Sketch Recognition via Part-based Hierarchical Analogical Learning

Kezhen Chen[1] , **Kenneth Forbus**[2] , **Balaji Vasan Srinivasan**[3] , **Niyati Chhaya**[3] and **Madeline Usher**[2]

[1]Northwestern University
[2]Adobe Research
kezhenchen@google.com, forbus@northwestern.edu, {balsrini, nchhava}@adobe.com, usher@northwestern.com

Abstract

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. Explainability also matters, so that users can see why a system got confused about something. This paper introduces a novel part-based approach for sketch recognition, based on hierarchical analogical learning, a new method to apply analogical learning to qualitative representations. Given a sketched object, our system automatically segments it into parts and constructs multi-level qualitative representations of them. Our approach performs analogical generalization at multiple levels of part descriptions and uses coarse-grained results to guide interpretation at finer levels. Experiments on the TU Berlin dataset and the Coloring Book Objects dataset show that the system can learn explainable models in a data-efficient manner.

1 Introduction

Sketching is an important 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] and [Hammond and Davis, 2007]) but remains far from being solved. Deep learning approaches have recently achieved some impressive results [Li *et al.*, 2013; Seddati *et al.*, 2015; He *et al.*, 2016; Lin *et al.*, 2019; Lin *et al.*, 2020; Yu *et al.*, 2016] but have several drawbacks. To achieve reasonable performance, they require massive amounts of data, and many epochs, for training. In supporting sketching in real-world situations, data-efficient learning is crucial. Sketching style varies widely across people, so adaptation must not require massive data for retraining. 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 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. To support real-world applications, models of sketch understanding should provide rapid ways to correct what the models learn via understandable representations.

Analogical learning via qualitative representations has been shown as an effective approach to sketch understanding [Chen *et al.*, 2019]. It can achieve reasonable results with high data-efficiency and strong explainability. However, the analogy-based model has two problems. First, the performance of the model is still low with a small number of training samples. For example, [Chen *et al.*, 2019] can only achieve 29.47% on a sketch recognition dataset with 9 training examples per category. Second, that system encodes the whole sketch as one qualitative representation. With more encoded details, the number of facts in a sketch representation increases quickly. With a greater number of facts, analogical learning requires more computation time for learning and classification. Also, the qualitative representation of a sketch becomes more complicated and harder to explore with more facts.

This paper introduces a novel approach for sketch recognition, PHAL, based on **P**art-based **H**ierarchical **A**nalogical **L**earning. 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 and using coarse-grained results to guide the interpretation at finer levels, we show that PHAL has better accuracy than traditional analogical learning while keeping high data-efficiency and explainability. Figure 1 shows the pipeline of PHAL, as described in section 3.

This paper makes four contributions. (1) We introduce a novel approach, hierarchical analogical learning, on hierarchical encodings. To our knowledge, this is the first time that analogical learning is used in a hierarchical way. Experimental results show that hierarchical analogical learning outperforms traditional analogical learning. (2) We describe a novel multi-level part-based encoding scheme to describe the geometric information of sketched objects. This encoding scheme splits a sketch into multiple qualitative representations, which improves the scalability over just one large representation. Specifically, we also introduce a new texture-detection method to describe sketched textures. (3) Our ap-

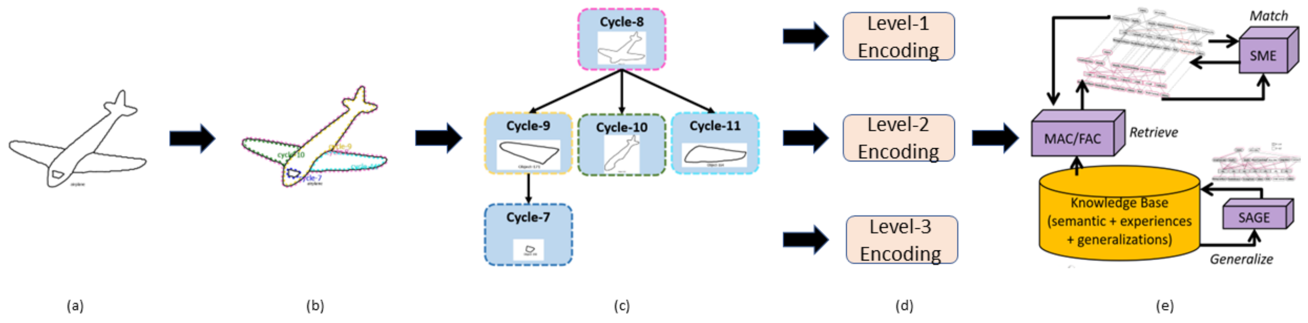


Figure 1: The pipeline of PHAL. (a) The system takes a sketched object as the input. (b) The sketch is decomposed into edges and edge-cycles. (c) Edges and edge-cycles are organized into a hierarchical decomposition tree. (d) The system constructs three-level part-based qualitative representations. (e) Hierarchical analogical learning is used to perform training and classification.

proach has better data-efficiency than deep learning models while being competitive in accuracy or outperforming them (4) Hierarchical analogical generalizations provide intuitively understandable explanations of what was learned.

We begin by introducing the most relevant related work on sketch recognition and analogical learning. Then we present PHAL, including the qualitative encoding scheme and hierarchical analogical learning. Next, we show results for experiments on the TU Berlin dataset and the Coloring Book Objects dataset. The explainability of models is examined next, ending with conclusions and future work.

2 Related Work

Sketch recognition has been studied for decades, making a detailed review beyond the scope of this paper. Consequently, we 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. [Lake *et al.*, 2015] uses Bayesian program learning (BPL) to develop a new human-like system to learn visual concepts. PHAL also uses qualitative visual relations but also computes a geon representation [Biederman, 1987] for each segmented part to provide a stable intermediate representation and uses a different approach to recognizing textures. The Ising model for texture detection only works on connected edge cycles, thereby missing many textures. By contrast, our new texture detection method can detect texture in connected or nonconnected edges or edge-cycles.

Deep learning models have achieved impressive results on sketch recognition [Li *et al.*, 2013; Seddati *et al.*, 2015; He *et al.*, 2016; Lin *et al.*, 2019; Lin *et al.*, 2020; Yu *et al.*, 2016]. They designed large-scale deep learning models and perform data augmentation or pretraining to achieve state-of-the-art performance. For example, [Lin *et al.*, 2020] designed SketchBERT, using BERT as the backbone, and pretrained it with many sketches. Then, they finetuned the model for downstream sketch recognition/retrieval tasks. However, this is the opposite of data-efficient learning, and as Section 4.2 shows, with small training datasets, deep learning models fail to achieve reasonable performance even with pretraining. Also, PHAL constructs explainable, rather than opaque, models.

Analogical learning has been used in other tasks. For example, [Crouse *et al.*, 2018] applied it to learning to answer questions via self-supervised learning, demonstrating better data efficiency and competitive accuracy compared to other state-of-the-art Geoquery systems. Similarly, [Chen and Forbus, 2018] applied analogical learning over qualitative representations of movements to perform human action recognition with competitive accuracy. The most similar work compared with our PHAL is [Chen *et al.*, 2019]. This work also uses analogical learning over qualitative representations for sketch recognition. However, it used single-level analogical learning and retrieval, whereas PHAL uses multi-level hierarchical analogical learning & retrieval, which improves the scalability of complicated qualitative representations. Also, our novel part-based encoding scheme enhances the explainability of the previous analogical learning method.

3 Part-based Hierarchical Analogical Learning

Psychological evidence from infant visual learning suggests that infants perform recognition by encoding objects from coarse-grained contours to finer details [Chen *et al.*, 2020]. This inspired us to explore using hierarchical analogical learning to perform sketch recognition. In Part-based Hierarchical Analogical Learning (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 in 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 results to generate a broad range of candidates, which are then refined by using more fine-grained models.

This section begins by describing the analogical models it uses. Then it describes CogSketch [Forbus *et al.*, 2011], an open-domain sketch understanding system used to help generate representations. The multi-level part-based representations are presented next, followed by how hierarchical analogical learning works.

3.1 Analogical Learning

Our analogy stack uses three processes. Analogical matching is handled by the Structure Mapping Engine (SME) [Forbus *et al.*, 2017], analogical retrieval by MAC/FAC [Forbus *et al.*, 1995], and analogical generalization is performed by the Sequential Analogical Generalization Engine (SAGE) [McLure *et al.*, 2015b]. We summarize each in turn.

SME is a computational model of analogical matching and similarity based on Structure Mapping Theory [Gentner, 1983]. Given two cases consisting of structured, relational representations, called the base and target, SME computes a mapping between them. A mapping includes a set of correspondences that align entities and relations in the base and target, a similarity score that indicates 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 as a similarity metric and a means of combining cases into generalizations, as described below.

The MAC/FAC algorithm models analogical retrieval. Given a probe (a case) and a library of cases, MAC/FAC retrieves cases that are highly similar to the probe from that library. When cases are added, a content vector is automatically constructed from the case, where each dimension represents the number of occurrences of a predicate in that case. The first stage, MAC, is a data-parallel map/reduce operation accumulating the best N results from the dot product of the content vectors of each library case and the probe. The corresponding structured cases are passed to FAC, which is also map/reduce, but using SME to compare the MAC outputs to the probe, returning the most similar case as the reminding. MAC provides scalability, while FAC provides human-like sensitivity to the structure. MAC/FAC is used for retrieval during both training and testing.

SAGE models analogical generalization. Each concept is represented by a generalization pool (aka gpool), which, given an incremental stream of examples, constructs a set of probabilistic generalizations and outliers that constitute an analogical model of that concept. Each item in a gpool is disjunct in the model. There are two basic operations: adding an example and classifying an example. When adding a training example to a gpool, MAC/FAC is used to retrieve the most similar item, treating the gpool as a case library. An assimilation threshold is used to determine when an example is assimilated into a generalization. If the similarity is too low, the new example is added to the gpool as an outlier. Otherwise, if the reminding is another example, then a new generalization is formed. This involves replacing nonidentical aligned entities with new unique symbols (i.e., skolems) and taking the union of the statements involved. The probability of each statement is 1 if aligned, or 0.5 otherwise. If the reminding is a generalization, its probabilities are updated based on what facts from the example align with it, and new skolems added as needed. Statements whose probability gets too low are eventually deleted. Since SAGE can accumulate multiple generalizations and outliers, it is like k-means with outliers, except that there is no a priori determination of the number of clusters: SAGE automatically derives that from the data. This paper describes a novel way to use analogical learning hierarchically, building SAGE gpools for every concept at each level.

3.2 CogSketch

CogSketch is an open-domain sketch understanding system that automatically constructs visual and spatial relational representations from digital ink, similar to what people use in visual problem-solving (e.g., [Lovett and Forbus, 2013; Lovett and Forbus, 2017]). It decomposes ink into edges and junctions by separating the ink into segments at its discontinuities and overlaps. It computes a variety of properties for edges, including their length, curvature, orientation, position, topological relations, and shape (i.e., type of junctions, such as T-junction). Edges can be assembled into edge-cycles that form closed shapes, providing larger units out of which representations of surfaces can be constructed. Edge-cycles have similar properties to edges and polygons, but, unlike polygons, can also have curved edges.

CogSketch operates directly with SVG format inputs. When the input is a bitmap, a preprocessing step is required to produce digital ink. Bitmaps are converted to digital ink in SVG format by a three-step procedure. First, the image is resized so that its longest dimension is below 300 pixels, to speed up subsequent processing. Second, the image is blurred and filtered to black and white using a threshold of 70. Finally, a combination of Potrace [Krenski and Selinger, 2003], a software tool for tracing bitmaps, and Zhang-Suen's thinning algorithm [Zhang and Suen, 1984] are used to generate an SVG file for input into CogSketch.

3.3 Multi-level Structured Representation

PHAL constructs multi-level structured representations for sketched objects. Given digital ink for an object, CogSketch generates a decomposition tree based on the containment of edges and closed edge cycles, as illustrated in Figure 1. That is, each edge or edge-cycle is stored in a tree node forming a hierarchical structure from the outside to the inside of the object. Thus, the root node contains the contour edge-cycle of the whole sketched object. Figures 1 (b) and (c) show the edge-cycles for an airplane and the corresponding decomposition tree. Cycle-8 in the root node is the outer contour edge-cycle of the airplane. As most sketches tend to only have two or three layers of encodings, we choose a three-level encoding scheme, i.e. elements of the first three levels in the decomposition tree.

The 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. In Figure 1, Level 1 is the outer contour (Cycle-8), Level 2 consists of three parts, the fuselage, and the two wings, and Level 3 consists of the windshield. Some sketches have complex textures consisting of

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**, **spindleSystemShape**
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 graph, 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 1: Edge-cycle encoding attributes

many edges or edge-cycles. Detecting and representing such textures is important for sketch recognition. Thus, we use 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 1 (d) illustrates what is included in the airplane sketch at each level. For each level, we utilize a part-based encoding scheme for representation construction. Texture encoding and part encoding are described next.

3.4 Texture Encoding

The decomposition tree is used to guide 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. Here, we introduce a novel texture encoding method using analogical generalization. We formalize the texture encoding problem as finding clusters consisting of sets of edges or edge-cycles that have similar properties. To solve it, we first compute a set of properties for the items to be clustered, and then add them to a SAGE generalization pool. Each generalization it finds is treated as a visual element representing a texture.

We use three features for edges, namely curvature, length, and direction. Curvature describes whether the edge is curved or straight, length describes the relative length of the group of edges, and direction describes whether the edge is vertical, horizontal, upward, or downward. For edge-cycles, we use six features: curvature, symmetry, shape estimation, orientation, size, and rectangularity, as described in Table 1. We use 0.7 for the SAGE assimilation threshold with a cutoff probability of 0.2 in texture encoding.

3.5 Part Encoding

We generate a set of parts for each hierarchical level based on the decomposition tree. There are three types of parts defined in the encoding: *edge part, edge-cycle part, and convex-hull part*. To generate these parts, we first filter tiny edges and edge-cycles from all components in each encoding level. The convex hull of these tiny components is regarded as the convex-hull part. Each large edge is an edge part. Each large edge-cycle is divided into qualitatively distinct pieces using a segmentation algorithm described in [Chen *et al.*, 2019]. The segmentation algorithm is inspired by [Biederman, 1987] recognition-by-components theory [Blum, 1967]. To segment an edge-cycle, we first compute the medial-axis transform, an algorithm from computer vision. Each point on the medial-axis has at least two closest points on the edge-cycle. The pairs of closest points are iterated over, to find closures of the edge-cycle, where a closure has at least one concave point relative to the edge-cycle. After finding all closures, we add a segmentation line to each closure. Each segment is regarded as an edge-cycle part. Once we have all the parts, a qualitative representation is constructed for each part to represent the local information of sketched objects. CogSketch computes geometric properties for each part and spatial relations between nearby parts. (Two parts on the same level are near to each other if they are connected or the distance between them is smaller than a threshold) 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. During encoding, the isa predicate in Cyc is used to express these properties as:

$$(isa \quad Part1 \quad \langle Properties \rangle) \tag{1}$$

where the $\langle Properties \rangle$ is a type of property, such as triangleSystemShape, or CurvedEdge. Between nearby parts, above and rightOf statements are used to represent the positional relations, e.g.

$$(above \quad Part1 \quad Part2) \tag{2}$$

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, we also encode the spatial relations between parts and textures. RCC8 relations [Randell *et al.*, 1992] 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, we add a part called Empty Part with only one fact:

$$(emptyEncoding \quad EmptyPart1) \tag{3}$$

which facilitates the prediction process introduced next.

Algorithm 1 Hierarchical Analogical Retrieval

Input: The set of first-level part representations of a sketch, P1. The set of second-level part representations of a sketch, P2. The set of third-level part representations of a sketch, P3. Concepts: C[1...n]
Parameter: The numbers of categories retrieved in each level: K, Q, V
Output: Classification Category

```
1: define function LevelRetrieval(Ci,N,P):
2:     Let $O = [], G = Gpools(C_i, N)$.
3:     for Gi in G[1...n] do
4:         T=[]
5:         for pi in P do
6:             T.append(MAC/FAC(pi,Gi))
7:         end for
8:         O.append(average(T))
9:     end for
10:    return O
11: O1=LevelRetrieval(C,1,P1)
12: C1=sort(O1)[:K]
13: O2=LevelRetrieval(C1,2,P2)
14: C2=sort(O2)[:Q]
15: O3=LevelRetrieval(C2,3,P3)
16: C3=sort(O3)[:V]
17: for ci in C3 do
18:     r1=C1[ci]*len(C1[ci])
19:     r2=C2[ci]*len(C2[ci])
20:     r3=C3[ci]*len(C3[ci])
21:     C3[ci]=r1+r2+r3
22: end for
23: result=sort(C3)[0]
24: return result
```

3.6 Hierarchical Analogical Learning on Parts

This section presents a novel approach to using analogical learning hierarchically on the representations just described. Our goal was to improve performance 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 that type of object. Classification of a new example is performed by first computing the part representations in three levels for it, and then using the Hierarchical Analogical Retrieval algorithm described in Algorithm 1. It works as follows: each part representation in the first level is used as a probe for MAC/FAC, with each of the Level-1 gpools being used as a case library. The top K categories, i.e., those with the highest average scores of parts coming out of MAC/FAC, are used as an initial pool of candidates (lines 11-12). Next, each part representation of the second level is used as a probe for MAC/FAC, but with each Level-2 gpool corresponding to the K categories found in the level-1 retrieval being used as a case library (lines 13-14). 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 (lines 15-16). Particularly discriminative information can appear at any level; hence in classification, 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 a same gpool of concepts. Thus, we compute a final score to decide the final prediction. 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 (lines 17-23). This ranked list provides the classification answers.

4 Experiments

We performed experiments on two datasets, the TU Berlin dataset [Eitz *et al.*, 2012] and Coloring Book Objects dataset [Chen *et al.*, 2019]. We compare PHAL with several baselines. In the experiments, we performed the hyper-parameter search by using a small subset of training samples from the training set as the validation set. Results show that our approach can achieve competitive or new state-of-the-art performance while retaining advantages in data/training efficiency and inspectability.

4.1 TU Berlin Dataset

The TU Berlin dataset is a sketch dataset containing 250 object categories. For each category, there are 80 hand-drawn sketches, for a total of 20,000 sketches. We 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 identify the object category of a sketch 73% of the time. We performed hyperparameters search, settling on 0.8 as the assimilation thresholds and 0.2 as the cutoff probability for all three encoding levels. On the full dataset, the numbers of categories we keep at each level are 20, 10, and 5. TU Berlin dataset provides both a bitmap version and an SVG version. For convenience, we used the SVG version so CogSketch can directly load the images.

We compared PHAL with existing baselines including traditional machine learning methods and recent deep learning models. Table 2 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 TU Berlin dataset. PHAL only looks at overall training data once

Approaches	Accuracy
[Eitz *et al.*, 2012]	56.00%
[Li *et al.*, 2013]	61.50%
[Hochreiter and Schmidhuber, 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 2: Results on TU Berlin dataset

instead of multiple epochs like deep learning models. Also, our approach only uses up to 10 CPUs to encode sketches and 1 CPU computer to perform hierarchical analogical learning. By contrast, deep learning baselines require multiple computation resources (GPUs/TPUs and CPUs) to support large model and data training. This provides evidence of the training-efficiency of our approach. Figure 2 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. We find that the training set does not have front view of an airplane nor a sketch of a bear face per se. Thus, data augmentation might be why SAN makes correct predictions in these cases. We argue that hierarchical analogical learning can achieve reasonable accuracy with good training-efficiency. Moreover, our method is incremental, an advantage that is not tested by batch-oriented datasets, but is crucial for integrating models into applications requiring adaptability.

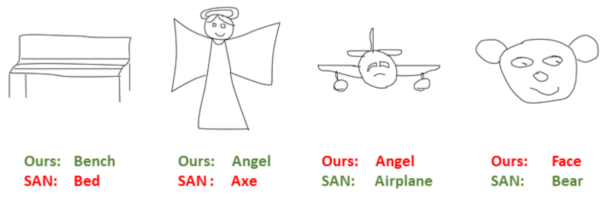


Figure 2: Our system predicts correct labels on the left two examples, but Sketch-A-Net (SAN) does not. Our system predicts incorrect labels on the right two examples, but Sketch-a-Net gets correct answers.

4.2 The Coloring Book Objects Dataset

The Coloring Book Objects dataset (CBO) [Chen *et al.*, 2019] consists of images from open-license coloring books. This dataset contains 10 bitmap examples for each of 19 different categories of animals and everyday objects. The images in each category vary widely in style (e.g., realistic vs. cartoon) and in view (e.g., profile vs. frontal). The small number of samples and high variety in contents makes this dataset a useful challenge for evaluating data-efficiency in sketch recognition. If a model has a higher performance, it has better data-efficiency on the sketch recognition task.

We use the same cross-validation method described in [Chen *et al.*, 2019] for evaluation. At each round out of ten rounds, a random image in each category is used as the testing samples and the other nine images in each category are used as training samples. Then, we compute the average accuracy on the ten rounds. After a hyper-parameter search, we use 0.7 as the assimilation threshold and 0.2 as the cutoff probability for all three levels. During hierarchical analogical retrieval, we keep the top 10, 5, and 3 categories in Levels 1, 2, and 3 respectively.

Approaches	Accuracy
CNN	5.26%
ResNet50	10.53%
ResNet50 (pretrained)	21.05%
[Chen *et al.*, 2019]	29.47%
PHAL	37.37%

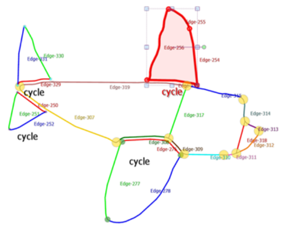
Table 3: Results on CBO dataset

Given the small number of examples, PHAL outperforms the deep learning models. We tested two different CNN-based deep learning models as baselines. First, models with fewer parameters seem more likely to converge, given the small number of training examples in this dataset. Consequently, we first evaluated a vanilla convolutional neural network (CNN) trained on it from scratch. We choose the same architecture as the LeNet [LeCun *et al.*, 1998]. The second model tests the ResNet50 baseline which is the same as in the TU Berlin experiment. We evaluated both the pretrained ResNet50 and the raw model trained from scratch. After hyper-parameter searching, vanilla CNN only achieved accuracy about chance. ResNet50 achieved 10.53% accuracy if we trained it from scratch. Pretrained ResNet50 could achieve 21.05% accuracy, which is still lower than our approach. We also compare our approach with the traditional analogical learning method described in [Chen *et al.*, 2019]. Table 3 shows the overall results for each model. These results demonstrate that our novel hierarchical analogical learning outperforms deep learning models and the traditional analogical method, providing a new SOTA for this dataset. This provides additional evidence that PHAL is good at data-efficient learning and can capture hierarchical structured information of sketched objects

5 Explainability

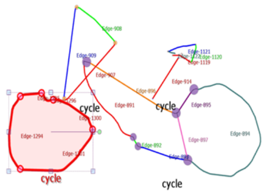
Analogical generalization provides explainability because the qualitative relational representations used are very similar to the human visual structure and are easily tied to language. Each generalization or outlier in a generalization pool represents a disjunct of the model and can be explored by users. For example, Figure 3 shows two generalizations from Airplane and Bicycle gpools. In each example, the left is a part case in the corresponding 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



Airplane-Level2: SageGen1

(areaTiny Part-Gen1)	1.0
(nonSymmetryAxesCycle Part-Gen1)	1.0
(isNeighbor Part-Gen1 Part-Gen2)	1.0
(curvedCycle Part-Gen1)	0.75
(ellipseSystemShape Part-Gen1)	0.75
(isNeighbor Part-Gen1 Part-Gen3)	0.5
(above Part-Gen1 Part-Gen2)	1.0
(above Part-Gen1 Part-Gen3)	0.5
(areaMedium Part-Gen2)	0.75
....	



Bicycle-Level1: SageGen0

(areaSmall Part-Gen1)	1.0
(ellipseSystemShape Part-Gen1)	1.0
(curvedCycle Part-Gen1)	1.0
(isNeighbor Part-Gen1 Part-Gen3)	1.0
(acutelyOriented Part-Gen1)	0.75
(hasTexutre Part-Gen1 Texture-Gen-0)	0.75
(above Part-Gen3 Part-Gen1)	0.75
(triangleSystemShape Part-Gen3)	0.625
....	

Figure 3: Figure 3: 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.

is a case of the upper wing of an airplane (the part is in red). Based on learned statements, the upper wing is a relatively small, nonsymmetric, 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 explanations are hard to extract from deep learning models. Being able to inspect the generalizations and outliers could help trainers (or systems using the models) to guide active learning.

6 Conclusion and Future Work

This paper introduces a novel approach for sketch recognition. For each sketch, a three-level part-based representation is constructed and used in analogical learning in a novel hierarchical way. The new state-of-the-art result on the Coloring Book Objects dataset shows that our system learns with high data-efficiency. The experiment on the TU Berlin dataset shows that our system is competitive in terms of accuracy with deep learning models, with higher training-efficiency. The explainability of analogical models is also a considerable advantage, as in the ability to operate incrementally, a capability not tested in machine learning datasets, but one that we believe is very important in practice.

At least two interesting directions should be explored in future work. First, can the representations be improved to achieve a new state-of-the-art performance on TU Berlin dataset? This might involve more sophisticated representations for better features and/or extending the geon representation, for example. Second, can this hierarchical analogical learning method be applied in other domains? Detecting events and relational patterns in story understanding, for instance, might be enhanced by using this technique since they already tend to use hierarchical event representations

Acknowledgments

This research was sponsored by the US Office of Naval Research under grant #N00014-20-1-2447 and by Adobe Research.

References

[Biederman, 1987] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 23(2), 1987.

[Blum, 1967] Harry Blum. *A transformation for extracting new descriptors of shape*. MIT Press, 1967.

[Chen and Forbus, 2018] Kezhen Chen and Kenneth Forbus. Action recognition from skeleton data via analogical generalization over qualitative representations. *Proceedings of the AAAI Conference*, 32, 2018.

[Chen *et al.*, 2019] Kezhen Chen, Irina Rabkina, Matthew McLure, and Kenneth Forbus. Human-like sketch object recognition via analogical learning. *Proceedings of the AAAI Conference*, 33, 2019.

[Chen *et al.*, 2020] Kezhen Chen, Kenneth Forbus, Dedre Gentner, Susan Hespos, and Erie Anderson. Simulating infant visual learning by comparison: an initial model. *Proceedings of the 42nd Annual Conference of the Cognitive Science Society*, 2020.

[Crouse *et al.*, 2018] Maxwell Crouse, Clifton McFate, and Kenneth Forbus. Learning from unannotated qa pairs to analogically disambiguate and answer questions. *Proceedings of the AAAI Conference*, 32, 2018.

[Eitz *et al.*, 2012] Mathias Eitz, James Hays, , and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics*, 2012.

[Forbus *et al.*, 1995] Kenneth Forbus, Dedre Gentner, and Keith Law. Mac/fac: A model of similarity-based retrieval. *Cognitive Science*, pages 141–205, 1995.

[Forbus *et al.*, 2011] Kenneth Forbus, Jeffrey Usher, Andrew Lovett, Kate Lockwood, , and Jon Wetzel. Cogsketch: Sketch understanding for cognitive science research and for education. *Cognitive Science*, pages 648–666, 2011.

[Forbus *et al.*, 2017] Kenneth Forbus, Ronald Ferguson, Andrew Lovett, and Dedre Gentner. Extending sme to handle large-scale cognitive modeling. *Cognitive Science*, 2017.

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

[Hammond and Davis, 2007] Tracy Hammond and Randall Davis. Ladder, a sketching language for user interface developers. *ACM SIGGRAPH*, 2007.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of IEEE conference on computer vision and pattern recognition*, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[Krenski and Selinger, 2003] Karol Krenski and Peter Selinger. Potrace. *http://potrace.sourceforge.net/*, 2003.

[Lake *et al.*, 2015] Brenden Lake, Ruslan Salakutdinov, and Joshua Tenenbaum. Human-level concept learning through probabilistic program induction. *Proceedings of IEEE*, 2015.

[LeCun *et al.*, 1998] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 1998.

[Li *et al.*, 2013] Yi Li, Song Yizhe, , and Shaogang Gong. Sketch recognition by ensemble matching of structured features. *BMVC*, 2013.

[Lin *et al.*, 2019] Hangyu Lin, Peng Lu, Yanwei Fu, Shaogang Gong, Xiangyang Xue, and Yugang Jiang. Tc-net for isbir: Triplet classification network for in-stance-level sketch based image retrieval. *ACM Multimedia*, 2019.

[Lin *et al.*, 2020] Hangyu Lin, Yanwei Fu, Xiangyang Xue, and Yugang Jiang. Sketch-bert: Learning sketch bidirectional encoder rep-resentation from transformers by self-supervised learning of sketch gestalt. *Proceedings of IEEE conference on computer vision and pattern recognition*, 2020.

[Lovett and Forbus, 2013] Andrew Lovett and Kenneth Forbus. Modeling spatial ability in mental rotation and paper-folding. *Proceedings of the Annual Conference of the Cognitive Science Society*, 2013.

[Lovett and Forbus, 2017] Andrew Lovett and Kenneth Forbus. Modeling visual problem solving as analogical reasoning. *Journal of Psychological Review*, 2017.

[McLure *et al.*, 2015a] Matthew McLure, Scott Friedman, and Kenneth Forbus. Extending analogical generalization with near-misses. *Proceedings of the AAAI Conference*, 2015.

[McLure *et al.*, 2015b] Matthew McLure, Subu Kandaswamy, and Kenneth Forbus. Finding textures in sketches using planar ising models. *28th International Workshop on Qualitative Reasoning*, 2015.

[Negroponte, 1973] N. Negroponte. Recent advances in sketch recognition. *Proceedings of the National Computer Conference and Exposition*, 1973.

[Randell *et al.*, 1992] David Randell, Zhan Cui, and Anthony Cohn. A spatial logic based on regions and connection. *KR*, 1992.

[Seddati *et al.*, 2015] Omar Seddati, Stephane Dupont, and Said Mahmoudi. Deepsketch: Deep convolutional neural networks for sketch recognition and similarity search. *13th international workshop on content-based multime-dia indexing*, 2015.

[Yu *et al.*, 2016] Qian Yu, Feng Liu, Yizhe Song, Tao Xiang, Timothy Hospedales, and Chen-Change Loy. Sketch me that show. *Proceedings of IEEE conference on computer vision and pattern recognition*, 2016.

[Zhang and Suen, 1984] Tongjie Y. Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 1984.