Meta-interpretive Learning from Fractal Images

Daniel Cyrus(✉), James Trewern, and Alireza Tamaddoni-Nezhad

Department of Computer Science, University of Surrey, Guildford, UK
{d.cyrus,jt00988,a.tamaddoni-nezhad}@surrey.ac.uk

Abstract. Fractals are geometric patterns with identical characteristics in each of their component parts. They are used to depict features which have recurring patterns at ever-smaller scales. This study offers a technique for learning from fractal images using Meta-Interpretative Learning (MIL). MIL has previously been employed for few-shot learning from geometrical shapes (e.g. regular polygons) and has exhibited significantly higher accuracy when compared to Convolutional Neural Networks (CNN). Our objective is to illustrate the application of MIL in learning from fractal images. We first generate a dataset of images of simple fractal and non-fractal geometries and then we implement a technique to learn recursive rules which describe fractal geometries. Our approach uses graphs extracted from images as background knowledge. Finally, we evaluate our approach against CNN-based approaches, such as Siamese Net, VGG19, ResNet50 and DenseNet169.

Keywords: Meta Interpretive Learning · Learning recursive rules · Few-shot learning · Computer Vision · Fractals

1 Introduction

The concept of fractals originated from a paper by Benoit Mandelbrot [16], in which he discussed the paradox of measuring coastlines. The length of line segments used to represent a coastline greatly affects the measured length, the smaller the line segment the greater the measured length, and vice versa. This idea of the infinite complexity of the boundary (the change in detail over the change in scale) introduced the concept of a fractal dimension, which the name fractal was derived from. One of the earliest examples of a pure mathematical fractal is the Mandelbrot set [17], calculated by taking a value on the complex plain and recursively applying the function:

$$Z_{new} = Z_{old}^2 + C \tag{1}$$

In the Eq. 1, Z and C represent complex numbers, and the iterative process commences from an initial value set to zero. As this formula is repeatedly executed, it produces a sequence of complex numbers. By converting a grid point into a complex value C and tracking the iteration count using Z_{old} while evaluating the threshold range, typically -2 to 2, for Z_{new}, we can determine whether the point resides within the fractal or lies outside of it.

In nature, fractals are generally found in branching or spiral patterns (see Fig. 1). Examples of branching patterns include river beds, lightning, tree growth, and blood

E. Bellodi et al. (Eds.): ILP 2023, LNAI 14363, pp. 166–174, 2023.
https://doi.org/10.1007/978-3-031-49299-0_12

Fig. 1. The presence of fractals in nature and their inherent recursive nature. This recursive iteration persists as each component divides into smaller components until reaching the desired level of intricacy or intricateness.

vessels. Examples of spiral patterns can be found in both the animal and plant kingdom as well as fluid dynamics, and galaxies. As many patterns in nature can be described as fractals, any generalisable computer vision approach which can capture fractal geometries would be useful for describing these structures. The primary technique for detecting fractals is through the use of fractal dimension (FD), While FD can account for irregular shapes within its range. Furthermore, there is ongoing research in the field of evaluating fractal dimension [8], though it may not be suitable for all types of fractals.

To fully capture the essence of a fractal, an illustrative subject is needed for its definition. Our primary focus in this paper lies in examining Sierpinski fractals through the extraction of graphs from images. Sierpinski fractals possess a well-defined structure characterised by self-similarity, and their iteration can be assessed through computational analysis. Examples of Sierpinski fractals are shown in Fig. 2. We show how to create logical facts about images which can then be utilised by meta-interpretive learning for generating rules describing the structure and internal relations of these images. This approach allowed us to learn recursive rules describing fractals using only information extracted from pixel data.

2 Related Work

Fractals and Deep Learning. In the field of visual object recognition, recent studies have utilised Deep Learning to pre-train models from randomly generated fractal images. The authors of [22] introduced a fractal learning approach which uses a gradient descent algorithm to acquire the parameters that govern a fractal image. Larsson et al. [14] introduced a strategy for CNN to classify fractals. While previous work based on Deep Learning predominantly relies on 'black-box' classifiers, our approach can generate rules describing fractals and their intricate patterns.

Fractal and Inversion. The task of determining the parameters associated with a given fractal image has been an unresolved challenge for a considerable period of time, Vrscay et al. [23] provided an inverse method to find iterated function system (IFS) for fractal construction. In an earlier study [3], researchers explored the solution to the inverse problem of fractal trees, aiming to describe complex patterns using the principle of contraction mapping.

Fig. 2. (a) Quantum density of electrons on the surface of copper can be characterised as a Sierpinski fractal [5, 12] (b) Examples of Sierpinski fractals from our dataset

Other Methods. A technique to gauge fractal pattern utilising self-similarity with fractal dimension proposed by Luis et al. [15]. Bölviken et al. [4] proposed their work on geochemical and geophysical data to analyse their fractal properties. A neuro-symbolic approach [1] leverages logic programming to represent iterated function systems (IFS). In their research, recurrent radial basis function networks are employed to approximate a fractal's graph. They introduced a technique for generating a logic program from an IFS, although it does not involve direct learning from images.

3 Methodology

Our methodology for learning from fractal images involves feature extraction from the images and the creation of a background knowledge file. Subsequently, our learning algorithm detects shapes along with their distinct characteristics, culminating in the creation of a recursive rule through the utilisation of our MIL technique. Employing MIL offers the benefit of crafting a recursive rule with minimal complexity, enabling the resulting rule to succinctly depict the structure of fractals. We demonstrate our approach in the task of learning rules for Sierpinski triangle fractals.

3.1 Feature Extraction

Edge Discovery. The method for edge discovery was adapted from the logical vision approach [7]. Edges were discovered using a stochastic algorithm.

Firstly a collection of random edge points in an image would be found. Edge points were discovered through the application of a Sobel filter over the image. With the inclusion of an adjustable constant to act as a threshold value, each pixel above this value was classed as an edge point.

Edges would then be conjectured between two randomly selected edge points from this set, if an edge was found it was extended and then added to a set of edges and any consumed edge points would be removed from the first set. In the case where a conjectured edge failed a line which intersects the line segment between the two selected points would be created, and all edge points along this would be added to the set of edge points. This process was iterated until no edge points remained that were not consumed by a discovered edge.

Network Creation. In this section we describe the graph construction from the found edges. To achieve this, first, endpoints of edges close to one another were grouped together, to form nodes. Next, if any edge exists between these nodes representing groups of endpoints then these edges are added to the network. As there can be only one edge between nodes and the graph is bidirectional, this step needs no additional checks to avoid duplicate edges. Each node in this resulting graph was given a position property, calculated from the average position of the constituent endpoints.

Once this network was created the next step was to split edges where they passed through nodes, by checking each edge for nodes which lie on that line segment it describes. The nodes that lay on the edge were ordered by their distance from one node of the edge in case there was more than 1. Finally, the original edge was deleted and 2 or more new edges were created which when joined together formed this original edge.

Finally, nodes that were considered redundant were removed. Redundant nodes were defined as nodes that have only 2 edges when the angle between those edges is below a given threshold value. Once these nodes were found they could be removed and the edges which passed through the node could be merged into one edge.

3.2 Dataset and Background Knowledge

The dataset[1] comprises images with a resolution of 1000×1000 pixels, categorised as either positive or negative examples of Sierpinski fractals. Each constituent shape of the fractals was given a different fill colour randomly, and the rotation of the image was also randomly selected, along with the depth of the recursion giving us fractals of different orders. These random factors allowed for the generation of many different versions of the same fractals to be produced. Examples of Sierpinski fractals from our dataset are shown in Fig. 2.b. During the feature extraction phase, we create a list of background knowledge that includes factual information. Within each image, there are multiple nodes, each node characterized by its X and Y coordinates.

Listing 1: Sample background knowledge for a Sierpinski triangle

```
img_nodes(img_45, [n45_0, n45_1, n45_2]).
node_pos(n45_0, [916, 494]).
node_pos(n45_1, [367, 103]).
node_pos(n45_2, [288, 761]).
edge(n45_0, n45_1). edge(n45_0, n45_2).
edge(n45_1, n45_0). edge(n45_1, n45_2).
edge(n45_2, n45_0). edge(n45_2, n45_1).
```

Shape Analysis. To analyse the extracted nodes from the graph and measure their distance with positive samples like triangles, rectangles, and trees, we utilise Frechet Distance [9]. Our approach involves implementing Convex Hull [2] to identify the

[1] The dataset and the source codes used in this paper are available from https://github.com/hmlr-lab/MIL-Fractals.

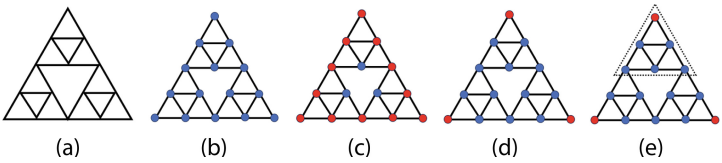

Fig. 3. (a) Feature extractions from Sierspenski fractal (b) we first extract nodes from each edges (c) a convex hull algorithm is employed to detect bounding area, (d) Douglas-Peucker algorithm removes extra nodes and detects corners (e) triangles connected to each nodes are re-scaled bigger to cover all inner nodes.

bounding region, followed by the application of the Douglas-Peucker algorithm [20] to simplify the nodes and locate corners. Ultimately, we employ the Frechet Distance technique to detect shapes. Please refer to Fig. 3 for further details. As shown in the code below, shape *A* is detected as a triangle using the predicate *shape* which calls predicate *shapeAnalysis* to match *A* with a known *shape* based on Frechet Distance.

Listing 2: Shape detection predicates.

```
shape(A, Shape):- convexHull(A, Nodes),
                  shapeAnalysis(Nodes, Shape).
innerShapes(A, B):- convex(A, Hull),
                    ramerDouglasPeucker(Hull, Corners),
                    innerShapes(A, Corners, B).
singleShape(A):- len(A, Len), Len =< 4.
```

3.3 Meta-interpretive Learning (MIL)

Meta-Interpretive Learning (MIL) is a form of Inductive Logic Programming (ILP) [18,19]. The learning process of MIL involves utilising a Prolog meta-interpreter to formulate hypotheses using background knowledge and training examples. This subsection elucidates our approach using a specific MIL implementation called Metagol.

Metagol. Metagol [6] is an implementation of MIL which uses background knowledge along with meta-rules to learn logic programs from positive and negative examples.

Metarules. When describing meta-rules for use by Metagol there a 3 required arguments, Subs, Head, and Body. Optionally the 1st argument can be a name for the meta-rule.

```
metarule(const_cond, [P,Q,B],[P,A],[[Q,A,B]]).
```

Symbols included in the subs list are substituted for some constant value such as an atom or predicate name, any symbols not included are considered variables for the rule built using the meta-rule. The head and body describe the two parts of the clause created using the meta-rule, where the head is implied by the body (Table 1).

Table 1. Table of Metarules used

Name	Substitutions	Meta-Rule
property	P,Q,R	$P(A) \leftarrow Q(A,B), R(B)$
name	P,Q,R,B	$P(A,B) \leftarrow Q(A,B), R(A)$
name	P,Q,R,F	$P(A) \leftarrow Q(A,B), F(B,R)$
name	P,Q,R,B	$P(A) \leftarrow Q(A,B), R(A)$
Identity	P,Q	$P(A) \leftarrow Q(A)$

Integrated Background Knowledge (IBK). Often, for learning simpler programs, standard dyadic metarules which are commonly used are good enough, in the case of attempting to learn more complex programs sometimes higher-order logic can be used. Integrated Background Knowledge (IBK) allows Metagol to learn higher-order logic programs, using predicates as arguments. This is especially useful when operating on lists which applies to the task of learning from lists of nodes, or lists of cycles. Useful predicates for IBK include quantification predicates *any* and *all*. The predicate *any* implements existential quantification, returning true if any element in a list returns true for a given predicate. This can also be extended by giving a second argument to pass a relational predicate to this higher-order predicate:

$$\exists a \in List(P(a)), \exists a \in List(P(a,b)) \tag{2}$$

The predicate *all* implements universal quantification, holding true only if all elements in a list are true for the given predicate. Also similarly to *any*, a second argument can be used to allow for relational predicates:

$$\forall a \in List(P(a)), \forall a \in List(P(a,b)) \tag{3}$$

3.4 Comparing with Neural Networks

To evaluate the performance of our method we compared our results against several neural networks, VGG, Densenet, Resnet, and Siamese Net. VGG is a deep convolutional network that performed well in the ImageNet 2014 challenge [21]. Resnet extends traditional Convolutional networks with residual connections skipping over layers allowing for deeper networks with less gradient loss [10]. Densenets improve upon residual networks by introducing the concept of densely connected residual blocks which allow them to achieve high performances with less required training parameters [11]. Deep convolutional neural networks can achieve high levels of accuracy when given large numbers of images to train on. Conversely, Siamese neural networks can be trained for one-shot image recognition [13].

4 Experiments

In this section we examine the following null hypotheses in order to evaluate our proposed MIL-based approach described in the previous section.

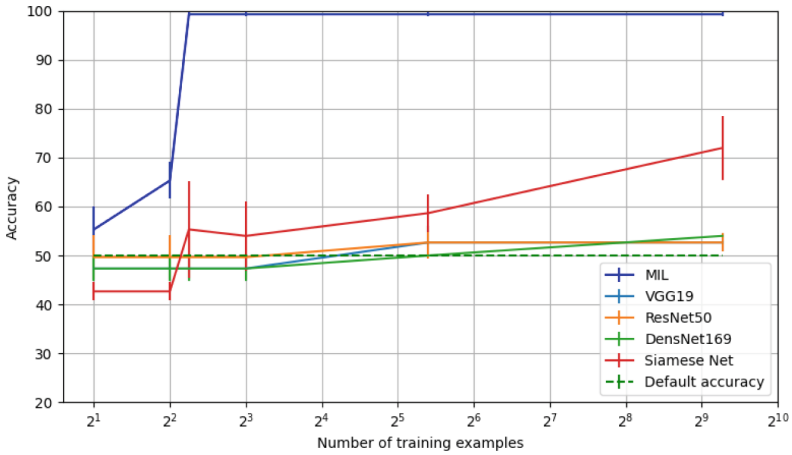

Fig. 4. Comparing the average accuracy of MIL and CNN in the task of learning from fractal images.

Null Hypothesis 1 Our MIL-based approach cannot generate rules to describe fractals.
Null Hypothesis 2 Our MIL-based approach cannot outperform neural networks in the task of learning from fractal images.

We initiate the process with a single-shot sample, initially comprising a positive and a negative example. Subsequently, we assess our models by incrementally augmenting the number of training examples (i.e., 2, 4, 6, 8, 100, and 950). During each learning episode, we regenerate training samples 5 times, while the test samples remain unchanged. The test dataset comprises 100 examples, evenly divided between 50 positive and 50 negative examples. Figure 4 compares the predictive accuracy of MIL vs. CNN algorithms. We measured the learning times on a MacBook laptop with 16 GB of RAM and an Apple M1 CPU with 10 cores. The average timings are shown in Table 2. We provide this table as a reference for the point where MIL accuracy approaches 100% after 6 examples (3 positives and 3 negatives) as shown in Fig. 4.

The rule shown in Listing 3 was learned by MIL from three positive and three negative examples. This recursive rule is a compact and accurate description of a Sierspenski triangle. The Null Hypothesis 1 is therefore rejected.

Listing 3: The final rule obtained by MIL after considering three positive and three negative training examples.

```
fractal(A, triangle):- shape(A, triangle),
                       fractal_1(A).
fractal_1(A):- innerShapes(A, B),
               any(B, fractal_1).
fractal_1(A):- singleShape(A).
```

According to Fig. 4, the Null Hypothesis 2 is also rejected as MIL achieved 100% accuracy after only six training examples but the best performing CNN algorithm in

this study (i.e. Siamese Net) achieved around 70% after around 950 training examples. MIL, with its unique combination of symbolic reasoning and learning techniques, exhibits remarkable capabilities in extracting meaningful patterns and insights from fractal images. By leveraging the power of meta-level reasoning, MIL can dynamically adapt its learning strategies, adjusting and optimising its models to effectively capture the intricate structures inherent in fractal data. Through the incorporation of interpretability and explainability into its learning process, MIL not only achieves high accuracy in modeling fractals but also provides insights into the underlying generative mechanism of the fractal.

Table 2. The average learning time for 6 training examples (3 positive and 3 negative)

Method	MIL	ResNet50	DensNet169	Siamese Net	VGG19
Learning time	**58** ms	570 ms	780 ms	1490 ms	1730 ms

5 Conclusions

We introduced a technique for acquiring knowledge about fractal geometries through graph extraction and MIL. By extracting edges and constructing graphs from them, we formed a representation of the image that could serve as background knowledge for MIL. Our findings demonstrate that even with a limited number of examples, our approach can effectively learn recursive logic programs that provide accurate descriptions of the Sierpinski triangle. When compared to various neural network architectures, our method surpassed them in performance while requiring fewer examples for learning. As future work, we will expand upon the method laid out in this paper to learn rules for a wide range of both artificial and naturally occurring fractals. Including branching fractals and the Sierpinski carpet, and natural structures such as river beds or veins in retinal imaging. By leveraging extended background knowledge utilising concepts of graph theory and geometry this representation has the potential to be used in broader applications in logical computer vision outside of classifying fractals.

Acknowledgments. The first and second authors would like to acknowledge the PhD scholarships from EPSRC and the University of Surrey. The third author would like to acknowledge the EPSRC Network Plus grant on Human-Like Computing (HLC) and the EPSRC grant on human-machine learning of ambiguities.

References

1. Bader, S., Hitzler, P.: Logic programs, iterated function systems, and recurrent radial basis function networks. J. Appl. Log. **2**(3), 273–300 (2004)
2. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. (TOMS) **22**(4), 469–483 (1996)

3. Barnsley, M.F., Ervin, V., Hardin, D., Lancaster, J.: Solution of an inverse problem for fractals and other sets. Proc. Natl. Acad. Sci. **83**(7), 1975–1977 (1986)
4. Bölviken, B., Stokke, P., Feder, J., Jössang, T.: The fractal nature of geochemical landscapes. J. Geochem. Explor. **43**(2), 91–109 (1992)
5. Conover, E.: Physicists wrangled electrons into a quantum fractal (2018). https://www.sciencenews.org/article/physicists-wrangled-electrons-quantum-fractal
6. Cropper, A., Muggleton, S.H.: Metagol system (2016). https://github.com/metagol/metagol
7. Dai, W.Z., Muggleton, S.H., Zhou, Z.H.: Logical vision: meta-interpretive learning for simple geometrical concepts. In: ILP (Late Breaking Papers), pp. 1–16 (2015)
8. Dubuc, B., Quiniou, J., Roques-Carmes, C., Tricot, C., Zucker, S.: Evaluating the fractal dimension of profiles. Phys. Rev. A **39**(3), 1500 (1989)
9. Eiter, T., Mannila, H.: Computing discrete fréchet distance (1994)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
11. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
12. Kempkes, S.N., et al.: Design and characterization of electrons in a fractal geometry. Nat. Phys. **15**(2), 127–131 (2019)
13. Koch, G., et al.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol. 2. Lille (2015)
14. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: ultra-deep neural networks without residuals. arXiv preprint: arXiv:1605.07648 (2016)
15. Louis, E., Guinea, F.: The fractal nature of fracture. Europhys. Lett. **3**(8), 871 (1987)
16. Mandelbrot, B.: How long is the coast of Britain? Statistical self-similarity and fractional dimension. Science **156**(3775), 636–638 (1967). https://doi.org/10.1126/science.156.3775.636
17. Mandelbrot, B.B.: Fractal aspects of the iteration of $z \rightarrow \lambda z(1 - z)$ for complex λ and z. Ann. New York Acad. Sci. **357**(1), 249–259 (1980). https://doi.org/10.1111/j.1749-6632.1980.tb29690.x
18. Muggleton, S.H., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. Mach. Learn. **94**, 25–49 (2014)
19. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Mach. Learn. **100**(1), 49–73 (2015)
20. Saalfeld, A.: Topologically consistent line simplification with the douglas-peucker algorithm. Cartogr. Geogr. Inf. Sci. **26**(1), 7–18 (1999)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
22. Tu, C.H., Chen, H.Y., Carlyn, D., Chao, W.L.: Learning fractals by gradient descent. arXiv preprint: arXiv:2303.12722 (2023)
23. Vrscay, E.R., Roehrig, C.J.: Iterated function systems and the inverse problem of fractal construction using moments. In: Kaltofen, E., Watt, S.M. (eds.) Computers and Mathematics, pp. 250–259. Springer, New York (1989). https://doi.org/10.1007/978-1-4613-9647-5_29