al., 1992] are used to describe the containing relations between the segments of edge-cycles and their children. For example, the following relations describe some spatial relations in Figure 15:

(above EdgeCycle-31 EdgeCycle-32)

(leftOf EdgeCycle-32 EdgeCycle-31)

(rcc8NTPP EdgeCycle-32 EdgeCycle-35)

The set of entities, attributes, and relations computed for a sketched object are combined to form a qualitative representation.

Analogical Learning

Sketched object recognition is a supervised learning task. Here I assume that each training example is labeled and added to the single SAGE generalization pool for each category. As described in the analogical learning process in Section 2.1, SAGE makes one or more generalizations for each concept.

Classification is performed using MAC/FAC, where the probe is the new testing example to be classified and the case library is the union of generalization pools representing the possible classifications. The generalization pool from which the best reminding comes is used as the label for that example.

3.3.2 Data-efficiency Experiments

The claim is that analogical learning over qualitative representations generated by GAL encoding provides understandability and data-efficiency. Regarding understandability, people can understand the

learned generalizations in SAGE to understand what the common geometric relations are for each object category. Data-efficiency means that my approach can achieve reasonable performance with small amounts of training data. To evaluate the data-efficiency of this approach, the system was tested on two very different sketch datasets, using a relatively small number of training examples.

Experiment 1: MNIST Dataset

The first dataset is the MNIST dataset [LeCun et al., 1998]. MNIST is popular for evaluating machine learning algorithms. Start-of-the-art deep learning models can achieve over 99% accuracy on the testing set. However, these approaches apply data augmentation on the training set and use more than 7 times of the number of training samples in the original dataset for training. In this experiment, I compare my approach with deep learning baselines and see whether the model can achieve competitive results if both models use a small number of training samples. To evaluate data efficiency, I use randomly selected subsets of 10, 100, and 500 images per category as the training set and test trained models on the full test set.

Method

All examples are converted into qualitative representations using the CogSketch pipeline described above. As most of the digits in MNIST dataset are very similar, I choose the SAGE assimilation threshold to be 0.9 and a cutoff threshold of 0.2 in all experiments. For each training set size with 10, 100, 500 per category, I create subsets by randomly select examples from training set. Our approach and deep learning baselines are trained on the subsets and tested on the full testing set.

Training Size (per digit)	Our Approach Accuracy	LeNet-5 Accuracy
10	54.9%	10.01% (chance)
100	76.24%	49.52%
500	85.03%	86.77%

Table 2: Accuracy and standard deviation per training size

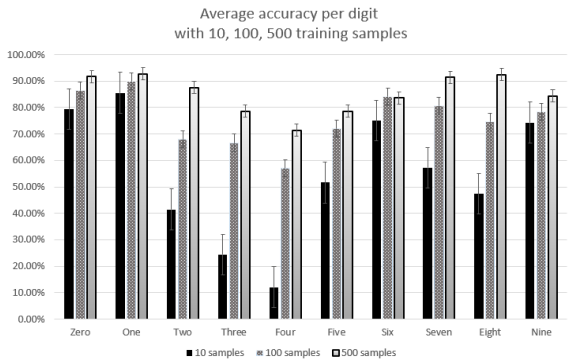


Figure 17: Average accuracy per digit of our approach [Chen and Forbus, 2019]

Results

See Figure 17 for the average accuracy per digit of our approach and Table 2 for overall accuracy and standard deviation information for each training set size compared with LeNet-5 [LeCun et al., 1998]. When I choose deep learning baselines, I focused on two aspects: the baseline model should achieve reasonable performance on visual inputs and the number of parameters should be relatively small (larger model with more parameters usually require more training data to converge). LetNet-5 has good performance on MNIST dataset and it is relatively a simple CNN model compared with recent CNN-based models such as ResNet or Transformer. LeNet-5 has similar performance on MNIST dataset compared to other methods. As our approach saw each example only once instead of multiple epochs like deep learning models, I trained LeNet-5 with only 1 epoch and choose the approperate learning rate after hyper-parameter search. LetNet-5 performs at chance with 10 samples per digit, 49.52% with 100 samples per digit, and 86.77% with 500 samples per digit. Compared with [Chen and Forbus, 2019], I performed more hyperparameter search with 4 Nvidia 1080Ti GPUs on LeNet-5. Although LeNet-5 achieves comparable accuracy to our approach with 500 samples per digit, our approach performs significantly better than LeNet-5 with 10 or 100 samples per digit. These results prove that our approach can achieve resonable accuracy. With very limited training data, such as 10 samples per digit, the model outperforms deep learning baselines.

Experiment 2: The Coloring Book Object Dataset

The second dataset is the Coloring Book Object dataset. As introduced in Section 2, this dataset has 19 categories, and each category has only 10 examples. Therefore, this experiment aims to evaluate whether the model can capture the object patterns using a very small number of training examples. Again, I compare analogy-based approach to LeNet-5.

Method

I use leave-one-out cross-validation to perform sketched object recognition. In each round, nine images are used as training data and one image is used for testing. Each animal or everyday object category is modeled as a generalization pool. A SAGE assimilation threshold of 0.9 assimilations and a SAGE cutoff threshold of 0.2 are used and average accuracy is computed.

As a baseline, I compared to results of the CNN model LeNet-5 trained using the same leave-one-out cross-validation technique. LeNet-5 has 2 convolution layers with a ReLU activation followed by max-pooling layers and a fully connected layer with softmax. As this model has good performance on the MNIST dataset, I assume that this deep learning model should be able to recognize sketched objects.

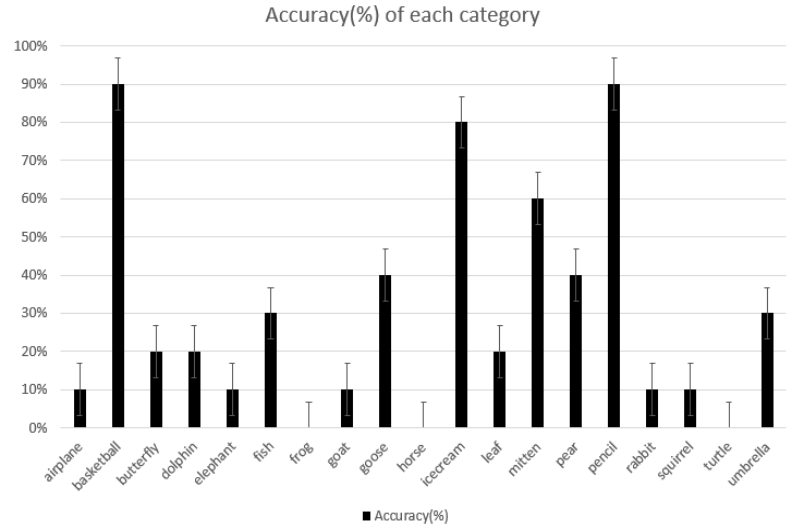


Figure 18: Average accuracy for each category in CBO [Chen and Forbus, 2019]

Results

Figure 18 shows the sketched object recognition accuracy for each category using our approach. Table 3 shows the overall accuracy and standard deviation of each model. Our approach achieves 29.47% accuracy, which is significantly above chance. The CNN model only achieves 5.26% accuracy, which does not differ from chance. Our model does not correctly recognize frog and turtle. The frogs and turtles

Methods	Overall Accuracy (%)	Standard Deviation
Our approach	29.47%	2.72%
LeNet-5	5.26%	1.19%

Table 3: Overall Accuracy on CBO and standard deviation results [Chen and Forbus, 2019]

in CBO dataset have very different styles, poses or textures. Thus, the model cannot capture the common patterns across the limited data.

Discussion

These results indicate that CogSketch plus analogical generalization can surpass 85% accuracy on the MNIST dataset using only 500 examples per concept and reaches 76.24% with just 100 examples per concept. We note that with LeNet-5 we were not able to get better than chance performance with only 10 examples per concept on standard MNIST inputs, but our approach can achieve 54.9% with the same number of training samples. We did not get very competitive results with the state-of-art because the

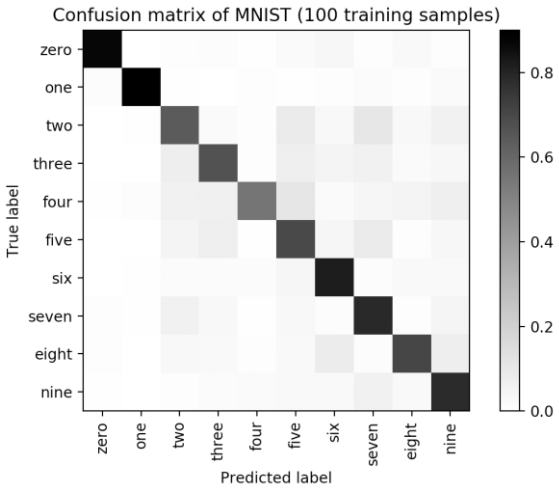


Figure 19: Confusion matrix of MNIST (100 training samples) [Chen and Forbus, 2019]

MNIST dataset is highly down-sampled, to fit the constraints of CNNs at the time, which introduces significant amounts of noise. Even though my preprocessing stage removes some noise, the object segmentation stage, and the attributes CogSketch for segment edge-cycles still have bias or errors. Thus, some images have similar segmentations to other digits. For example, Figure 19 shows the confusion matrix from when our system was trained on 100 examples per digit. A frequent failure mode is the digit two being mistaken for a five, and vice versa. This is an example of the segmentation problem—both twos and fives are sometimes interpreted as two segments (essentially a top curve and a bottom curve), connected in the middle.

As mentioned above, attribute selection is a tricky question that needs further exploration. When segments are similar, the selected attributes may lose information. Also, given the down-sampled images, CogSketch may not be able to compute correct attributes. The results might be better with the original NIST dataset, but I have not yet explored this option.

With the Coloring Book Objects dataset, which has extremely high variability, even with only 9 training examples as training data, our system has significantly better accuracy than chance, whereas a CNN model performs at only chance (Table 3). The variability in this dataset is extreme: Animals sometimes have hats, for example. Figure 20 shows the confusion matrix for this dataset. It shows that

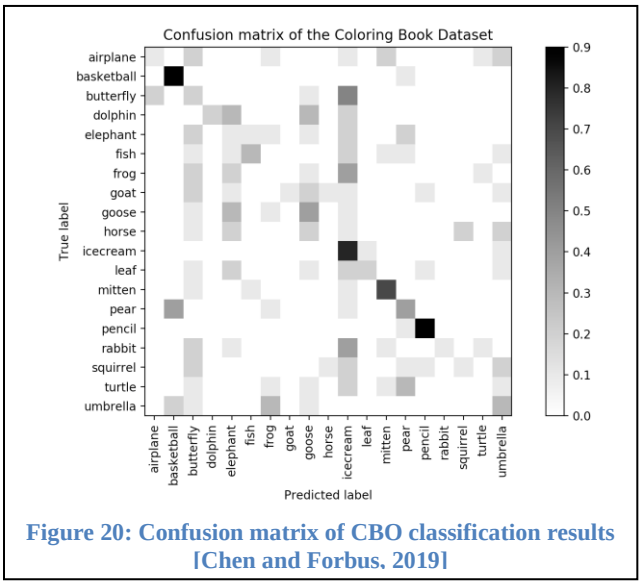


Figure 20: Confusion matrix of CBO classification results [Chen and Forbus, 2019]

our system has high accuracy on simple objects such as mittens and pencils but cannot distinguish butterflies and ice-cream—likely because they have complicated texture or shapes. Being able to recursively decompose recognition might be necessary to get very high accuracy on this dataset. Also, texture representation is also essential on this dataset. In section 3.5, I describe a novel encoding approach for texture representation that leads to improved results.

While the results do not yet approach the state of the art on the MNIST dataset and the performance on the Coloring Book Object dataset has plenty of room for improvement, these results already support the hypothesis that analogical learning over GAL qualitative representations performs more human-like ways, with far better data efficiency than deep learning models. In the future, near-misses in analogical learning [McLure et al., 2015a] could be integrated with this approach, which might provide accuracy improvements.

3.4 Simulating Infant Visual Learning by Comparison

Researchers have recently found that 3-month-old infants are capable of using analogical abstraction to learn the *same* or *different* relation, given the right conditions [Anderson et al. 2018]. Surprisingly, seeing fewer distinct examples led to more successful learning than seeing more distinct examples. This runs contrary to the prediction of standard learning theories, which hold that a wider range of examples leads to better generalization and transfer. However, this result is compatible with other findings in infant research [Casasola 2005; Maguire et al. 2008]. Anderson et al. [2018] propose that this is due to interactions between encoding and analogical learning. This section explores that proposal through the lens of cognitive simulation, using automatically encoded visual stimuli and our cognitive model of analogical learning. The simulation results are compatible with the original findings, thereby providing evidence for this explanation. The assumptions underlying the simulation are delineated and some alternatives are discussed. This work in this section was originally published in [Chen et al., 2020] in collaboration with other researchers.

3.4.1 Introduction

Relational learning and reasoning are central in human cognition [Gentner, 2003, 2010; Gentner & Markman 1997]. How does this ability arise? Is analogical ability built up gradually via maturational change, or by combining other component processes? Or is the structure-mapping ability an innate species-level adaptation? The first possibility may seem more plausible, given the abundant evidence that relational sophistication increases over development [Gentner & Rattermann, 1991]. But recent findings suggest that analogical processing ability may be present early on, and that developmental gains in analogical fluency are due to increases in relational knowledge [Gentner, 2010; Gentner & Rattermann, 1991] and/or executive ability [Richland et al., 2006; Thibaut et al., 2010]. For example, Ferry, Hespos and Gentner [2015] found evidence that 7-9-month-old infants can carry out analogical abstraction across a sequence of examples to derive an abstract *same* or *different* relation.

Anderson et al. [2018] recently reported that even 3-month-old infants can learn the *same* or *different* relations via analogical abstraction. A surprising aspect of the research was that the infants learned better when given fewer examples. In the first experiment, infants failed to learn these relations after being shown six distinct examples of either *same* or *different* repeated until habituation. (The exact number of habituation trials varied, ranging from 6 to 9 trials until infants' looking times declined by 50% from the first three trials to the last three, or until infants had completed nine trials.) In the second experiment, infants succeeded after being given repeated exposure to only two examples of the relation. As noted above, this result runs contrary to the predictions of standard learning theories, which predict that a wider range of examples leads to better generalization and transfer, but it is compatible with some prior findings on infant relational learning [Casasola 2005; Maguire et al. 2008].

Anderson et al. [2018] proposed that these phenomena are due to interactions between encoding and analogical processing. This section examines this proposal via cognitive modeling, using automatically encoded stimuli and a model of analogical learning. Specifically, we ask whether the 3-

month-old pattern can be modeled by assuming that the infants have structure-mapping ability, but that they are limited by their encodings of examples. We lay out a set of assumptions that provide a possible processing account and show that these assumptions could explain the generalization pattern. The modeling enterprise also reveals other possible encoding assumptions, which can be explored in future work.

We first review prior research on analogical abstraction, then describe the Anderson et al. [2018] experiments to be modeled. Then we describe our model of the infants' encoding and learning process. To preview, the model is constructed from pre-existing components (described below). This includes automatic encoding of visual stimuli based on photos of the objects shown to the infants. In the previous section, CogSketch encodes all edge-cycles in the decomposition tree into one qualitative representation. In this cognitive simulation, CogSketch encodes objects incrementally, with multiple levels from simple features to detailed geometric features. We describe the processing performed by the model, laying out the assumptions we are making and noting where alternative explanations are feasible. Then we present the results of the simulation. We end with a discussion of the implications and possible future work.

3.4.2 Background

There is evidence of analogical ability in children from early preschool through adulthood [Gentner, 2003; Gentner & Rattermann, 1991; Richland et al., 2006]. Two signatures of this ability are (1) the ability to perceive abstract relational matches can be enhanced by comparing instances of a relation, in both adults [Gick & Holyoak, 1983; Markman & Gentner, 1993] and children [Gentner, 2003; Kotovsky & Gentner, 1996]; and (2) the presence of salient objects can interfere with relational mapping, especially early in development [Gentner & Toupin, 1986; Paik & Mix, 2008; Richland et al., 2006]. These findings are consistent with other research suggesting that comparison entails a structural alignment process that highlights relational commonalities between the items compared [Markman & Gentner, 1993].

Recent research has explored relational learning in human infants [Anderson et al., 2018; Ferry et al., 2015; Gervain et al., 2012]. Ferry et al. [2015] found evidence that 7 to 9-months-old infants can engage in analogical abstraction. When shown a series of *same* pairs (using the method described below), infants afterward looked significantly longer at a novel *different* pair than at a novel *same* pair (and the reverse for habituation to *different*). This is evidence for the first signature of analogical processing—that comparing across examples promotes abstracting the common relational structure. They also found evidence for the second signature of analogical processing: that salient objects tend to distract from relational processing. When infants were shown a subset of objects prior to habituation, they performed poorly on test trials containing these objects, failing to distinguish *same* and *different*. Thus, Ferry et al. [2015] concluded that by 7-9-months, infants can use analogical generalization to form a relational abstraction.

3.4.3 Analogical Learning in 3-month-old Infants

To explore the origins of analogical ability, Anderson et al. [2018] asked whether 3-month-olds could abstract *same* and *different* relations. Infants were shown a series of pairs: half the infants saw *same* pairs and the other half saw *different* pairs[4]. The materials were pairs of colorful, distinctive objects (Figure

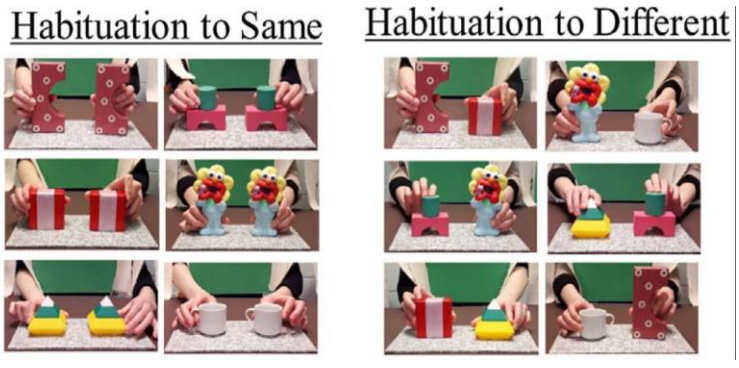


Figure 21: Examples of habituation pairs from Anderson et al. Expt. 1 [2018].

[4] To test for salient-object interference, the infants had previously seen some objects in the waiting room; this is not modeled here.

21). In order to engage infants' attention, on each habituation and test trial, the pair was moved together through a fixed motion path: up, then tilted left, then right, then down to the start point. This 8-second cycle was repeated continuously until the infant looked away for 2 seconds. Then the next pair was shown in the same way. The habituation trials continued until the infant's looking time declined by 50% from the first three trials to the last three, with a maximum of nine trials (range = 6 to 9 trials).

Both groups of infants then saw the same six test pairs—three depicting the *same* relation and three depicting the *different* relation. The pairs were shown one at a time, and the key dependent measure was how long the infant looked at each pair. The key test pairs had brand new objects instantiating either the *same* or *different* relation. If infants have abstracted the relation they saw during habituation, they should look longer at the novel relation. Looking time is a commonly used measure with preverbal infants; the idea is that the familiar relation will fit their expectations, whereas the novel relation will be more surprising.

In Experiment 1, infants were shown six distinct pairs (either all *same* or all *different*) during habituation. During testing, the infants failed to look longer at novel pairs on the key trials. Thus, there was no evidence for analogical learning. Although this could mean that 3-month-olds lack this ability, the experimenters explored another possibility: that the infants were overwhelmed by the variety of objects in the study, and thus failed to encode the relations between them (e.g., Casasola & Park, 2013). Consequently, in Experiment 2, only two distinct pairs were used during habituation (e.g., AA, BB, AA, BB...for *same*). The infants were then tested in the same manner as in Experiment 1. In this case, infants did indeed learn. They looked longer at pairs showing the novel relation, even with brand new objects—evidence that they had abstracted the relation.

3.4.4 Simulating the Infants' Learning

In order to abstract a *same* or *different* relation from a series of examples, two things must happen (not necessarily in a fixed order): (1) the learner must compare the objects *within* each pair to form some

initial representation of the *same* (or *different*) relation within the pair; and (2) the learner must compare *across* the pairs to arrive at a more abstract encoding of the relation. Only if both those things happen will the learner experience a brand-new *same* pair as familiar. Our simulation explores one path—by no means the only path—by which this could happen.

3.4.5 Simulation Design

Here we discuss our simulation. We begin by noting a critical point: in order to be informative about human cognition, a simulation must be constrained. Many simulations have used hand-coded representations to depict the learner's construal of a situation, and/or have implemented a simulation process specific to the situation being modeled. But this allows enormous latitude to tailor the representations and processes to fit whatever outcome is desired. To avoid this problem, (a) as input, the model is given representations that are automatically encoded from the visual stimuli given to the infants; and (b) our processing model is built out of pre-existing components that have successfully simulated prior findings in analogical processing.

We first describe the component models, then how they are combined.

Simulation of analogical processing

We use the Structure-Mapping Engine as a simulation of analogical mapping, and SageWM [Kandaswamy et al. 2014] as a simulation of analogical generalization in working memory. These models have been used to model a number of psychological phenomena already. We use 0.95 as the assimilation threshold in these experiments[5] which is the default for SageWM.

Simulation of visual encoding

The production of visual stimuli occurs via an automatic pipeline, starting with photographs the pairs of objects provided by the original experimenters. The photographs are blurred, and the Canny edge

[5] As descriptions are merged, frequency counts are kept for how often each statement is aligned. If the probability goes below a threshold (0.2 by default), the statement is eliminated.

detector is used to generate a sketched glyph describing each object. CogSketch decomposes the glyph into edges and edge-cycles. CogSketch automatically computes a variety of information about each edge, including its length, curvature, orientation, position, and topological relations with other edges. CogSketch computes the same attributes as described in Table 1 for each edge-cycle, which we assume are visually salient and likely to be encoded early in human processing. Since color is visually salient in those stimuli, we also use a color extraction library to extract up to eight of the most frequent colors for an object.

An important issue in this modeling effort is to consider the visual encoding processes available to 3-month-olds. In the first months of life, vision and attentional processes are becoming increasingly stable [Arterberry & Kellman, 2016; Colombo, et al., 1991]. Visual acuity improves steadily through the first several months. Especially relevant here, infants' habituation and fixation periods decrease dramatically during the first 6 months [Bornstein, 1985; Colombo & Mitchell, 2009] suggesting that young infants' encoding is slower and more variable than that of older infants.

To capture young infants' relatively inefficient encoding processes, here we have assumed *slow encoding*—that is, that not all the available information is encoded on first exposure. (Other assumptions

```
;; Overall properties
(WhiteColor A)
(RedColor A)
(highRectangularity R-1)
(ellipseSystemShape R-1)
(hasConcavedEdge R-1)
(someCurvedCycle R-1)
[…] ;; Internal regions
(LowRectangularity  R-34)
(CycleSystemShape R-34)
(NonConcavedEdge R-34)
(AllCurvedCycle R-34)
(above R-29 R-34)
(rightOf R-34 R-29)
[…]
```

Figure 22: Example of boundary and geon relational representations. These relational representations are automatically generated. For the readers' convenience, we use English words to indicate relations and entities. Example from [Chen et al., 2020]

are possible, including variable extraction of information.) Specifically, we assume that the boundary of an object, its shape properties, and color are encoded early. When given more time, we assume infants compute more detailed representations of the shape, including internal properties and relations. We use GAL encoding as described in the previous section. On the first exposure, we assume that infants can only encode the equivalent of the root node of decomposition tree, which is the boundary edge-cycle of the object. CogSketch is used to segment this shape as before. On the second exposure, we assume that infants have enough time to encode the details of the object and perform GAL encoding on the whole decomposition tree. Figure 22 shows examples of boundary and GAL representations for one of the objects. We further assume that, given sufficient time, infants encode representations of both objects.

In the original experiments, the pairs were moved in a uniform way throughout the habituation and test trials. We assume that the infants encode these motions, since motion is extremely salient for them. While we could use qualitative spatial representations to automatically represent the specific motions of the stimuli as part of the encoding process, using techniques described in Section 5, this would involve considerable complexity to gather the video data. Thus, we do not explicitly encode such motions in the present model.

We hypothesize that repeated motion influenced the infants' processing in two ways. First, within a trial, the two objects in a pair always move together. This gives rise to a perception of the unity of the pair and prompts the infant to compare the two objects in a pair. Over trials, as the object representations become more detailed, this will lead to perceiving many common attributes in a same pair (or few, in the case of a different pair). We call the representation of the two objects plus relations computed between them the *pair-level* description. We hypothesize that pair-level descriptions are only computed when both objects have been fully encoded. The second effect of the repeated motion is to invite comparison across trials: even though the individual pairs (say, AA and BB) are quite distinct, we hypothesize that the similarity in their motion leads the infant to compare them, as described below.

To represent the visual similarity of objects, we use one of two relations, depending on whether their similarity, as measured by SME, is above a particular threshold (here, 0.5). If their similarity is above the threshold, a statement using the **sameObject** relation is encoded, and otherwise, **differentObject**. We use these terms for convenience, but we do not assume that infants distinguish absolute sameness from high similarity (see [Smith, 1993]). It is also not clear whether infants are learning these relations de novo, or whether they already possess some kind of representation of *same* and *different*, either innately or through early learning. We return to this question in the Section 3.4.7.

Processing Assumptions

To recapitulate, we assume that infants encode the motion of the pair of objects and that this invites comparison both within and across trials. However, the comparison process also requires that the object representations be sufficiently detailed. We do not assume that infants encode everything about the objects in a trial at first exposure. As noted above, we assume that information about object boundaries and color are computed first, followed by information about the decomposition of the object into geons, and that these two levels of representation occur in that sequence. We assume that even partial object representations are stored in SageWM and retrieved the next time they are exposed to the pair. This retrieval speeds up the initial encoding process, allowing processing to move on to the next level of encoding.

It is not clear whether infants are encoding both objects on first exposure to a pair. Here we assume that objects are encoded independently in parallel, but with the levels of representations outlined above. We assume that having the objects placed into correspondence causes them to be compared, once their encodings are complete. The result of this comparison results in the description of the pair being augmented with a **sameObject** or **differentObject** statement, depending on the outcome of that comparison.

3.4.6 Experiment Simulation

Now let us reconsider the experiments in Anderson et al. [2018] through the lens of cognitive simulation. We discuss each experiment in turn. In both simulations, we did not simulate the infants' experience of some objects from the waiting room.

Simulation of Experiment 1

Following the original experiment, we simulated two habituation sequences: one with a sequence of six pairs of objects satisfying the *same* relationship (<A,A>, <B,B>, <C,C>, <D,D>, <E,E>, <F,F>) and one with a sequence of six pairs of objects satisfying the *different* relationship (<A,B>, <C,D>, <E,F>, <B,C>, <F,A>, <D,E>). Given our assumption of parallel object encoding, in the *same* condition only the first level of encoding occurs for each object in the simulation, and hence the objects are not compared, and no pair-level descriptions are generated. For the *different* conditions, there are repeated exposures to particular objects, but another comparison involving them would be needed to generate pair-level representations. Since there are no pair-level examples, they cannot be compared and generalized, and hence no analogical learning takes place, compatible with the infant results.

Simulation of Experiment 2

Following the original experiment, two sequences of alternating pairs of objects were used. For the *same* habituation trials, these were (<A,A>, <B,B>, <A,A>, <B,B>, <A,A>, <B,B>, and for the *different* habituation trials, these were <A,B>, <C,D>, <A,B>, <C,D>, <A,B>, <C,D>. Thus, for both habituation conditions, each pair was presented to the simulation three times, in alternation. In the first exposure to a pair, the first level of encoding occurs for its objects, which are stored in SageWM. In the second exposure, the second level of encoding occurs, building on the initial model stored in SageWM. In the third exposure, the fully encoded objects retrieved are used to construct a pair description, including the cross-object comparison (because of the assumed common roles in the motion perceived by the infants). That pair description is also stored in SageWM. The pair representations are generalized by SageWM

across pairs as they occur: that is, a generalization is formed that includes either a **sameObject** or a **differentObject** statement, depending on habituation condition. This new abstraction is relatively portable, since it has many fewer object details in common, and hence is retrieved when test pairs are presented. Even if these test pairs are not fully encoded (because of novel objects), alignment with the abstraction leads to a projection of a **sameObject** or **differentObject** statement as a candidate inference (depending on whether habituation was for *same* or *different*). When a test pair is compatible with the learned relation, the candidate inference fits. When a test pair is incompatible with the learned relation, the candidate inference is contradicted, and this novelty, we hypothesize, leads to greater looking times for the infant.

3.4.7 Discussion

The simulation captures the pattern of infant results across the two experiments: When given six different example pairs (Experiment 1), the simulation fails to form abstractions of *same* and *different* during habituation, and therefore fails to differentiate novel from familiar relations during test. When given two pairs (Experiment 2), the simulation forms abstractions of *same* and *different* during habituation, and therefore arrives at distinct matching scores for novel vs. familiar relations during test.

Thus, we have shown that a reasonable set of assumptions about the visual encoding of infants, along with pre-existing encoding algorithms and analogical process models, can be used to simulate Anderson et al.'s [2018] results on analogical learning in 3-month-old infants. This provides evidence for their proposed explanation, in terms of partial infant encoding.

This simulation assumed that something like **sameObject** and **differentObject** were already available to infants. How might such relationships be learned, even perhaps during the experiment? It is not unreasonable, given how ubiquitous analogy and similarity appear to be in human cognition [Gentner 2003], that infants can remember the qualitative feeling of high-similarity or low-similarity for pairs that they have just seen. In other words, the alignments during analogical generalization could provide the

basis for introducing a simple qualitative value on similarity, e.g., high or low [Forbus, 2019]. For example, given habituation on *same* trials, these similarity scores will tend to cluster quite high, and given habituation on *different* trials, these similarity scores will tend to cluster quite low (see Figure 23). Seeing a score for a pair in the same role that is substantially different, i.e., a different qualitative value, could also predict looking times and reifying such a difference into a pair of relationships would then make such information accessible in future comparisons. This provides a possible explanation for how such relationships can be learned.

Our general assumption is that the rather surprising pattern—that 3-month-olds can from an abstraction from two alternating pairs over six pairs but not from six different pairs—results from inefficiencies in their visual encoding process. In this simulation, we have focused on slow encoding to capture this inefficiency. Another interesting possibility is variable encoding. For example, different subsets of geons might be computed over different exposures, so that the perceived similarity of a pair over time would depend on the particular orders in which geons were found. Such models could be explored in future work.

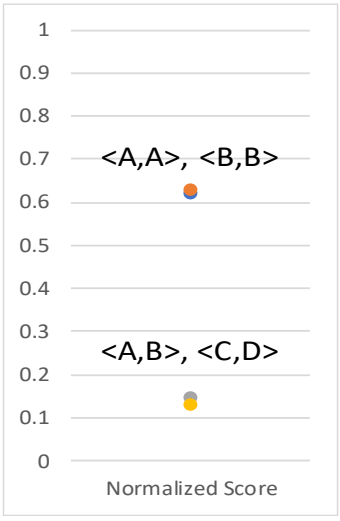


Figure 23: Each point represents the normalized similarity score for same vs. different pairs from Experiment 2. Distributional properties of numerical similarity estimates provide a possible signal for introducing qualitative values and learning same/different relationships. [Chen et al., 2020]

Despite the vast amount of research on analogical processing in children, there is very little research on how children learn relations in the first place. One exception is DORA [Doumas et al. 2008]. DORA begins with unstructured representations of objects as simple feature vectors. When DORA compares two or more objects, it forms explicit representations of any properties they share. These properties are then combined into relations. This contrasts with our model, in which the relations are formed from online differences in qualitative similarity.

3.4.8 Conclusion

Our results lend support to the idea that 3-month-old infants have structure-mapping ability but are limited by their encodings of examples. Here we have shown that a reasonable set of assumptions about encoding and the use of analogical generalization within working memory simulate the experiments from Anderson et al. [2018]. The simulation provides an explanation for why 3-month-old infants are able to learn, or not learn, *same/different* relations. In our assumptions, infants cannot encode object rapidly in the first exposure and they usually first encode visually salient features such as color and contour before

encoding object details. Thus, inspired by this idea, we propose a hierarchical encoding strategy in the next section.

We see several paths for future work. First, we think encoding variability may be an important factor in explaining the conditions under which infants can learn. Second, we want to simulate a wider range of experiments with this model, including experiments with older infants (e.g. Ferry et al. 2015). This will involve developing and testing plausible models for how encoding skills change across development with experience and building up models of long-term experiences and generalizations that infants accumulate.

3.5 Part-based Hierarchical Analogical Learning

In the previous section, cognitive simulation of infant visual learning suggests that infants seem to perform recognition by encoding objects from coarse-grained contours to finer details. This inspired us to explore using a hierarchical encoding scheme to describe objects for sketched object recognition. Given an object, instead of encoding all information in one qualitative representation, the idea is to encode it as multiple qualitative representations from coarse-grained information to details. However, this leads to the need to extend the way analogical learning is done, to combine multiple qualitative representations. This section introduces a novel way to use analogical learning hierarchically. I propose an alternative encoding

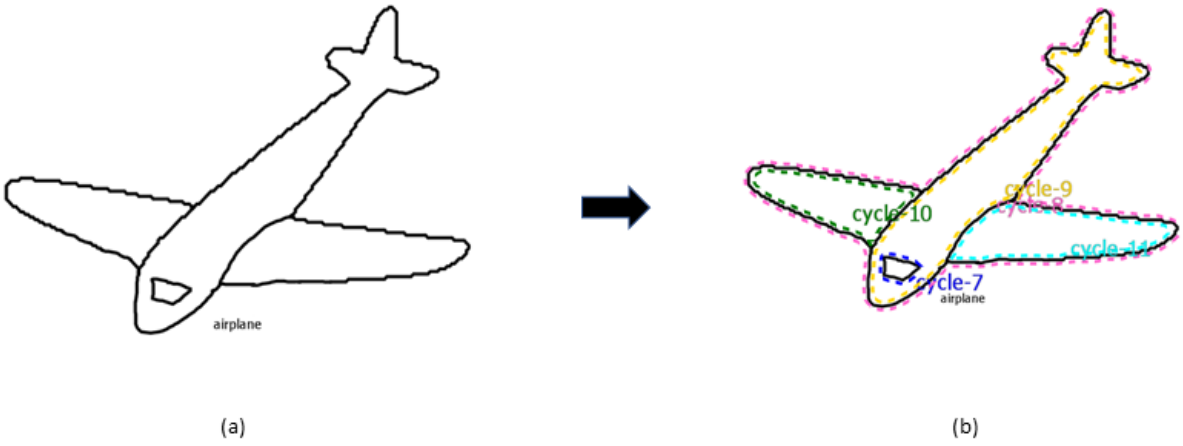


Figure 24: The pipeline of PHAL. (a) The system takes a sketched object as the input. (b) The sketch is decomposed into edges and edge-cycles.

scheme to represent object geometric features via describing properties of local parts. Thus, combining everything together, I develop a novel approach for sketched object recognition, *Part-based Hierarchical Analogical Learning (PHAL)*.

In PHAL, given a sketch, parts are generated from its digital ink, constructing a multi-level hierarchical structure from the outside contours inward. It constructs qualitative representations on parts at each level and performs analogical learning at multiple levels, to train models corresponding to the different levels of detail. In classification, analogical retrieval starts with using coarse-grained models to generate a broad range of candidates, which are then refined by using more fine-grained models.

3.5.1 Multi-level Structured Representation

PHAL constructs multi-level structured representations for sketched objects. Given a digital glyph for an object, CogSketch decomposes the glyph into edges and edge-cycles. For instance, Figure 24 (b) shows the edge-cycles for an airplane (Figure 24 (a)). Then, CogSketch generates a decomposition tree based on the containment of edges and closed edge cycles like the GAL scheme. The root node contains the

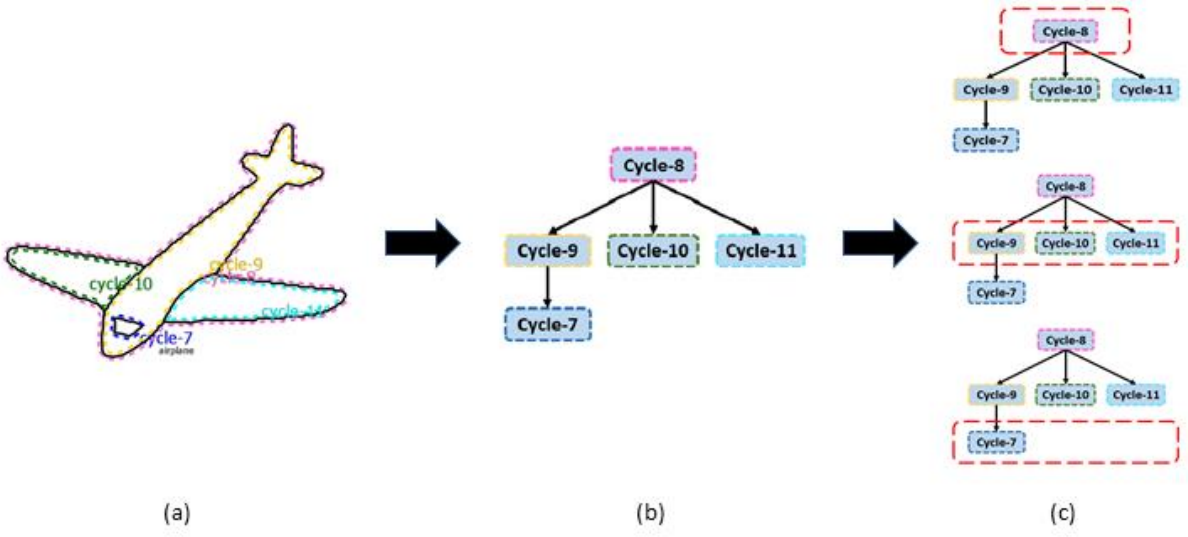


Figure 25: (a) The sketched airplane in Figure 24 (a) is decomposed into edges and edge-cycles. (b) Edges and edge-cycles are organized into a hierarchical decomposition tree. (c) The system constructs three-level part-based qualitative representations.

contour edge-cycle of the whole sketched object. Figure 25 (b) is the decomposition tree of this airplane. Cycle-8 in the root node is the outer contour edge-cycle of the airplane.

Three hierarchical levels of sketch representation are built from this decomposition tree. The first level representation consists of the information in the first layer of the decomposition tree. The second level representation consists of the second level of the tree, and the third level consists of the rest of the levels of the tree. (Most sketches tend to only have two or three layers.) Some sketches have complex textures consisting of many edges or edge cycles. Detecting and representing such textures is important

Algorithm 3: Texture Encoding

Input: Elements in a level of decomposition tree, R

Functions:
EncodeEdge (E): encode the facts of an edge E and return all encoded facts.
EncodeEdgeCycle (C): encode the facts of an edge-cycle C as in Table 4 and return all encoded facts.
SageWMGeneralization (D, L)**:** add all statements in D to case library L via SageWM.
Elements (G)**:** return all elements (edges or edge-cycles) in a generalization G.
isEdge (D)**:** check whether an element D is an edge.
isEdgeCycle (D)**:** check whether an element D is an edge-cycle.

Output: A list of textures T (each texture contains a list of edges or edge-cycles)

```
1:    T = []
2:    For Ri in R do
3:        E = []
4:        C = []
5:        If IsEdge (Ri) do:
6:            Ei = EncodeEdge (Ri)
7:            E.append (f) for f in Ei
8:        If IsEdgeCycle (Ri) do:
9:            Ci = EncodeEdgeCycle (Ri)
10:           C.append (f) for f in Ci
11:       SageWMGeneralization (E, LE)
12:       SageWMGeneralization (C, LC)
13:       For Gi in LE do
14:           If Length (Gi) > 3 do
15:               TEi = Elements (Gi)
16:               T.append(TEi)
17:       For Gi in LC do
18:           If Length (Gi) > 3 do
19:               TCi = Elements (Gi)
20:               T.append(TCi)
21:   return T
```

for sketch recognition. Thus, I also propose a novel texture encoding scheme to detect textures on edges and edge cycles. Textures found for each level are included in each representation. Edges or edge-cycles that are generalized into a texture are removed from the tree. Figure 25 (c) illustrates what is included in the airplane sketch at each level. For each level, I utilize a part-based encoding scheme for representation construction. Texture encoding and part encoding are described next.

Texture Encoding

The decomposition tree is used to guide the process of texture encoding. Algorithm 3 describes the process of texture encoding. All the edges and edge-cycles that have the same parent node are candidates for inclusion in textures at that level of the decomposition tree. Thus, textures can be detected at each level of the decomposition tree. My approach to texture encoding uses analogical generalization. It formalizes the texture encoding problem as finding clusters consisting of sets of edges or edge-cycles that have similar properties. Thus, a twostep process is used: (1) compute a set of properties for the items to be clustered, then (2) add them to a SageWM generalization pool. Each generalization SageWM finds that has three or more items is treated as a visual element representing a texture. The set of items to cluster is based on two assumptions. Firstly, all elements in a texture must be in the same level of the decomposition tree. Secondly, each texture can only contain either edges or edge-cycles. There may be

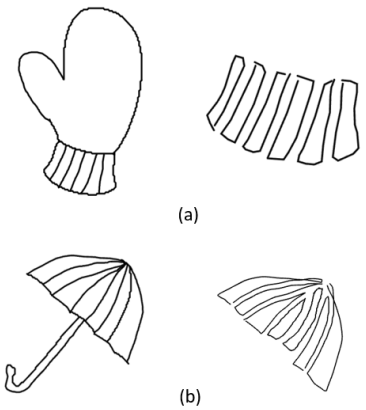


Figure 26: Two examples of texture detection

textures that violate these assumptions, but these assumptions have been robust with regard to the datasets used so far.

The texture algorithm uses three features for edges, namely curvature, length, and direction. Curvature describes whether the edge is curved or straight, length describes the length of the edge related to the group of edges, and direction describes whether the edge is vertical, horizontal, upward, or downward. For edge-cycles, it uses six features: curvature, symmetry, shape estimation, orientation, size, and rectangularity, as described in Table 4. Figure 26 shows two examples of texture detection, where (a)

Attribute	Description
Curvature	Whether all edges of the edge-cycle are curved or straight, including **concavedCycle** or **nonConcavedCycle**.
Symmetry	Whether the edge-cycle has symmetry axes, including **symmetryAxesCycle** or **nonSymmetryAxesCycle**.
Shape Estimation	The shape estimation of the edge-cycle, including **ellipseSystemShape**, **triangleSystemShape**, **rectangleSystemShape**, **polygenSystemShape**
Orientation	Whether the edge-cycle is horizontal, vertical, or acute, including **horizontallyOriented** or **verticallyOriented**, and **acutelyOriented**
Size	The relative size of the edge-cycle comparing with the whole glyph, including **areaTiny**, **areaSmall**, **areaMedium**, **areaLarge.**
Rectangularity	The ratio between the area of the edge-cycle and the area of its bounding box, including **lowRectangularity**, **middleRectangularity**, and **highRectangularity**.

Table 4: Edge-cycle encoding attributes

is a glove with detected texture and (b) is an umbrella with its texture. After using SageWM to make generalizations, I generate a texture encoding for each generalization. The learned statements in a generalization are used as the statements to describe the texture. For each statement, the skolem is

replaced by a texture id. Also, I use a predicate **edgeTexture** or **edgeCycleTexture** to describe whether the texture is constructed from edges or edge-cycles. Thus, the statements could be:

(edgeCycleTexture Texture-1)

(curvedCycle Texture-1)

(verticallyOriental Texture-1)

(lowRectangularity Texture-1)

……

Algorithm 4: PHAL Encoding

Input: The decomposition tree D, Object Glyph G

Functions:
EncodePart (P): encode a part P (edge part, edge-cycle part or convex-hull part).
EncodePartRelation:(P1, P2): encode connected and positional relations on a pair of edge-cycles P1, P2.
TextureDetection (E): detect textures on the set of elements in a decomposition tree level (Algorithm 3).
FindNeighbors (P): find all neighbors of part P.
EncodePartRelation (P, PN, T): encode positional relations between part P and all neighbor parts in the set PN plus the RCC8 relations between P and its textures in the set T.
isEdgeCycle (E): check whether the element E is an edge-cycle.
isEdge (E): check whether the element E is an edge.
Area (E): compute the area of an edge-cycle E.
ConvexHull (S): generate the convex hull on a set of elements S.
Segmentation (C): perform segmentation algorithm on edge-cycle c and return a set of segments.

Output: A list of encoded parts, O

```
1:    O = [], P = [], T = []
2:    For level Li from 0 to Depth (D) do
3:        E = D [Li]
4:        Ti = TextureDetection (E)
5:        T.append(Ti)
6:    For level Li from 0 to Depth (D) do
7:        E = D [Li]
8:        Pi = [], PVi = []
9:        For Ei in E and not in T do
10:           If IsEdgeCycle (Ei) do
11:               If Area (G) / Area (Ei) > 60 do
12:                   For Eij in Segmentation ( Ei ) do
13:                       PVi.append(Eij)
14:               Else do
15:                   Pi.append(Ei)
16:           If IsEdge (Ei) do
17:               If Length (G) / Length (Ei) > 6 do
18:                   PVi.append(Ei)
19:               Else do
20:                   Pi.append(Ei)
21:       Pi.append( ConvexHull (PVi) )
22:   For Li from 0 to Length (D) do
23:       Pi = P[Li]
24:       Ti = T[Li]
25:       Oi = []
26:       For Pij in Pi do
27:           PNij = FindNeighbors (Pij)
28:           Oi.append (EncodePartRelation (Pij, PNij, Ti) + EncodePart (Pij) )
29:       O.append (Oi)
30:   return O
```

Part Encoding

PHAL encoding constructs a representation for each object part. The algorithm for PHAL encoding is presented in Algorithm 4. It generates a set of parts for each hierarchical level of the decomposition tree. Firstly, the texture detection algorithm is performed at each level of the decomposition tree to accumulate textures. (Alg. 4 line 2-5). Then, it splits the components that do not construct textures into a set of edge-cycles and a set of edges (Alg. 4 line 10 and 16). There are three types of parts defined in the encoding: *edge part*, *edge-cycle part*, and *convex-hull part*. These three types of parts are constructed as follows.

For each edge-cycle, CogSketch computes the ratio between the edge-cycle area and the area of the glyph (i.e. the original set of digital ink). If the ratio is larger than 60 (area of the glyph/area of the edge-cycle), it is regarded as a tiny edge-cycle (Alg. 4 line 14-15). For each edge, CogSketch computes the ratio between the length of the glyph and the length of the edge. If the ratio is larger than 6 (length of the glyph/length of the edge), it is regarded as a tiny edge (Alg. 4 line 19-20). CogSketch generates the convex-hull of tiny components (edges and edge-cycles) and the edge-cycle of the convex-hull is the convex-hull part in this layer (Alg. 4 line 21).

Each large edge is an edge part. For each large edge-cycle, I use the segmentation algorithm from the previous GAL encoding scheme. After segmentation, each segment is an edge-cycle part in the rest of the components (Alg. 4 line 12-13).

Figure 27 shows an example of encoded parts of a palm tree. As the decomposition tree of the palm tree only has two layers, I only show the first two layers of encoded parts. In the first layer, the decomposition tree has three components, the edge-cycle of the fronds contour and two edges for the trunk. The two edges encode two edge parts. For the contour edge-cycle, CogSketch performs segmentation, and each segment is an edge-cycle part. In the next layer, four fronds are regarded as tiny edge-cycles, and one frond is a large edge-cycle. Thus, the system computes the convex-hull of the four

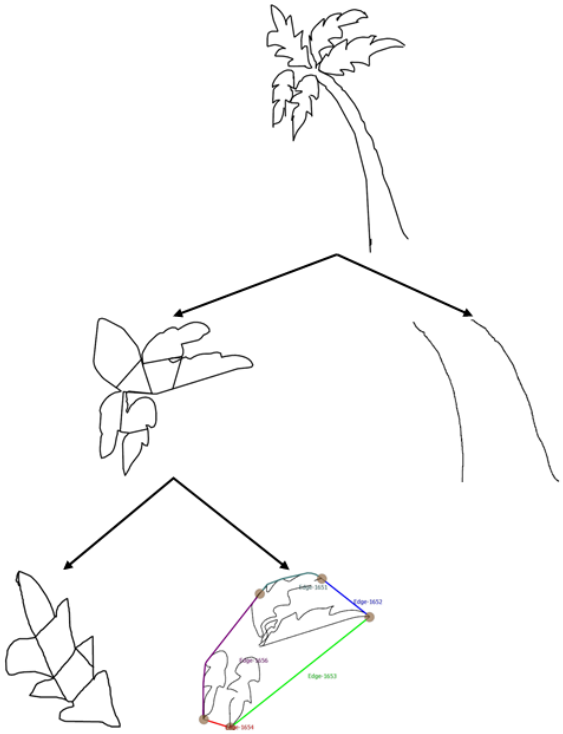


Figure 27: Encoded parts of a palm tree. During encoding, the edges on the leaves have some noise so that CogSketch cannot recognize the four leaves as edge-cycles. Thus, CogSketch computes the convex-hull of these leaves.

tiny edge-cycles; and constructs the convex-hull as a part. On the large edge-cycle, the system performs segmentation, with each segment becoming an edge-cycle part.

Once it has all the parts, a qualitative representation is constructed for each part to represent their local information. CogSketch computes geometric properties for each part and spatial relations between nearby parts. (Two parts in the same level are near with each other if they are connected or the distance between their closest points is smaller than the one over eightieth of the perimeter of the object.) The three features, curvature, length, and direction in texture encoding are used to encode edge parts. For edge-cycle parts and convex-hull parts, the same properties in Table 1 are used to describe their geometric features (Alg. 4 line 22-29).

Between nearby parts, **above** and **rightOf** statements are used to represent positional information. The structured representation for a part is the union of the properties of the part, the properties of nearby parts and the relations between them.

Once the part representations are generated, CogSketch also encodes the spatial relations between parts and textures. RCC8 relations are used to describe the containing relations between parts and textures. If a part has a texture, its representation includes the texture representations and RCC8 relations.

If the hierarchical level is empty, a special part is added, called "**EmptyPart**". with only one fact:

(emptyEncoding EmptyPart-1)

This facilitates the learning process introduced next.

3.5.2 Hierarchical Analogical Learning on Parts

This section presents a novel approach to use analogical learning hierarchically on the representations just described. Our goal was to improve performance and speed up the retrieval process while maintaining the data efficiency of traditional analogical learning.

Each sketched object category is modeled with three different generalization pools, one gpool for each level of representation. Thus, for example, the concept of Airplane would be represented by the contents of three gpools, **AirplaneGpoolLevel1**, **AirplaneGpoolLevel2**, and **AirplaneGpoolLevel3**. Training consists of constructing part representations for the three hierarchical levels for each example and adding them to the appropriate gpool. Ideally, each generalization in the gpools represents a common part of its objects.

Classification of a new example is performed by first computing the part representations in three levels for it as shown in Algorithm 4, and then using the Hierarchical Analogical Retrieval algorithm described in Algorithm 5. It works as follows: each part representation in the first level is used as a probe for MAC/FAC, with each Level-1 gpool being used as a case library. Thus, for each category, a set of items with corresponding scores are retrieved. The top *K* categories, i.e., those with the highest average

Algorithm 5: Hierarchical Analogical Retrieval

Input: The three-level part representations of a sketch: P1, P2, P3, Concepts: C[1,…,n]

Parameters: The numbers of categories retrieved in each level: K, Q, V

Functions:
Gpools (Ci, N): return the gpools of all concepts in Ci at level N.
MAC/FAC (p, G): use MAC/FAC to retrieve an item from gpool G with the probe p.

Output: Classification Category

```
1:    Define Function LevelRetrieval (Ci, N, P):
2:        O = []
3:        G = Gpools (Ci, N)
4:        For Gi in G[1,…, n] do
5:            T = []
6:            For pi in P do
7:                T.append(MAC/FAC(p1, Gi))
8:                O.append(average(T))
9:        return O
10:   O1 = LevelRetrieval (C, 1, P1)
11:   S1 = sort(O1)[:K]
12:   O2 = LevelRetrieval (S1, 2, P2)
13:   S2 = sort(O2 )[:Q]
14:   O3 = LevelRetrieval (S2, 3, P3)
15:   S3 = sort(O3) [:V]
16:   For class Ci in S3 do
17:       r1 = S1[Ci] * len(S1[Ci])
18:       r2 = S2[Ci] * len(S2[Ci])
19:       r3 = S3[Ci] * len(S3[Ci])
20:       S3[Ci] = r1 + r2 + r3
21:   result = sort(S3)[0]
22:   return result
```

scores of parts coming out of MAC/FAC, are used as an initial pool of candidates (Alg. 5 lines 10-11). Next, each part representation of the second level is used as a probe for MAC/FAC, but with each Level-2 gpools corresponding to the *K* categories found in the level-1 retrieval being used as a case library (Alg. 5

lines 12-13). The top *Q* categories with the highest average scores of all parts form the pool of candidates for the third level retrieval, where each part representation is used as a probe for MAC/FAC with each Level-3 gpool corresponding to the *Q* candidates retrieved by the previous step being used as a case library, keeping the top *V* candidates for this step (Alg. 5 lines 14-15). Particularly discriminative information can appear at any level; hence it is useful to consider scores from all three levels instead of only the last level. Also, ideally, each part probe should retrieve a different item from gools of concepts. Thus, it combines information across levels to compute a final score for classification. The final score for each of the *V* candidates is found by adding together the product of the average score and the number of distinct retrieved items computed for that category from all three levels (Alg. 5 lines 16-22). This ranked list provides the classification answers.

3.5.3 Experiments

Similar to the experiments of GAL encoding, I performed experiments on two sketched object datasets, the TU Berlin dataset and the Coloring Book Objects dataset, comparing PHAL with several baselines. On the TU Berlin dataset, I compared PHAL with several deep learning baselines, to evaluate whether PHAL can achieve competitive performance. On the Color Book Object dataset, I compare PHAL with

Scenario	Accuracy
[Eitz et al., 2012]	56.00%
[Li et al., 2013]	61.50%
[Hochreiter et al., 1997]	62.35%
[He et al., 2016]	69.35%
[Lin, et al., 2019]	73.95%
[Lin et al., 2020]	76.30%
[Yu et al., 2016]	77.95%
PHAL	69.85%

Table 5: Results on TU Berlin dataset

GAL, and I aim to show that PHAL outperforms GAL with same number of training samples. Thus, PHAL has strong understandability, high data-efficiency, and good performance. Results show that our approach can achieve competitive or state-of-the-art performance.

TU Berlin Dataset

The TU Berlin dataset contains 250 object categories. For each category, there are 80 hand-drawing sketches, for a total of 20,000 sketches. I used the popular training/testing splits, where each category has 16 testing sketches, and the rest of the sketches are training samples. The authors of the dataset performed a perceptual study and found that humans could correctly identity the object category of a sketch 73% of the time. I performed hyper-parameters searching, settling on 0.8 as the SAGE assimilation threshold and 0.2 as the cutoff probability for all three encoding levels. On the full dataset, the numbers of categories kept at each level are 20, 10, 5. Our approach takes about 15 seconds to encode a sketch from TU Berlin dataset and the training process takes about 8 hours (each object has multiple parts).

I compared PHAL with existing baselines including traditional machine learning methods and recent deep learning models. Table 5 presents the results. Our system achieves 69.85% accuracy on the full dataset, which is competitive compared to the baselines. The performance is only lower than Sketch-A-Net (SAN) [Yu et al., 2016], TCNet [Lin, et al., 2019], and SketchBERT [Lin et al., 2020]. However, these three deep learning approaches either perform data augmentation to generate more training data or leverage models pretrained on a large amount of data before fine-tuning on Berlin dataset. PHAL only

Figure 28: Our system predicts correct labels on the left two examples, but Sketch-A-Net (SAN) does not. Our system predicts incorrect labels on the right two examples, but Sketch-a-Net gets correct answers.

looks over all training data once instead of multiple epochs like deep learning models, which provides evidence for training efficiency of our approach. Figure 28 shows some examples comparing PHAL and SAN. In the left two examples, our method generates correct labels, but SAN does not. In the right two examples, our method does not predict correct labels, but SAN does. The training set does not have a front view of the airplane and the bear is a bear face. Thus, data augmentation might be why SAN makes correct predictions in these cases. This evidence suggests that hierarchical analogical learning can achieve reasonable accuracy with good data efficiency. Moreover, our method is incremental, an advantage that is not tested by batch-oriented datasets, but crucial for integrating models into many applications.

The Coloring Book Objects Dataset

I test the PHAL on the Coloring Book Objects dataset (CBO) to show data efficiency. I use the same cross-validation method as evaluating GAL. At each round out of ten, a random image in each category is used as the testing sample and the other nine images in each category are used as training samples. Then, I compute the average accuracy of the ten rounds. After a hyper-parameter search, I used 0.7 as the assimilation threshold and 0.2 as the cutoff probability for all three levels. During hierarchical analogical retrieval, I keep the top 10 categories in level-1, the top 5 categories in level-2, and the top 3 categories in level-3.

Given the small number of examples, PHAL outperforms the deep learning models. I tested two different deep learning models as baselines. First, models with fewer parameters seem more likely to

Scenario	Accuracy
LeNet-5	5.26%
ResNet50	10.53%
ResNet50 (pretrained)	21.05%
[Chen et al., 2019]	27.48%
PHAL	37.37%

Table 6: Results on CBO dataset

converge, given the small number of training examples in this dataset. Consequently, I chose LeNet-5 [LeCun et al., 1998] rather than more complex CNN models and trained them from scratch. The second model tests the ResNet50 baseline which is the same as in the TU Berlin experiment. I evaluated both the pre-trained ResNet50 and raw model trained from scratch. After hyper-parameter searching, LeNet-5 only achieved accuracy about chance. ResNet50 achieved 10.53% accuracy if I trained it from scratch.

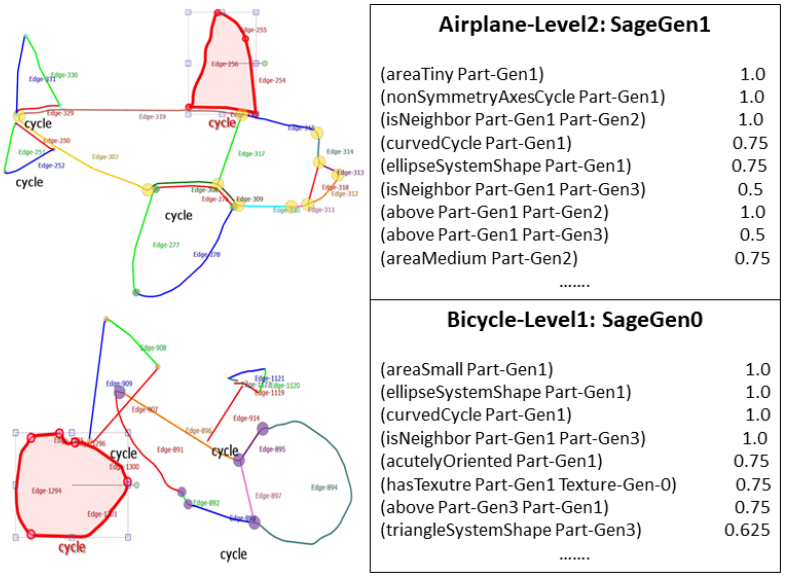


Figure 29: The top is a generalization in Airplane-level2 gpool and a possible corresponding part (in red). The bottom is a generalization in Bicycle-level1 gpool.

Pretrained ResNet50 achieved 21.05% accuracy, which is still lower than our approach.[6] I also compare our approach with GAL. Table 6 shows the overall results for each model. These results demonstrate that our novel hierarchical analogical learning outperforms deep learning models and the traditional analogical learning, providing a new SOTA for this dataset. This provides additional evidence that PHAL is good at data-efficient learning and can capture hierarchically structured information of sketched objects.

3.5.4 Understandability

Analogical generalization provides understandability because the qualitative relational representations used are very similar to human visual structure and are easily tied to language. Each generalization or outlier in a generalization pool represents a disjunct of the model and can be explored by users. For example, Figure 29 shows

[6] As there are not public source codes for SketchBERT, I did not test it on CBO dataset.

two generalizations from the Airplane and Bicycle gpools. In each generalization, the left is a part visualization in an sample from the generalization and the right is the learned descriptions, with the probability for each statement. The entities in the descriptions are skolems introduced by SAGE, e.g., <*type*>-*Gen*-<*index*> indicates a generalized entity construct from a part or a texture as <*type*>, with <*index*> being an integer id within that generalization. The top generalization is from the Airplane level-2 gpool, which has 36 generalizations and 19 outliers. In the top example, the left is a case of the upper wing of an airplane (the part is in red color). Based on learned statements, the upper wing is a relatively small, non-symmetric and curved edge-cycle, which connects with two other parts. Similarly, the bottom example is a generalization from Bicycle level-1 gpool, which has 41 generalizations and 26 outliers. The left figure is a case of a front wheel of a bicycle. The wheel is usually small and like an ellipse. It is acutely oriented and sometimes has textures. These representations easily map to natural language, providing intuitive explanations. Such descriptions are hard to extract from deep learning models. Being able to understand the generalizations and outliers could help trainers (or systems using the models) to guide active learning.

3.6 Conclusion and Future Work

I have shown that analogical learning over relational representations is a viable and promising path for sketch recognition. Our approaches are based on human-like encoding schemes and they achieve solid results with a small number of training examples on different types of sketches. I note that deep learning systems require from 60,000 examples [Ciresan et al. 2011] to 420,000 examples [Ciresan et al. 2012] over hundreds of epochs to achieve the performance that they report on MNIST. Moreover, the Coloring Book Objects dataset illustrates that deep learning models have poor performance with small numbers of training examples, whereas analogical generalization, despite the high variability of the examples, performs much better.

This chapter introduces two novel encoding schemes and a cognitive simulation. Both encoding methods show high data efficiency learning when used with analogy on these datasets. Thus, geon-based encoding and part-based encoding can describe object geometric information and analogical learning can easily be applied on them. GAL approach was originally published in [Chen and Forbus, 2019]. Also, in

PHAL, I use analogical learning hierarchically to improve the performance of traditional analogical learning like GAL while maintaining data-efficiency.

While the data efficiency of analogical generalization is already very encouraging, I plan several lines of future work to improve it further. First, I plan to explore dynamic attribute selection, using statistics gleaned from SAGE to control the choice of attributes in subsequent encoding. Secondly, I plan on using near-miss learning [McLure et al. 2015a], which provides additional discrimination to analogical generalization and has been beneficial in other datasets.

4 Hybrid Primal Sketch

The previous chapter showed that on sketched objects, analogical learning over qualitative representations has high data efficiency, strong understandability, and can achieve competitive or better performance than deep learning systems. However, sketches are simpler than natural images. Sketches do not have the same kind of fractal visual structure found in natural images, nor complex lighting effects, nor color. This chapter focuses on adapting the qualitative/analogical approach to high-level vision to natural images. This involves understanding color, lightness, and complex visual structure, including significantly more noise than sketches. I introduce a hybrid system, the *Hybrid Primal Sketch Processor* (HPSP), that combines deep learning models and qualitative representations. Deep learning models are used for low-level perception, and various encoding schemes are used to generate qualitative representations for perception based on the requirements of different tasks. These qualitative representations can be used with analogical learning.

I first present an overview of HPSP architecture. Then, I describe how to adapt the HPSP to two visual understanding tasks, *visual relationship detection* and *visual question answering*. For both tasks, it uses Mask-RCNN to detect objects and generate input representations for CogSketch, which then constructs visual representations for each task using the same kinds of techniques used in the previous chapter. These two tasks are presented in sections 4.2 and 4.3.

4.1 Hybrid Primal Sketch Overview

The HPSP extends the ideas in Marr's [1982] Primal *Sketch*, which constructed edges as a form of initial symbolic representation that also included metric information, as a foundation for representations for scene understanding. Marr's Primal Sketch contained a *raw primal sketch* and a *full primal sketch*. The raw primal sketch encoded initial visual components such as edges and blobs as symbolic representations. The full primal sketch performed grouping procedures on the representations from raw primal sketch and generated representations for larger semantic components such as objects or shadows. The Hybrid Primal

Sketch Processor extends these ideas by adding deep learning components to produce information about objects in the image, including their boundaries and semantic category. The boundaries and category information provides inputs to CogSketch – each object is a glyph, with the ink of the glyph being the boundary. The same kinds of qualitative visual representations discussed in previous chapters can then be used to build further representations needed for high-level vision tasks.

Figure 30 presents an overview of HPSP architecture. The HPSP architecture has three stages. The first stage is low-level visual processing. In this stage, HPSP uses deep learning models to perform low-level perception. HPSP can use different deep learning models based on encoding strategies for different tasks. For example, if I only want to encode the locations of objects, Faster-RCNN is a good choice because this model detects object bounding boxes. Suppose the encoding scheme requires representing geometric features of objects. In that case, it might need to use Mask-RCNN to detect the contours of objects and even perform edge detection to find object textures.

The second stage is high-level visual encoding. In this stage, HPSP uses several encoding schemes to generate qualitative representations for the inputs. HPSP has two representation levels for images: *object-level representation* and *attribute-level representation*. Object-level representations only encode the spatial information on bounding boxes of objects, including locations and their categories. Attribute-level representations encode both the spatial information of bounding boxes and the visual

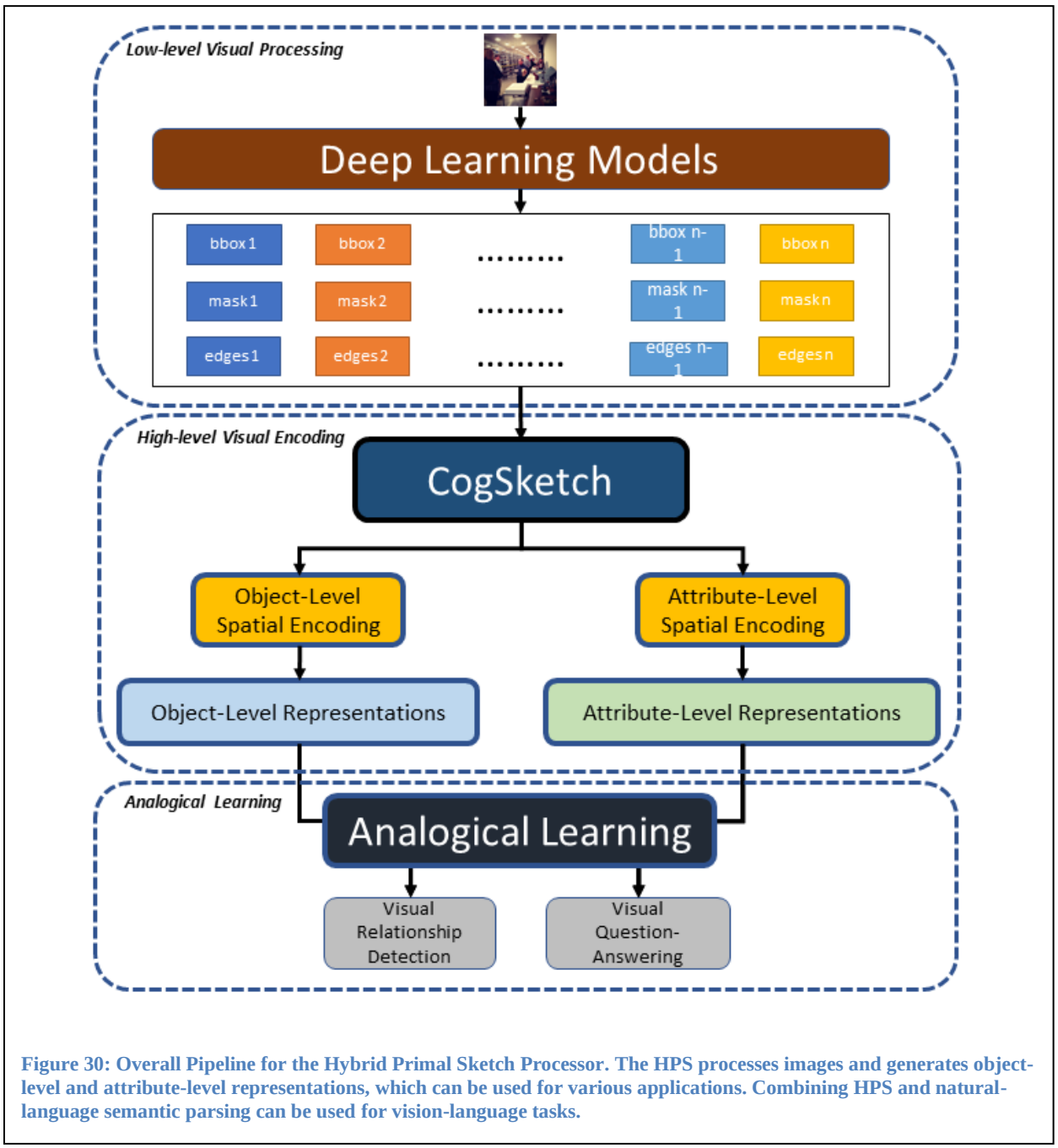


Figure 30: Overall Pipeline for the Hybrid Primal Sketch Processor. The HPS processes images and generates object-level and attribute-level representations, which can be used for various applications. Combining HPS and natural-language semantic parsing can be used for vision-language tasks.

features inside the bounding boxes, including object boundary, edges, color, etc. Thus, attribute-level representations have finer details than object-level representations. Both representations can be adapted to pair-level encoding or scene-level encoding. Pair-level encoding aims to describe the visual features of a pair of objects. A scene-level encoding aims to describe the visual features of multiple objects in an image. For example, suppose attribute-level representations are used to describe the scene information of an image, the qualitative representation should include the spatial relations between image objects, and the attribute information of each object. CogSketch encodes object geometric features (encoded via PHAL and GAL encoding schemes), color features, and material features.

The last stage is analogical learning. Analogical learning over the qualitative representations generated from the second stage can be used on various tasks. Also, HPSP can either use traditional analogical learning or hierarchical analogical learning. In the following sections, I introduce two visual understanding tasks and how to adapt the HPSP for these tasks.

4.2 Visual Relation Detection and Visual Question Answering

Visual relation detection and visual question answering are two popular visual tasks. Both tasks require AI models to perform perception on images and understand the scenes they depict.

The goal of the visual relation detection task is to detect objects in images and predict semantic relations between them. For example, we could describe Figure 31 (a) as “a man holds a plate” and Figure 31 (b) as “a girl wears a dress”. Visual relation detection is thus an important task for artificial intelligence and computer vision.

Figure 31: Two image samples from Visual Relationship Dataset.

In recent years, many approaches have been proposed for visual relation detection, based on recognizing objects, constructing a *scene graph* of visual relationships between objects, and then predicting semantic relationships based on the scene graph and object identifications. The visual relations between two objects are written as a triple < Subject, Relation, Object>, where the detected objects can either be the Object or the Subject in the triple. Most systems use a two-step pipeline that detect objects from the image first and then classifies the relations between objects. For example, in Figure 31 (a), the model needs to detect the person and the plate, and then it should predict a relation "holds" exists between them. In the first step, almost all methods use deep learning detectors, either off-the-shelf detector [Lu et al., 2016; Zhuang et al., 2017; Dai et al., 2017] or fine-tuning with the relationship datasets [Xu et al., 2017]. In the second step, the visual relation task is usually regarded as a classification task with a relation is predicted for each pair of objects. Various architectures have been used for encoding and classification. For example, [Zellers et al., 2018] uses stacks of LSTMs to encode the features of object pairs. [Yang et al., 2019] uses an attentional graph convolutional network for encoding the context information of objects. Most of the deep-learning models operate end-to-end, combining the two steps into one single

neural network. They have broad coverage on the input data, especially on visual data. However, given the single neural network, these models have some problems such as low training efficiency and lack of understandability. The HPSP approach uses the two-step scheme, with deep learning modules identifying objects for which a scene graph is constructed using CogSketch, with the second step being performed via analogical generalization over the qualitative representations computed by the first step. Thus HPSP gets the same coverage advantages of deep learning models, but provides the understandability of probabilistic qualitative models. Moreover, the HPSP only goes over all data once, compared with multiple epochs used in deep learning.

Another task is visual question-answering (VQA). In VQA tasks, AI models answer questions based on the information in images. This task requires models to ground language inputs in images, constructing a shared representation suitable for reasoning. VQA is typically modeled as a generation task, where the model constructs the correct answer, or a classification task, where the model selects the correct choice from a candidate answer set. Deep learning models have achieved some impressive results on VQA tasks. For example, [Anderson, et al 2018] extracted visual features from images using a pre-trained CNN and used an LSTM model to encode the questions as embeddings. Then they designed bottom-up and top-down attention to combine the visual features and word embeddings for answer prediction from candidate answers. Researchers have recently explored using large-scaled pre-training models with transformers as the backbone for vision and language tasks [Yang et al.; Liu et al., 2019; Radford et al., 2021]. Given a pretraining corpus containing many image-text pairs, the large-scaled models are pre-trained with novel training objectives to learn the statistical alignments between visual features and language inputs. Then, the pre-trained models are fine-tuned on downstream vision and language tasks. This setting has an efficient learning ability on downstream tasks and improves accuracy. However, understandability is also essential for VQA tasks, since people want to know why an answer was generated (or selected). Pure deep learning models to date have failed to provide understandability, whereas the HPSP approach does.

Next we discuss how the HPSP is used on each task in turn.

4.3 Visual Relation Detection using HPSP

This section introduces how to use the HPSP for visual relation detection tasks by combining deep-learning models and analogical generalization. Object bounding boxes and masks are detected using deep-learning models and analogical generalization over qualitative representations is used for visual relation detection between object pairs. Experiments on the Visual Relation Detection dataset indicate that our hybrid system gets comparable results on the task and is more training-efficient and explainable than pure deep-learning models. This approach was originally published in [Chen and Forbus, 2021].

The HPSP architecture is adapted to this task as follows. The HPSP uses a two-step pipeline of object detection followed by pairwise relation detection (Figure 32). Given an image, object detection uses deep learning models to detect bounding boxes, instance segmentation, and their categories of objects (low-level visual processing). Object bounding boxes provide positional information, object masks give pose information and object categories provide the semantic category of objects. The second

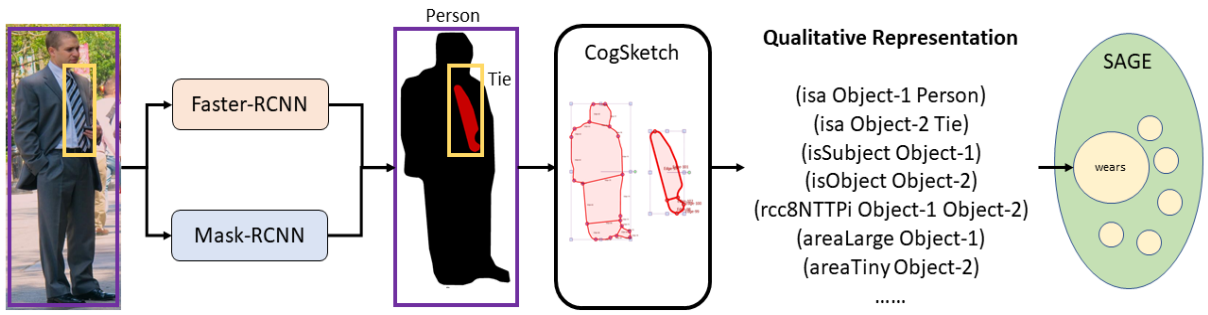


Figure 32: The pipeline overview to learn visual relation classification. Given an image, our system uses Faster-RCNN and Mask-RCNN to detect bounding boxes, instance segmentations, and categories of objects. Then, detection results are imported into CogSketch. CogSketch computes qualitative representations for each pair of objects based on an encoding scheme, including object encodings and pair encodings. Generated qualitative representations are added into targeted generalization pools in our analogical generalization model, SAGE. SAGE learns probabilistic relational generalizations, which are used for classification [Chen and Forbus, 2021].

step automatically computes qualitative visual representations based on our encoding scheme for pairs of objects (high-level visual encoding). During training, analogical generalization is used to learn analogical

models for semantic relations, and during testing, analogical retrieval is used to classify the relationship for each pair of objects. We discuss each step in turn next.

4.3.1 Low-level Visual Processing

In the first step, we need to detect objects from images. The bounding box, mask, and category of each object are generated to provide enough information for encoding object pairs. We use Faster-RCNN [Ren et al., 2015] with VGG16 backbone to detect the object bounding boxes and categories. Faster-RCNN has two modules for object detection. The first module is a deep fully convolutional neural network that proposes possible regions, and the second module is the Faster-RCNN detector that performs object recognition on each proposed region. Given an image, Faster-RCNN generates a set of bounding boxes. Each box has an object prediction label and confidence score. The bounding boxes are filtered with a threshold to generate the final set. In our experiments, Faster-RCNN uses the VGG16 backbone that is pre-trained on COCO dataset [Lin et al., 2014] and is fine-tuned on targeted visual relation datasets.

To detect object masks, we utilize Mask-RCNN model [He et al., 2018] with Resnet-50 as the backbone for instance segmentation. Mask-RCNN also has two modules. The first module is similar to Faster-RCNN, proposing regions for objects. The second module uses proposed regions to generate instance segmentation masks for objects. As most of the visual relation datasets do not have instance segmentation annotations, we directly detect object masks using a Mask-RCNN trained on the COCO dataset.

During testing, Faster-RCNN detects the object bounding boxes and their categories. Mask-RCNN detects object segmentations and the bounding boxes of corresponding segments. Here, the Faster-RCNN is trained on each visual relation dataset, but Mask-RCNN is pretrained from COCO dataset. As the labels from COCO dataset may not be same as the visual relation datasets, we merge the detection results from Mask-RCNN to Faster-RCNN based on the overlap ratio of bounding boxes. HPSP computed the intersection of union (IoU) between all the detected bounding boxes detected from Faster-

RCNN and detected Mask-RCNN. If the IoU is larger than 0.7, the corresponding instance segments from Mask-RCNN are combined with the bounding box from Faster-RCNN to construct the final detection result. Otherwise, we regard the bounding boxes as the masks of the objects. After this step, each detected bounding box from Faster-RCNN has a paired mask, which is either detected from the Mask-RCNN or is the bounding box itself. We describe how the detection results are used for pairwise relation detection.

4.3.2 Pairwise Relation Detection

During training, we use the ground truth triples, i.e. <Object, Relation, Subject>, as training data for analogical generalization. Given the detected bounding boxes, categories, and segmentation of the Object and Subject of the triple, our system is trained to learn analogical models of the Relation, based on automatically constructed qualitative visual representations. During inference, analogical retrieval is used

Algorithm 6: Object-pair-based Encoding (OPE)

Input: A pair of object glyphs, O1 and O2.

Functions:
EncodeSpatialRelations (O1,O2): encode the spatial relations between the pair of objects, O1, O2.
GAL (O): encode an object O with GAL encoding algorithm.
Category (O): encode the category information of an object O.

Output: A list of encoded statements for object pairs, E

```
Define Function Obj-Rep (O1, O2):
    EO = []
    C1 = Category (O1)
    C2 = Category (O2)
    F1 = GAL (O1)
    F2 = GAL (O2)
    EO.append( C1 )
    EO.append( C2 )
    EO.append( F1 )
    EO.append( F2 )
  return EO
Define Function Spa-Rep (O1, O2):
        ES = []
    S = EncodeSpatialRelations (O1, O2)
    ES.append( S )
  return ES
E = Obj-Rep (O1, O2) + Spa-Rep (O1, O2)
return E
```

to identify the relationship with highest score for each pair of entities detected from the detection stage. To model the information between two detected objects, we first describe our visual encoding scheme for generating the symbolic representations. Then we describe how analogical learning for relation detection works. Finally, the inference process is presented.

Symbolic representations for object pairs: For each pair of objects, we build symbolic representations for each object and the spatial relations between them via the *object-pair-based encoding scheme* (OPE) as described in Algorithm 6.

OPE has two parts: *Obj-reps and Spa-reps*. Obj-reps consists of facts for each object generated from their masks and categories. Spa-reps consists of the spatial relations between the pair of objects generated from their bounding boxes.

Attribute	Description	Example
Cross-sectional curvature	Whether the edges of the shape are straight or curved.	(allCurved EC-1)
Edge concavity	Whether the shape has concave edges.	(hasConcavedEdge EC-1)
Shape Estimation	The simple geometric shape closest to the shape.	(ellipseSystemShape EC-1)
Rectangularity	How much the shape looks like a rectangle.	(highRectangularity EC-1)

Table 7: Detailed description of four attributes that describe geon shapes.

In Obj-reps, object category and pose are encoded. The semantic category is produced by using lexical lookup from the word produced as the category label to map that into the OpenCyc ontology (Alg. 6 line 3-4). We use the predicate **isa** from OpenCyc to describe an object's category, for example,

(isa Object-1 Person)

indicates that the object *Object-1* is an instance of the category *Person*.

For object pose, we use the segmentation algorithm from GAL to generate a set of segments on object masks. Recall that in GAL the geometric features of each segment are described. In this step, each segment is represented using four different attributes: *cross-sectional curvature, edge concavity, shape estimation, rectangularity*. As we only encode the object contour (mask) and ignore the structures inside the object, four geometric features are enough to distinguish them (Alg. 6 line 5-6). Table 7 describes each attribute. Connection and positional relations are computed between each pair of segments. For each segment in an object, we use a predicate **isSegment** to indicate that the segment is part of the object. For example,

(isSegment Segment-1 Object-1)

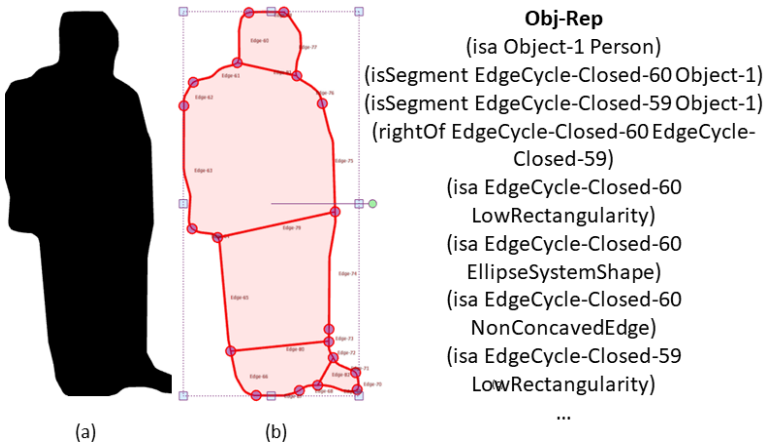


Figure 33: (a) the mask of a Person. (b) the geon segmentation of the mask (c) Obj-reps for the object. [Chen and Forbus, 2021]

Figure 33 shows an object mask labeled as Person, the segmentation result from geon-based encoding, and its corresponding Obj-reps. Obj-reps are computed for both Object and Subject of a triple.

For each pair of objects, each has a role in a relation triple, either Object or Subject. We use the unary predicates **isObject** and **isSubject** to indicate the roles for each object, for example,

(isObject Object-1)

(isSubject Object-2)

Spa-reps encode three types of spatial information between the pair of bounding boxes: RCC8 information [Cohn, 1996], positional information, and size information. RCC8 is widely used in qualitative spatial reasoning to describe topological relationships between regions. We use six CogSketch positional relations: **above, rightOf, enclosesHorizontally, enclosesVertically, centerAbove** and **centerRightOf** (as described in Section 2.3.2). The first four predicates describe the positional information on bounding boxes and the last two describe the positional information for their center points (Alg. 6 line 12-18).

Size information is encoded with four CogSketch predicates: **areaTiny, areaSmall, areaMedium** and **areaLarge**. The larger box in the pair is encoded as **areaLarge** and the smaller box is encoded based on the size relative to the larger box. If the area of smaller box is less than ¼ the area of the larger box, it is encoded as **areaTiny**. If the area of smaller box is between the ¼ and ½ the area of the larger box, it is encoded as **areaSmall**. Similarly, if it is between ½ and ¾ the area of larger box, it is encoded as **areaMedium** and above ¾ is encoded as **areaLarge**. The Obj-reps and Spa-reps are combined to create the relational representation for an object pair. Next, we describe how these representations are used in analogical learning.

For visual relation detection, the model needs to compute a relation category for a pair of objects. Since the geon representations have many facts when object contours are complicated and each gpool has many cases, we use a two-level hierarchical process for relation classification to speed up the retrieval and scope relations to improve performance. In the first level, only object categories in Obj-reps and the Spa-reps are used to provide a rough estimation for relations. We call this the *rough case* for an example. In the second level, full Obj-reps and Spa-reps are combined to predict the final relation from the estimated relations in first step. We call this the *full case* for an example. For every relation, there are two gpools, one for rough cases and one for full cases. During training, each relation triple has its rough and full cases computed, which are added to the appropriate gpools.

Relation Detection: Given a pair of objects detected in an image by the first step, we use OPE to build its rough case and full case, as per above. To detect a relation, we use a modified analogical retrieval method, Layered Analogical Retrieval (LAR). The LAR algorithm is shown as Algorithm 7. The rough case is used as a probe to MAC/FAC, with all gpools for rough cases serving as the case library (Alg. 7 line 6-7). The relations corresponding to the top five retrievals are used as filters for retrieval over full cases. That is, the full case is used as a probe with MAC/FAC over the union of all full case gpools whose relations were retrieved in the prior step. The relation associated with the highest similarity score

Algorithm 7: Layered Analogical Retrieval (LAR)

Input: Rough case and full case of a sketch: P1, P2, Analogical Gpools: G[1,…,n]

Parameters: The numbers of categories retrieved in each layer: K, Q

Output: Classification Category

```
1:    Define Function LayerRetrieval (A, P):
2:        O = []
3:        For Gi in A[1,…, n] do
4:            O.append(MAC/FAC(P, Gi))
5:        return O
6:    O1 = LevelRetrieval (G, P1)
7:    S1 = sort(O1)[:K]
8:    O2 = LevelRetrieval (S1, P2)
9:    S2 = sort(O2 )[:Q]
10:   result = sort(S2)[0]
11:   return result
```

retrieved by MAC/FAC is assigned to the pair of entities as its classification (Alg. 7 line 8-10). This process is run over every pair of detected objects in the image.

Models	Recall@50	Recall@100
VRD (Lu et al., 2016)	47.87	47.87
VTransE (Zhang et al., 2017)	44.76	44.76
Zoom-Net (Yin et al., 2018)	50.69	50.69
LK (Yu et al., 2017)	55.16	55.16
CAI+SCA-M (Yin et al., 2018)	55.98	55.98
MMLFM-LC (Ma et al., 2019)	56.65	56.65
Ours	**52.38**	**52.38**

Table 8: Results of PREDT condition on VRD dataset [Chen and Forbus, 2021]

4.3.3 Experiments

We evaluate the HPSP on the Visual Relationship Dataset (VRD) [Lu et al., 2016]. We show that our model can get competitive results with lower training cost. Detecting a visual relational tuple involves classifying both object entities, the predicate between them, and the bounding boxes of both entities. Consequently, the performance of models relies on both the accuracy of object entity detectors and the visual relationship classifiers.

Following [Lu et al., 2016], we measure two conditions to evaluate the performance of our model. The first condition is predicate detection (PREDT). In PREDT, the input is an image, a set of localized objects in the image and their labels. The task is to predict a set of possible predicates between pairs of objects. This condition shows how relation detection via analogical learning performs on ground truth inputs. The second condition is relationship detection (RELDT). In RELDT, the input is only an image. The task is to output a set of triples <Object, Relation, Subject> and localize both entities in the image having at least 0.5 overlap with their ground truth boxes simultaneously. This condition evaluates

Models	Recall@50	Recall@100
VTransE (Zhang et al., 2017)	14.07	15.20
SA-Full (Peyre et al., 2017)	15.80	17.10
Zoom-Net (Yin et al., 2018)	21.37	27.30
CAI+SCA-M (Yin et al., 2018)	22.34	28.52
KL (Yu et al., 2017)	22.68	31.89
Large-Scale VLU (Zhang et al., 2019)	26.98	32.63
Ours	16.12	18.41

Table 9: Results of RELDT condition on VRD dataset [Chen and Forbus, 2021].

how the whole pipeline performs on visual relation detection. We use the same evaluation metrices Recall@50 and Recall@100 as in [Zhang et al., 2019]. The details of VRD dataset and implementations are introduced below.

Visual Relationship Detection Dataset

This dataset contains 5,000 images with 100 object categories and 70 predicates. There are 37,993 triple combinations total. We follow the popular train/test split, using 4,000 images for training and the other 1,000 images for testing. We trained the Faster-RCNN model using the method from (Zhang et al., 2019). For Mask-RCNN, since the VRD dataset lacks instance segmentation annotations, we directly use the checkpoint pre-trained on COCO dataset[7]. In SAGE, we use 0.8 for the assimilation threshold and 0.2 for the cutoff threshold. Table 8 shows the results on PREDT task compared with [Lu et al., 2016]. Table 9 shows the results compared with other state-of-art models on RELDT dataset. Besides the results in the two tables, we also compute Recall@1 of our model, since that is a more rigorous test. Our system achieves 32.26 for Recall@1 in PREDT task and 8.91 in RELDT task.

[7] We use the pre-trained model from https://github.com/facebookresearch/detectron2.

Discussion

In Table 8, we compare our system with several existing models. From the results, our system outperforms three baseline models. In PREDT task, the ground truth object bounding boxes and categories are passed into model. The results show that our hybrid system has good performance when the object detection results are good. In Table 9, our results are better than VTransE and SA-Full. In RELDT task, detected object bounding boxes and categories from object detection model are used. Thus, the relation predication results depend on both the performance of object detections and relation classification. The results suggest that our hybrid system has reasonable adaptation on noisy object detection results.

Indeed, although our model does not outperform all baselines, we use much less training to achieve this performance. Firstly, all baselines use pre-trained object detector models in their first stage, which has a similar cost as our model. However, using analogical learning, our model learns the generalization pools in the second stage with only one epoch on the whole training dataset. All other deep learning baselines require 7 to 30 epochs on the whole training dataset to converge. Thus, our model uses less training time to achieve the results. Also, analogical learning does not require to use of expensive hardware resources such as GPUs, but all deep learning baselines need to use GPUs to speed up the training process. Besides less training cost, analogical learning produces models that are easier to understand than deep learning models. In the next section, we discuss understandability further.

4.3.4 Understandability

The use of analogical learning for relation detection provides strong understandability. As noted previously, the contents of SAGE generalization pools consist of schema-like descriptions which can be easily understood by people. For example, Figure 34 shows the descriptions of the largest generalizations in the Above and Wears gpools. In the generalization of Above**, (centerAbove O1 O2)** has probability score 1.0. O1 and O2 are the skolems for two different objects. This fact shows that, in many cases for this relation, the center of one object bounding boxes is above the center of the other object. Also, this generalization reveals information about common object types, e.g. that objects of type Sky have the relation Above with objects of types Building, Tree, Mountain, etc. Similarly, objects of type Sky tend to have **(areaLarge O-1)** and all other objects have **(areaTiny O-2),** which means Sky is much larger than other objects. In the generalization of Wears, one of the objects is Person and the other object is clothes for lower body, such as Pants, Jeans or Shorts. Therefore, Person has a large area and cloth objects have a small area. The RCC8 relation PO (i.e. Partially Overlaps) has a high score in this generalization, which means the two objects intersect each other. These probabilistic generalizations provide new insights,

Above: SageGen0	
(centerAbove O-1 O-2)	1.0
(isObject O-2)	1.0
(isSubject O-1)	1.0
(rcc8-PO O-1 O-2)	1.0
(areaLarge O-1)	0.9802955
(isa O-1 Sky)	0.9359606
(above O-1 O-2)	0.7832512
(areaTiny O-2)	0.5270934
(isa O-2 Building)	0.2364532
(isa O-2 Tree)	0.0985222
(isa O-2 Mountain)	0.0837438
……	

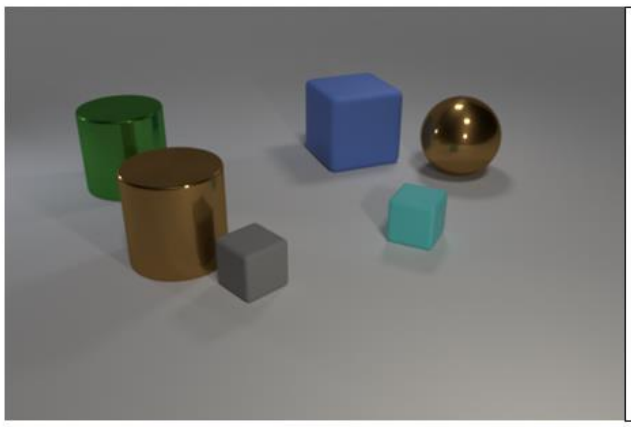
Wears: SageGen0	
(areaLarge O-2)	1.0
(centerAbove O-2 O-1)	1.0
(isa O-2 Person)	1.0
(isObject O-1)	1.0
(isSubject O-2)	1.0
(rcc8-PO O-1 O-2)	0.9674796
(areaSmall O-1)	0.6585366
(isa O-1 Pants)	0.4146341
(isa O-1 Shoes)	0.2926829
(isa O-1 Jeans)	0.1463414
(isa O-1 Shorts)	0.10569106
……	

Figure 34: Scheme-like descriptions and corresponding probabilities in largest generalizations of the Above and Wears gpools [Chen and Forbus, 2021].

including possibly into dataset bias. Moreover, one interesting possibility is tuning learned knowledge, via trainers manually editing facts, something which is difficult for deep learning models.

4.3.5 Conclusions about VRD

This section showed how to adapt the HPSP architecture to build a hybrid system for visual relation detection by representing the object information and spatial information between objects. Our system achieves competitive results by combining deep learning models, qualitative representations, and analogical learning. Results on the PREDT task indicate that, given accurate object detections, analogical learning is a promising approach to detect relations in images. Furthermore, analogical learning is more efficient for training than deep learning and has better understandability. People can easily explore the learned generalizations to understand the high-probability generalizations and outliers, which provides a more solid foundation for building reliable and human-like visual systems. Results on RELDT also shows that analogical learning is flexible enough to combine with other methods. These results indicates that HPSP is a promising architecture for combining deep learning models and qualitative representations on visual understanding tasks. In the next section, I describe how to use the HPSP on another visual task and further explore the effectiveness of the HPSP.



Question: Are there more big green things than large purple shiny cubes?

Program:

1. (Filter_Size Large Scene)
2. (Filter_Color Green Result_1)
3. (Count, Result_2)
4. (Filter_Size Large Scene)
5. (Filter_Color Purple Result_4)
6. (Filter_Material Metal Result_5)
7. (Filter_Shape Cube Result_6)
8. (Count, Result_7)
9. (Greater_Than Result_3, Result_8)

Figure 35: An example in CLEVR dataset, including an image, a text question and its program.

4.4 Visual Question Answering

I have shown how to use HPSP architecture on the visual relationship detection task. This section focuses on a more complicated visual understanding task, visual question answering. This task requires models to extract scene information from images and use that information to answer questions. Thus, models should generate comprehensive and correct representations of images for questions. Specifically in this section, I describe how HPSP is used to generate scene representations on CLEVR dataset (introduced in section 2.7.5) and generated representations are applied for QA.

The objects in the CLEVR universe are simple geometric shapes. Figure 35 shows an example. It includes an image, a question, and its corresponding program to compute the answer. Following the HPSP architecture, deep learning models are used to perform low-level perception and construct qualitative representations for visual scenes on the perceptual results. The questions in CLEVR require models to understand multiple features of objects including color, material, and size. For color, the average RGB values are converted to the nearest color entity defined in NextKB. For instance, the two large cylinders in Figure 35 are Green and Brown. For size information, CogSketch is used to compute relative size information. For material information, analogical learning over the GAL encoding is used to recognize the material of the object (see Section 4.3.2). The scene representation starts with these object properties. Next, I describe the details of my approach and experiments.

4.4.1 Low-level Visual Processing

The HPSP starts by applying the same Mask-RCNN model from [Yi et al., 2019] to object detection. For each image, Mask-RCNN is used to generate segment proposals of all objects. The network also predicts category labels such as Cube or Sphere along with the segmentation mask. Proposals with bounding box scores less than 0.9 are dropped. Removing low confidence proposals reduces prediction noise. The implementation of the object proposal network (Mask-RCNN) is based on Detectron2 [Wu et al., 2019], a package with implementations of multiple CNN-based neural networks. I use ResNet-50 FPN as the

backbone and train the model for 30,000 iterations with eight images per batch. Instead of training on the whole CLEVR dataset, this network is trained on 4,000 CLEVR images generated by the CLEVR simulator as described in Yi et al., 2018, which provides a fair comparison in later experiments.

After I generate the object masks from the image, each object mask is used on the original image to filter out the object. Each mask is a matrix with same shape as the original image and the number in the mask is either 0 or 1. To filter out objects, we compute the element-wise multiplication between each

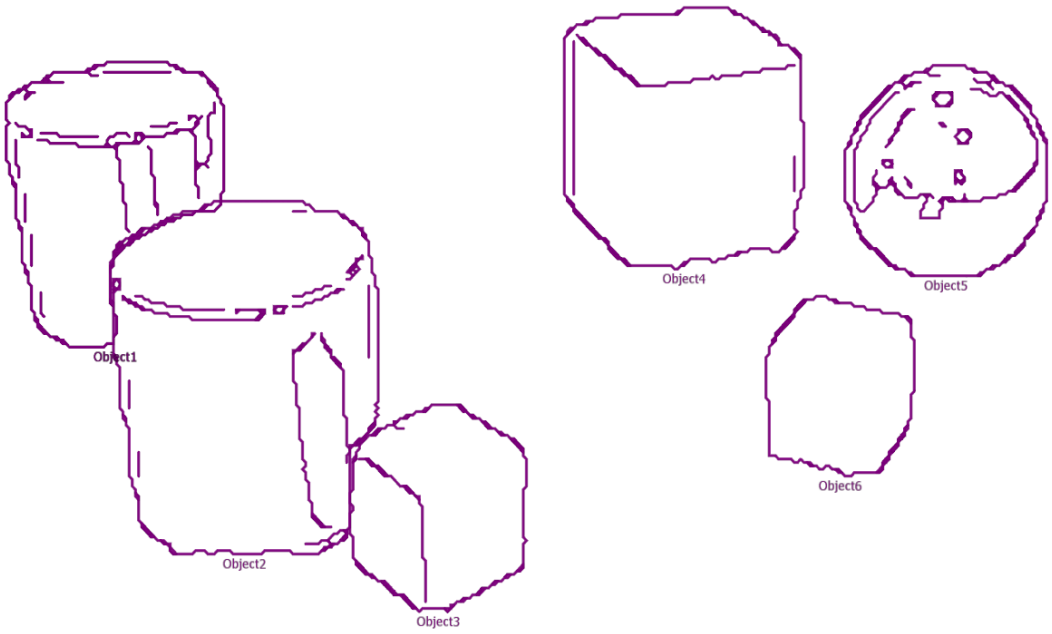


Figure 36: Reconstructed sketch in CogSketch of the image in Figure 35.

mask and the original image. I run the Canny edge detection algorithm on each filtered object based on the mask segmentation and reconstruct these objects in CogSketch. Figure 36 shows the reconstructed CogSketch sketch of the image in Figure 35.

4.4.2 High-level Visual Encoding

From the low-level visual processing stage, the Mask-RCNN model generates the object masks and corresponding categories. To represent the shape category of each object, I use the predicate **isa** to indicate the shape of objects. For example,

(isa Object-1 Cylinder)

Besides the object categories, I need to encode each object's color, material, and size information. CogSketch computes the color and size information, and I use a different learning approach to recognize the material.

To encode color, I extract the average pixel values and find the nearest color defined in NextKB, including Gray, Blue, Brown, Yellow, Red, Green, and Cyan. The predicate "**hasColor**" is used to indicate the color of an object. For example:

(hasColor Object-1 Green)

This statement means that the object "Object-1" is a green object.

To encode size, CogSketch computes the relative size relations between objects in an image. As CLEVR only has two size labels, Small and Large, I represent the objects as small objects if they have **sizeSmall** and **sizeTiny** properties computed by CogSketch. If objects have **sizeMedium** and **sizeLarge** computed by CogSketch, they are regarded as large objects. Then, I use the predicate **hasSize** to represent the size information of objects. For example,

(hasSize Object-1 Large)

Finally, I need to recognize objects' material attributes (metal or rubber). In the CLEVR universe, the images are generated by a simulator which models lighting effects on objects. If the material of an object is metal, light reflections will cause many edges or edge-cycles on the object's surface, depending on the surrounding shapes. On the other hand, if the material of an object is rubber, the object surface is not influenced by light reflection and the surface would be clean. I explore the light reflection for the three types of shapes and use an algorithm to detect edges or edge-cycles caused by reflection.

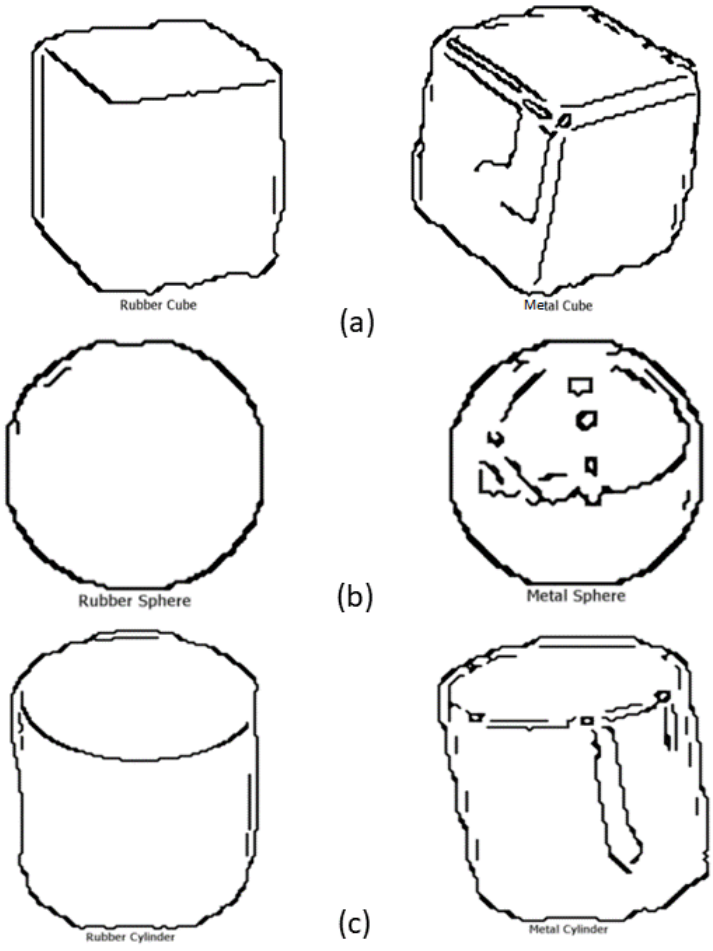


Figure 37: The edge detection results of different material. (a) The left object is a rubber cube and the right is a metal cube. (b) The left is a rubber sphere and the right is a metal sphere. (c) The left is a rubber cylinder and the right is a metal cylinder.

Figure 37 (a) shows the glyphs computed for two cubes. The right object is a metal cube, and the left object is a rubber cube. Based on the observation, light reflection on the boarders of metal cubes can cause multiple close edges that have similar gradients with each other. Thus, the decomposition tree is used to look for within-object visual structure involving multiple edges with similar gradients. If close to one another, they are treated as reflection elements. The objects in Figure 37 (b,c) are handled similarly. Analogical learning on these qualitative representations is used to recognize the material of an object. Each type of material is modeled as a gpool in SAGE, i.e., one for rubber and one for metal. All of the objects in 100 images are used as training data to build generalizations. Then, these learned gpools are applied as needed to classify what an object is made of, using the predicate "**hasMaterial**". For example,

(hasMaterial Object-1 Metal)

After encoding the properties of each object, the spatial relations between them must also be encoded. Specifically, the spatial relations in CLEVR include right/left and front/behind. CogSketch computes positional relations between each pair of adjacent objects. **rightOf** is already computed by CogSketch. To determine when front or behind applies, the Y coordinates of the bounding boxes are used. If the Y coordinate of one object is larger than the other object, that object is in front of the other object. The statements about each object and the spatial relations between objects constitute the scene representations of images.

4.4.3 Question Answering

The CLEVR dataset does not require models to use any external knowledge besides the information from the image. Thus, I formalize the question answering task as finding the answers from images given question queries. Here, I take the program of each question as the query to search for the answer from the corresponding image representation. A script is used to convert the question program into a query for the FIRE reasoning engine, which is built into CogSketch and thus is a natural choice for this kind of reasoning.

The conversion script works as follows. Each CLEVR program is a sequence of functions, and a function can use the results from previous functions. Therefore, the script needs to convert each CLEVR function into a query statement, and then a query is converted to a sequence of statements. Table 10

CLEVR Function	**Fire Query**
Filter <color / shape / size / material>	(hasColor ?object <color>) / (isa ?object <shape>) / (hasSize ?object <size>) / (madeof ?object <material>)
Count Object-Set-1	(LengthOfListFn (SetToListFn Object-Set-1))
Union Object-Set-1 Object-Set-2	(SetOrCollectionUnion Object-Set-1 Object-Set-2)
Intersect Object-Set-1 Object-Set-2	(SetOrCollectionIntersection Object-Set-1 Object-Set-2)
Query <color / shape / size / material> Object-1	(hasColor Object-1 ?color) / (isa Object-1?shape) / (hasSize Object-1 ?size) / (madeof Object-1 ?material)
Equal_integer Number1 Number2	(EqualTo-UnitValuesFn Number1 Number2)
Equal <color / shape / size / material> Attribute-1 Attribute-2	(StringEqualFn (StringFn Attribute-1) (StringFn Attribute-2))
Less than Value-1 Value-2	(LessThan-UnitValuesFn Value-1 Value-2)
Greater than Value-1 Value-2	(GreaterThan-UnitValuesFn Value-1 Value-2)
Relate <right / front> Object-1	(rightOf ?object1 Object-1) / (front ?object1 Object-1)

Table 10: Each type of CLEVR function and its corresponding Fire query.

shows each type of CLEVR function and the resulting FIRE query. Generating FIRE queries involves generating nested queries, as shown by Algorithm 8. After all functions are converted, these statements are wrapped into an **and** predicate, so the FIRE reasoner can find the answer that satisfies all constraints. The FIRE reasoner executes **ask** to retrieve the answer based on the generated query from the image scene representation. Figure 38 displays an example of a CLEVR question, its corresponding program, and its converted FIRE query for the image displayed in Figure 35.

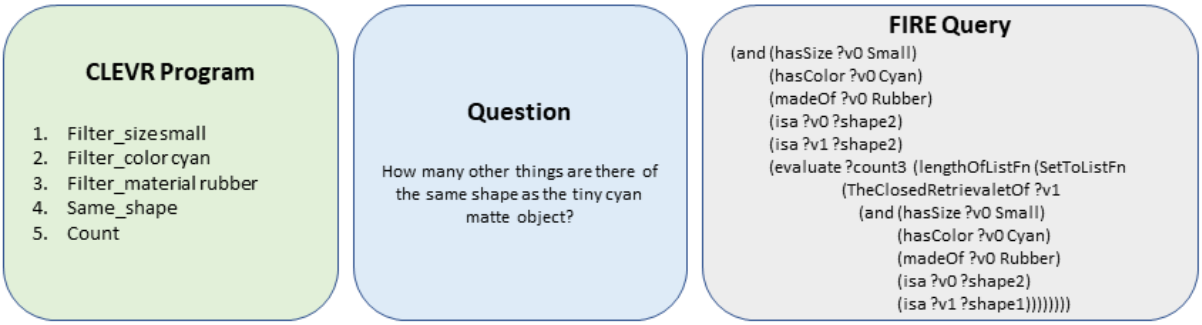


Figure 38: An example of a CLEVR question, its corresponding program, and its converted FIRE query for the image displayed in Figure 35.

Algorithm 8: FIRE Query Generation

Input: A CLEVR program, P

Functions:
ClevrFuncMap (F): map a CLEVR function F to a FIRE query.
CombineQuery(Q1, Q2): combine one FIRE query Q1 with another FIRE query Q2 to construct a new query.

Output: A converted FIRE query
1: S = []
6: For Fi in P do
7: Qi = **ClevrFunctionMap** (Fi)
8: **If Empty** (S) **do**
8: S.**append** (Qi)
8: **Else do**
8: Q1 = S[-1]
8: Q2 = **CombineQuery** (Q1, Qi)
8: S.**append** (Q2)
11: **return** S[-1]

Models	Overall Accuracy
QGHC+Att+Concat [Gao et al., 2018]	65.90%
NMN [Hu et al., 2018]	71.1%
IEP [Shi et al., 2019]	96.9%
FiLM [Perez et al., 2018]	97.6%
MAC [Hudson & Manning, 2018]	98.9%
NS-CL [Mao et al., 2019]	98.9%
TbD [Mascharka et al., 2018]	99.1%
MDETR [Kamath et al., 2021]	99.7%
NS-VQA [Yi et al., 2018]	99.8%
Ours (Ground truth material labels)	94.5%
Ours (Recognized material labels)	86.7%

Table 11: Results on CLEVR dataset.

4.4.4 Experiments

I evaluate the approach on the CLEVR validation dataset, containing 15,000 images and 149,991 questions (each image can have multiple questions). For each image, the scene representation is stored as a microtheory used as the context for FIRE reasoning. Thus, the FIRE query is evaluated against the representation for the scene. Table 11 shows the results of the approach compared with other deep learning baselines. The HPSP-based approach achieves 86.7% accuracy on the question answering task. In some examples, the HPSP-based approach fails because the object detection results are not correct (missing an object or predicting wrong locations). Also, this approach generates the front/behind relations based on the coordinates of object bounding boxes instead of analyzing the 3D locations of objects. Thus, our approach sometimes computes incorrect front/behind relations leading to generating wrong answers for some questions. Adding better 3D reasoning is a promising future research direction. Another problem

is that the material recognition is based on edge detection and texture of the objects. This can introduce noise from edge detection, which leads wrong material labels. I also tested using ground truth material labels for objects, and HPSP can achieve 94.5% accuracy given them. Thus, the scene representations of HPSP are effective and can provide comprehensive information for question answering. If the object perception models are improved, the performance on CLEVR could be improved significantly.

4.5 Conclusion

This chapter focused on the visual understanding of natural images. Processing natural images can introduce a lot of noise during low-level visual processing. Thus, I designed a hybrid architecture, the Hybrid Primal Sketch Processor, to take advantage of deep learning models for low-level perception. I showed that the HPSP could perform two tasks from the literature. First, in the Visual Relationship Detection task, pre-trained deep learning models were used to detect object bounding boxes, object masks, and object categories. Then, a pair-level encoding scheme guides the system to generate qualitative representations for object pairs and analogical learning was used to learn relation detection. Experiments show that the HPSP architecture can represent information on the object pair level and can be used to construct image scene graphs. This approach was originally published in [Chen and Forbus, 2021] In the second task, I applied the HPSP architecture to a visual question answering task. Deep learning models were used to construct qualitative scene representations for each image and the FIRE reasoner was used to derive answers to questions from these scene representations. Experiments on the CLEVR dataset illustrate that the HPSP architecture can generate promising scene representations for visual understanding and reasoning.

5 Human Action Recognition

5.1 Introduction

This chapter describes how analogical learning over qualitative representations provides high accuracy and understandability in recognizing human actions from Kinect skeleton data. This work was originally published in the paper [Chen and Forbus, 2018]. Instead of computing frame-based features [Wang et al. 2016; Ye, 2016; Li and Chen. 2016], our approach is to decompose the video stream into *sketch graphs*, consisting of multiple sequences of snapshots. Each snapshot is like a panel in a comic strip: It consists of a motion segment described by a single qualitative state, which might correspond to many frames. Each body point has its own sequence of such states. The trajectories within these states and relationships across these states are described qualitatively, using automatically constructed visual representations. The sketch graphs for each instance of a behavior type are combined via analogical generalization using SAGE. This automatically constructs probabilistic relational schemas (plus outliers) characterizing that behavior type. Given a new behavior, a set of sketch graphs is computed for it, and analogical retrieval is used across the entire set of behavior models to retrieve the closest schema (or outlier). We describe the learning pipeline and how classification works. Next, we show how these representations support understandability of recognition decisions made via analogy. Results on three public Kinect action datasets are described. We close with related and future work.

5.2 Approach

Our approach is implemented as a pipeline with four stages: Action Segmentation, Relational Enrichment, Action Generalization, and Classification. A dynamic feature selection process picks reasonable additional features for different actions before the final training. All sketches and relations are computed by our system automatically. Figure 39 shows the pipeline of our system. We describe each stage in turn and describe how sketch graphs support understandability.

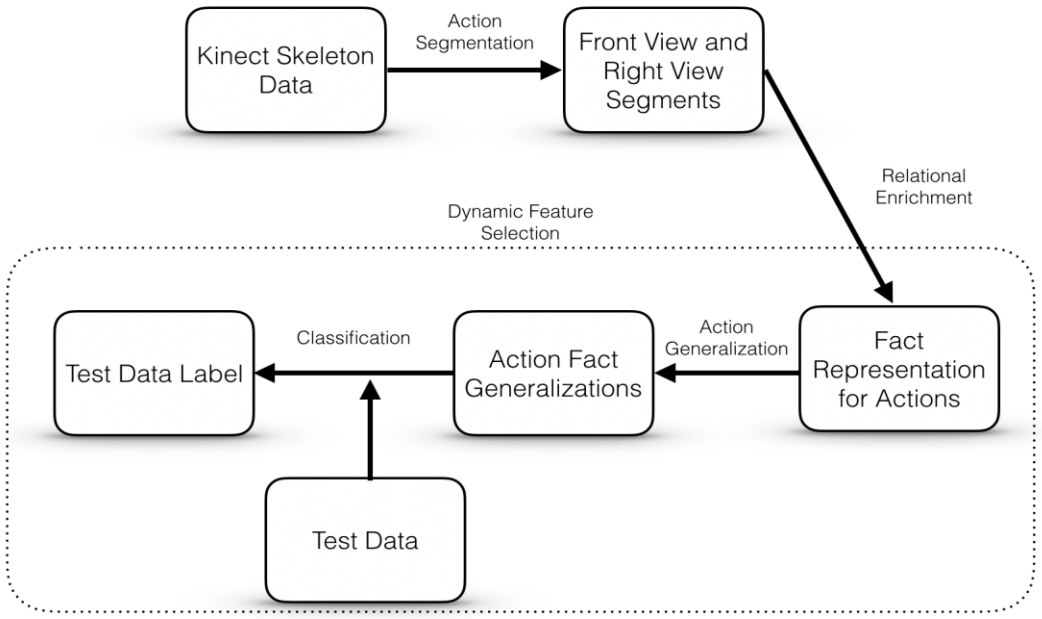


Figure 39: Pipeline of human action recognition algorithm [Chen and Forbus, 2018].

Action Segmentation

The skeleton data produced by a Kinect (or other 3D sensor) contains many points per frame, representing each body part such as the head or right-hand. We use 20 body points tracked by Kinect V1 to represent 20 body parts and connect these body points to provide a concise body skeleton graph, as shown in Figure 40.

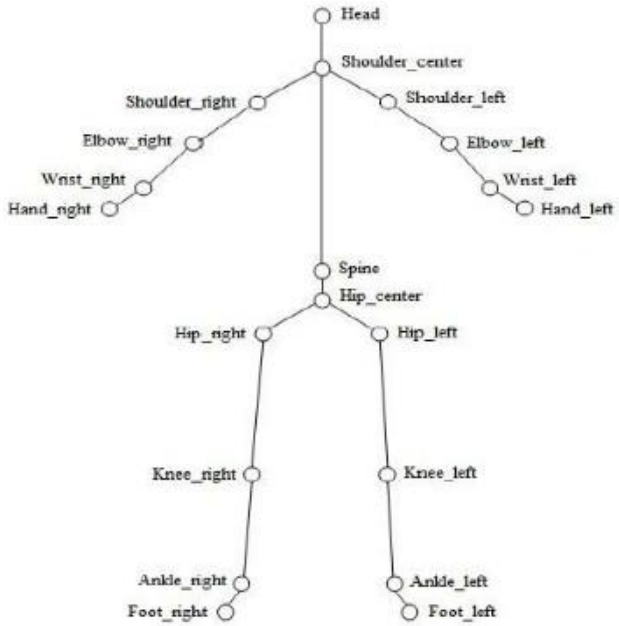


Figure 40: Kinect human skeleton graph.

Each instance of an action consists of a continuous movement stream, sampled via many frames, each frame containing coordinates for these points. The first step of our pipeline abstracts away from frames into qualitatively distinct intervals describing the motion of particular body parts. A *track* is a sequence of point coordinates from each frame for a specific body point. As CogSketch needs 2D sketches, we map each 3D coordinate into a front-view and a right-view sketch. To segment movements of a track (a body point) in a view, we compute the *azimuth* (the angle that is clockwise relative to the north) changes of the track frame by frame to find the direction change. Intervals of time over which the motion has a similar azimuth are grouped into one segment. In the experiments reported here, we use only the right-hand, left-hand, right-hip, and left-hip in front-view and right-view, because the motions in the datasets used can be described as the movements of these four body parts.

For each track in a view, we first compute the spatial relations *Moving or Stationary* (MOS), with 0.02 quantization factor via *QSRlib* [Gatsoulis et al. 2016], a library of Qualitative Spatial Relations and Calculi, for the four main body points. MOS relations can show whether a point in a frame is moving (label ‘m’) or stationary (label ‘0’). An MOS relation sequence could be as following:

[0,0,0,0,m,m,m,m,m,m,m,0,0,]

After motion detection, eight MOS sequences corresponding to four points in two views are extracted. All frames with label ‘m’ are segmented by computing the cartographical azimuth changes between each pair of two consecutive moving points. When the azimuth change is larger than ninety degrees, a track is segmented into two parts. After action segmentation, we get eight sequences of segments, four points in the front-view and four points in the right-view.

We use two techniques to reduce segmentation noise. First, after action segmentation, all segments in a track are merged again when the average azimuth between the start-point and the end-point is smaller than fifty degrees. Second, segments are also merged when the distance between the start-point and end-point of the segment is smaller than half of the average distance between start-point and end-point of all segments.

Relational Enrichment

The relational enrichment stage involves automatically adding additional relationships via CogSketch, to provide more information about the motions within and between segments. Each example of a behavior is imported into CogSketch as a set of sketches, one per track, with each panel (segment) within a track being represented by a separate subsketch. Within each panel, the skeleton is represented by a set of glyphs, including an arrow from start-point to end-point of the track panel to represent the direction of motion. Figures 41 and 42 show a sketch representation of raising the right hand and putting it down. Figure 41 shows the segmentation of the right-hand track in the front view and Figure 42 shows the segmentation of the right-hand track in the right view.

To summarize, each action is represented by eight sketches in CogSketch: right-hand in front-view, right-hand in right-view, left-hand in front-view, left-hand in right-view, right-hip in front-view, right-hip in right-view, left-hip in front-view and left-hip in right-view. In each sketch and subsketch,

CogSketch is used to compute relationships between body parts, e. g. the relative position of the right-hand to the head.

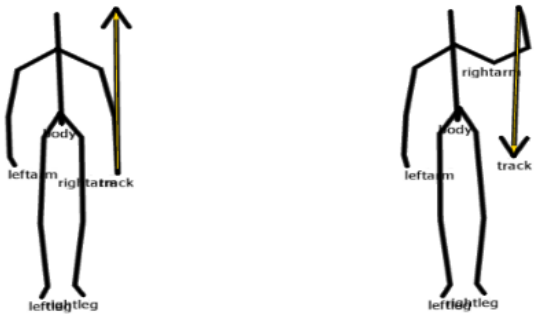


Figure 41: front-view of raising hand [Chen and Forbus, 2018].

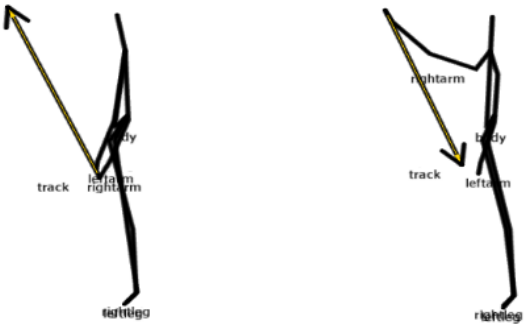


Figure 42: right-view of raising hand [Chen and Forbus, 2018].

We use the following logical function to denote panels in a sketch graph:

(KinectMotionFn <body-part> <view> <move-type> <token>)

<body-part> is from the four main body points: right-hand, left-hand, right-hip, left-hip. <view> is front or right view. <move-type> describes the type of movement: single-hand, two-hand or full-body. <token> is a unique identifier denoting the segment.

In each segment subsketch, additional details can be provided via Cyc's **holdsIn** relation. For example, spatial relations are helpful to determine the locations of body points, so these relations are added. In each segment motion, we use entities from CogSketch's qualitative representation of orientation to describe the direction of motion, i.e., the quadrant representations **Quad1**, **Quad2**, **Quad3**, **Quad4**, the

pure directions **Up**, **Down**, **Left**, and **Right**, plus the constant **NoMotion** indicating lack of motion. The direction of motion is linked to the motion segment via **holdsIn**. For example:

(holdsIn

(KinectMotionFn RightHand Front SingleHand D1RHFS2)

(trackMotionTowards Quad1))

Sequence information between segment panels is represented using the **occursAfter** and **occursTogether** relations. **occursAfter** indicates that two segments occur successively and is used to connect segments from same track in same view. **occurTogether** means that two segments have eighty percent time overlap and connects the segments from different track in same view, e. g.

Relations	Descriptions
(trackMotionTowards <Quad1/Quad2/Quad3/Quad4 >)	The track moving direction from start-point to end-point.
(quadDirBetween <right-hand/left-hand> <right-elbow/left-elbow> <Quad1/Quad2/Quad3/Quad4>)	The elbow direction with respect to corresponding hand.
(bodyStatus <Bend/Straight>)	Whether the body bends larger than 45 degrees.
(handPosition <RaiseHand/PutDown > <right-hand/left-hand>)	Whether the hand bends raising larger than 90 degrees.
(armStatus <Bend/Straight > <right-arm/left-arm>)	Whether the arm bends larger than 90 degrees.
(motionRange <Large/Small>)	Whether the movement range is larger than half of body length.
(twoArmRela <Cross/NoCross>)	Whether the two arms are cross with each other.
(legStatus <Bend/Straight> <right-leg/left-leg>)	Whether the leg bends larger than 45 degrees.
(moveRespectArm <Inside/OutSide> <right-hand/left-hand>)	Whether the hand is moving towards inside of the arm or outside of the arm.
(distRespectBody <Large/Small>)	Whether the distance of x coordinates between hand and spine is larger than 25.

Table 12: Basic features for sketch graph descriptions

(occursAfter

(KinectMotionFn RightHand Front SingleHand D1RHFS2)

(KinectMotionFn RightHand Front SingleHand D1RHFS4))

This representation enables facts from different segments to be included in one unified case representation and is extracted totally automatically. Table 12 provides the full set of relationships that we compute for every track.

Action Generalization

All facts for each segment are combined into a case representing the entire action. Each such action instance is added to the generalization pool being used to learn that concept. For all experiments reported here, we used an assimilation threshold of 0.7. The default SAGE probability cutoff of 0.2 was used in all experiments. Each action type is represented as a distinct generalization pool and all action type generalization pools are combined into a case library for classification.

Classification

By treating the union of generalization pools for action types as one large case library, our system can classify new examples based on which library the closest retrieved item came from. Given a new case, MAC/FAC is used to retrieve the closest generalization or outlier from the union of the generalization pools. The action label of the generalization pool it came from is assigned to the test instance.

Dynamic feature selection

As shown in Table 12, ten basic relations are extracted to represent each segment. However, representing the direction of motion more precisely relative to different reference points can be very useful. The relation **qualDirBetween** represents the direction of motion with respect to a reference point. Its first argument is the start or end point of the motion. Its second argument is the reference point. Its third argument is the direction. For example,

(holdsIn

(KinectMotionFn RightHand Front SingleHand D1RHFS1)

(qualDirBetween RightHand-StartPoint Head Quad4))

indicates that the start-point of the motion D1RHFS1 is in the Quad4 direction, with respect to the head, in the first segment of the right-hand track within front-view.

In these experiments, we use head, shoulder-center, spine, and hip-center as the possible reference points. Directions are described either in terms of quadrants or broad directions, i.e. Left/Right or Up/Down, each of which are the unions of two quadrants. For conciseness, we will abbreviate subsets of this representation via the template <reference point>-<direction type>, i.e. the statement above would be an example of Head-Quad.

Dynamic feature selection is used to select which families of direction representations are used for a dataset. Given the distinctions above, there are 12 families of **qualDirBetween** relations that can be computed. The algorithm starts with the basic set of features plus a single family of optional features, doing training and testing with each independently. The highest accuracy optional feature is retained. On subsequent rounds, the search is constrained by limiting it to the two unused features that perform best where the choices so far perform the worst. The search stops either when a cutoff is reached (here, the cutoff is four optional features, which provides a reasonable tradeoff between accuracy and efficiency) or when all the additions lead to lower accuracy.

Methods	Features	Accuracy results (%)
Manual feature selection	Head-Quad, Spine-Quad, Hip-Center-Quad	63.6
Dynamic feature selection	Head-Quad Spine-Up-Down Hip-Center-Up-Down	74.2

Table 13: Recognition results with and without dynamic feature selection

We evaluated dynamic feature selection on the Florence 3D Action dataset [Seidenari et al. 2013], which contains nine activities: wave, drink from a bottle, answer phone, clap, tie lace, sit down, stand up, read watch, bow. Ten subjects were asked to perform the nine actions two or three times. Two groups of additional features are tested: one is picked manually and the other one is picked via the algorithm above. Cross-subject validation was used. The results, shown in Table 13, show that dynamic feature selection improves accuracy by ten percent.

Understandability of Sketch Graphs

Sketch graphs carve motions up into a discrete set of snapshots, much like comic strips, an easy to apprehend visualization for people. The generalizations are also sketch graphs, enabling them to be inspected more easily than, for example, large vectors. The analogical mapping that justifies a recognition decision can be displayed by drawing the correspondences between panels in the two sketch graphs and their constituents.

This is the basis for the *explanation sketch*, which depicts how a retrieved sketch graph for a generalization (or an outlier) aligns with a sketch graph for a new behavior. The skeleton glyphs from corresponding segments are visualized in two boxes side-by side. Correspondences within the segments are indicated by dashed lines. Thus, the explanation sketch provides a visualization for how a model explains a behavior, based on their overlap. Figures 43 and 44 show two examples. Figure 43 shows a perfect match with five dashed lines for all five different parts. Figure 44 shows a different movement corresponding, which only has same facts for left-arm, left-leg and right-leg.

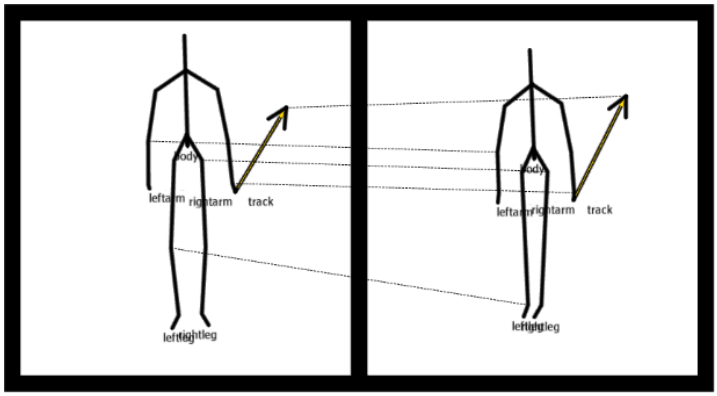


Figure 43: Perfect matching [Chen and Forbus, 2018].

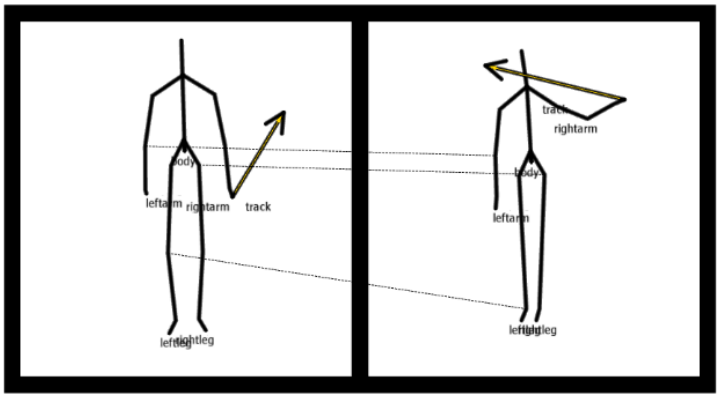


Figure 44: Partial matching [Chen and Forbus, 2018].

5.3 Experiments

While we view the ability to produce understandable explanations as an important part of our approach, we note that other approaches do not explore explanation, so we confine ourselves here to comparing with others using their metrics. Three datasets are tested, and we describe each in turn.

UTD-MHAD Dataset

The first dataset we test is UTD-MHAD. This dataset contains eight different subjects (4 females and 4 males) performing twenty-seven actions in a controlled environment, with each action repeating four times, collecting data from a Kinect V1 sensor. As our research only focuses on skeleton data, we only use skeleton Files from skeleton videos. As Kinect only tracks 20 body points and some motions have

very similar movements, such as “Clap” and “Arm curl”, we removed 6 skeleton similar actions and removed two actions with large noise in skeleton data. The other nineteen actions are tested here.

Qualitative spatial relations are computed for all target body points and dynamic feature selection is used. From dynamic feature selection, Head-Quad, Head-Up-Down, Hip-Center-Up-Down and Spine-Quad were picked as four additional features. We used the same cross-subject testing method from Chen, et al’s (2015) paper, which uses four subjects (1,3,5,7) for training and four subjects (2,4,6,8) for testing. Our method achieves 80.3% accuracy with the cross-subject testing method. The result of each action is shown in Figure 44. The average result of existing methods is in Table 14 (all methods are tested on the 19 actions for fair comparison).

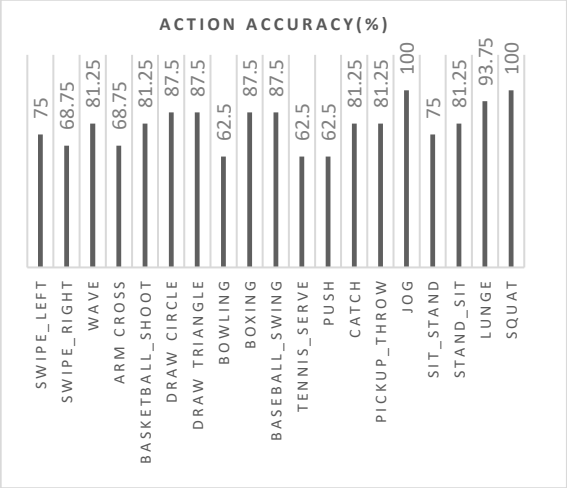


Figure 44: Recognition rates (%) for each action in UTD-MHAD [Chen and Forbus, 2018].

Method	Accuracy (%)
Kinect [Chen et al. 2015]	66.1
Inertial [Chen et al. 2015]	67.2
Kinect & Inertial [Chen et al. 2015]	79.1
Adaboost M2 [Zhang et al. 2017]	83.0
Our Method	80.3

Table 14: Recognition rates (%) on the UTD-MHAD [Chen and Forbus, 2018].

As shown in the table above, our method has better accuracy than the Kinect & Inertial and a little lower than the Adaboost M2. One reason is that our method uses qualitative relations instead of numbers. This can cause information loss if the available relationships do not provide fine enough distinctions. For example, the “swipe-right”

action can be segmented into three parts: raise hand, swipe, and put down hand. But these three segments could form a triangle in the air, which is similar to the "draw-triangle" action. With spatial relations we defined here, some instances of "draw triangle" with larger movement range may be represented by same relational facts of "swipe-right". This resolution/accuracy tradeoff is worth exploring in future work.

Florence 3D Actions Dataset

We ran an experiment on this dataset again but followed the leave one out cross-validation protocol (LOOCV) [Seidenari et al. 2013] to compare with other methods. Dynamic feature selection picked Head-Quad, Spine-Quad, Hip-Center-Up-Down as additional features. The average accuracy compared with other methods is shown in Table 15.

Method	Accuracy (%)
Seidenari et al. 2013	82.0
Devanne et al. 2014	87.04
Vemulapalli et al. 2014	90.88
Our Method	86.9

Table 15: Recognition rates (%) on the Florence dataset [Chen and Forbus, 2018].

As Table 14 shows, our method has comparable results with Devanne et al.'s [2014] results and a little lower than Vemulapalli et al.'s [2014] results. In this dataset, our algorithm has relatively low accuracy on "drink from a bottle" and "answer phone" among all nine actions because some instances of them cannot be distinguished from the "wave" action. All three can be segmented into a motion that subject raises the right-hand to the position near the head, and our qualitative representations did not have sufficient resolution to distinguish them. However, we note that when people are asked to review the skeleton data for these two actions, they also find it hard to describe the differences. Consequently, we do not necessarily view our system's performance on these actions as negative.

UTKinect-Action3D Dataset

To further evaluate our method, we ran an experiment on the UTKinect-Action3D dataset [Xia et al. 2012]. This Kinect dataset has 10 actions performed by 10 different subjects. For each action, each person performs it twice so there are 200 behaviors in this dataset. The ten actions are: walk, sit down, stand up, pick up, carry, throw, push, pull, wave, and clap hands.

Action	Xia et al. 2012	Theodorakopoulos et al. 2014	Ours
Walk	96.5	90	100
Sit down	91.5	100	90
Stand up	93.5	95	85
Pick up	97.5	85	100
Carry	97.5	100	85
Throw	59.0	75	60
Push	81.5	90	70
Pull	92.5	95	95
Wave	100	100	100
Clap hands	100	80	100
Overall	90.92	90.95	88.50

Table 16: Recognition rates (%) on the UTKinect-Action dataset [Chen and Forbus, 2018].

With dynamic feature selection, Head-Up-Down, Spine-Quad, and Hip-Center-Up-Down are picked as the additional features for this dataset. We follow the leave one-out cross-validation protocol (LOOCV) [Xia et al. 2012] for comparability. Table 16 shows the recognition rates corresponding to the different actions and compares our accuracy with two other methods.

As shown in Table 16, the three algorithms present comparable performance for different actions. Our average accuracy is 88.5%. Our system has relatively lower accuracy on some actions such as throw and push. In this dataset, some subjects did not face the Kinect directly when they performed the actions. As our method needs to extract front-view and right-view sketches from the data, this noise could have influenced our algorithm.

5.4 Summarization

This chapter presents a new approach for human action recognition from skeleton data, with high accuracy and novel explanation ability, based on qualitative representations and analogical generalization. This approach was originally published in [Chen and Forbus, 2018]. Our pipeline uses azimuth changes to segment tracks, a cognitive model of human high-level vision to enrich descriptions of motion and configuration, and analogical generalization to provide learning via understandable, relational models. Explanation sketches are used to visualize the correspondences and mappings between different segments. Experiments on three public datasets provide evidence for the utility of this approach, i.e. that sketch graphs provide an effective representation for time-varying human behavior, and that analogical learning over sketch graphs provides competitive performance to other ML techniques, with the added advantage of explanations.

There are several avenues to explore next. The first is to experiment with additional datasets, both to explore noise and dynamic encoding issues. The second is to examine the effectiveness of explanation sketches in helping system trainers improve performance. The third is to examine whether near-miss learning [McLure et al. 2015] provides better results and explanations.

6 Conclusion and Future Work

In this work, I present novel ideas for using qualitative representations and analogical learning for visual understanding. The investigations were specifically concerned with encoding strategies for visual inputs. These encoding strategies provide object-level, scene-level, and temporal-level information for visual understanding. I also developed a novel way to use analogical learning hierarchically on qualitative representations, which provides training efficiency and improved performance. This hierarchical analogical learning process can potentially be applied to many different domains or tasks. Furthermore, I presented a new architecture, Hybrid Primal Sketch Processor, that combines deep learning with

qualitative representations for visual processing. This architecture takes advantage of the broad coverage and flexibility of deep learning models along with the advantages of understandability, reasoning ability, and incremental learning ability from qualitative representations and analogical learning.

This research provides evidence that analogical learning over qualitative representations is promising for visual understanding. Analogical learning is designed to extract from examples a set of understandable, structured generalizations. Qualitative representations provide structured representations that are human-like and flexible. In this work, I explored four important visual tasks and discussed the bottlenecks of current off-the-shelf deep learning models on these tasks. In each task, I introduced novel encoding strategies and how to use analogical learning for those tasks. Experiments on public datasets provide evidence that our method is effective. This section revisits our claims and contributions. I then suggest some of the most exciting avenues of future work.

6.1 Claims Revisited

1. Geon-based and part-based qualitative representations can describe object geometric information and support analogical learning.

In Section 3, I described experiments using analogical learning over qualitative representations based ony geon-based and part-based encoding schemes. For geon-based encoding, the experiment on the MNIST dataset shows that analogical learning over geon representations can achieve competitive results compared with off-the-shelf deep learning models but with fewer training samples. On the CBO dataset, our approach using geon-based encoding can achieve nearly 29% accuracy with only 9 training samples per category, whereas deep learning models fail to capture the patterns with the same amount of data. For part-based encoding, experiments on TU Berlin and CBO datasets illustrate that the part-based encoding scheme has even better performance than geon-based encoding.

Both encoding schemes achieve competitive results compared with baselines but with a much smaller amount of training data, or single-epoch versus many epochs, by using analogical learning. This high data and training efficiency shows that these two encoding schemes can effectively describe the object's geometric information because geometric features are most important for recognizing sketched objects.

2. Using analogical learning hierarchically can improve performance while maintaining the data efficiency and training efficiency of traditional analogical learning.

In Section 3, the PHAL encoding scheme represents object geometric features in a three-level hierarchical representation. This enables analogical learning to be hierarchical, with models learned at coarser encoding levels used to guide retrieval for finer-grained encoding levels. Experiments showed that PHAL is competitive with other approaches on the TU Berlin dataset and provides a new state of the art on the CBO dataset.

3. The Hybrid Primal Sketch Processor is a promising architecture for combining deep learning models and qualitative representations on visual understanding tasks.

Visual understanding of real images is a challenging problem. Deep learning has been used to produce many improved components for vision tasks, albeit requiring massive amounts of training data and lacking understandability of their outputs. Given visual entities from sketches, CogSketch computes qualitative representations that can be used for analogical reasoning and learning, including understandable models and results. Chapter 4 introduces the Hybrid Primal Sketch Processor, which combined deep learning components with CogSketch's qualitative visual capabilities to gain the advantages of both. The HPSP is evaluated on two complex visual tasks, visual relationship detection and visual question answering. I leveraged the Mask-RCNN model for object detection and CogSketch to construct qualitative representations over object detection results. On both tasks, analogical learning with HPSP-produced representations achieved competitive results compared to deep learning models.

Furthermore, analogical generalizations provide strong understandability as symbolic representations can be understand by human. Our approach has better training efficiency since analogical learning only looks over all training data once instead of multiple epochs like deep learning models. Therefore, I argue that the HPSP is a promising architecture to combine deep learning models and qualitative representations.

4. Qualitative representations can effectively represent temporal behaviors for analogical learning.

Some visual understanding tasks involve temporal inputs, such as videos or skeleton data, as inputs and perform reasoning on them. Chapter 5 shows how qualitative representations can be extended to describe temporal behaviors in human action recognition tasks. Specifically, human skeleton data is generated by a Kinect camera, where each skeleton sample has a sequence of frames with 20 human body tracking points. Rather than working frame-by-frame, as many approaches do, I follow [Gatsoulis et al., 2016] to segment sequences of frames into meaningful units, based on when qualitative descriptions between frames change. This is used to generate *sketch graphs*, a comic-strip like representation of behavior over time, that can be used with analogical learning to provide competitive results. It also supports *explanation sketches*, where corresponding elements of sketch graphs (e.g. a new example and a candidate generalization of it) can be highlighted to understand a classification decision.

6. Analogical learning over qualitative representations has high data-efficiency and strong understandability for visual tasks.

This thesis has demonstrated the advantages of analogical learning over qualitative representations using multiple visual tasks. The experiments on sketched object recognition included tests of data-efficiency. On the MNIST dataset, only 500 training images sufficed to achieve 85.03% accuracy. On the CBO dataset, only 9 images per category suffice to achieve about 29% accuracy with the geon-based encoding scheme and about 34% accuracy with the part-based encoding scheme. The structured relational representations provide concise statements, in probabilistic schemas or outlier examples, that are easily understood by people.

6.2 Future Work

6.2.1 Combining analogical learning and deep learning models in an end-to-end fashion

Chapter 4 proposes the Hybrid Primal Sketch Processor to combine deep learning models and qualitative models for visual understanding. However, as described in section 4.2.4, some deep learning baselines outperform our approach on the RELDT task, because pre-trained deep learning models cannot always predict correct object categories or locations. While I keep tracking the state-of-the-art deep learning models, I am also interested in using the generalizations from analogical learning as feedback to optimize deep learning models in an end-to-end fashion. The essential problem is that analogical learning over qualitative representations is in the discrete space. Therefore, we cannot apply backpropagation algorithm on it because our approach is not differentiable. I propose a possible research direction to combine analogical learning and deep learning.

As the analogical learning process on qualitative representations is discrete, we need to figure out how to convert it to a continuous space while keeping the understandable representations. The Tensor Product Representation (TPR) mechanism [Smolensky, 1990] is a method to create a vector space embedding of complex symbolic structures and can model symbolic algorithms. TPR has two important operations: binding and unbinding. The TPR binding operation encodes a symbolic structure into a tensor. The TPR unbinding operation decomposes the encoded tensor back to original structures. Thus, TPR can entangle and disentangle a dense representation in continuous space without losing information. I used TPR to perform structure transformation and learn understandable rules from knowledge graphs [Chen et al., 2020; Chen et al., 2021]. My research on TPR suggests that it could be a possible way to explore encode qualitative representations and model analogical learning processes in the future.

6.2.2 Video representation with HPSP architecture

In this thesis work, I have shown how to represent scene-level information of images and temporal information of skeleton data. Combining them could be a good model for constructing human-like relational representations of videos, extending the approach described in Chapter 5 for skeleton data.

The scheme might work like this. First, detect objects from video frames using state-of-the-art deep learning object detection (OD) models, such as Faster-RCNN or Mask-RCNN, producing object bounding boxes, masks, and categories. Second, encode both spatial relations and semantic relations per frame, using the same types of spatial relations used here (e.g. positional, topological, and size), from which semantic relations can be computed, as per my work on the visual relation detection task. Doing this exhaustively would probably be too inefficient, so filtering mechanisms (e.g. using closeness as determined by Voronoi diagrams) would need to be developed. The positions of objects in this semantic graph would be used as one temporal segmentation clue, just as was done with skeleton data, along with changes in the semantic relations. The sketch graphs produced by such a scheme should still be amenable to analogical learning, with the same advantages in terms of efficiency (both data and training) and understandability.

6.2.3 Formal evaluations for understandability

We only explored the learned facts in SAGE generalizations to show the understandability of our approach but did not design a formal method to evaluate understandability. Such methods could help guide the development of qualitative representations and visualizations for making better interactive learning systems. Some of these experiments would be within-approach, others might compare alternate approaches. For example, visualizations of attention maps of network weights for a deep learning model might be compared with natural language generated summaries of SAGE generalizations, to see which people prefer and/or which are actually useful for tasks such as improving/debugging a system.

References

Anderson, E. M.; Chang, Y.; Hespos, S.; and Gentner, D. 2018. Comparison within pairs promotes analogical abstraction in three-month-olds. *Cognition*, pages 176:74-86.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L. 2017. Bottom-up and Top-down attention for image captioning and visual question answering. arXiv: 1707.07998

Arterberry, M. E., & Kellman, P. J. 2016 Development of perception in infancy: The cradle of knowledge revisited, 2nd Edition. New York: Oxford University Press.

Barbella, D., and Forbus, K. 2013. Analogical word scene disambiguation. *Advances in Cognitive Systems*, 2(1):297-315.

Biederman, I. 1987. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2), p.115.

Blum, H. 1967. A transformation for extracting new descriptors of shape. *Models for Perception of Speech and Visual Forms*, 362-380.

Brown, T., Mann, B., Ryder, N., …, Suskever, I., and Amodei, D. 2020. Language models are few-shot learners. arXiv:2005.14165

Bornstein, M. H. 1985. Habituation of attention as a measure of visual information processing in human infants: Summary, systematization, and synthesis. In G. Gottlieb & N. A. Krasnegor (Eds.), *Measurement of audition and vision in the first year of postnatal life: A methodological overview* (pp. 253–300). Norwood, NJ: Ablex.

Buhrmester, V.; Munch, D.; and Arens, M. 2019. Analysis of explainers of black box deep neural networks for computer vision: a survey. *arXiv preprint arXiv:1911.12116.*

Cantoni, V. 1994. Human and Machine Vision.

Casasola, M. (2005). When less is more: How infants learn to form an abstract categorical representation of support. *Child Development*, 76(1), 279–290. http://dx.doi.org/10.

1111/j.1467-8624.2005.00844.x.

Casasola, M., & Park, Y. (2013). Developmental changes in infant spatial categorization: When more is best and when less is enough. *Child Development*, 84(3), 1004–1019.

Chen, C.; Jafari, R.; & Kehtarnavaz, N. 2015. UTD-MHAD: A Multimodal Dataset for Human Action Recognition Utilizing a Depth Camera and a Wearable Inertial Sensor. IEEE International Conference on Image Processing.

Chen, K., Huang, Q., Palangi, H., Smolensky, P., Forbus, K. and Gao, J., 2020a, November. Mapping natural-language problems to formal-language solutions using structured neural representations. In *International Conference on Machine Learning* (pp. 1566-1575). PMLR.

Chen, K., Huang, Q., Smolensky, P., Forbus, K. and Gao, J., 2021. Learning Inference Rules with Neural TP-Reasoner.

Chen, K.; Rabkina, I.; McLure, M. D.; and Forbus, K. D. 2019. Human-like Sketch Object Recognition via Analogical Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 1336-1343.

Chen, K.; and Forbus, K.D. 2018. Action Recognition from Skeleton Data via Analogical Generalization over Qualitative Representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

Chen, K., Forbus, K. 2021. Visual Relation Detection using Hybrid Analogical Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chen, K.; Forbus, D.; Gentner, D.; Hespos, S.; and Anderson, E. 2020b. Simulating Infant Visual Learning by Comparison: An Initial Model. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society.*

Chen, X. And Zitnick, C.L., 1997. Mind? S eye: A recurrent visual representation for image caption generation. *Neural computation*, *9*(8), pp.1735-1780.

Chen, X., Wang, X., Changpinyo, S., …, Houlsby, N., and Soricut, R. 2022. PaLI: a jointly-scaled ultilingual language-image model. arXiv:2209.06794.

Crouse, M.; McFate, C.; and Forbus, K.D. 2018. Learning from Unannotated QA Pairs to Analogically Disambiguate and Answer Questions. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Ciresan, D. C.; Meier, U.; and Schmidhuber, J. 2012. Multi-column deep neural networks for image classification supplementary online material. *Computer Vision and Pattern Recognition (CVPR)*, 3642-3649.

Ciresan, D. C.; Meier, U.; Gambardella, L. M.; and Schmidhuber, J. 2011. Convolutional neural network committees for handwritten character classification. *Document Analysis and Recognition (ICDAR)*, 1135-1139.

Cohn, A. G., Bennet, B., Gooday, J., and Gotts, N. M. 1997. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica* 1: 275-316.

Colombo, J., Mitchell, D. W., Coldren, J. T., & Freeseman, L. J. (1991). Individual differences in infant visual attention: Are short lookers faster processors or feature processors? *Child Development*, 62(6), 1247–1257. Doi.org/10.2307/1130804

Dai, W.; and Zhou Z. 2017. Combining logical abduction and statistical induction: Discovering written primitives with human knowledge. *AAAI*, 4392-4398.

Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; and Li F. F. 2009. Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition*.

Devanne, M.; Wannous, H., Berretti, S.; Pala, P.; Daoudi, M.; & Del Bimbo, A. 2014. 3D human action recognition by shape analysis of motion trajectories on Riemannian manifold. *IEEE trans.*

Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., Zweig, G. and Mitchell, M., 2015. Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809*.

Doumas, L., Hummel, J., & Sandhofer, C. 2008. A theory of the discovery and prediction of relational concepts. *Psychological Review, 111:*1-43.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2625-2634).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929.

Eitz, M.; Hays, J.; and Alexa, M. 2012. How do humans sketch objects? *ACM trans. Graph*. 31.4: 44-1.

Ferry, A. L., Hespos, S. J., & Gentner, D. 2015. Prelinguistic relational concepts: Investigating analogical processing in infants. *Child Development*, 86, 1386–1405.

Forbus, K. 1995. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 141-205.

Forbus, K.; Usher, J.; Lovett, A.; Lockwood, K.; and Wetzel, J. 2011. CogSketch: Sketch understanding for Cognitive Science Research and for Education. *Cognitive Science*, 648-666

Forbus, K.; Ferguson, R. W.; Lovett, A.; and Gentner, D. 2017. Extending SME to handle large-scale cognitive modeling. *Cognitive Science*, 1152-1201.

Forbus, K., & Hinrichs, T. (2017). Analogy and Qualitative Representations in the Companion Cognitive Architecture. *AI Magazine*, 38(4):34-42

Forbus, K.; Liang, C.; & Rabkina , I. 2017. Representation and Computation in Cognitive Models. *Cognitive Science*.

Forbus, K.; Usher, J.; Lovett, A.; Lockwood, K.; & Wetzel, J. 2011. CogSketch: Sketch Understanding for Cognitive Science Research and for Education. *Conitive Science*, 648-666.

Forbus, K. D., Chang, M., McLure, M. & Usher, M. (2017). The Cognitive Science of Sketch Worksheets. *Topics in Cognitive Science*. DOI: 10.1111/tops.12262

Forbus, K. D. 2019. Qualitative representations: how people reason and learn about the continuous world. MIT Press.

Forbus, K. D., Hinrichs, T., De Kleer, J., and Usher, J. 2010. FIRE: Infrastructure for experience-based systems with common sense. *AAAI Fall Symposium on Commonsense Knowledge*, Arlington VA.

Gatsoulis, Y.; Alomari, M.; Burbridge, C.; Doudrup, C.; Duckworth , P.; Lightbody, P.; Gohn, A. 2016. QSRlib: a software library for online acquisition of Qualitative Spatial Relations from Video. *QR*.

Gao, J., Galley, M. and Li, L., 2019. Neural approaches to conversational ai. *Foundations and trends® in information retrieval*, *13*(2-3), pp.127-298.

Gao, P., Li, H., Li, S., Lu, P., Li, Y., Hoi, S.C. and Wang, X., 2018. Question-guided hybrid convolution for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 469-485).

Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*.

Gentner, D. 2010. Bootstrapping the mind: Analogical processes and symbol systems. *Cognitive Science*, 34 (5). 752-775

Gentner, D. 2003. Why we're so smart. In *Language in Mind: Advances in the study of language and thought* (pp. 195-235). MIT Press.

Gentner, D., & Rattermann, M. J. (1991). Language and the career of similarity. In S. A. Gelman & J. P. Byrnes (Eds.), *Perspectives on thought and language: Interrelations in development* (pp. 225-277). London: Cambridge University Press.

Gentner, D., & Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. *Cognitive Science*, 10, 277-300.

Gentner, D., and Markman, A. B. 1997. Structure mapping in analogy and similarity. *American Psychologist*, 52, 45-56

Gervain, J., Berent, I., & Werker, J. F. (2012). Binding at birth: The newborn brain detects identity relations and sequential position in speech. Journal of Cognitive Neuroscience, 24(3), 564-574.

Gibson, E. J. 1963. Development of perception: Discrimination of depth compared with discrimination of graphic symbols. *Monographs of the Society for Research in Child Development*: 5-24.

Gick, M. L., & Holyoak, K. J. 1983. Schema induction and analogical transfer. *Cognitive Psychology*, 15, 1-38.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524.

He, K.; Gkioxari, G.; Dollar, Piotr.; and Girshick, R. 2017. Mask R-CNN. *arXiv preprint arXiv:1703.06870*

He, K., Zhang, X., Ren, S., and Sun, J. 2015. Deep residual learning for image recognition. arXiv:1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv: 1406.4729.

Hammond, T.; and Davis, R. 2007. LADDER, a sketching language for user interface developers. *ACM SIGGRAPH*

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *In Neural computation.*

Hu, R., Andreas, J., Darrell, T. and Saenko, K., 2018. Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 53-69).

Hudson, D.A. and Manning, C.D., 2018. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*.

Kamath, A., Singh, M., LeCun, Y., Synnaeve, G., Misra, I. and Carion, N., 2021. MDETR-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1780-1790).

Kandaswamy, S.; Forbus, K.; and Gentner, D. 2014. Modeling Learning via Progressive Alignment using Interim Generalizations. *Cognitive Science Society*, Vol. 36, No. 36.

Kiros, R., Salakhutdinov, R. and Zemel, R., 2014a, June. Multimodal neural language models. In *International conference on machine learning* (pp. 595-603). PMLR.

Kiros, R., Salakhutdinov, R. and Zemel, R.S., 2014b. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Kotovsky, L., & Gentner, D. (1996). Comparison and categorization in the development of relational similarity. *Child Development*, 67, 2797-2822.

Kunze, L.; Burbridge, C.; Alberti, M.; Tippur, A.; Folkesson, J.; Jensfelt, P.; & Hawes, N. 2014. Combining Top-down Spatial Reasoning and Botton-up Object Class Recognition for Scene Understanding. *IROS.*

Krenski, K.; and Selinger, P.; Potrace: http://potrace.sourceforge.net/.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *IEEE*, 2278-2324.

Li, J.; & Chen, J. 2016. Joint Motion Similarity (JMS)-based Human Action Recognition Using Kinect. *DICTA.*

Li, Y.; Song, Y.; and Gong, S. 2013. Sketch Recognition by Ensemble Matching of Structured Features. In *BMVC, 2013*

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. And Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 10012-10022).

Liang, C.; and Forbus, K. 2015. Learning Plausible Inferences from Semantic Web Knowledge by Combining Analogical Generalization with Structured Logistic Regression. *Proceedings of AAAI15.*

Lu, C.; Krishna, R.; Bernstein, M.; Li F.F. 2016. Visual Relationship Detection with Language Priors. *European Conference on Computer Vision., Springer, Cham.*

Lin, T. Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollar, P. 2014. Microsoft COCO: Common Objects in Context. *Springer, Cham.*

Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. 2016. Feature pyramid networks for object detection. arXiv:1612.03144

Lin, H.; Lu, P.; Fu, Y., Gong, S., Xue, X., and Jiang, Y. 2019, TC-Net for isbir: Triplet classification network for instance-level sketch-based image retrieval. ACM Multimedia, 2019.

Lin, H.; Fu, Y., Xue, X., and Jiang, Y. 2020. Sketch-Bert: Learning Sketch Bidirectional Encoder Representation from Transformers by Self-Supervised Learning of Sketch Gestalt. *Computer Vision and Pattern Recognition (CVPR)*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., and Berg, A. 2015. SSD: Single shot multibox detector. arXiv:1512.02325.

Lovett, A., and Forbus, K. 2011. Cultural commonalities and differences in spatial problem solving: A computational analysis. *Cognition* 121, pp. 281-287.

Lovett, A.; and Forbus, K. 2013. Modeling spatial ability in mental rotation and paper-folding. *Annual Meeting of the Cognitive Science Society*, 25.

Lovett, A.; Forbus, K. 2017. Modeling visual problem solving as analogical reasoning. *Psychological Review*, 124(1).

Lovett, A., Tomai, E., Forbus, K., and Usher, J. 2009. Solving geometric analogy problems through two-stage analogical mapping. *Cognitive Science*, 33(7), 1192-1231.

Lowet, A. S., Firestone, C., & Scholl, B. J. (2018). Seeing structure: Shape skeletons modulate perceived similarity. *Attention, Perception, & Psychophysics*, 80(5), 1278-1289.

Ma, X.; Bao, B.; and Yao, L; Xu, C. 2019. Multimodel Latent Factor Model with Language Constraint for Predicate Detection. *In 2019 IEEE International Conference on Image Processing (ICIP)*.

Maguire, M. J., Hirsh-Pasek, K., Golinkoff, R. M., & Brandone, A. C. (2008). Focusing on the relation: Fewer exemplars facilitate children's initial verb learning and extension. *Developmental Science*, 11(4), 628–634.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z. and Yuille, A., 2014. Deep caption with multimodal recurrent neural networks (m-rnn). arXiv. *arXiv preprint arXiv:1412.6632*.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B. and Wu, J., 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*.

Markman, A. B., & Gentner, D. (1993). All differences are not created equal: A structural alignment view of similarity. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 682-686.

Marr, David. 1982. Vision: A computational approach.

Mascharka, D., Tran, P., Soklaski, R. and Majumdar, A., 2018. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4942-4950).

McCulloch, W., and Pitts, W. 1943. A logical calculus of the ideas imminent in nervous activity.

McLure, M. D., Friedman, S. E., Lovett, A. and Forbus, K. D. 2011. Edge-cycles: A qualitative sketch representation to support recognition. *Proceedings of the 25th International Workshop on Qualitative Reasoning*. Barcelona, Spain.

McLure, M.; Friedman, S. E.; and Forbus, K. 2015a. Extending Analogical Generalization with Near-Misses. *AAAI*, 565-571.

McLure, M.; Kandaswamy, S.; and Forbus, K. 2015b. Finding Textures in Sketches using Planar Ising Models. *28th International Workshop on Qualitative Reasoning (QR2015)*, Minneapolis, MN.

Negroponte, N. 1973. Recent Advances in Sketch Recognition. *In Proceedings of the National Computer Conference and Exposition*.

Neimark, D., Bar, O., Zohar, M. and Asselmann, D., 2021. Video transformer network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3163-3172).

Paik, J. H., & Mix, K. S. (2008). It's all relative: Different levels of relational similarity used in children's comparisons. *British Journal of Developmental Psychology*, 26(4), 499–505.

Palmer, S. 1999. Vision science: Photons to phenomenology. *MIT Press.*

Perez, E., Strub, F., De Vries, H., Dumoulin, V. and Courville, A., 2018, April. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

Peyre, J.; Laptev, I.; Schmid, C.; and Sivic, J. 2017. Weakly-supervised learning of visual relations. *In ICCV.*

Radford, A., Kim, J., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. 2021. Learning transferable visual models from natural language supervision. arXiv:2103.00200.

Randell, D.A; Cui, Z; Cohn, A.G. 1992. A spatial logic based on regions and connection. *3rd Int. Conf. on Knowledge Representation and Reasoning, pp. 165-176.*

Ren, S.; He, K.; Girshick, R.; Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. 2015. You only look once: unified, real-time object detection. arXiv:1506.02640.

Richland, L. E., Morrison, R. G., & Holyoak, K. J. (2006). Children's development of analogical reasoning: Insights from scene analogy problems. *Journal of Experimental Child Psychology*, 94, 249-271.

Sagi, E.; Gentner, D.; and Lovett, A. 2012. What difference reveals about similarity. *Cognitive Science*, 36(6), 1019-1050.

Samek, W.; Wiegand, T.; and Muller, K. 2017. Explainable artificial intelligence: understanding, visualizing, and interpreting deep learning models. *arXiv preprint arXiv:1708.08296.*

Seddati, O.; Dupont, S.; and Mahoudi, S. 2015. DeepSketch: deep convolutional neural networks for sketch recognition and similarity search. In *Content-based Multimedia Indexing (CBMI), 13th International Workshop on (pp. 1-6).*

Seidenari, L.; Varano, V.; Berretti, S.; Bimbo, A.; & Pala, P. 2013. Recognizing actions from depth cameras as weakly aligned multi-part bag-of poses. *CVPRW*, (pp. 479-485).

Smith, L. B. (1993). The concept of same. In H. W. Reese (Ed.), *Advances in child development and behavior* (Vol. 24, pp. 215-252). San Diego, CA: Academic Press.

Smolensky, P., 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, *46*(1-2), pp.159-216.

Shen, Z., Liu, Z., Li, J., Jiang, Y., Chen, Y., and Xue, X. 2017. DSOD: Learning deeply supervised object detectors from scratch. arXiv: 1708.01241.

Shum, H.Y., He, X.D. and Li, D., 2018. From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, *19*(1), pp.10-26.

Shi, J., Zhang, H. and Li, J., 2019. Explainable and explicit visual reasoning over scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8376-8384).

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *ArXiv*.

Theodorakopoulos, I.; Kastaniotis, D.; Economou, G.; & Fotopoulos, S. 2014. Pose-based human action recognition via sparse representation in dissimilarity space. *Journal of Visual Communication and Image Representation, 25,* 12-23.

Thibaut, J. P., French, R., Vezneva, M. 2010. The development of analogy making in children: Cognitive load and executive functions. *Journal of experimental child psychology*. 106(1), 1-19.

Van de Weghe, N., Cohn, A., De Tre, B., & De Maeyer, P. 2005. A Qualitative Trajectory Calculus as a basis for representing moving objects in Geographical Information Systems. *Control and Cybernetics*, 97-120.

Vemulapalli, R.; Arrate, F.; & Chelappa, R. 2014. Human action recognition by representing 3D skeletons as points in a lie group. *CVPR,* (pp. 588-595).

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3156-3164).

Wang, P.; Li, W.; Li, C.; & Hou, Y. 2016. Action Recognition Based on Joint Trajectory Maps with Convolutional Neural Networks. *IEEE Transactions on Cybernetics.*

Wetzel, J. 2014. Understanding and critiquing multi-model engineering design explanations. Doctoral disertation, Northwestern University.

Wilson, J., Chen, K., Crouse, M., Nakos, C., Ribeiro, D., Rabkina, I., and Forbus, K, 2019. Analogical question answering in a multimodel informaiton kiosk. *In Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems.*

Wu, Y., Kirillov, A., Massa, F., Lo, W., and Girshick, R. 2019. Detectron2. https://github.com/facebookresearch/detectron2

Xia, L.; Chen, C.; & Aggarwal, J. 2012. View invariant human action recognition using histograms of 3D joints. *CVPRW*, (pp. 20-27).

Xu, D.; Zhu, Y.; Choy, C. B.; Li, F. F. 2017. Scene graph generation by iterative message passing. *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Page. 5410-5419.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R. and Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

Ye, J. 2016. Spatial and Temporal Modeling for Human Activity Recognition from multimodal Sequential Data

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P. and Tenenbaum, J., 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, *31*.

Yin, G.; Sheng, L.; Liu, B.; Yu, N.; Wang, X.; Shao, J.; and Change Loy, C. 2018. Zoom-net: Mining deep feature interactions for visual relationship recognition. *In The European Conference on Computer Vision (ECCV).*

Yu, R.; Li, A.l Morariu, V. I.; and Davis, L. S. 2017. Visual relationship detection with internal and external linguistic knowledge distillation. *In the IEEE International Conference on Computer Vision (ICCV)*

Yu, Q.; Liu, F.; Song, Y.; Xiang, T.; Hospedales, T.; and Loy, C. 2016. Sketch me that show. *In Proceedings of IEEE conference on computer vision and pattern recognition.*

Yuan, L., Chen, D., Chen, Y., Codella, N., …, Zhou, L., and Zhang, P. 2021. Florence: a new foundation model for computer vision. arXiv:2111.11432.

Zhuang, B.; Liu, L.; Shen, C.; and Reid, I. 2017. Towards context-aware interaction recognition for visual relationship detection. *In proceedings of the IEEE International Conference on Computer Vision*, Page. 589-598.

Zhang, H.; Kyaw, Z.; Chang, S. F.; Chua, T. S. 2017. Visual Translation Embedding Network for Visual Relation Detection. *arXiv preprint arXiv:1802.08319.*

Zhang, B.; Yang, Y.; Chen, C.; Yang, L.; Han, J.; Shao, L. 2017. Action Recognition Using 3D Histograms of Texture and A Multi-Class Boosting Classifier. IEEE.

Zhang, J.; Shih, K. J.; Elgammal, A.; Tao, A.; and Catanzaro, B. 2019. Graphical Contrastive Losses for Scene Graph Parsing. *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Page. 11535-11543.

Zhang, J.; Kalantidis, Y.; Rohrbach, M.; Paluri, M.; Elgammal, A.; and Elhoseiny, M. 2019. Large-Scale Visual Relationship Understanding. *In AAAI 2019.*

Zhang, H,. Kyaw, Z., Chang, S.F. and Chua, T.S. 2017. Visual Translation Embedding Network for Visual Relation Detection. *In CVPR 2017*.

Zellers, R.; Yatskar, M.; Thomson, S.; and Choi, Y. 2018. Neural Motifs: Scene Graph Parsing with Global Context. *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Page. 5831-5840.

Zhang, T.; and Suen, C. 1984. A fast parallel algorithm for thinning digital patterns. In *Communications of the ACM 27.3: 236-239.*