Determining the number of roots of a given polynomial through its newton fractal and deep-learning techniques

Roque Mula, Marco Nieto, Pablo Pérez
Physics engeneering degree, Polythecnic University of Valencia.

Abstract

In this work, a database of Newton fractals has been generated using online tools. A post-processing has been performed using image resizers for the data normalization, and a copy of the original images has been used to generate a parallel database where all the images have been filtered with a Sobel filter. The main objective of this work is to determine whether the usage of colored images is better or not than using filtered images for determining the grade of the polynomial when a newton fractal is given to the model. An XCeption model (a variation of a convolutional neural network) was used to perform the image classification, unfortunately not being good enough to perform the requiered task, as can be seen in the 'training' and 'testing' sections.

Keywords: newton fractal; machine learning; neural network

1 Context and motivation

The main motivation of this paper is the curiosity on the fractal behaviour that emerges from a extremely simple function such as a polynomial. The brand new techniques and practical uses that are being developed in deep learning and AI suits perfectly this context, as image recognizing is a well-studied topic in this field.

1.1 About the Newton's Fractal

The Newton fractal[1] is a classification of the points that comprise the complex plane, characterized by the application of Newton's method to a fixed polynomial [1]. When there are no attractive cycles, it divides the complex plane into $\{n_0, n_1, ..., n_i, ...n_{-k}, n_k\}$ regions , each of which is associated with a root i of the polynomial of order k. In this way, Newtonian fractals resemble Mandelbrot sets and, like other fractals, exhibit complex appearances derived from simple descriptions. It is relevant to numerical analysis because it shows that outside the region of quadratic convergence [2], Newton's method is very sensitive to the starting point choice.

The Newton-Raphson method is performed in this way: An initial point is arbitrarily chosen, and this point is used as the initial value z_0 for Newton iterations

$$Z_{n+1} = Z_n - \frac{p(z)}{p'(z)}$$

giving a sequence of points $\{z_1, z_2, ...\}$. If the sequence converges to the root i , then z_0 is an element of the region n_i. However, for every polynomial of degree at least 2 there are points for which the Newton iteration does not converge to any root. This method is widely used to compute the roots of polynomial functions of arbitrary orders, not only in one dimension but higher. Some of its applications are estimating linear regression models, optimization problems or adjusting the parameters of many simulations or machines. [3]

The fractal behaviour emerges with polynomials of order equal or higher than 3. This is because the boundaries of every region n_i are always adjacent to each other root regions, and the only way to preserve this property, regardless of the scale, is the image to have infinite complexity on its boundaries. That's why complex patterns do not appear with polynomials of order 2, because the boundary between the two regions is a straight line, as it's the simplest way to have the two regions in contact.

[1] which Newton did not see in life despite having his name

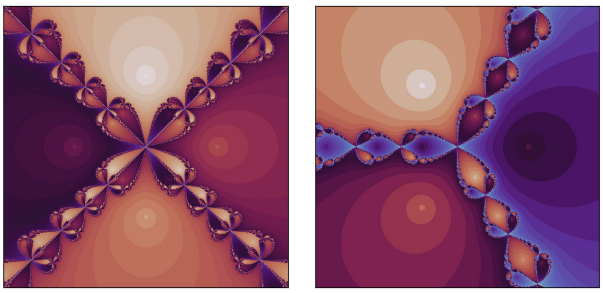

Figure 1: Examples of Newton Fractals, computed from the functions $\frac{(x^2-1)(x^2+1)}{2x((x^2-1)+(x^2+1))}$ (left) and $\frac{x^3-1}{3x^2}$ (right), between the limits 2,-2, $2i$, $-2i$.

2 Technical tools and resources

For this work, a recently created software has been fundamental in some parts. The chatbot 'ChatGPT' developed by the company OpenAI has been used to generate some parts of the code, to give a general overview of the topics that we had to deepen on and to generate some examples in order to make our contributions as similar as possible to already stablished standards in the field. More information about this tool can be found at OpenAI's chatGPT main page.

For the data generation, the javascript program 'Interactive Newton Fractal' [4],developed by the user 'SingularPTS' has been used. It consists in an interactive app that lets the user choose the desired number of roots and colour them in any colour palette For the deep-learning and implementation of neural networks, the python library 'TensorFlow', and more specifically the 'Keras' model has been used.

For the model training, the use of Google Colaboratory was crucial because of the option to use free GPU processing, which significantly reduces training time. All the files containing the code needed for this article can be found inside the `GitHub` repository newton-fractal-deep-learning.

3 Dataset description

3.1 Image processing

As the interactive fractal builder used to generate the images did not have an option to directly save the generated plots, images have been stored using manual screenshots and a posterior resizing. An online image resizer was necessary in order to preserve the same ratio among the images, 360x360 pixels.

It is well known that training an artificial intelligence has an add-on of difficulty because of the extreme care that has to be put in the quality of training data sources. In this work, we want to test two different approaches for obtaining the grades of the polynomials. The first one consists in training the model with the default RGB images, where each region of the fractal has a different color from the adjacent regions. This is the easiest approach, but for our results to be more precise we have considered that providing the model with colored images would not be the best option, because we don't want the model to learn how to distinguish between colors, but figures and patterns that emerge with Newton fractals. That's why the second appproach consists in filtering all the data provided to the model with a Sobel filter [5], which converts the original image to a black and white picture where the edges of the shapes are remarked.

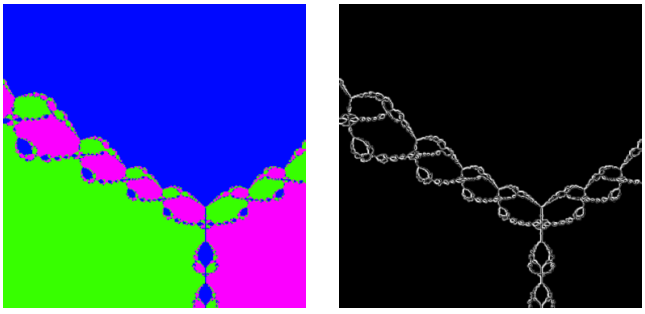

Figure 2: Example of a Sobel filter applyed to a Newton fractal.

The sobel filter technique consists in convolving the original image with two 3x3 matrix kernels:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$\mathbf{G}_x$ is used to calculate derivative approximations in the horizontal changes, and $\mathbf{G}_y$ in the vertical changes. This computations have been performed automatically using functions from the 'cv2' and 'numpy' python libraries. The 'os' library has also been used to access all imgages of a directory at once, in order to speed up the process of filtering.

3.2 Data structuring

Training data

The dataset has been divided in two separate folders, colored and filtered fractals. Inside both folders there is the same amount of images, divided into four directories, each one for a different polynomial grade and with 40 images inside. In this work, four different grades have been computed and processed in order to train the neural network. Specifically, polynomials of grade 3, 4, 5 and 6. In total, there are approximately 160 colored and 160 Sobel-filtered images in the dataset.
A technique to increase the size of the training data has been used, and it consists in applying random rotations and reflections to all the images in the dataset, so the original size is multiplied up to a factor of 8 (4 possible rotations, each one with a possible reflection). As the function responsible for doing this modified copies uses random numbers, not always the eight possible variations are created , thus the model is trained with repeated images which doesn't contribute to a better accuracy. This technique is introduced in the python code that trains the Keras model.

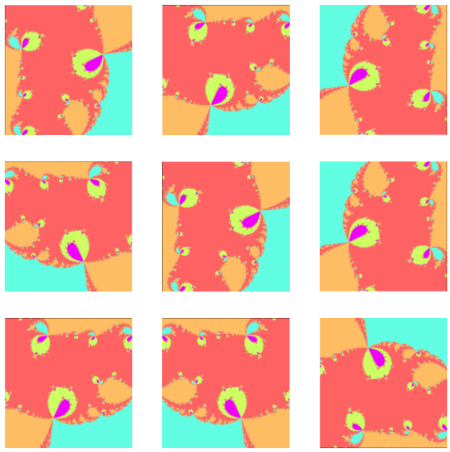

Figure 3: Example of the dataset augmentation technique acting over a 5th grade polynomial (without the Sobel filter, so it's easier to view the differences). In this case, there are 6 different images.

Testing data

Another separate folder has been created with 16 images, 8 non-filtered and 8 with the Sobel filter. Their function has been to test the trained model and observe the results of the experiment.

4 What is a neural network?

A neural network is a structure conformed of artificial neurons, which resembles and is inspired in the shape of a human brain. It's formed of a series of layers, where the neurons in each layer process the inputs of the previous layer and pass the results to the next [6]. Mathematically, each neuron is a simple processing unit that receives input from other neurons, processes the input, and produces an output using different activation functions, such

as the sigmoid , the hyperbolic tangent or the rectified linear unit (ReLU) function. The model is tuned through some parameters that are assigned to each neuron (weights and biases), and are modified as the model keeps being trained until they converge to a state where it's accurate enough for the task needed.

A popular technique for tuning neuron's parameters is called backpropagation. It's an algorithm that adjusts the weights and biases of the connections between neurons in order to minimize the error between the predicted output of the network and the desired output. It consists of taking the derivative of the error with respect to the neuron's parameters in order to update them in the opposite direction of the gradient, until a minimum is reached.

The model used in this work is a variation of a convolutional neural network called XCeption [7], and consists in an input layer, followed by the data augmentator preprocessor, a reescalator and some separable convolutional layers, interspersed with batch normalization and activation layers. The output of these combinations is then passed through a dense layer to produce the final prediction (A dense layer has all of its neurons connected to every one of the previous layer). This model has been selected because is proven to learn complex features using fewer parameters, which reduces the computational cost of the model and makes it more time and resource efficient.

4.1 Training

The training is performed in iterations named 'epochs'. In every iteration, the model learns using all the images on the dataset. Three models have been trained, one from an example to test that the model was correctly assembled, one with colored fractals and another with Sobel filtered fractals. At the end of this section a comparision between the three models is shown.

Example training

Before training the neural network with our fractals database, the XCeption model has been tested with a tutorial example, consisting in a image classificator between two groups, cats and dogs [8]. This example case had a database of 23410 images divided into two folders. After a 25 epoch training, the accuracy reached has been of 0.9823, which means less than a 2% of error.

```
Epoch 24/25
586/586 [==============================] - 257s 439ms/step - loss: 0.0433 - accuracy: 0.9829 - val_loss: 0.8758 - val_accuracy: 0.6837
Epoch 25/25
586/586 [==============================] - 257s 439ms/step - loss: 0.0484 - accuracy: 0.9823 - val_loss: 0.9622 - val_accuracy: 0.6969
```

Figure 4: Results of the example neural netkwork after a 25 epoch training

Using colored database

After testing that the model was correctly assembled and functional, we've passed the colored database into the model. The image recognition from the directories, the categorization in four different groups and the data augmentation has worked perfectly. However, the model training has failed, only achieving a peak of 0.376 accuracy and dropping into 0.272 where it has remained stable until the end of the 25 epochs.

Using filtered database

The result of training the model with a filtered database has been similar. Although the model has reached higher peaks in his accuracy, the training process has failed, converging into the same low result as before, an accuracy of 0.272.

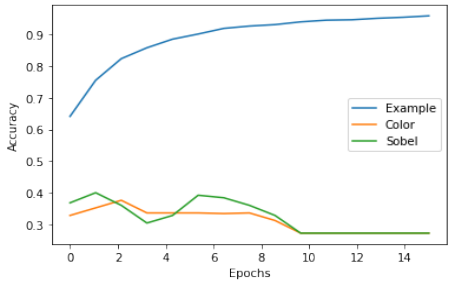

Figure 5: Comparison of the three training models after 15 epochs

4.2 Testing

We have tried to use the three models testing them on a set of images. The first model has shown an absolute domain of the task. Four images have been passed, two of a dog and two of a cat, with an output of 84.25% dog, 98.97% dog for the first two images, and an output of 67.54% cat and 91.22% cat for the last two. However, for the Newton fractal's models, the outputs of the predictions had no sense and we can affirm that both trained models were incompetent in the task of classifying the images into their correspondent classes.

5 Discussion and possible applications

Although a 0.272 accuracy is a bit higher than random guessing, it has no practical utility, and we have not been able to determine whether one data coloring is better or worse.

We can not discard the case where the model choice or even the training process has been performed incorrectly. Another possible reason for the model failing could be the scarcity of database images. 40 examples for each class is a very low number, even after the data augmentation process, in comparision with the example database where each class had a number of 11705 photos, considering the complexity of the task demanded.

Some possible applications of this work could be discarding this model in a future approach to this topic, or teaching artificial intelligence students why a wide and adequate database is fundamental for a model to work properly. Also, our database of fractals is open to use for everyone studying the topic, and we believe it can be very useful. It can be found inside the repository shared before.

References

[1] Interactive Newton Fractal, developed by 'SingularPTS'. https://p5js.org/es/

[2] Roland Hildebrand: Local convergence behaviour of the Newton method. https://mipt.ru/

[3] Michael McAleer: Applications of the Newton-Raphson Method in Decision Sciences and Education. https://iads.site/

[4] J. Hubbard, D. Schleicher, S. Sutherland: How to find all roots of complex polynomials by Newton's method. https://www.semanticscholar.org/

[5] University of Edinburgh, School of informatics: Sobel Edge Detector. https://homepages.inf.ed.ac.uk

[6] IBM: What are neural networks? https://www.ibm.com/

[7] Mael Fabien: XCeption Model and Depthwise Separable Convolutions https://maelfabien.github.io/

[8] Fchollet: Image classification from scratch. https://keras.io/