Human-like Sketch Object Recognition via Analogical Learning

Kezhen Chen, Irina Rabkina, Matthew D. McLure and Kenneth D. Forbus

Northwestern University

{KezhenChen2021 | irabkina | mclure}@u.northwestern.edu | forbus@northwestern.edu

Abstract

Deep learning systems can perform well on some image recognition tasks. However, they have serious limitations, including requiring far more training data than humans do and being fooled by adversarial examples. By contrast, analogical learning over relational representations tends to be far more data-efficient, requiring only human-like amounts of training data. This paper introduces an approach that combines automatically constructed qualitative visual representations with analogical learning to tackle a hard computer vision problem, object recognition from sketches. Results from the MNIST dataset and a novel dataset, the Coloring Book Objects dataset, are provided. Comparison to existing approaches indicates that analogical generalization can be used to identify sketched objects from these datasets with several orders of magnitude fewer examples than deep learning systems require.

Introduction

Deep learning approaches have, in recent years, become very popular within artificial intelligence. This excitement is reasonable given that such systems often do provide impressive results given enough training data. On the MNIST handwritten digit recognition task (LeCun et al., 1998), for example, several convolutional neural network techniques have achieved error rates below 0.3% (e.g. Ciresan et al., 2011; Ciresan et al., 2012).

This result is remarkably close to the estimated human error rate of 0.2% on this dataset (LeCun et al., 1998) and even better than the human classification accuracies from two experimental trials 96.8% and 97.8% (Harding et al., 2018). However, the approach itself is not at all human-like. The network with the lowest error rate (Ciresan et al., 2012), 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.

Human learning is also more stable. Neural networks are easily fooled: not only do state of the art neural networks classify white noise as, for example, a robin with extremely high confidence (Nguyen, Yosinski, and Clune, 2015), but slight perturbations to correctly classified images—called adversarial examples—can cause a neural network to no longer classify the image correctly (Szegedy et al., 2013; Goodfellow, Shlens, and Szegedy, 2015; Carlini and Wagner, 2017). These perturbations are small enough that a human cannot detect them, let alone be fooled by them.

Furthermore, convolutional neural networks do not model human spatial cognition. When classifying images, they learn discriminative patterns that are driven by low-level relationships between nearby pixels. On the other hand, vision psychologists have ample evidence for structured, relational models in human vision (Marr, 1982; Palmer, 1999). Perhaps computational approaches that are based on structured, relational information can demonstrate human-like learning, in terms of both number of examples and stability, within and across datasets.

Indeed, it has been demonstrated that such models can match human learning on tasks that involve higher order cognition. For example, Kandaswamy, Forbus, and Gentner (2014) showed that the Structure Mapping Engine (SME) can match the learning performance of 4-year-olds on a higher-order pattern matching task. Similarly, Lovett and Forbus (2013) showed that CogSketch (Forbus et al., 2011), a model of human visual perception that relies on relational structure, can solve mental rotation and paper folding tasks using SME. Furthermore, these representations and analogical comparison have been used to model human performance on several visual problem-solving tasks, including Ravens' Progressive Matrices (Lovett &

Forbus 2018), performing at the 75th percentile, which is better than most adult Americans.

This paper demonstrates that the combination of the human-like visual system of CogSketch and analogical generalization can also perform sketched object recognition. Sketch data has high variability and is relatively hard to collect in large quantities (Eitz, Hayes and Alexa, 2012), which is likely to pose problems for deep learning models. However, we show that our approach has reasonable recognition results on two different types of sketches despite having only sparse data with high variability.

We begin by describing our approach, including data encoding and analogical learning. We then present results for experiments on object recognition in sketches using two datasets: the MNIST handwritten digit dataset (LeCun et al., 1998) and a novel dataset, the Coloring Book Objects dataset, which consists of drawings of everyday objects and animals. Finally, we look at related work and discuss future directions for this line of research.

Approach

Sketch understanding can start with digital ink or bitmaps. Here we start 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 spatial representations. The second step is analogical learning using the relational representations as cases. Figure 1a shows a sketch of a fish from the coloring book dataset, used as an example.

Bitmap to Structured Representation

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

Bitmap Preprocessing

Given a sketched object bitmap, we first convert the sketches to ink vectors for further processing. To reduce noise and speed up the encoding algorithm, each original image is resized such that the resized length is below 300 pixels. Then, the image is blurred and filtered to black and white using a threshold of 70. Potrace, a software tool for tracing bitmaps and Zhang-Suen's thinning algorithm (1984), are used to generate SVG for input into CogSketch.

CogSketch Visual Processing

CogSketch (Forbus et al., 2011) is an open-domain sketch understanding system that automatically constructs relational representations based on visual and conceptual information. CogSketch is capable of computing spatial properties (attributes and relations) at multiple representational levels on digital-ink sketches. Its basic level representations concern *glyphs*, which are visual objects. Glyphs are 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 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, this is somewhat under constrained.

Object Segmentation

Each digital-ink sketch imported into CogSketch is decomposed into closed edge-cycles and edges. The edge-cycles and edges are sorted and stored in a *decomposition tree*. From outside to inside of the object, each edge or closed edge-cycle is stored in a node of the decomposition tree from root to leaves, so the root node contains the contour edge-cycle of the whole sketched object. Figure 1b shows the edge-cycle decomposition of the sketched fish depicted in Figure 1a. Figure 1c shows the corresponding

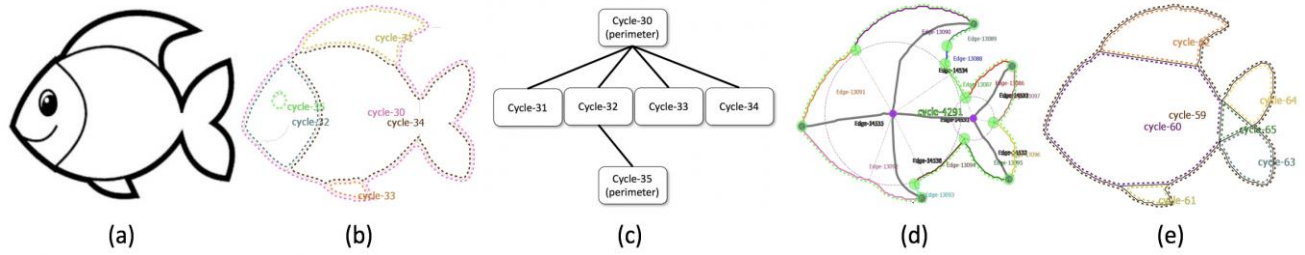

Figure 1: (a) is a sketched fish from Coloring book dataset. (b) is the edge-cycle decomposition of (a) from CogSketch. (c) is the corresponding decomposition tree. (d) is the medial-axis transform of the contour edge-cycle of (a). (e) is the segmentation result.

decomposition tree. Note that the inner edge-cycles correspond to the eye, fins, body, and head of the fish.

To represent properties of edge-cycles, we draw on Biederman's (1987) recognition-by-components theory, that people seem to encode visual input as a combination of simple shapes. Thus, each edge-cycle is segmented and described by several attributes. For example, the medial-axis transform is found by computing the grassfire transform (Blum, 1967). For each medial axis point, a pair of closest points on the edge-cycle is generated. 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.

To reduce segmentation noise, we add 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 smallest angle sum is detected in a certain range (i.e. $1/20^{th}$ 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. Figure 1d shows the medial-axis transform of the Figure 1a contour. Figure 1e shows the edge-cycle segmentation of the Figure 1a contour. Notice that each edge-cycle in the decomposition tree (Figure 1c) is segmented into several pieces for later encoding. There are five closures detected in Figure 1e.

Spatial Encoding

After decomposition and segmentation are completed, the spatial relations between edge-cycles and segments are encoded. Each edge-cycle is described as a combination of the attributes of its segments, as well as the positional relations and connection relations between segments.

Attribute selection for segments 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, we use 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 Geons (Biederman, 1987). All attributes are converted to a qualitative size description (i.e., small, medium and large) 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 EdgeCycle-Seg-149 HighSolidity)
```

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 second edge-cycle. For example:

```
(segmentsConnectToViaEdge
    (leftOfEdgeOfCycleFn EdgeCycle-Seg-149)
     EdgeCycle-Seg-152)
```

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.

The set of entities, attributes, and relations computed for a sketched object are combined to form a case, for use in analogical learning and classification, as described next.

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

Analogical Learning

Human learning is broad and general, while being data-efficient. An important advantage of our approach is that we are using off-the-shelf analogical learning models, without modification, for this task. The analogical processing models introduced below have considerable psychological evidence supporting them and have been used for building AI performance systems for a variety of tasks (Forbus & Hinrichs, 2017).

Structure Mapping Engine (SME)

The Structure Mapping Engine (Forbus et al., 2017) is a computational model of analogical matching and similarity based on Structure Mapping Theory (Gentner, 1983). Given two cases of structured, relational representations, called a *base* and a *target*, SME computes one (or up to three) *mappings* 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 both as a similarity metric and as a means of combining cases into generalizations, as described below.

MAC/FAC

The MAC/FAC algorithm (Forbus, Gentner, and Law, 1995) is a model of analogical retrieval. Given a *probe* case and a *case library*, it retrieves up to three examples from the case library that are 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 for it 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[1]. 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, which is used as a pre-filter. The MAC stage is a map/reduce operation, where dot products for a content vector of the probe is computed in parallel with the vectors for all items in the case library, with the top three scoring cases passed on to the FAC stage as output. The FAC stage also is map/reduce but using SME on the probe and the retrieved cases, keeping the best (or up to all three, if they are very close to the top). The MAC stage provides scalability, since vector dot products are quite cheap. The FAC stage provides the sensitivity to structure that human retrieval demonstrates, probably because structural similarity leads to useful conclusions. Each case returned from FAC is called a *reminding*. We use MAC/FAC for retrieval during both training and testing, as described below. Only the top reminding is used.

SAGE

The Sequential Analogical Generalization Engine (SAGE; McLure et al., 2015a) is a model of analogical generalization. Each concept to be learned by analogy is represented by a *generalization pool*, which potentially holds both generalizations and outlying examples. The generalizations and examples in a concept's generalization pool represent alternative models of the concept. There are two basic operations: adding an example and classifying an example.

Here we assume that each training example is labeled and added to a single SAGE generalization pool. The example is merged to the most similar case retrieved from the pool via MAC/FAC. If nothing is retrieved, or the similarity score associated with the top retrieval is below the *assimilation threshold*, 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*, 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 new skolems, and updating the probabilities for each statement (statements whose probability gets 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 probe is the new 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.

Experiments

We test our approach on two very different sketch datasets, using a relatively small number of training examples.

Experiment 1: MNIST

The MNIST handwritten digit dataset (LeCun et al. 1998) is constructed from NIST's Special Database 3 and Special Database 1. It consists of 60,000 training images and 10,000 testing images of handwritten digits. Each image is a 20x20 pixel bitmap centered on a 28x28 pixel field. We use randomly-selected subsets of 10, 100, and 500 images for training and the full test set.

Method

All examples were converted into relational representations using the CogSketch pipeline described above. A SAGE assimilation threshold of 0.9 was used in all experiments. For each training set size, three random subsets were se-

[1] In a large knowledge base like OpenCyc, this leads to very sparse vectors, since there are on order of 10-100 non-zero dimensions out of roughly 10^5.

lected, and the system was trained on each subset and tested on the full test set.

Results

See Figure 2 for the average accuracy per digit and Table 2 for overall accuracy and standard deviation information for each training set size compared with the LeNet-5 (LeCun et al., 1998). While the LeNet-5 also performed well using less than the full MNIST training dataset, note that it learned over 20 iterations. Our approach saw each example once.

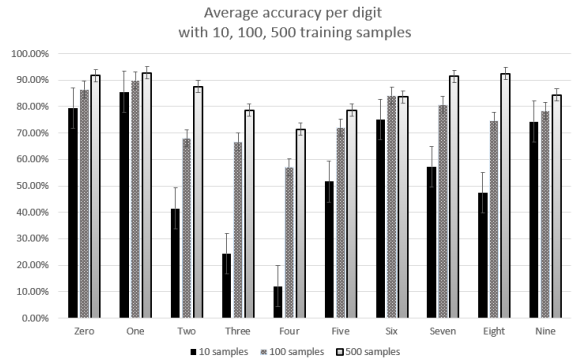

Figure 2: Average accuracy per digit

Methods	Training Size (per digit)	Overall Accuracy
Our Approach	10	54.9%
Our Approach	100	76.24%
Our Approach	500	85.03%
LeNet-5	1500 x 20 iter	98.3%
LeNet-5	6000 x 20 iter	99.2%

Table 2: Accuracy and standard deviation per training size

Experiment 2: Sketched Object Recognition

Dataset

We created the Coloring Book Objects dataset[2] (hereafter CBO) by collecting images from a collection of open-license coloring books. It contains 10 bitmap examples for each of 19 different categories of animals and everyday objects. Each image is a roughly 900x550 pixel field. The images in each category have very high variety including style (e.g. realistic vs. cartoon) and view (e.g. profile vs. frontal). Figure 4 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 3 illustrates, they provide significant variability nonetheless.

Figure 3: A subset of examples in CBO

Method

We 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. As some animals or objects have texture or noise, only closed edge-cycles as perimeters are encoded into representations. Each animal or everyday object category has a generalization pool. A SAGE assimilation threshold of 0.9 assimilation is used and average accuracy is computed.

As a baseline, we compare to results of the CNN model LeNet-5 (LeCun et al., 1998) trained using the same leave-one-out cross-validation technique. The model has 2 convolution layers with a ReLU activation followed by max-pooling layers and a fully connected layer with softmax. This CNN model performed above 99% accuracy on the full MNIST training set.

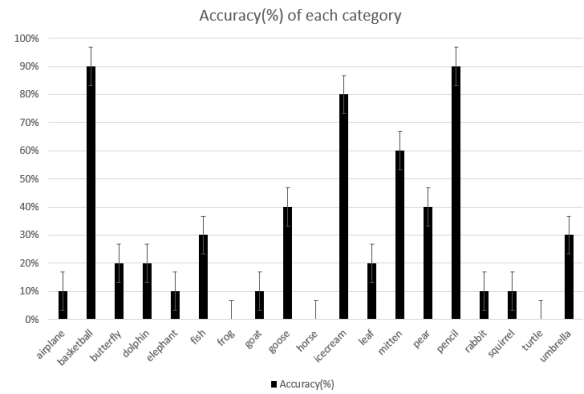

Figure 4: The accuracy (%) for each category

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

Results

Figure 4 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.

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

Table 3: Overall Accuracy and standard deviation results

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 until the system was given 1,000 examples per concept on standard MNIST inputs. We did not get very competitive results with the state-of-art because the MNIST dataset is a highly down-sampled, to fit the constraints of CNNs at the time, which introduces significant amounts of noise. Even though the preprocessing stage removes some noise, the object segmentation stage and the attributes CogSketch computations for segment edge-cycles still have bias or errors. Thus, some images have similar segmentations to other digits. For example, Figure 5 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.

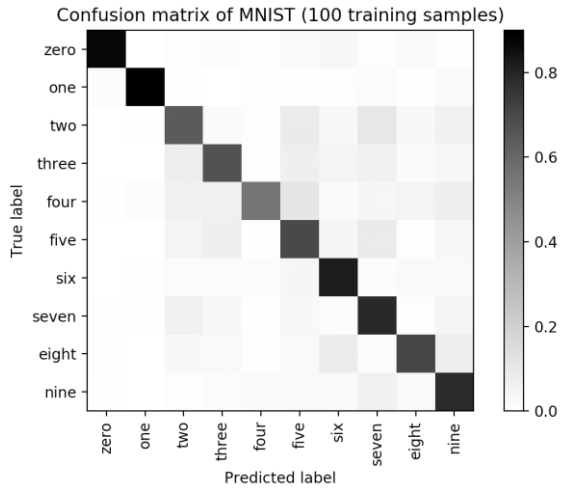

Figure 5: Confusion matrix of MNIST (100 training samples)

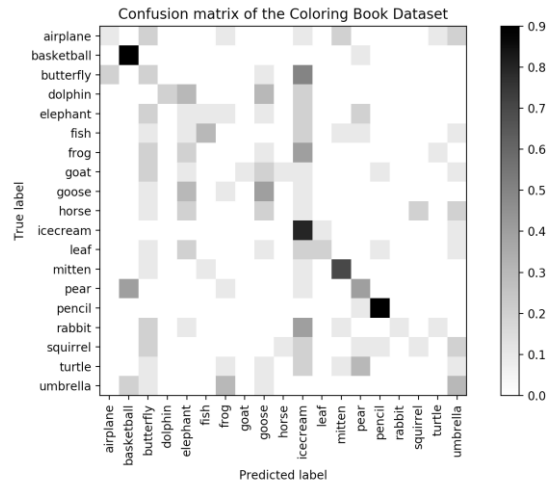

Figure 6: Confusion matrix of CBO classification results

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. For example, Figure 5 shows that some nines are recognized as fours. This is because the computed attributes of the edge-cycles in these images sometimes cannot distinguish between the upper triangle of a four and the upper circle of a nine—both are closed edge cycles. Our results might be better with the original NIST dataset, but we 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 6 shows the confusion matrix for this dataset. It shows that 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.

While our 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 structured relational representations and off-the-shelf analogical learning models can be used to produce systems that learn to recognize object from sketches in more hu-

man-like ways, with far better data efficiency than deep learning models.

Related Work

Here we discuss four existing approaches on learning sketched objects. We highlight where these approaches overlap with ours, and how they differ.

Eitz, Hays and Alexa (2012) created a large dataset of human object sketches containing 80 sketch bitmaps per category from 250 different categories, called the Berlin dataset. They represented sketches using local feature vectors that encode distributions of image properties. Specifically, the distribution of line orientation within a small local region of a sketch is encoded. With the local feature vectors, they partitioned the vectors into k disjunct clusters via k-means clustering. A frequency histogram of the k clusters is generated as the feature representation of sketches. With the frequency histograms, KNN and SVM were tested on the whole dataset. KNN could achieve around 25% accuracy with 10 training examples and 43% accuracy with 80 training examples. SVM could reach 31% accuracy with 10 training examples and 55% accuracy with 80 training examples. Prior work with analogical generalization on this dataset (McLure et al., 2015b) achieved similar levels of performance on a subset of that database by introducing an Ising model to handle textures over edge-cycles. The integration of Biederman's recognition-by-components model with CogSketch encoding, introduced here, could be combined with texture encoding to improve performance on this dataset as well.

Seddati, Dupont, and Mahmoudi (2015) presented a deep convolution neural networks (ConvNets) model for sketch recognition, which they tested on the Berlin dataset. The model contains 15 layers, which are a combination of convolution layers with ReLU followed by Maxpool layers. Each sketch is rescaled from 1x1111x1111 to 1x180x180 and the black and white pixels are reversed. During each iteration of training, 64 samples from 64 different sketches categories were randomly selected. With 0.1 learning rate and a momentum equal to 0.9, the model could reach 75.42% average accuracy after 80 epochs (epoch = 13056 examples presented to the ConvNet). While this accuracy on that corpus is impressive, it uses far more data than people require on such tasks.

On the other hand, Lake et al. (2015) used Bayesian program learning (BPL) to learn to recognize handwritten symbols and generate new, similar examples after seeing only one example of each symbol, using their framework. Symbols were represented as probabilistic programs—sequences of movements based on pen strokes. Having seen a single such example of a symbol, the model was able to match human learning on a similar symbol matching task. It was also able to generate similar images that humans matched to the original with high fidelity. Unfortunately, it is well-known in handwriting recognition that stroke data is easier to recognize than bitmap data, so we do not see it as applicable here.

Dai and Zhou (2017) used an approach that combined logical abduction and statistical induction (LASIN) to learn encodings for hand-written symbols from several datasets. For each dataset, LASIN learned dictionaries of primitive concepts combined with background knowledge, such as strokes or ink clusters, that were then used for encoding the symbols. The utility of the learned dictionaries was tested using support vector machines (SVM) with a linear kernel. MNIST was one of several datasets used for testing. With a dictionary of 200 strokes, an SVM reached 97% accuracy on 5-fold cross-validation of a randomly selected subset of 1000 MNIST training examples (100 per digit). A dictionary of 20 strokes achieved approximately 91% accuracy on this task. While our approach differs in terms of its encoding strategy and learning method, Dai and Zhou's results provide evidence that thousands of training examples are not necessary for robust learning. Rather, it is important to determine appropriate representations for the forms being learned. We argue that the representations used by humans are a good place to start.

Conclusions and Future Work

We have shown that analogical learning over relational representations is a viable and promising path for sketch recognition. Our approach is based on a human-like encoding scheme and achieves solid results with a very small number of training examples on different types of sketches. We 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.

While the data efficiency of analogical generalization is already very encouraging, we plan several lines of future work to improve it further. First, we plan to explore dynamic attribute selection, using statistics gleaned from SAGE to control the choice of attributes in subsequent encoding. Second, we plan to integrate texture representations, as per McLure et al (2015b). Finally, we plan on using near-miss learning (McLure et al. 2015a), which provides additional discrimination to analogical generalization and has been beneficial in other datasets.

Acknowledgments

This research was supported by the Machine Learning, Reasoning, and Intelligence Program of the Office of Naval Research.

Reference

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.

Cantoni, V. 1994. Human and Machine Vision.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. *Security and Privacy (SP) IEEE Symposium on. IEEE.*

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.

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

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

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

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

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.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *ArXiv*

Harding, S. M.; Rajivan, P.; Bertenthal, B. I.; Gonzalez, C. 2018. Human Decisions on Targeted and Non-Targeted Adversarial Samples. *In 40th Annual Meeting of the Cognitive Science Society (CogSci 2018)*, pp. 25-28.

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

Lake, B. M.; Salahutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266), 1332-1338.

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

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).

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

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.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep Neural Networks are Easily Fooled: High Confidence Pre-dictions for Unrecognizable Images. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 427-436.

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

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.*

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).*

Su, B. 1983. Affine Differential Geometry. *CRC Press.*

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

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