EVOLUTIONARY NEURAL NETWORKS
2. SIMPLIFICATION

José del Carmen Rodríguez Santamaría
https://www.ejristos.com
Bogotá, Colombia

May 30, 2024

Abstract

We investigate the possibilities of Evolution to simplify a perfect but possibly redundant man-made Neural Network.

Contents

1 REVIEW 3

2 THE OR-EXPANSION 8
2.1 In hindsight 13

3 EVOLUTION COMES TO HELP US 13
3.1 A Genetic Algorithm for the optimization of the NN architecture 14
3.2 AN EXPERIMENT 18

4 CONCLUSION 29

5 ANSWERS TO EXERCISES 29

6 THE BIBLIOGRAPHY 47

INTRODUCTION

Our global aim is to research the possibilities of Evolution to design Neural Networks (NN) and to extract conclusions that may be relevant for the assessment of the Evolutionary Theory (ET).

In section 1, REVIEW, we revise the important concepts concerning Neural Networks (NNs) that were introduced in our Intro (Rodriguez, [1] 2023). What we called Perfection in there was just the best that a specific NN with two perceptrons could do. It could classify without mismatches the half of bitmaps among a total of 512. So, our perfectionism was indeed mediocrity. To improve performance is the very first challenge undertaken in this article.

In Section 2, THE OR EXPANSION, we extend the capabilities of the initial NN by adding more perceptrons to open more recognizing channels, which are joined by and OR-perceptron. Following this directive, we get perfection, a NN that can recognize all 512 bitmaps without one single mistake. We found that perfection has for us a cost that is exponential in the number of bits of the bitmaps under study. This means that our approach is untenable for the real world.

Evolution is called for help in section 3, EVOLUTION COMES TO HELP US. It indeed can help, but not enough to apply to the real world. The reason is that we consider that simplified NNs also have a number of perceptrons that is exponential in the number of bits of the bitmaps.

Unless otherwise stated, programs were run over a computer with 8 Gigabytes of RAM.

1 REVIEW

Our general aim is pattern recognition with the help of Evolution.

We worked in our Intro a toy model in which images consisted of black-or-white pixels that were arranged in a 3×3-grid. We defined two archetypes, one represented number 1 and the other 0:

```
010      111
010      101
010      111
```

The corresponding bitmaps were 010010010 and 111101111. A bitmap is a binary string that represents a sign.

The grid makes place for 512 different images or bitmaps. The Bit-by-Bit distance (BbB or BBB distance) classified each bitmap as a deformation of 1 else 0 depending on the shortest distance to the archetypes. In most cases, this classification matches that of common sense. For instance, a deformation of 1 is followed by a deformation of 0:

```
010      111
000      100
010      111
```

Let CLASS1 be the family of bitmaps that are classified by the BBB distance as a deformation of 1, and CLASS0 as the deformations of 0.

To implement pattern recognition, we abstracted the neuron as a generalized noise-tolerant boolean gate, that was simulated by a perceptron or cell.

1 ***Exercise.*** *Clearly explain why the logic of a perceptron is or not boolean. What kind of logic does it obey?* *Answer*

We joined two perceptrons in a Neural Network (NN) with the architecture presented in figure 1, and showed that it can correctly classify 260 of the 512 possible bitmaps where the BBB distance defined the correct assignation.

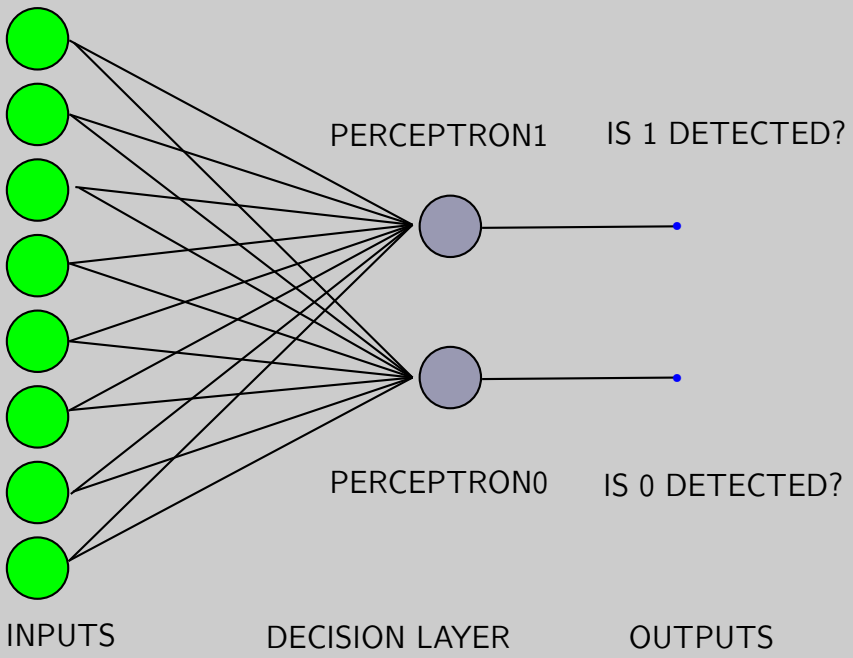


Figure 1: This NN was designed by human intelligence. Its task is to recognize distorted images of 1 else 0. How does it work? It has 9 inputs, one for each pixel in the 3×3-binary-grid. The inputs represent a possibly distorted bitmap of the images of 1 else 0. All inputs are connected to each cell or perceptron of the decision layer, which contains just 2 cells or perceptrons. The upper cell must output 1 when an image of 1 is presented, and zero otherwise. The lower cell must output 1 when 0 is presented and 0 otherwise. Images can be a bit distorted by noise and/or variability. The overall output is a string, with the output of Perceptron 1 followed by that of Perceptron 0. Warning: this output must be analyzed to see whether it contains a contradiction. This happens when both perceptrons output 1, that means that the first perceptron has perceived a 1 and the other a 0. If the output is 10, it says that the first perceptron perceived a 1, while the other found nothing. The NN perceived a 1. If the output is 01, the input was unknown for the first perceptron, while the second perceived a 0. The NN perceived a 0. There is here no contradiction. The same happens for the output 00, which is interpreted as UNKNOWN. Thus, the logic of this NN is three-valued: 1, 0, UNKNOWN.

We also showed in our Intro that Evolution can fine-tune some isolated parameters of the perceptrons of our NN to maximize the matches but without one single mismatch. This result was valid for biases and thresholds, but we failed with weight vectors. In the last case, performance wandered preferentially around under mediocrity, a fact that we took as the signature of Evolution. In our Intro, the ratio 260/512 was our definition of Perfection. But here, it is no more than mediocrity, and our challenge is to leave it to embrace a quest for real perfection. To undertake this enterprise, let us recall the way to design a perceptron to recognize a given bitmap.

To design a perceptron that recognizes a given bitmap that we call the archetype, we follow the following prescription:

1. The perceptron must decide if a stimulus is strong enough to fire it. The force of the stimulus is given by a total sum S, and it must be stronger than a certain threshold T. So, we must decide if $S \geq T$.

2. To recognize the bitmap of the archetype, we transformed it into an algebraic expression of sums of the form $we + b$ with weight w, bias b, and e is the value of the given bit, 1 else 0. When the bit is $e = 1$, $w = 1$ and $b = 0$. When the bit is $e = 0$, $w = -1$ and $b = 1$.

3. In the total sum $S = \sum w_i e_i + b_i$, the term $\sum b_i$ is the *initial bias*. S must be next compared with the threshold T that is equal to the number of bits.

4. To extend the noise tolerance or recognizing power in the presence of noise, 0.2 was added to the bias for each input or bit. Why? The recognizing power means that the perceptron is able to recognize slightly distorted images. These would elect weaker responses than the pristine specimen. To ensure that they also fire the perceptron, its intensity is raised a bit by brute force.

Thus, a positive number a_i is added for each bit. Finally, we get $S = \sum w_i e_i + b_i + a_i$. The term $\sum a_i$ is the additional bias.

5. The output of the perceptron is read from the inequality $S > T$ that says that the perceptron must recognize the archetype, i.e., the total sum must be strong enough to fire it.

2 *Example.* *Let us reproduce a result of our Intro with this procedure. This receipt will be used everywhere all along.*

Suppose that our task is to recognize a 1, whose bitmap is 010010010. This is the archetype or, more formally, the bitmap of the archetype. The AND-perceptron to recognize it is built as follows:

- The perceptron must decide if a stimulus is strong enough to fire it. The force of the stimulus is given by a total sum S, and it must be stronger than a certain threshold T. So, we must decide if $S \geq T$.
- The sum S is equal to $\sum w_i e_i + b_i$ where w_i is the weight associated to stimulus i that must multiply e_i, the bit i of the bitmap while b_i is the $i-th$ initial bias.
- The weight associated to a 1 in the input is 1 and the bias is 0. The weight associated to a 0 is -1 and the bias is 1. So, for the archetype 010010010 we get $S = -e_0 + 1 + e_1 - e_2 + 1 - e_3 + 1 + e_4 - e_5 + 1 - e_6 + 1 + e_7 - e_8 + 1 = -e_0 + e_1 - e_2 - e_3 + e_4 - e_5 - e_6 + e_7 - e_8 + 6$ where we counted bits from 0 to 8. Since there are 6 zeroes, the initial bias is 6.
- The threshold is the number of bits in the archetype or the length of each bitmap. It is 9.
- To enhance the tolerance or recognizing power, let us subtract 1.8 from the threshold, 0.2 for each bit, a value that was chosen tentatively. This is equivalent to adding 1.8 to the bias. It is called the **additional bias**. Total bias $B = 6 + 1.8 = 7.8$.

- The decision executor decides whether $S = -e_0 + e_1 - e_2 - e_3 + e_4 - e_5 - e_6 + e_7 - e_8 + 7.8 \geq T = 9$.
- This inequality must be fulfilled for the archetype 010010010. Is that true? Replacing the variables e_i by the corresponding bit in the archetype , we get: $S = 1 + 1 + 1 + 7.8 = 10.8 \geq T = 9$ that is true. The perceptron recognizes the archetype. Let us notice that if we delete the additional bias, 1.8, the inequality reads: $9 \geq 9$ that means that when there is no tolerance, the perceptron recognizes anyway the bitmap of the archetype.

3 *Exercise.* *Prove that when there is no additional bias, the perceptron recognizes only the archetype. Thus, the additional bias is the responsible term for fault tolerance.* Answer

4 *Exercise.* *Is it possible to implement a boolean logic with perceptrons?* Answer

2 THE OR-EXPANSION

In our Intro, all bitmaps that represent a sign in a 3×3-grid were classified by a distance method as deformations of 1 else 0, which were the archetypes. The BBB distance assigned each bitmap to the closest archetype: the CLASS1 is the family of bitmaps that are classified as deformations of 1, while CLASS0 consists of those that count as deformations of 0. We designed an NN with two perceptrons to recognize them. After optimization, this NN was able to correctly classify roughly the half of all possible bitmaps. Thus, *the performance of our NN was mediocre*.

What shall we do to improve quality? If a pair of perceptrons were able to recognize half of the bitmaps without classification errors, What would happen if we added another pair? Thus, we will add another pair of AND-perceptrons to recognize a previously

unrecognized pair of bitmaps. So, each archetype will have two perceptrons to its service. We shall research how to maximally expand their recognizing power without errors and after that, they might be joined by an OR-perceptron because the two perceptrons collaborate in the task of recognition. See figure 2.

5 ***Exercise.*** *We plan to join the two perceptrons associated to each one of the archetypes by an OR-perceptron? Would it not be better to use an OR-gate?* *Answer*

We must be ready to add many new perceptrons to improve the recognition power of our NN. Since, all perceptrons associated to each of the two archetypes are inserted into an OR-perceptron, we denominate this addition the **OR-expansion**. That is why we need a method that receives as input a specific bitmap and outputs an AND-perceptron that recognizes it. The resulting perceptron is called *the generated perceptron by the given bitmap*. It is included in program `compareTwoClassifications4.html`.

6 ***Exercise.*** *Run program* `compareTwoClassifications4.html`*, which contains an NN with two perceptrons that were automatically devised, and test it by its results: it must reproduce the very same output of program* `compareTwoClassifications3.html` *of our Intro. Optimize performance by choosing among the options of the select element. Originally, we set the* ***additional bias*** *that enhances the recognizing power to* $0.2 \times n$*, where* n *is the number of entries of each bitmap (all with the same length). Do you have a better proposal? Do you expect it to be firm for the OR-expansion?* *Answer*

We can start our OR-expansion by adding two new perceptrons to the NN with the two perceptrons that recognize the 2 archetypes. The new ones must recognize two bitmaps that the old pair of perceptrons cannot. The bitmap associated with the first new perceptron must be classified as 1 by the BBB distance, so it must be

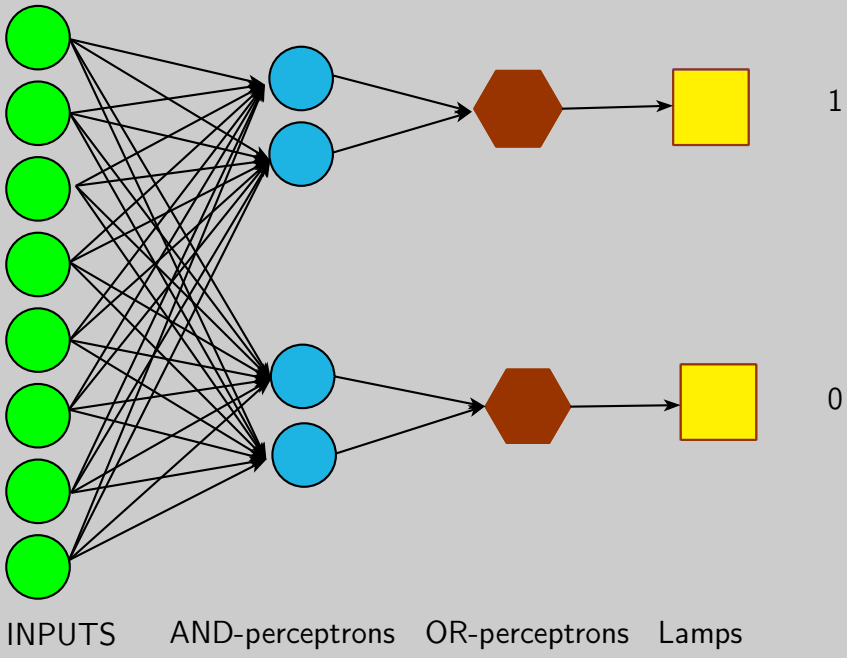


Figure 2: A perceptron is targeted to recognize a bitmap that plays the role of archetype. When it is furnished with a margin of tolerance, it can recognize other bitmaps as deformations of the given archetype. A former NN with two perceptrons was able to recognize 260 bitmaps out of 512 and without a single miss-classification. To improve the classifying capacity or the fault-tolerance, we add another pair of perceptrons. The first new perceptron was targeted to recognize a bitmap of the most distant in CLASS1 to the archetype of 1. The two cells were joined by an OR-perceptron. The corresponding task was solved by a second perceptron for the archetype of 0. We rely on the BBB distance to define what is distant. Each archetype ends with two perceptrons to its service, which might be joined by and OR-perceptron. Te problem is to find the maximal recognizing capacity without mismatches or contradictions.

in CLASS1, while that of the second must be classified as 0 and must be in CLASS0. The problem is that there are more than two hundred bitmaps that the two original perceptrons cannot recognize, while we must pick only two bitmaps. Which ones? Let us choose two bitmaps in the class of the most distant ones. By running program `compareTwoClassifications4.html` and looking at the Text Area TRIPLE CLASSIFICATION, we decided at first to choose the bitmap 101110000 which was classified by the BBB distance as a deformation of 1. For the deformations of 0, we took 101111000. The distance of both bitmaps to their respective original archetypes was 5.

These ideas were implemented in program `addTwoPerceptrons.html`. In this program, the biases of perceptrons P1 and P0 are those that maximize the matches without one single mismatch. In their turn, the biases of the new pair of perceptrons can be changed by choosing an option in the corresponding select element. The perceptron that helps P1 is P1h, while P0h is intended to help P0.

7 *Exercise.* *Run program `addTwoPerceptrons.html` adding the two bitmaps 101110000 and 101111000 of the 5-class, at a distance 5 from the archetypes. The program automatically generates the two corresponding perceptrons that recognize these bitmaps. Is the addition of the two perceptrons of any help? Explain your results.* *Answer*

8 *Exercise.* *Run program `addTwoPerceptrons.html` adding two target bitmaps of the 4-class, at a distance 4 from the archetypes. To choose a pair, help yourself with a list of the elements of this class that is produced by the program `compareTwoClassifications4.html` and published in the Text Area TRIPLE CLASSIFICATION. Are the additional two perceptrons of any help?* *Answer*

9 *Exercise.* *Run program `addTwoPerceptrons.html` adding two target bitmaps of the 3-class, at distance 3 from the archetypes. Are the additional two perceptrons of any help?* *Answer*

We dreamed of a great success if only we added a few perceptrons. But in our former tries in which two perceptrons were added, we got a maximum of 7 additional matches. We have failed to produce a simple and direct solution to the problem of pattern recognition. Thus, let us get prepared for a long war against complexity. The first step is to improve our JS code.

10 *Example.* *Aiming at elegance and simplicity, the fundamental JS classes, Perceptron and NN, were redefined in program* `addTwoPerceptrons2.html` *. The NN was divided into two semi-layers that group companion perceptrons that attack the same type of task, to detect a 1 else a 0. An OR-perceptron was included for each semilayer to join the outputs of its perceptrons. Thus, we have implemented the promised architecture of figure 2. We consider that scalability is guaranteed -with some minor adjustments. This will be used below to implement the Evolution of the architecture of a given NN.*

As a second step to improve our machinery, let us expand our NN to include not just two perceptrons per semilayer, but a whole family or class of perceptrons, which are associated to a class of bitmaps. For each bitmap of the family, a perceptron targeted to recognized it, is added. Its additional bias can be 0 else 1. This was also implemented in program `addTwoPerceptrons2.html`.

11 *Exercise.* *Program* `addTwoPerceptrons2.html` *expands our NN by adding the perceptrons associated with the two 3-classes (it adds all bitmaps at a distance 3 from the archetypes). Run it and publish results.* *Answer*

12 *Exercise.* *What happens if we add all bitmaps of the 4-class only? Hint: to define the class, go to line 1282 of the code of program* `add3Class.html`. *Answer*

13 *Exercise.* *What happens if we add the 5-class? Keep in mind that the additional bias, responding for enhanced recognizing, was a random value amidst 1 and 0.* Answer

14 *Exercise.* *Add some numbers to the following objection against triumphalism: the adding of whole classes helps, but a high cost must be paid because these classes are indeed huge. More to the point, the aim of this research is not to improve matches but to improve matches at zero mismatches. Thus, there is no reason to be glad because while the number of matches grow up, the number of mismatches was also huge.* Answer

2.1 In hindsight

We have looked for an easy and fast answer to the problem of getting perfection. We have failed. So, it is necessary to know whether perfection can be achieved.

15 *Exercise.* *Use your acquired know-how until here to prove that perfection is possible for our model or toy problem. Do you consider that your methodology is applicable to real problems?* Answer

Our conclusion is simple: perfection is achievable but extremely expensive. Our task is therefore to simplify the perfect but costly NN. To that aim, we give to the additional bias of an arbitrary perceptron a value of one if only that causes no mismatches. In that way, a single perceptron can recognize more than one bitmap and the corresponding perceptrons can be deleted. Let us ask for help to Evolution to implement that idea!

3 EVOLUTION COMES TO HELP US

We already have an NN that can perfectly reproduce the classification of a group of bitmaps that is dictated by a distance function. The

NN has 254 perceptrons, all of which have additional bias equal to 0 except for those two that recognize the archetypes, which are privileged. That is why each perceptron can recognize only one bitmap, precisely that that was used to generate it. We can give to the additional bias the value 1 and see if some other bitmaps are recognized but without mismatches. If that is the case, the corresponding perceptrons can be deleted. This checking procedure is what Evolution must maximize.

Let us remark that apart from the additional bias, everything in the NN is frozen. So, Evolution makes two tasks: to choose a subset of perceptrons and to play with their additional bias, 1 else 0, to see if perfection is retained while the NN gets simpler. Thus, Evolution refers in the first instance to the architecture of the NN, more exactly, to its composition of included perceptrons. Why do we expect that this will function?

We expect that smaller samples of perceptrons would do it better because they have less opportunity to create contradictions. Nevertheless, the border condition of zero mismatches seems to be very stringent. That must be cleared out by direct experimentation because we lack such a powerful insight to predict what will be the output.

3.1 A Genetic Algorithm for the optimization of the NN architecture

Our task is to find the simplest NN that perfectly discriminates bitmaps as pertaining to either of two classes, a deformation of the archetype of 1 else 0. The classification of each bitmap is dictated by the BBB distance. The NNs consist of two semilayers of perceptrons that recognize the aforementioned archetypes, plus a set of helping perceptrons that recognize a specific bitmap of the 4- or 5-class. These classes consist of those bitmaps whose distance to the closest archetype is 4 or 5 and that are naturally divided in two subclasses,

one whose members count as deformation of 1 and the other as of 0. That is why we have two semilayers.

The encoding of the architecture and the possibility of its evolution are readily understood. Let us imagine that we have a list of bitmaps. An architecture is encoded in a first trial as a binary string of presence (1) else absence (0) of each bitmap. Such a string defines an architecture because each bitmap generates a perceptron that recognizes it, and the string determines which perceptrons enter into the NN.

Example. If a subclass of bitmaps has 3 members, the string 101 means that the NN has two perceptrons, one to recognize the first bitmap and another to recognize the third. No perceptron is targeted directly to recognize the second bitmap, possible because it is recognized by one of the other two.

Thus, the Evolution of the architectures of NNs is fairly traced by the Evolution of the corresponding binary string encoding them by the presence-absence correspondence. What we shall see is an evolutionary process of binary strings, and the NNs appear only in the evaluation process of their fitness. Nevertheless, simplification is enabled by a non-zero additional bias, thanks to which one perceptron can recognize more than its generating bitmap. But this is not as yet encoded in the binary strings.

16 ***Exercise.*** *Propose and implement a solution for the following problem: to encode presence-absence, we need a binary string. But, besides, we need to manage the additional bias that take in the values 0 or 1. ¿What to do?* *Answer*

Evolution is simulated by a Genetic Algorithm. The evolving population of chromosomes consists of tertiary strings that encode for the architecture of NNs together with the additional bias of each perceptron except those that recognize the archetypes, which are privileged.

The algorithm consists of the following procedures:

1. Initialization: Our algorithm begins with an NN that perfectly reproduces the BBB distance over all bitmaps. That is why all chromosomes of the population at the beginning are strings of ones, 111...1, that means that, except for those perceptrons that recognize the archetypes, all perceptrons are present, and their biases are equal to 0 and, so, each perceptron can recognize at most its generating bitmap. The NN is possibly redundant. Next, the population is subject to a recurrent evolutionary cycle that is expected to simplify it while keeping perfection. The cycle consists of

2. Assessment of the score or fitness of each chromosome. It is defined as follows: Each chromosome, a tertiary string, defines the architecture of an NN by the code: [0 = absence; 1 = present with additional bias 0; 2= present with additional bias 1]. The bitmaps are those of the 4+5-classes. The performance of the NN is measured and it is given a score dictated by the number of correct matches minus a chastisement due to its classification errors. The number of zeroes in the tertiary string is added to the score because it represents the achieved simplicity that results from deleting some perceptrons. Every NN is tested over all bitmaps and the corresponding score is given, which directly determines its fitness.

   The fitness function is

```
score =
-50 * (512 - numberOfMatches) [chastise for  imperfect
- 5 * numberOfMismatches  [chastise for mistakes]
+ 0.1 * numberOfDeletions [reward for simplification]
+ 30000 [constant to make the result positive]}.
```

   where the number of deletions is the number of perceptrons that were deleted from the original NN. It is exactly the number of

zeroes of the given chromosome or tertiary string. The score augments when the number of matches and deletions rises, and decreases when the number of mismatches grows.

3. The fittest is cloned to generate a new population with the same number of individuals.

4. The population is subject to mutation: some changes are made to the clones. Mutation begins with 111...1 and step by step explores what happens when zeroes and twos are introduced.

5. The population undergoes recombination or interchange of information.

6. The cycle ends after a given number of generations.

This algorithm was implemented in program `ENN2Simplify.html`. The program selects the 4+5-classes to expand the NN to improve its performance. The program was started from perfection, i.e., with an NN that recognizes, correctly classifies, all 512 bitmaps.

17 *Exercise.* *Make sure that the program `ENN2Simplify.html` is correct, i.e., that it indeed simulates Evolution. To that aim, manage the printing flags of the program to make it printing the effect of mutation, recombination and selection.* *Answer*

The main exploratory results of program `ENN2Simplify.html` are:

1. Evolution is unable to keep perfection because the number of matches declines to reach a turbulent plateau that represents an equilibrium point + fluctuations.

2. The number of matches tends to stabilize around 70-80% of the initial optimal value. So, an equilibrium value seems to exist.

3. The number of mismatches grows from 0 to more than 100.

4. The number of deletions grows from 0 to nearly 100.

5. The number of matches seems to fit an exponential decreasing function. The number of mismatches and of deletions seems to fit exponential increasing functions. All functions are subject to fluctuations.

The program can be followed through the console which must be activated before starting the program.

These observations need to be tested. So, let us make an experiment to verify else reject the existence of an equilibrium point in the number of matches and to evince the role of mutation and of selection. The existence of an equilibrium value of the number of matches is surprising because the behavior of perceptrons is chaotic: its answer to a stimulus just below the threshold is totally different from that to a stimulus just above it.

3.2 AN EXPERIMENT

To shed light on the validity of the exploratory results that were just formulated, we carried out a suitable experiment, whose characteristics are:

The number of matches was registered for a population size of 10, during 3000 generations for diverse values of the mutation rate and when selection was turned on and off. To turn off selection, all architectures received an score equal to zero. In this case, the program creates variability with mutation, recombination and strong sampling effects. When selection is turned on, the scores of architectures are by merits, such as it was expressed above. To know if there is an equilibrium value, the program was run from two initial conditions: from total perfection when the number of matches is maximal, 512, and from total imperfection when the number of matches is minimal, 0. In the first case, all perceptrons of the NN are active and have additional bias equal to 0. The number of matches in this case must

be 512. In the second, no perceptron is active and hence no bitmap is recognized, the number of matches must be 0. If an equilibrium value exists, both dynamics must converge to the equilibrium value somewhere in the middle. The program outputs the maximal value of matches for each generation.

Results follow. See next page.

EXPERIMENTS WITH NO SELECTION

1. Mutation rate = 0.001.

```
From below
Generation numberOfMatches
 0       0
300     101
600     248
900     259
1200 260
1500 342
1800 332
2100 330
2400 331
2700 334
3000 338


From above
0        512
300     434
600     388
900      377
1200 363
1500 270
1800 281
2100 347
2400 340
2700 269
3000 278
```

2. Mutation rate = 0.01.

```
From below
Generation numberOfMatches
 0      0
300     273
600     269
900     340
1200 201
1500 335
1800 266
2100 269
2400 208
2700 205
3000 262


From above
 0      512 intel
300     275
600     271
900     275
1200 276
1500 350
1800 259
2100 343
2400 338
2700 205
3000 270
```

3. Mutation rate = 0.1.

```
 From below
 Generation numberOfMatches
 0       0
300      262
600      337
900      329
1200 342
1500 269
1800 342
2100 335
2400 266
2700 331
3000 337

From above
 0       512
300      274
600      330
900      331
1200 260
1500 271
1800 273
2100 265
2400 209
2700 336
3000 270
```

4. Mutation rate = 0.5

```
 From below
 Generation numberOfMatches
 0      0
300     278
600     267
900     338
1200 271
1500 338
1800 336
2100 335
2400 197
2700 336
3000 274


From above
0       512
300     262
600     335
900     345
1200 334
1500 337
1800 278
2100 334
2400 271
2700 267
3000 335
```

EXPERIMENTS WITH SELECTION

5. Mutation rate = 0.001.

```
From below
 Generation numberOfMatches
 0      0
300     358
600     368
900     369
1200 378
1500 386
1800 394
2100 401
2400 405
2700 407
3000 403

From above
0       512
300 491
600 479
900 473
1200 468
1500 456
1800 459
2100 450
2400 443
2700 432
3000 434
```

6. Mutation rate = 0.01.

```
 From below
 Generation numberOfMatches
 0      0
300     392
600     408
900     399
1200 412
1500 401
1800 398
2100 392
2400 406
2700 397
3000 401

From above
0      512
300     401
600     403
900     401
1200 385
1500 400
1800 404
2100 404
2400 393
2700 396
3000 409
```

7. Mutation rate = 0.1.

```
From below
Generation numberOfMatches
 0      0
300     357
600     348
900     342
1200 357
1500 349
1800 346
2100 357
2400 351
2700 350
3000 360

From above
0      512
300     353
600     354
900     351
1200 357
1500 346
1800 352
2100 350
2400 358
2700 361
3000 360
```

8. Mutation rate = 0.5.

```
 From below
 Generation numberOfMatches
 0      0
300     345
600     343
900     331
1200 333
1500 339
1800 332
2100 344
2400 339
2700 326
3000 346

 From above
0      512
300     342
600     336
900     334
1200 338
1500 345
1800 349
2100 341
2400 339
2700 336
3000 337
```

18 *Exercise.* *Use a spreadsheet to draw the presented data and draw your conclusions in regard to the possible existence of an equilibrium value, and the role of selection and mutation rate over it. Make sense of results.* Answer

19 *Exercise.* *Implement an exit to the following accusation: if Evolution cannot keep perfection while working in simplification, it is because it is bad programmed. So, the definition of the score function must be changed.* Answer

Our analysis until this point possible contains a contradiction. On one hand, we say that the mutation rate is disconnected from the level of the equilibrium point. On the other, we observed that the velocity of approaching the equilibrium value is lower the lower is the mutation rate. What could happen when the mutation rate is very low?

20 *Exercise.* *Using the newly defined score, test the idea that a lower mutation rate will produce a less imperfect result. Begin with* 0.0005*. Would you agree that for very low mutation rates, perfection is the rule?* Answer

The results of the last exercise show that **Evolution can create and maintain perfection for the problem of simplifying a NN**, 70 perceptrons were deleted out of 254.

21 *Exercise.* *Do you consider that the good news about the simplifying power of Evolution over NNs applies to the real world?* Answer

22 *Exercise.* *Make your best to organize all results along the Official Scientific View on Evolution.* Answer

23 *Exercise.* *Do you consider that our results falsify else support the Official Scientific View on Evolution?* Answer

4 CONCLUSION

We built with our human intelligence an NN with 254 perceptrons that can correctly recognize all 512 possible bitmaps of a given family that were classified by a distance as deformation of an image of number 1 else of 0. We gave to Evolution the task to simplify the NN while keeping perfection, i.e., without mismatches or unknowns. Evolution readily can simplify the NN, if imperfection is welcome: we achieve a simplification of 70 perceptrons together with many mismatches and unknowns. But to get perfection, 512 matches while no mismatches or unknowns were produced, we needed to lower the mutation rate. A value of 0.0005 per char per generation was sufficient for us. This is a poor result for those persons that want to sustain the ET because it solves a detail of an extremely complex problem. In contrast, it fits the needs of those that want to falsify the ET because we see that Evolution is an ordinary tool that burns time and resources and therefore tons of imperfections is what must remain behind perfection when it is achieved. Nevertheless, this is not a falsification because it lacks predictions in regard with the luck of fossils. That is why we have recurred to abstraction, as it was shown in previous works.

5 ANSWERS TO EXERCISES

1, page 4. A perceptron fires when the stimulus surpasses a given threshold. Firing is interpreted here as an affirmation that a pattern has been found. But if a stimulus does not cause firing, the perceptron remains as if there were no stimulus. Thus, the perceptron ignores it because of two possible causes. First: the stimulus does not represent the target pattern. Second: the insufficient discriminant capability of the perceptron.

We consider that when the perceptron fires, it represents a YES. But if it does not fire, it is proclaiming that the classification of the

stimulus is UNKNOWN to it. Hence, the overall behavior is two-valued, YES and UNKNOWN, but it is not boolean because from its silence we cannot conclude that the stimulus does not correspond to the target pattern.

3, page 8. Let us suppose that a perceptron is designed to recognize the archetype of 1 whose bitmap is 010010010. The corresponding inequality is $S = -e_0 + e_1 - e_2 - e_3 + e_4 - e_5 - e_6 + e_7 - e_8 + 6 \geq T = 9$. If we plug in the archetype, we get $S = 0 + 1 + 0 + 0 + 1 + 0 + 0 + 1 + 0 + 7.8 = 10.8 \geq T = 9$ that says that the archetype is recognized. Other bitmap would have at least one mutation. Let us consider, for instance, 110010010, which renders $S = -1 + 1 - 0 - 0 + 1 - 0 - 0 + 1 - 0 + 6 = 8 < 9$, which means that this bitmap is not recognized as 1. Or, consider 000010010, which renders: $S = -0 + 0 - 0 - 0 + 1 - 0 - 0 + 1 - 0 + 6 = 8 < 9$ so, it is not recognized as a deformation of 1. And so on: a mutation in the archetype of 0 into 1, changes a 0 by a -1 in the sum, while a change of a 1 into 0, changes a 1 by a 0 in the sum. So, the new sum is lower than that of the archetype. We conclude that without additional bias, the perceptron recognizes only the archetype. Therefore, the additional bias is the responsible term for the tolerance of the perceptron.

4, page 8. Suppose we have to discriminate between two patterns. We devise an NN with two terminal lamps, the first shines when the first pattern is caught, the second when the second pattern is evinced. In a boolean logic, if the first pattern is not caught, the second must. And vice versa. As we will witness, such certainty is tremendously elusive because it is easy to commit classifying errors, in which both terminal lamps fire, meaning that they both claim to have found its corresponding pattern. From the boolean perspective, the usual one for an important sector of our lives, this is a contradiction. Or when the wrong lamp shines while the correct one remains turned

off. This is a mismatch and also a contradiction against the correct classification. Thus, one must get content with an approximate boolean logic but trying to diminish as uncertainty as contradictions as much as possible. This implies that we deal with an optimization problem, a topic that welcomes the Genetic Algorithms, which are simulations of Evolution, which we have explored and will continue to do so.

5, page 9. Yes, it would be simpler to join the And-perceptrons by an OR-gate. But, the important aim for us is to ease the coming recursive expansion and this will be simpler with perceptrons instead of gates. In fact, we plan to add a pair of perceptrons to expand the recognizing capabilities of the NN but we have no certainty that this will be enough to reach perfection. So, we must be ready to add more pairs and is in that way that we get an OR-expansion. An OR-perceptron will be the indicated to deal with this situation.

6, page 9. The optimal values of the biases were found to be 9.8 for perceptron P1 and 4.8 for P0. The number of matches were 260 and mismatches 0 out of 512 possible bitmaps. To fit these values, we set the additional bias to $0.4 \times n$, where n is the number of entries of each bitmap. To do this, we used the definition of the total bias: it is the number of zeroes in the bitmap, or of -1 in the weight vector, plus the additional bias to enhance the recognizing power. Now, can we use this formula indiscriminately?

No. This formula, that always can be questioned, might be acceptable for isolated perceptrons. But in the OR-expansion we have two modules, each one with a main perceptron, targeted to one of the archetypes, and helper ones that must recognize additional bitmaps but without contradicting the BBB distance. That is a totally new problem. Hence, the optimal value must be researched. We must enable the perceptron to accept those bitmaps that differ in one place from its target. Thus, the additional bias might be uniformly set to

1. By the same reason, the bias might grow by one unit by step when looking for its optimization when the question of adding or not a new perceptron is at stake.

7, page 11. The best performance for additional perceptrons P1h and P0h with bitmaps 101110000 and 101111000 respectively was 262 coincidences with 0 mismatches. This was achieved by selecting the bias of P1h as 5 and that of P0h as 4. This means a gain of 2 matches among more than 250 that lack. The result is insignificant. Why it is so tiny?

The distance between these two bitmaps, 101110000 and 101111000, is 1, but according to the BBB distance, the first represents a 1, while the second a 0. This means that the respective AND-perceptrons that recognize either of them cannot have a margin for tolerance when 0 errors are mandatory. Therefore, the new perceptrons recognize the new target bitmaps, and nothing else. Is that true? This is a test for our whole programming. Is that true?

Yes. It is true. By looking at the results that appear in the Text Area QUINTUPLE CLASSIFICATION, one can see that with one exception, all matches belong in classes 0, 1, 2, and 3, which were already accepted by the original NN with just two perceptrons. The exception corresponded to 101110000, the helper target 1. The same happens with the second helping target 101111000: the optimal total bias is the initial bias. But, why only they? To know that, we noted that the optimal total bias of the first perceptron was 5, which is just the bias corresponding to 101110000, which has 5 zeroes, without margin for enhanced recognizing. What does it happen if we increase the bias? By looking at the results with total biases 6 and 4, we find in the aforementioned Text Area that the NN correctly classifies 6 new bitmaps that are classified by the BBB distance as deformations of 1. No mismatch for this case. However, 4 bitmaps that are deformations of zero were classified by perceptron P1 as 1 and by perceptron P0 as 0. Thus, the best performance has no tolerance.

8, page 11. With target bitmaps 000000001 and 000001111 of the 4-class, 262 matches were obtained with 0 errors by selecting the bias of P1h as 8 and that of P0h as 5. If the biases are 9 and 5, the number of matches is 267 without mismatches. This is not an announced failure because the distance between these two bitmaps is 3. The code was modified circa line 280.

9, page 11. With target bitmaps 000000000 and 000101111 of the 3-class, only 260 matches were obtained with 0 errors by selecting the bias of P1h as 9 and that of P0h as 4. But if the biases are 9 and 5, the number of clean matches is 266. This is not an announced failure because the distance between these two bitmaps is 5.

11, page 12. We chose the 3-class and the corresponding expansion was consigned in program `add3Class.html`. From program `addTwoPerceptrons2.html` we learned that the additional bias could be 0 else 1. Thus, we assign 0 else 1, chosen at random, as additional bias of each perceptron of the network. After some tests, the number of matches was very close to 360, that of mismatches 67 and unknowns 85 for a total of 512, which is the number of possible bitmaps. Therefore, we will forget the whole 3-class.

12, page 12. If we add the 4-class, the number of matches was very close to 376, that of mismatches 133 and unknowns 3 for a total of 512. Unknowns were sacrificed in favor of matches and mismatches, but the great favored was the last.

13, page 13. Program `add3class.html` was modified in line 1282 to select the 5-class, composed of those bitmaps that are at a distance of 5 from the corresponding archetypes. This is the output :

```
 The chosen class is the = 5-class.
 Number of bitmaps in \textsc{class1} = 35
 Number of bitmaps in \textsc{class0} = 35

runNN(myNNClass)
 numberOfMatches 380, 74.21875%
 numberOfMismatches 68, 13.28125%
 numberOfUnknowns 64, 12.5%
 Total 512
 Score = 40
```

We declare that a random assignation of 1 to the additionalBias does not work because it can increase matches but at a high price in mismatches.

14, page 13. The number of bitmaps in the 3-class is $84+84=168$ while the gained matches were nearly a hundred. The number of bitmaps of the 4-class is $91+91=182$ while gained matches were a bit more than a hundred. If we add the fifth class, we gain some 120 matches, but these classes have a size of $35+35=70$. So, the objection is right.

15, page 13. Program `perfection.html` shows that it is possible to make 512 bull-eyes at zero mismatches. This is what we did:

We built a NN with two semi layers. The first was addressed to recognize the archetype of 1 and its deformations, while the second those of 0 and its deformations. Each semi layer consisted of one perceptron to recognize the corresponding archetype and its deformations at distance 1, 2 and 3. For each bitmap at distance 4 or 5 from each one of the two archetypes, a helper perceptrons was added to recognize it. Each bitmap had a consecrated perceptron with additional bias equal to zero that implied that the recognizing

relation was one-to-one in the 4 and 5 classes. The total number of perceptrons was 254 which is roughly the half of 512 the number of bitmaps to be re-classified. Thus, it is not realizable to use this method to achieve perfection in real applications because the number of needed perceptrons grows exponentially in the number of bits of the bitmaps.

16, page 15. We solved the problem with the additional bias as follows: we do not use a binary string to denote the presence-absence of a bitmap. Instead, we use a tertiary string whose entries can take on the values 0, 1 or 2. Option 0 is interpreted as absence of the corresponding bitmap and perceptron. Number 1 means that the corresponding bitmap is present and that the additional bias of the associated perceptron is 0. Number 2 means that the corresponding bitmap is present and that the additional bias of the associated perceptron is 1. Example: if we have a chromosome of the form 021, it means that we have a family of bitmaps with three members. Of them, the first one was not taken, while the second was taken and granted an additional bias of 1 to enable recognizing of more than the generating bitmap. In its turn, the third member was taken and was given an additional bias of 0.

17, page 17. In program `ENN2Simplify.html`, the number of generations to be run was put equal to 5, line 351, and the population size, line 357, equal to 10. The printing flags are below line 369. Mutation became visible with the change `$PRINT_MUTATION_IND$=true`. No fault was found. The printing of the effect of recombination was activated with `$PRINT_RECIND$=true`. It was very easy to witness that its work is correct. The methods that execute selection are various and complex. The principal methods were `function assessFitnessPop(myNN, CLASS1, CLASS0)` in line 1719 and `function assessFitnessChro(myNN, chromosome, CLASS1, CLASS0)` in line 1662. No extra correction was made.

In conclusion, no error was caught by the Author.

18, page 28. With the help of graphics but no statistical analysis, we interpret these results as follows:

- Evolution is powerful enough to create function up to 0.8 of perfection in less than 3000 generations with a population of 10 individuals.
- The number of matches has an equilibrium value around which the process fluctuates.
- The equilibrium value is undetermined but is bounded from above. In fact, the program reports the maximum number of matches of each generation. Its number ranges from 250 to 450. Thus, the average value is bounded above by 450. This proves that Evolution is incapable of reaching or keeping perfection, which is represented by 512 matches.
- Apart from rare exceptions, deletions are accompanied by large numbers of unknowns and mismatches (not reported in the precedent tables).
- The average value of matches without selection is nearly 300 but when selection is turn on, it is 400. We see that the equilibrium value is highly sensitive to selection.
- The value of the mutation rate does not affect the equilibrium value but affects the velocity of approaching it, i.e., the velocity of approaching is lower for low mutation rates.
- Paths that lead to imperfection are very common.

We rationalize our inferences as follows:

1. The behavior of perceptrons is chaotic: the answer to a stimulus just below the threshold is quite distinct from that of another a

bit higher than the threshold. Our observations say that chaos is visible through the ensuing fluctuations, which dance around an equilibrium value.

2. Mutation and recombination are continuously creating and destroying matches. The quantity of creations by mutation is proportional to the number of no-matches, and that of destruction to that of matches. We have a creation-annihilation process. Equilibrium is attained when creation balances annihilation. The equilibrium value is blurred by fluctuations due to the chaotic nature of the behavior of perceptrons and to sampling effects that must diminish when the population size grows.

3. The equilibrium value is stable. It is not like a needle that spins on its spike, but like a ball that fluctuates around the bottom of a bowl. This means that the equilibrium value absorbs as initial conditions as fluctuations that may appear by a combined effect of selection and lucky mutations. The stability of the equilibrium value implies that Evolution gets locked to imperfection.

19, page 28. We have been working with the score given by

```
score =
-50 * (512 - numberOfMatches) [chastise for  imperfection
- 5 * numberOfMismatches  [chastise for mistakes]
+ 0.1 * numberOfDeletions [reward for simplification]
+ 30000 [constant to make the result positive]}.
```

Let us try now the following definition:

```
score = number of matches
        - 5 * number of mismatches
        + 0.1*number of deletions
```

The program was run over 10000 generations beginning with total perfection and with mutation rate equal to 0.001 and population size 10.

```
Generation  Number of matches
0       512
1000 511
2000 509
3000 510
4000 510
5000 512
6000 512
7000 511
8000 508
9000 510
10000 511
mismaches were always zero,
unknowns were less than 4,
deletions nearly 60.
```

From these results, we conclude that the Evolution of this NN is highly sensible to the form of selection because such high stability of quasi-perfection was not observed with the previous form of the score. Additionally, Evolution cannot keep perfection for the mutation value of 0.001, but almost-perfection is the rule.

20, page 28. We tested a mutation rate equal to 0.0005 over a population of 10 individuals. Our results were:

1. Experiments without selection

```
From below
MUT_RATE = 0.0005
Generation matches
0    0
1000 158
2000 181
3000 199
4000 273
5000 267
6000 268
7000 262
8000 266
9000 263
10000 329
11000 330
12000 339
13000 334
14000 342
15000 197
16000 200
17000 258
18000 277
19000 282
20000 282
```

From above
without Selection
MUT_RATE = 0.0005
Generation matches
0 512
1000 401
2000 287
3000 355
4000 345
5000 343
6000 335
7000 334
8000 201
9000 204
10000 194
11000 271
12000 268
13000 263
14000 333
15000 339
16000 341
17000 337
18000 344
19000 343
20000 266

2. Experiments with selection

```
From below
MUT_RATE = 0.0005
Generation matches
0    0
1000 351
2000 400
3000 503
4000 511
5000 511
6000 512
7000 512
8000 512
9000 512
10000 512
11000 512
12000 512
13000 512
14000 512
15000 512
16000 512
17000 512
18000 512
19000 512
20000 512
Number of deletions 70
Generation = 20000
numberOfDeletions = 70
numbOfMatches = 512
numbOfUnknowns = 0
numbOfMismatches = 0
score = 519
```

```
From above
MUT_RATE = 0.0005
Generation matches
0       512
1000 512
2000 512
3000 512
4000 512
5000 512
6000 509
7000 512
8000 512
9000 512
10000 512
11000 512
12000 512
13000 512
14000 512
15000 512
16000 512
17000 512
18000 512
18000 512
19000 512
20000 512
Generation = 20000
numberOfDeletions = 70
numbOfMatches = 512
numbOfUnknowns = 0
numbOfMismatches = 0
score = 519
```

With the help of graphics, it became evident that there is a huge difference between outputs without selection and with selection. When there is no selection, the equilibrium value of the maximal number of matches is around 265. When selection operates, it was 512, i.e., perfection and fluctuations are quite few.

21, page 28. We think that our methodology does not apply to real life because the number of bitmaps is an exponential function of the number of bits and our simplifying power is just a fraction of that cost. Thus, the number of perceptrons in the simplifying NNs continues to be exponential, i.e., our approach is untenable for real applications. Our reasoning is based on our intuition and not on mathematics. This warning is worth researching because we do not know how scales the simplification fraction with respect to the length of chromosomes.

22, page 28. We can accommodate our results about the Evolution of NNs and the brain along the very simple scientific credo of Evolution: NATURAL EVOLUTION BEGAN WITH SIMPLE PROPOSALS THAT AROSE BY MERE RANDOMNESS THAT WERE REFINED WITH THE COURSE OF GENERATIONS. THERE IS A NATURAL TENDENCY TO PERFECTION THAT WAS ENABLED BY A DIMINISHING OF THE MUTATION RATE THAT ENSUED A REFINEMENT IN REPLICATION FIDELITY AND IN REPAIRING MECHANISMS.

23, page 28. According to our results until here, Evolution explains a tiny fraction of the whole problem (the moderate optimization of the architecture of an NN) but its core (the origin of the architecture and its components) is untouched. So, it is shameful to claim something in favor of the ET (Evolutionary Theory). But a light of hope appeared in the last experiment, in which the mutation rate was given a very low value. That is why, we felt forced to rerun a program of our Intro in which Evolution failed to emulate the

achieved quality of a man made NN with 2 perceptrons.

Program EvolNN4.html was targeted to have Evolution optimize the weight values of the two perceptrons of the NN, a fundamental item of perceptrons that, in our Intro, we were unable to specify with the use of Evolution. We ran it using a lower mutation rate, 0.0001, and a recombination rate of 0.1. The program was run on a laptop with 16GB of RAM and we halted it after 14 hours of work. What did we get?

Let us highlight some results:

1. An always-rising tendency was observed during 156367 generations.

2. In general, the more perfected the NN, the longer the waiting time for a higher state. It may be interesting to look at the numbers that show the generations in which an improvement happened:

Genertn	score	matches	mismatches	cost
0	44	64	4	100
719	64	104	8	72000
737	72	72	0	73800
904	112	112	0	90500
40222	152	152	0	4022300
61972	212	212	0	6197300
156367	212	212	0	15636800

The cost is the number of left fossils until the given generation. If the experiment is repeated, the numbers may change, but the pattern remains. If the mutation rate rises, the segments in which Evolution always goes up get shorter, but recovery begins at once.

3. Perfection does not grow steadily, rather it grows by jumps, say, from 152 coincidences it passes to 212 in one generation, but after a very long waiting time. This is compatible with the claim that Natural Evolution was responsible for the observed jumps in the morphology of the shells of snails (Wikipedia, [3] 2024). But in our case, mutations in the weight values were not macro, as we can see here:

Genertn	Weight values of P1	Weight values of P0
0	0,0,-1,-1,0,0, 0, 0,-1	1,1,0,1,-1,0,1,1,-1
719	-1,0, 0, 0,1,0,-1,-1,-1	1,0,0,0, 1,1,1,0, 1
737	-1,0, 0,-1,1,0,-1,-1,-1	1,0,0,0, 1,1,1,0, 1
904	-1,0, 0,-1,1,0,-1, 0,-1	1,0,0,0, 1,1,1,0, 1
40222	-1,0, 0,-1,1,0,-1, 0,-1	1,0,0,1, 1,1,1,0, 1
61972	-1,0, 0,-1,1,0,-1, 0,-1	1,0,1,1, 1,1,1,0, 1
156367	-1,0, 0,-1,1,0,-1, 0,-1	1,0,1,1, 1,1,1,0, 1

4. In 156367 generations with a population size of 100, the top achieved number of matches was 212 of 512 at zero mismatches.

Thanks to this and many other experiments that we have done, we recognize the creative power of Evolution. Now, does Evolution have a signature that leaves on its products? Yes, and very firmly: Together with the proposal of solutions to posed problems, Evolution leaves abundant witness of its many, many trials done before halting. Our results pile naturally around our usual accusation against the Evolutionary Theory: Evolution is not magic, it cannot resolve problems immediately. Rather, it is an ordinary tool that employs time and resources to propose approximated solution to posed problems. Thus, we ask over and over: where are in the fossil record the footprint of the too many trials of Evolution, dressed of imperfections of every sort, before attaining the perfection that I see in most children? Since the number of attempts needed to rise to the next level of quality grows disproportionately, we consider that we cannot

claim that sampling effects can be responsible for the unobservability of the Evolution of Perfection.

Nevertheless, our results in isolation are debatable. This truth together with the fact that Evolution is an algorithm have propelled us to abstract the problem of the Evolution of Perfection along the general theory of the complexity of algorithms (Rodriguez [2] 2021). The idea is that Evolution would require many eternities to develop life such as it exists, and therefore in the present universe -with an extremely short life - everything made by Evolution must carry its trademark of imperfection. Thus, a mandatory prediction arises: imperfection shall be visible and abound in my own body. This is false. I am a very old man and, nevertheless, the quasi-perfection of my body marvels me.

The Evolutionary Theory is false.

But given that Evolution is a crude reality, it must cause mutations that impair the genome. This poses the problem of how to live together with Evolution when perfection is demanded. We consider that this is done by fault-tolerant design, such as our biology is, in which pre-prepared solutions solve specific failures. For example, coagulation is necessary to control hemorrhages, but it can happen inside our vessels and this would cause blood clots. Fault-tolerance means in this case that we have a machinery that dissolves internal clots. This machinery is encoded in our genomes. That is why, a bruise caused by a blow disappears in within one week or two.

6 THE BIBLIOGRAPHY

References

[1] José Rodríguez (2023)
Evolutionary Neural Networks 1. Introduction.
https://ejristos.com/Evol%20Neural%20Networks/EvolNNintro.pdf 3

[2] José Rodríguez (2021)
Falsification of the Evolutionary with flipp.
https://ejristos.com/GPfalsifiesEvo/FalsificETwithFlipp.pdf 46

[3] Wikipedia contributors. (2024).
Punctuated equilibrium. *In Wikipedia, The Free Encyclopedia. Retrieved 13:53, April 2, 2024,.*
https://en.wikipedia.org/wiki/Punctuated_equilibrium 45

These links were active by 30/V/2024