ISSN (e): 2250 – 3005 || Volume, 15 || Issue, 10|| Oct. – 2025 ||
International Journal of Computational Engineering Research (IJCER)

Enhanced Intelligence Models based on the CSN Algorithm

Catalin Silviu Nutu
Constanta Maritime University
Constanta, Romania

ABSTRACT
Present classical AI models, mainly based on supervised and unsupervised learning, may be appropriate to simulate human brain learning processes corresponding to the intelligence stage of children, when learning basics about their surroundings, but they are completely insufficient and totally inadequate to simulate highly sophisticate intelligence processes in the human brain when finding extremely innovative solutions or when making breakthrough scientific discoveries.
Therefore, in order to do both, to decrypt codes embedded in the structure of Nature, and to simulate sophisticate and complex human brain processes, the present AI models should either be replaced or they should be extended and completed with new enhanced intelligence models, taking into account the randomness factor involved in those sophisticate thinking processes.
An enhanced intelligence model (EI model) is defined by author as an intelligence model which takes into account of an increased degree of randomness, because when solving highly complicated problems, the information is not purely procedurally processed, but it depends of a serious degree of randomness, which is seriously disregarded by the present classical AI models.
The CSN Algorithm is used for this purpose, as starting point and as foundation for EI models, such as presented and described in this paper. The CSN Algorithm which is used to generate procedural randomness, simulates much better and more realistically than the classical AI, the way in which information is processed. This is because many scientific discoveries and breakthroughs in science take actually place by aleatory processing information or even accidentally or randomly, or by trial and error. The CSN Algorithm could also be used to complete, extend and enhance the classical AI models, in the way it is presented in this paper.
This paper expands on the previous work and research of the author presented in the References of this paper, and on another more recent work of the author [13], paper which has been already presented at the Springer Conference XR Salento 2025, in Otranto, Italy.
This paper mentioned above introduces some new concepts: the concept of evolutionary fractal, the concept of procedural randomness, and the concept of detraining. For explanatory purposes, the definitions of these concepts will be given again, in this paper

Keywords: *AI, evolutionary fractal, procedural randomness, CSN Algorithm, training and detraining, disruptive transitions, randomness, EI models*

Date of Submission: 08-10-2025 Date of acceptance: 19-10-2025

I. INTRODUCTION

The idea that the intelligence of Nature may be embedded in fractals has already been stated by author, since 2016, as in [2] and [4]. This paper is expanding on previous papers of the author [1] and [3], by providing alternative enhanced intelligence models based on the CSN Algorithm, to complete the present models of AI.

The AI, in its present stage, as presented in the literature in [5], [6], [7], [9] and [10], is hardly to be called „intelligence" since it is not possessing at least the intelligence of the most primitive life forms of Nature. In this context, by Nature it is to understand the non-animal or animal life structures. Excepting some basic mechanisms, like it is the perceptron's model, for instance, primitively simulating the functioning of brain cells, the stage of understanding human brain and human intelligence, despite all advances in science and technology,

has remained pretty much the same since the ancient times of the first Greek thinkers like Socrates, Plato and Aristoteles, as presented in [11].

Present AI theory, seriously oversimplifies complicated processes in the human brain, by reducing them to calculus and to analysis of huge amounts of data based on computer subroutines designated to find patterns, classifications, etc. in the respective data. Hence, AI should be completed and extended with the element of randomness, of paramount significance in major scientific breakthroughs and discoveries.

When dealing with or generating new science, completely new ideas in science can neither be generated based exclusively on previously available information and knowledge. Many of these ideas in science, occur instead somehow as results of aleatory brain processes in the thinking activity of extremely creative minds, or as being randomly generated, either by mere chance and sheer luck. These creative ideas, have many times, little in common with previous science, and although they are somehow based on previous science, often times they even seem to contradict already existent science, such as in the case of Theory of Relativity, for instance.

The codes embedded in the structure of Nature, cannot be deciphered using simple algorithms which disregard the importance of chance and randomness. In the very same way, when making scientific breakthrough discoveries or giant leaps in science, almost always, the existing information is not purely procedurally processed, as in present classical AI models which are mostly deterministic, but it is processed by brain aleatory, in random processes, as, for example, in the model based on the CSN Algorithm presented in [1] and [3].

The CSN Algorithm proposed in previous papers is based on two different matrices, both having the same number of lines and columns: the CSN Matrix of Elements and the CSN Matrix of Operations. One of them contains as information in its cells elements, and another one contains in its cells operations. Based on a random input number, the algorithm selects the position of the corresponding operation in the matrix to be applied to the element on the very same position, and thus outputs one certain value, depending on the random input Tkey and its corresponding position in both matrices above: Output = CSN Algorithm (Tkey_Random_Input_Value,CSN_Matrix_of_Elements,CSN_Matrix_of_Operations)

Evolutionary fractals may be deemed as structures in which is embedded the Intelligence of Nature and are defined as fractals structures evolving from previous states into subsequent other different ones. In opposition with evolutionary algorithms which are characterized by small, incremental successive changes and improvements, mostly implying fitting and adapting in order to survive, such as in the case of the Darwinian law of evolution, the evolutionary fractals are characterized by sometimes, sudden disruptive transitions, and hence disruptive changes of the fractal rules. Evolutionary fractals are defined as fractals evolving from a certain shape into another one, by changing shapes because of moving through different fractal rules, or in a particular case, from a life form into another, from an initial state of the life form into a final state of life, by passing through a multitude of phases and transformations. More exactly it means a structure based on and constructed using a set of many fractals, each one having its own fractal rule.

The CSN Algorithm above, is the main tool used to obtain procedural randomness, meaning that a random input value determines the position of both, of the element and of the operation, and hence determines the output of the algorithm, based on the same respective position in both matrices: of the element from the CSN Matrix of Elements and of the operation from the CSN Matrix of Operations.

Procedural randomness means that a random input value determines the positions of the element and the position of the operation which are selected to generate the output of the CSN Algorithm. The output of the algorithm will be generated using both, random input data and procedure to access the pertaining information in both matrices, and hence this algorithm will generate the respective procedural randomness.

The process of detraining is the process of generating procedural randomness, by using the CSN Algorithm, and is pretty much the opposite of the very known and used process of training of a model. In other words, the detraining actually means to run the CSN Algorithm in order to either decrypt fractal rules or to find innovative solutions to complex problems, by applying this algorithm to a set of given elements and operations, elements and operations organized in the CSN Matrix of Elements and in the CSN Matrix of Operations.

If the intelligence model presented by the author in this paper, based exclusively on the CSN Algorithm, will prove as insufficient to simulate and to decrypt human brain processes and human intelligence, an alternative solution may be a combination between this intelligence model based on the CSN Algorithm and classical AI models, in the way it is presented in a further section of this paper.

However, until the full decryption of human brain processes and until human intelligence is fully understood, the EI models presented in this paper can be also used as intelligence models to enhance and to complete the existing traditional AI models. The EI models presented in this paper take into account an increased degree of randomness and accidents, randomness and accidents which are present in both, in Nature and in the aleatory thinking processes, when making breakthrough discoveries in science.

II. METHOD

2.1 The Research Question: How to generate enhanced intelligence models more appropriate to decrypt evolutionary fractals, and which hence simulate better and mirror more adequately complex and sophisticate human brain processes?

The idea that secret codes of Nature, and hence also the secrets of functioning of the human brain, may be embedded in the evolutionary fractals has already been stated, and the evolutionary fractals have been also defined as being fractals evolving from a shape or in this case from a life form into another, from an initial state of the life form into a final state of life, by passing through a multitude of phases and transformations, as in [1].

With almost no other better means than their minds and their intuition, scholars of the Middle Age and even some scholars of the Ancientry, like Pythagoras and others [11], had the common sense to feel and to admit the existence of secret undeciphered codes embedded in Nature, facts confirmed sometimes thousands of years thereafter, by the discovery of the DNA, for example.

In the present, using the available technology and the huge computing power available today, some of those encrypted codes, still remained elusive until now, can be decrypted using this huge computing power, jointly with the enhanced intelligence models taking account of randomness, proposed by this paper, for instance.

Using the EI models proposed, one can, on one hand decrypt secret undeciphered codes embedded in the structure of Nature, and on the other hand, one can better and much more adequately simulate sophisticate thinking processes taking place in human brain, when making breakthrough discoveries or giant leaps in the scientific knowledge.

The EI models proposed can also be employed in the following way: a database with many various fractals should be created. This database will then be mined using different existent mining techniques to identify similar fractal shapes in Nature. This database should then be used to train a certain fractal shape from the database.

After the training process above, the fractals recorded in this database will be detrained by means of a CSN Algorithm. Randomly generating different fractals rules using the algorithm, fractal rules of evolutionary fractals can be thus identified. In this way, both concepts, the traditional one of training and the newly introduced one of detraining can be combined to identify a general valid set of rules corresponding to certain evolutionary fractal, based on the shape of the respective evolutionary fractal.

2.2 Objectives and Data used

The main objectives of this paper are: on one hand, to explain and understand better how the human mind works and why present AI models are insufficient to simulate complex and sophisticate human brain processes. On the other hand, based on this better understanding of thinking processes, the second objective is to enhance and improve the existent AI models, which are mainly purely procedural and take less account of the randomness, trial and error processes, features which are very characteristic and inherent to many scientific discovery processes.

Another objective, besides simulating more accurately complex and sophisticate thinking processes, is to use the EI models thus generated to solve the special issue of decrypting the evolutionary fractals, an issue which is also related to intelligence itself and its foundations.

The data used in this paper to generate new science and knowledge, let us say, of finding a new theory comprising a certain number of scientific discoveries, discoveries denominated in short as TRUTHs, is represented by the infinite set of outputs which can be generated using the CSN Algorithm. The Python computer script in the Appendix outputs such infinite results, used to identify the respective TRUTHs by identifying their corresponding TRUTH keys (Tkeys).

III. EI MODELS BASED ON THE CSN ALGORITHM

3.1 Understanding better human mind processes

In order to better understand how human mind works, let us think a little bit about thinking. Let us suppose that someone has to memorize seven words or seven ideas. The first step is to memorize each of the seven ideas, but then, when he/she wants to remember and to check the correctitude and completeness of information stored in one's memory, the respective person checks remembering those ideas by checking their number, respectively 7.

And this is exactly how human mind works and processes information, in the most efficient and most effective way possible, continuously optimizing huge amounts of data and permanently reducing them to very small bits of information.

This is also one of the main reasons, why the proposed CSN Algorithm is so powerful and so appropriate to simulate complicate brain processes, because it does the very same thing: it reduces big amounts

of data stored in both CSN Matrices to a single corresponding number, Tkey. And therein lies the value and the usefulness of the CSN Algorithm, in reducing that huge information and large and complicate chunks of data stored and necessary to be processed, to a single vector of identified Tkeys, corresponding to a new scientific theory, for example.

The facts presented above about the human thinking process, also explain another more important thing, namely they explain how human brain is able to create, invent and come up with the most brilliant new ideas, most of those ideas having nothing or little to do with past knowledge or information. In the very same way in which one's inexplicable and incomprehensible life, is not the same with the sum of the events in one's life, brilliant new ideas do not represent a logical combination or a sum of past information and knowledge, but these new ideas go far beyond previous knowledge, such as presented in [12].

Thus, while present classical AI models are working by rummaging through enormous amounts of data, human mind manages to get by and to reach far better results than AI, with a lot less information and with considerable less effort, through this kind of optimization processes.

3.2 The basis for the enhancement of artificial intelligence models, based on the CSN Algorithm, explained

The Python programming environment [8], is one of the most appropriate, powerful and versatile tools, when it comes to model and simulate the behavior of artificial neural networks and hence, also when to show the actual application of the CSN Algorithm. In the paper [1], a MATLAB script has been previously developed and used for the same purpose, but Python programing environment is more adequate and hence, the Python script is more suitable for this purpose. This Python script, corresponding to the CSN Algorithm is presented as an appendix to the body of the paper.

The CSN Algorithm and its corresponding script in Appendix, which generates infinite outputs, is used to find only one TRUTH, corresponding to a certain identified Tkey. This algorithm, however, can be easily extended to find a certain number of Tkeys, in order to generate an EI model, for a certain CSN Matrix of Elements, in the following manner:

Step 1: Use the CSN Algorithm to generate one CSN Matrix of Operations for a given CSN Matrix of Elements

Step 2: Try all possible Tkeys, this is to say apply all the operations in the CSN Matrix of Operations, to each of the elements of the CSN Matrix of Elements

Step 3: If Tkey not found, permutate the positions of operations in the CSN Matrix of Operations and revert to Step 2

Else, go to Step 1, to generate another Matrix of Operations to search for the next Tkey

Step 4: If all possible permutations in the CSN Matrix of Operations have been tried and still no Tkey has been found, go to Step 1, and generate another Matrix of Operations and use it with the same Matrix of Elements

This EI model presented above, could either be applied as an intelligence model to find new knowledge in a certain specific area of science or it can be used to decrypt a certain evolutionary fractal taking into account the following conditions:

1) specific specialized knowledge about the respective evolutionary fractals is required
2) specific knowledge about the algorithms involved in the respective fractal structure is required
3) specific knowledge about past values of the previous stages of evolutionary fractal structures

3.3 Various different additional alternative EI models based on the CSN Algorithm

The following various different additional alternative EI models presented in this section are using the theory regarding the perceptron's model with its related concept of activation function, jointly with the above EI model based on the CSN Algorithm. Hence, these EI models represent possible combinations between the perceptron's model and the EI model proposed, in the following ways presented.

On one hand, the input of the classical activation functions used, can be generated using the process of detraining. On the other hand, the usual activation function used to trigger or not reactions in the classical neural network can also be generated by detraining. In this second case, the cells of CSN Matrix of Operations will be populated with different activation functions.

3.3.1. First EI model proposed based on the CSN Algorithm

I) Choose a certain activation function

II) Choose a vector or a set of (n) different numbers: $x_1, x_{2,\dots} x_n$

III) Calculate the outputs $CSN(x_1), CSN(x_2), \dots CSN(x_n)$ using the CSN Algorithm

IV) Given a certain vector or set of weights $w_1, w_{2,\dots} w_n$, calculate the output in the form of the weighted sum: $\sum_{i=1}^{n} w_i \, CSN(x_i)$

V) Using the respective activation function, decide about the output of the respective activation function (activation state of the perceptron)

3.3.2. Second EI model proposed based on the CSN Algorithm

I) Choose a certain activation function

II) Randomly generate a value for TKey

III) Use the nth output $CSN(CSN(CSN\dots(TKey)$ as input for the chosen activation function

IV) Depending on the value obtained at III), decide about the output of the respective activation function (activation state of the perceptron)

3.3.3. Third EI model proposed based on the CSN Algorithm

I) Choose a certain activation function

II) Choose a vector or a set of (n) different numbers: $x_1, x_{2,\dots} x_n$

III) Choose a vector or a set of (n) weights: $w_1, w_{2,\dots} w_n$,

IV) Calculate the values:

$CSN(x_1), CSN(CSN(x_2)), CSN\left(CSN(CSN(x_3))\right), \dots CSN(CSN(\dots CSN(x_n)))$ and use them in the weighted sum:

$$\sum_{i=1}^{n} w_i \, CSN(CSN(CSN \dots CSN(x_i)))$$

V) Using the above value as input for the chosen activation functions, decide about the output of the respective activation function (activation state of the perceptron)

3.3.4. Fourth EI model proposed based on the CSN Algorithm

I) Choose a certain activation function

II) Depending on a certain randomly generated TKey, a number of n outputs can be generated:

$CSN(TKey), CSN(CSN((TKey)), CSN(CSN(CSN((TKey)))$, and so on

III) The sum of these can be used as the input for the chosen activation function

IV) Decide about the output of the respective activation function (activation state of the perceptron)

IV. CONCLUSION

This paper explains, in short, how human brain works and processes information, using effective and efficient optimizations, and also underlines the fact that when making scientific breakthrough discoveries the importance of randomness in thinking processes cannot be discounted. It also explains why the EI models proposed are more appropriate and more adequate when sophisticate and complicated human thinking processes should be simulated.

The EI models proposed by this paper are using the process of detraining based on procedural randomness generated with the CSN Algorithm.

If the proposed intelligence model based only on CSN Algorithm, presented by the author in this paper is not sufficient several different approaches may be possible:

1) To further expand and to improve the proposed intelligence model based on the CSN Algorithm, and thus to create a more complicated E.I. model solely based on the CSN Algorithm

2) A combined and enhanced model of intelligence which completes the classical AI models with the CSN Algorithm as presented in the section 3.3. in this paper, used, either for the task of decrypting intelligence of Nature or for the purpose of improving the existent classical AI models

3) To use the CSN Algorithm as an EI model generator, by placing different EI models in the cells of its corresponding CSN Matrix

Depending on the problem which needs to be addressed, the EI models proposed in this paper, can be tested regarding their efficiency and the most effective and appropriate one is to be chosen. Hence, in a certain situation or in a certain area, a certain EI model can be preferred over the others.

The research and the EI models proposed by this paper can be extended and improved in the following ways:

1) By expanding the Python computer script in the Appendix with the section which permutes all the positions in both CSN Matrix of Operations

2) By filling in the cells of both CSN Matrices with certain specific information (elements and operations), in order to apply this EI model to a certain specific scientific area or a certain domain of human knowledge, or for decryption of still undeciphered evolutionary fractals in Nature

3) By finding various other ways to link present AI models with the EI model proposed, and thus improving the present classical AI with the element of procedural randomness

The EI model based on the CSN Algorithm can be extended in the following manner:

1) By filling in the cells of the CSN Matrix of Operations with other matrices of operations
2) By filling in the cells of the CSN Matrix of Operations with algorithms instead of simple functions
3) By placing knowledge in the cells of the CSN Matrices, to generate more knowledge and science

Another important area in which the enhanced intelligence based on the process of detraining and on its corresponding CSN Algorithm could be employed, is in finding various other different types of EI models. By placing different information for different EI models in the cells of its corresponding matrices, and using the process of detraining, the EI model presented can either be used as an EI generator of models to identify further enhanced intelligence models, or it could also further be used to decrypt different evolutionary fractal structures in Nature.

REFERENCES

[1] Nutu, C. S., Axinte T.: "Communication with Nature based on a Combined Multiplicative/Additive Encryption Model", Advanced Topics in Optoelectronics, Microelectronics and Nanotechnologies XI, 124931H-10, Proceedings of SPIE, Vol. 12493, 2022: 369-378

[2] Nutu, C. S., Axinte T.: "Microelectronics and Nanotechnology and the Fractallike Structure of Information, Knowledge and Science", Advanced Topics in Optoelectronics, Microelectronics and Nanotechnologies VIII, 10010 100101I-1, Proceedings of SPIE, Vol. 10010, 2016: 394/402

[3] Nutu, C. S.: "Decryption of Evolutionary Fractals by means of Procedural Randomness generated with the CSN Matrix", International Journal of Computational Engineering Research (IJCER) ISSN 2250-3005, Vol.14, Issue 4, Jul-Aug 2024

[4] Nutu, C. S., Axinte T.: "Fractals. Origins and History", Journal of Marine Technology and Environment, Vol. 2, 2022, 48-51

[5] M. P. Deisenroth, A. A. Faisal, C.S. Ong: "Mathematics for Machine Learning", Cambridge University Press, 2020

[6] Mitchel T.M.: "Machine Learning", McGraw-Hill Science/Engineering/Math, 1997

[7] Mehlig B.: "Machine learning with neural networks", University of Gothenburg, Göteborg, Sweden 2021

[8] Halvorsen H.P.: "Python Programming", ISBN:978-82-691106-4-7, 2020

[9] Thomas A.: "An introduction to neural networks", 2019, https://coursehero.com

[10] Ezekiel S., Pearlstein L., Alshehri A.A, Lutz A., Zaunegger J., Farag W.: "Investigating GAN and VAE to Train DCNN", International Journal of Machine Learning and Computing, Vol. 9, No. 6, 2019, 10.18178, 774/781

[11] Dumitriu A.: "Istoria Logicii (The History of Logic)", Editura Tehnica, Bucharest, 1993

[12] Nutu C. S.: "Hunting for Genius. What is Brilliance? Can A.I. be brilliant?", Journal of Marine Technology and Environment, Vol. 1, 2024, 29-34

[13] Nutu C.S.: "Decryption of Evolutionary Fractals by means of Procedural Randomness generated with an Intelligence Model based on the CSN Algorithm", International Conference XR Salento 2025, Otranto, Italy, June 17-20, 2025, Proceedings, Part VII, 117-130

Appendix: *The Python computer script corresponding to the CSN Algorithm*

```
import numpy as np
import sympy
from sympy import*

m=Symbol('m')
n=Symbol('n')
i=Symbol('i')
j=Symbol('j')

x=Symbol('x')

print('Enter the number of lines of the CSN Matrix')
m = int(input('Enter m>5:'))
print('Enter the number of columns of the CSN Matrix')
n = int(input('Enter n:'))
p=m*n;
print('The total number of possible Tkey values is:',p)

Tkey = np.random.randint(0,p-1)
print('The Tkey value is: Tkey =',Tkey)

print('The number of first lines of functions f is:')
a = np.random.randint(1,m//3)
print('a=',a)

print('The number of next lines of functions g is:')
b = np.random.randint(1,m//3)
print()
CSN_a_elements = np.array([poly_f(j,x)])
CSN_a_operations = np.array([Derivative(poly_f(j,x),x)])

print('The first "a" lines of the CSN Matrix of elements are:')
print(CSN_a_elements)
print()
print('The first "a" lines of the CSN Matrix of operations
to transform the CSN Matrix of elements are:')
print(CSN_a_operations)
print()

for j in range (b):
   print('The coefficients of polynoms g in line',a+j+1,'are:')
   print(g(j))
   print()
   def poly_g(j,x):
      return sum(a*x**(i+1) for i,a in enumerate(g(j)))
   print('The polynoms g in line',a+j+1,'are:')
   print()
   print(poly_g(j,x))

CSN_b_elements = np.array([poly_g(j,x)])
CSN_b_operations = np.array([Integral(poly_f(j,x),x)])
```

```
print('b=',b)

print('The number of next lines of functions h is:')
c = np.random.randint(1,m//3)
print('c=',c)

print('The number of last lines of functions i is:')
d = m-a-b-c
print('d=',d)

F = np.random.uniform(0,3, size=(a,m+n+1,n))
G = np.random.uniform(0,3, size=(b,m+n+1,n))
H = np.random.uniform(0,3, size=(c*n))
I = np.random.uniform(0,3, size=(d*n))

print('The matrix of coefficients of f polynomial
functions is:')
print(F)
print()
print('The matrix of coefficients of g polynomial
functions is:')
print(G)
print()
print('The matrix of coefficients of h logarithmic
functions is:')
print(H)
print()
print('The matrix of coefficients of i exponential
functions is:')
print(I)
print()

for j in range (a):
   def f(j):
      return F[j,:,:]
for j in range (b):
   def g(j):
      return G[j,:,:]
for j in range (a):
   print('The coefficients of polynoms f in
line',j+1,'are:')
   print(f(j))
   print()
   def poly_f(j,x):
      return sum((a*x**(m+n+1-i) for i,a in
enumerate(f(j))))
   print('The polynoms f in line',j+1,'are:')
   print()
   print(poly_f(j,x))

print('The second "b" lines of the CSN Matrix of
elements are:')
print(CSN_b_elements)
print()
print('The second "b" lines of the CSN Matrix of
operations
to transform the CSN Matrix of elements are:')
print(CSN_b_operations)
print()

CSN_c_elements = np.array(H)
CSN_c_operations = np.log(CSN_c_elements)

print('The arguments of exponential functions h in line
are:')
print()
print(CSN_c_elements)
print()
print('The third "c" lines of the CSN Matrix of
operations
to transform the CSN Matrix of elements are:')
print(CSN_c_operations)
print()

print('The arguments of logarithmic functions i in line
are:')
print()
CSN_d_elements = np.array(I)
print(CSN_d_elements)
CSN_d_operations = np.exp(CSN_d_elements)

print('The last "d" lines of the CSN Matrix of elements
are:')
print(CSN_d_elements)
print()
print('The last "d" lines of the CSN Matrix of
operations
to transform the CSN Matrix of elements are:')
print(CSN_d_operations)
print()

if (Tkey<a*n):
   CSN_Alg = np.array(CSN_a_operations[Tkey])
else:
      if (a*n<=Tkey<(a+b)*n):
            CSN_Alg =
np.array(CSN_b_operations[Tkey-a*n])
      if ((a+b)*n<=Tkey<(a+b+c)*n):
            CSN_Alg =
np.array(CSN_c_operations[Tkey-(a+b)*n])
      if ((a+b+c)*n<=Tkey<m*n):
            CSN_Alg =
np.array(CSN_d_operations[Tkey-(a+b+c)*n])

print('The output of the CSN Algorithm is ',CSN_Alg)
```