The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.69 e-ISSN: 2251-7545

"Visual" Mathematics Used in a Novel Machine Learning Architecture

Neven Dragojlovic
CEO
EYEYE Sarl au
Tangier, Morocco
eyeye689@gmail.com

***Abstract*— At some point or another, everyone has surely had the experienced of, staring at mottled surroundings without any thought in mind and finding how one's visual system started detecting patterns - like faces or animals or objects – in the mottled background. Once those patterns have been detected, moreover, it becomes difficult to observe the mottled surroundings and not perceive the patterns without engaging in active repression. This spontaneous organization of input into recognizable molds starts with just a few patches of visual input that activate a schema that is already present in the memory structure. The visual system then queries the surrounding features to see if they fit into the activated schema, thus constructing a richer and more compelling pattern. Psychology would interpret this process as a Rorschach test that shows what our unconscious mind is primed to perceive at any moment.**

This type of experiences suggeststhat, in order for intelligence to emerge, controlled chaos is necessary. Self-feedback and multiple calculation points, each of which follows simple rules, permit the emergence of chaotic attractor-networks, which bring stability to a system, and which, if networked together can create intelligence. This paper describes a computational system that is capable of such a feat, depending only on a mechanical process that does not require thought or consciousness. The systemonly requires local processing units, their associated memory, and simple software that interprets its immediate environment, (that is, the activity of surrounding processing units). In order to be functional, such a system cannot limit itself to the simplest possible case (such as letters or simple geometric shapes), but must be able to process all types of input and form active networks out of it.

The system described in the following article uses a fully parallel pattern-type language that can be used in multiple, easily joined modules, where each module can be used in processing a specific type of information. As it uses simple programs in each computing element, the information is easily integrated and debugged. Complex statistical models, which form the foundation of most current search and recognition algorithms, are not necessary in this system as it automatically uses simple search and recognition strategies at each computing component.

(Based on U.S. Patent 7,426,500 and pending patent US13/117,176)

*Keywords:**Cellular parallel architecture; distributed memory; distributed computing; swarm intelligence; hexagonal framework; chaotic attractors in complex systems; networks; nodes with relationships; clusters; matrix vectors; multi-level up and down information flow; language analysis; language analysis***

INPUTUNIT

The basic component in this architecture is the Processing Cell (PC), a roughly hexagonal structure in with one central hexagon ($H^{(level)}$ – the colored cells in Figure 1)connects with the 6 surrounding hexagons ($h_{(binary)}$) in a 2-dimensional hexagonal lattice. Each $H^{(level)}$ is also connected to cells in higher – or lower-level lattices in which $h_{(binary)}$ of a PC in a higher-level layer are each connected to the $H^{(level)}$ of a corresponding PC in the level immediately below it. Information is processed as it flows from a base-level Input Layer through to successive higher-level layers in which schemata of increasing generality are formed.Each PC is identified by its central hexagon address (for example PC_{000}).

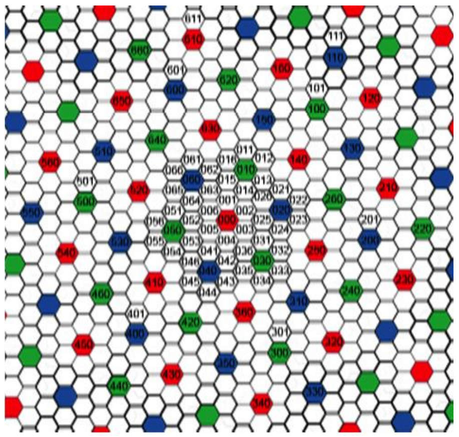


Figure 1

In the input layer, the 6 $h_{(binary)}$ in each PC detect the stimulus covering their respective area of the input field (for example, light intensity in visual processing), and compare it to the threshold level that the stimulus must reach in order to change the $h_{(binary)}$'s state from 0 to 1. If X_i and Y_i represent the (x, y) coordinates of input pixels found in a rectangular input field (like photographic memory in digital cameras, or an internet page), N represents the number of those pixels in the area covered by a specific $h_{(binary)}$, and (X_i, Y_i)represents each pixel's activation status (0 or 1), The stimulus intensity for $h_{(binary)}$ can be represented as:

$$S = (\sum (X_i, Y_i))/N \text{ for } i = 1 \text{ to } N$$

Given a threshold constant K, the value of $h_{(binary)}$ is set such that:

If $S \geq K$,
Then $h_{(binary)} = 1$ (Activated)
Else $h_{(binary)} = 0$ (Not activated)

The values of the 6 peripheral hexagons in each PCare recorded its central hexagon ($H^{(level)}$) using the following algorithm. Each peripheral hexagon in a given PC is assigned a unique binary identifier based on its position in the tile . As Figure 2 shows, starting with the hexagon at the top of the tile, each hexagon is identified by 2^K, where K ranges from 0 to 5. To record the pattern of activated hexagons in each PC, then, the activation value of each peripheral hexagon (1 or 0) is multiplied with the hexagon's identifier (2^0 through 2^5 in base-ten notation), and the binary values for each all six hexagons are summed, resulting in a single six-digit binary value ranging from 000000 to 111111 (i.e., 64 possibilities). For any given value, each "1" indicates the activation of the peripheral hexagon corresponding to that place-holder, meaning that each specific binary value represents a specific spatial arrangement or pattern of activated peripheral hexagons in the PC.

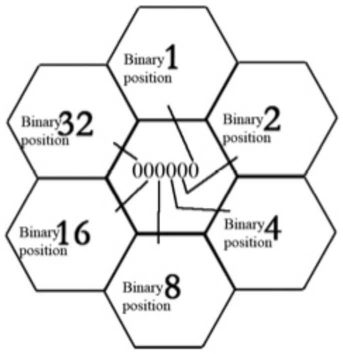


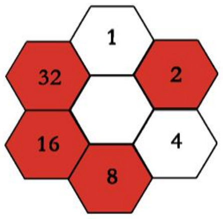
Figure 2

Each hexagon in the full hexagonal lattice, has its own unique identifier, as shown in Figure1. These identifiers are 3-digit decimal values in which the first two digits identify the central hexagon of each PC, and the last digit takes on the values 1-6 and identifies the peripheral hexagons in each PC, starting at the top of the tile and proceeding clockwise.

Figure 1 shows an arrangement of seven Processing Cells in the input layer along with their addresses (PC_{000}, PC_{010}, PC_{020}, PC_{030}, PC_{040}, PC_{050}, and PC_{060}). Together, seven PCs together form a Patch ($P^{(level)}{}_{(address)}$) whose address is taken from the central PC's address (in this case $P^0{}_{000}$). The Patch is a unit that sends and receives information from one Memory Unit (MU). The Memory Unit (MU_{000}), which is described in detail below, has an address based on the Patch address, in this case 000.

The addressing system extends to surrounding patches, and with a further addition of binary places can extend to Patches-of-Patches as far as necessary. Lee Middleton and Jayanthi Sivaswamy have developed this Hexagonal Image Processing (HIP) numbering system, and the associated mathematics [6]. It permits representation of hexagonal images as vectors in which the ordering of elements (which excludes the central hexagons for each PC) is established by the unique identifiers for each hexagon, and the elements themselves hold the 0 (inactive) or 1 (active) values.

In the following example, the hexagons where the stimulus exceeds the threshold are represented by a color, and their binary number changes from 0 to 1. The central hexagon represents that pattern in a binary and a decimal form (see Figure 3).



Binary form: 111010
Decimal form: 58
Figure 3

The two ways of representing this activated pattern are interchangeable. The decimal number represents its binary equivalent and is used more often for simplicity and greater readability, though both are presented because the binary representation allows for a direct reconstruction of the activated pattern. As noted earlier, this system's architecture includes an input layer and an arbitrary number of hexagonal lattice processing layers (the algorithms operating in the input and higher-level layers differ, however). To identify a specific hexagon in the system, the hexagon's level is represented by a superscript ($h^{(y)}$ for peripheral hexagons and $H^{(y)}$ for the central hexagons in each Processing Cell; $H^{(0)}$, for example, is a central hexagon in the input (0) layer. The position of the hexagon on the layer is indicated by a subscript using the unique identifier discussed above. This designation system allows us to precisely describe the connections between the input layer and higher-level processing layers.

To begin processing the stimulus, the activation pattern from each PC in the input layer is sent by the central hexagons ($H^0{}_x$) to a uniquely associated hexagonal cell in the upper layer PC, such that each Patch of 7 PCs provides input to a single higher-level Processing Cell. In the example shown in Figure 4, the seven H^0s (H_{000}, H_{001}, H_{002}, H_{003}, H_{004}, H_{005}, H_{006}) send their output (PCO) to the associated hexagon in the higher-level lattice (in a sense, it overlaps the lower-level PC).

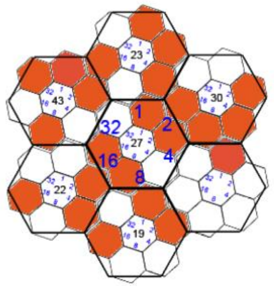


Figure 4

Upper level binary position	*Lower level PC output (PCO)*	*Lower level H_x address*	*Initial activation threshold*	*Initial Upper level activation*
H^1_{000}	27	000	1/6	1
h^1_{001}	23	060	1/6	1
h^1_{002}	30	010	1/6	1
h^1_{003}	01	020	1/6	1
h^1_{004}	19	030	1/6	1
h^1_{005}	22	040	1/6	1
h^1_{006}	42	050	1/6	1
Total number of 1s in the input	23	Average activation 23/7= 3	Revised activation threshold: 3/6	Initial APN: 127 Revised APN: 123

Table 1

The processing of input in this layer is, however, different than that in the input layer. Specifically, the higher-level Processing Cell engages in an iterative process in which only the strongest signals of the input Patch are retained. Initially, each hexagon in PC^1_{000} (including the central hexagon H^1_{000}) determines whether the number of activated hexagons in its associated lower-level PC^0_x meets its activation threshold, which is initially set at 1/6. In Table 1, all of the inputs PC^0_x have at least one activated hexagon, meaning that all the upper-level PC's cells are activated.

To extract more information from the signal, however, H^1_{000} resets the activation threshold for the whole PC using the following algorithm. It averages the total number of activated hexagons in the input layer Patch, rounding to the nearest whole number. For the input in Table 1, this results in a new threshold of 3, which results in the deactivation of h^1_{003}, since the associated PC^0 only had one activated hexagon.

Once the the activation threshold is adjusted, the upper layer PC (in this case, PC^1_{000}) switches from an Input Mode to an Output Mode, and creates an Active Patch Number (APN) showing which PC^0s activated its component hexagons. The APN of PC^1_{000}, in this case 123 (1111011), is formed by adding the binary positional representations of its activated hexagons, ranging from h^1_{001} (1, or 000001) to h^1_{006} (32, or 111111) and H^1_{000} (64, or 1000000). Each APN, then, is associated with a unique activation pattern for PC^1_{000}. The APN for each higher-level PC is then sent for storage to a dedicated Memory Unit (MU), which is described in the following section.

MEMORY UNIT

Each MU is composed of 64 Dedicated Cells (DC), where each DC corresponds to a unique PC^0 activation pattern (primitive). These DC primitives are shown in an "exploded" form in Figure 5. All seven hexagonal cells in a PC^1 send output to their associated MU in the following format: PCO^0 / APN / H^1_xbinary position*x*. H^1_{000}, for example, would send 27/123/000,or in binary form 011011/1111011/000, to all MU's DCs in a binary position 000 (center of each DC number), whereas h^1_{001} would send 23/123/000 to binary position 001, or in binary form 010111/1111011/000. Each DC is only activated (activation status set to 1 instead of 0) if the first 6 bits of at least one input number correspond to its number (the yellow hexagons in Figure 5 represent activated DCs). The activating numbers are stored in the DC's binary place, which is the same as the binary position of sending PC in the patch.

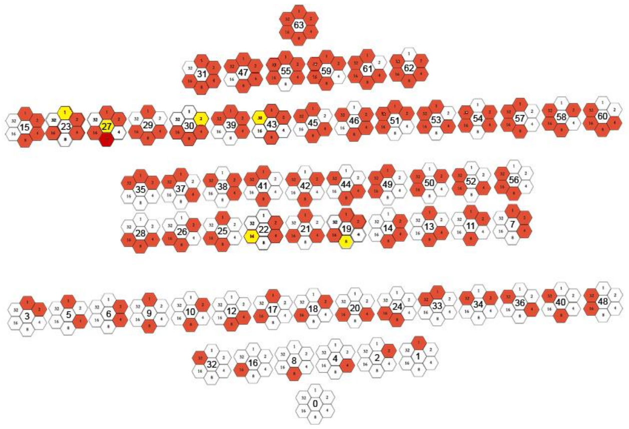


Figure 5

The key innovation in this memory system is the 3D structuring of the MU, which can be visualized as a set of 8 truncated octahedrons connected at their truncation planes (see Figure 6, which shows three MU and how they would connect in the horizontal plane). This yields 64 hexagonal surfaces, which correspond to the 64 DCs, and can be thought of as equivalent to the H^0_x's in the associated input layer Patch. Specifically, the edges of each DC correspond to a unique digit's place and activation status in the 6-digit binary number associated with the activation pattern the DC corresponds to. The activation state of this digit is preserved across DCs, such that each shared hexagonal border in the MU represents either a 1 or a 0. The implication of this is that each DC hexagon connects only with DCs whose PC^0 activation pattern numbers differ only in one binary place. For example: the hexagon containing number 30 (011110) would connect to the hexagons representing numbers 31 (01111**1**), 28 (0111**0**0), 26 (011**0**10), 22 (01**0**110), 14 (0**0**1110), and 62 (**1**11110)(see Figure 6).

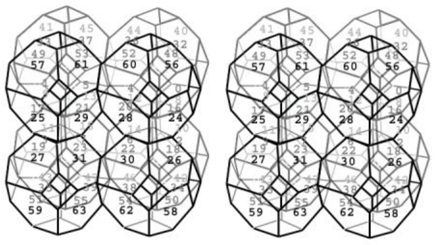


Figure 6

The ends of each MU attach to other MUs corresponding to the Patches that surround the original patch (H^0_{000}) in the input layer. In effect, this transposes the 2-dimensional structure of the input field into a 3-dimensional memory representation (which, as is described below, is what allows the activation of previously stored "schemata" by related, but not identical input patterns). In this case, for example, the patches H^0_{100} and H^0_{400} (see Figure 1) are represented by neighboring MUs in the y-axis direction, H^0_{200} and H^0_{500} in the z-axis direction, and H^0_{300} and H^0_{600} in the x-axis direction. As the set of MUs are expandable in 3D, allowing for the absorption of input from as large an input layer as necessary, and because no MU's activation state is fully determined only by its own patch input (as described below), different sets of meaning structures in the overall memory can interact with each other at a distance, leading to a unified semantic space in the system's memory.

The activation state of each DC can be influenced by the surrounding DCs. Each DC is connected to surrounding DCs through zeroes and ones. When a DC gets activated from the patch, a strength number is assigned to it ($S_{DC} = y$).Then it would stimulate DCs that are connected to its zeroes ($S_{DC} = y + 1$), which is a union, (DC 25 (011001) would stimulate DC 57 (111001), DC 29 (011101), and DC 27 (011011). The reason for that is that the more abstract DC would contain the activated DC as well as others. The activated DC would also inhibit the DCs connected to it through ones ($S_{DC} = y - 1$), which is a meet of simpler patterns (DC 25 would inhibit DC 9 (001001), DC 17 (010001), and DC 24 (011000)).

In order not to lose information about the original pattern, each activated DC would memorize APN in a matrix P(APN, DCN), with a maximum P(127, 127). The first 127 represent the possible APNs, and the second represents possible positions at which this DC was located in a patch (for example, if DC 25 occurred at binary place 1, 4 and 64(center) in a given patch DCN would be 69).

NODES

As this still leaves the patterns dispersed across MU complex and does not give them a context or unify them, further joining into hash like node boxes is achieved through Inner nodes (IN) and Outer nodes (ON) that are activated by DCs as shown in Figure 7.

Each DC has a 'bottom' and a 'top' side. For example, if a DC were pictured as a coin, one would call the topside head and the bottom side tail. An inside node (IN) can be considered as an octahedron formed from the DC heads facing it, and an outside node (ON) as an octahedron formed from the tails side facing it.

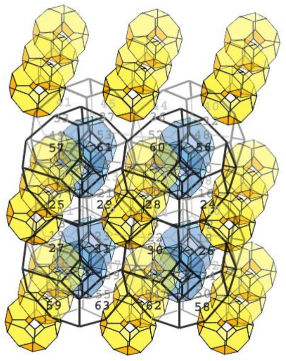


Figure 7

The network formed from INs can be considered a hole in a Swiss cheese, and the network formed by ONs as a cheese itself. The INs would create forms, and ONs the surround in which those forms were created. In order not to limit the number of memories that this system can hold, a double system of INs and ONs interact separately with their neighboring INs or ONs in a handshake manner (querying the adjacent IN/ON if it is active, and if so activating that binary place in its number structure), memorizing patterns of interactions and creating networks that would represent each possible input pattern.

One MU contains 8 IN nodes (labeled from 000 to 111 following the xyz 3D scheme), and each node is activated by 8 DCs. Each group of DCs activating a given node has a maximum binary distance of 2 as well as three identical binary positions in all 8 DCs of that group. It is an equivalent of a hash code for some missing and some present binaries (for example IN 001 is activated by 8, 9, 10,11, 24, 25, 26 and 27, and in all of them binary 4 and 32 are zero, and binary 8 is 1). Table 2 shows all 8 INs with same binary positions in all the DCs that activate them.

IN	Binary 32	Binary 8	Binary 4
000	0	0	0
001	0	1	0
010	1	0	0
011	1	1	0
100	0	0	1
101	0	1	1
110	1	0	1
111	1	1	1

Table 2

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.69 e-ISSN: 2251-7545

The MU structure also forms 8 ON nodes (also labeled from 000 to 111 following the xyz 3D scheme), and each node is activated by a different set of 8 DCs. The DCs activating each ON node are different from the INs, and also exhibit the presence and absence of certain binaries, as seen in Table 3.

ON	Binary 16	Binary 2	Binary 1
000	0	0	0
001	1	0	0
010	0	1	0
011	1	1	0
100	0	0	1
101	1	0	1
110	0	1	1
111	1	1	1

Table 3

These groupings create passive analysis of input patterns, and represent 'archetypes' of the input that also allows recognition of variations within limits. For example; IN 011 means that binary position 8 and 32 are 'present', and binary position 4 is 'absent', which points to 8 possible DCs being activated (40, 41, 42, 43, 56, 57, 58, 59). If ON 101 is also activated, that means that binary position 1 and 16 are 'present' and binary position 2 is 'absent'. Presence of both IN 011 and ON 101 limits the basic pattern to 57 (111001). Therefore, if the input came from another module as a pattern of INs and ONs, the recall of the original input is guaranteed.

Each node has binaries associated with each one of its faces, which is connected to specific DCs. This permits each node to calculate its DC activation number (DCA). For example, if IN 011 gets stimulated by DC 43, 56 and 59 (which correspond to binary places 8, 16 and 128), the DCA for IN 011 would be 152 (see Figure 8).

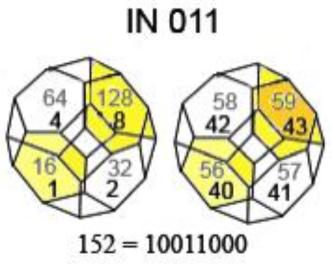


Figure 8

In the above example of IN O11, the corresponding ONs activated by DC 43, 56 and 59 would be ON 001, ON 110 and ON 111 (see Figure 9).

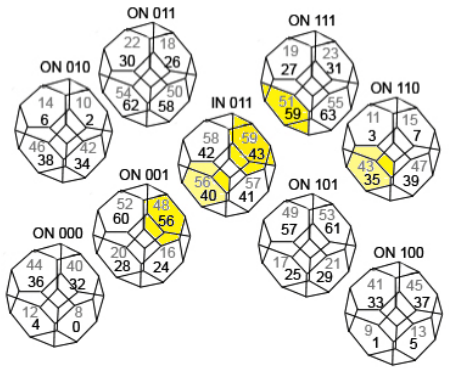


Figure 9

The ON nodes activated by DCs change from 0 to 1 their binary positions connected to DCs. In a given example ONs activated by DCs connected to IN 011 are all activated in ON's binary position 64, therefore the DCA for each activated ON in this case will be 10111111 = 191.

Each node also connects with 6 other nodes through binary numbers (two of each kind in each given direction). For example IN 011 connects to IN 001 through binary 2 and 16, to IN 010 through binary 1 and 8, and to IN 111 through binary4 and 32, as shown in Figure 10.

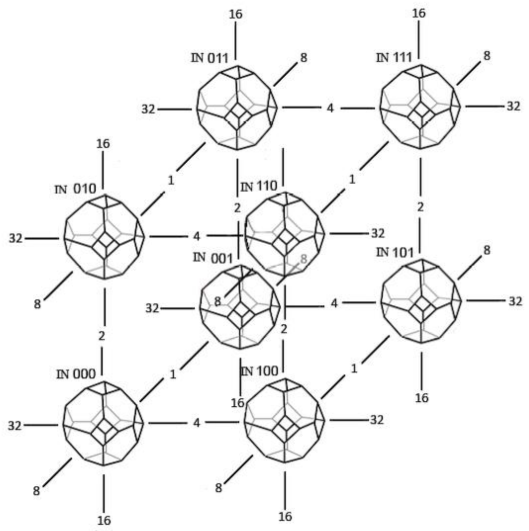


Figure 10

The IN nodes register 'forms', and ON nodes the 'surround' in which those forms were created. Thus the same form can belong to different surrounds. For example, IN 011 has a surround of 8 ON nodes that are activated by the IN 011 DCs. The possible activations of IN 011 by DCs are 256, therefore creating 256 possible surrounds if the ONs are activated only by IN 011. Yet, as other INs can also activate the ONs, that increases the possible surrounds to 256^8 for any given IN node.

When an IN node is activated by DCs, it sends a handshake-query to its 6 surrounding IN nodes. If the surrounding node is active it answers the query and an information link is opened between the two nodes. When the node receives the answers to its handshake query, it establishes a node activation number (NA). In this example IN 011 could have:

$$NA_0 = 0_1 + 1_2 + 0_4 + 0_8 + 0_{16} + 0_{32} = 2.$$

The nodes send their NA to surrounding active nodes, and receive NAs from them. In this example we assume that only IN 001 in binary position 2is active. The NA_0 represents NA of the IN 011 node itself, and NA_2 the received NA from binary position 2. Let us assume $NA_2 = 7$ (that is IN 001 has an active connection with IN 011 in binary position 2, with IN 000 in binary position 1, and with IN 101 in binary position 4).

Connection between two nodes is set in a strength variable W_X (where x represents the binary position link between the two nodes) equal to the sum of the two NAs.

$$W_X = NA_x + NA_y$$

In this example IN 011 $W_2 = 2 + 7 = 9$. W_1, W_4, W_8, $W_{16}$and W_{32} would all be 0 because the surrounding INs are not active.

W_X variable sets the duration during which the information link between the two nodes remains opened, depending on the number of binary 1s present. In this example W_2= 3 clock cycles (000111).

The W_Xs are added together by the node into a total connection number (W_T), which represents the state of activation of surrounding nodes.

$$W_T = \sum W_X$$

In this example IN 011's $W_T = 0_1 + 9_2 + 0_4 + 0_8 + 0_{16} + 0_{32} = 9$. An additional Identity number (ID) is calculated by the node (ID = DCA + NA), which represents total activation of the node by DCs and by its surrounding nodes. In this example ID for IN 011 = 10 (see Figure 11). The binary connection between two nodes exchange their ID numbers, so that in a recall memorized networks and their variations can be reformed. These numbers are memorized in a 3D matrix position M_{IN}(DCA, NA, (W_T, ID)) where most of the matrix will have zero value (sparse coding). In this example M_{IN}(8, 2, (9,10)), would be memorized, representing the state of activation of the node and its surroundings.

The M_{IN} matrix would store information only until the arrival of next I_X inputs, and then would erase it. This is equivalent to short-term memory. If, during the I_X inputs, the same position is triggered again, that position's memory becomes permanent. The same system would be used for M_{ON} matrix. Any repetition of these numbers increases the strength of that memory position by the number of memorized variations in M_{IN}(DCA, NA) position.

In a sequence of events, those matrix positions can record the consecutive states, thus anticipating what the nextstate should be.

NETWORKS

The eight IN nodes belonging to one MU, stimulated by the input patch in this example, would be activated as follows: the IN 000 would be activated by DC 19, which gives DCA = 128; IN 001 would be activated by DC 27, which gives DCA = 128; IN 010 would not be activated; IN 011 would be activated by DC 43, which gives DCA = 8 (see Figure 8); IN 100 would be activated by DC 22 and 23, giving DCA = 192; IN 101 would be activated by DC 30, giving DCA = 64; IN 110 and IN 111 would not be activated. The IN-nodes-network is shown in Figure 11, along with DCA, NA and W_X values.

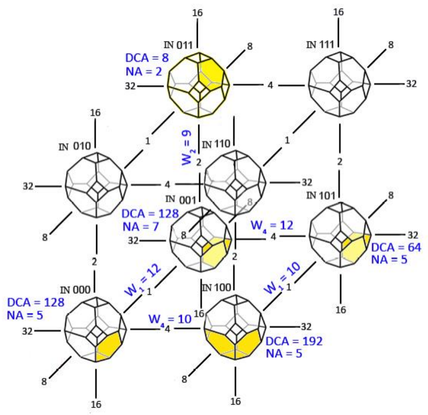


Figure 11

There are 27 ON nodes surrounding the IN-nodes-set of one MU, arranged in three vertical layers of 9 nodes (Back, Middle and Front layers). A Back ON node can be in Top, Middle or Bottom row, and on Left, Middle or Right position in the row. Therefore, in order to specify each ON node a subscript initials will be used (layer, row, position). For example ON 011_{BTL} indicates an ON 011 node found in Back layer, Top row, and Left position (Figure 12).

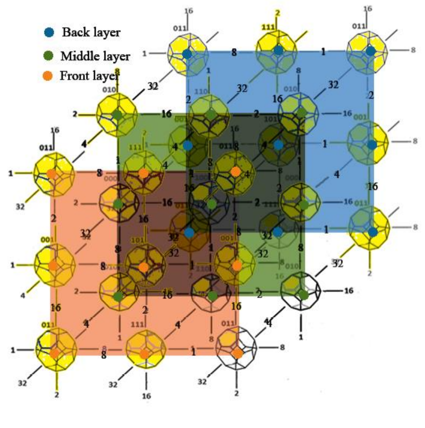


Figure 12

In order to 'tie' all the active ONs together and create a 'container' enclosing the activated INs, the ONs inhibited directly by DCs change their binary activation 63 (111111) to 0 (000000), and then send a query to all surrounding ONs through its binary 0s. The connections between ON nodes through binary 1s create a unified background around IN network.

If the ON receiving a query in a given binary position has 1, it changes it to 0 and forwards the query through the rest of binary 1 positions. If the ON receives a query in a binary position 0 it establishes a W_X connection with the querying ON, and stops query propagation. This indicates that the loop has been closed, or that different but separated parts of IN network have been joined (see Figure 13).

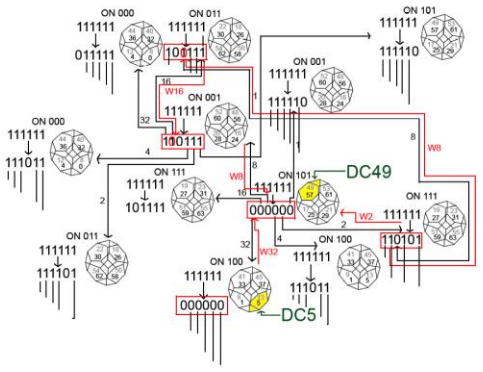


Figure 13

When a node receives and sends information from the same binary direction <u>in the same time period</u>, it forms a W_X connection with that node. For example, ON 111_{BBM} sends a query to ON 011_{BBR}, through a binary 1 position, at time t2, and ON 011_{BBR} sends the query to ON 111_{BBM} at the same time. As they both have 0 in that position, the W_1 is set. This way a 'container' is created, a container that can accommodate variations of IN activation.

The spread of activation in this example is done in several time units. At time 1 (t1) the ONs that are inhibited by DCs change their binary 63 to 0. At time 2 (t2) the 'inhibited' ONs send a query through their binary zeroes to surrounding ONs. ONs activated at t2 change query receiving binary position to 0 and forward a query to ONs through remaining binary 1s at time 3 (t3), and so on.

ONs whose 2 or more binary positions are changed to zero <u>in the same time period</u> stop the propagation of the query, and create W_X with querying ONs. The 2 zeroes threshold was chosen because it means that input to the node came from two different directions, therefore that node would join the querying ONs by establishing a W_X connection with them, making a path from one activated node to another. White octagons (Figure 14) are octagons inhibited through DCs and the red lines represent W_X between partially inhibited ONs.Each inhibited ON would memorize a matrix M_{ON}(DCA, NA, (W_T, ID)) in the same way as IN nodes.

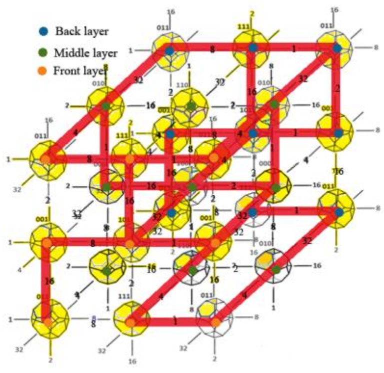
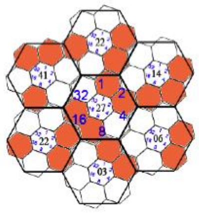
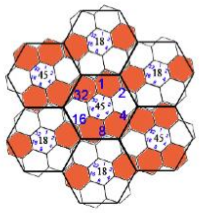


Figure 14

In this example the calculations for 'container' stopped at 27 ONs surrounding the eight INs of the MU stimulated by the example patch. Actually the connections of ONs go beyond one MU considered here, and extend throughout the MU-complex, joining through a shortest path 'container' network of the whole input field.

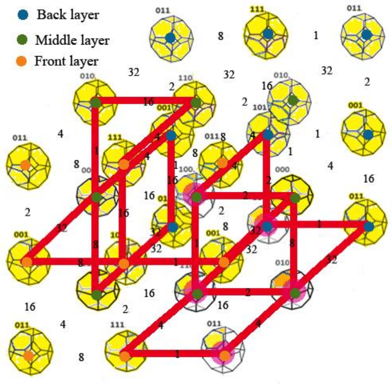
LEARNING

What would occur if the next input were slightly different (Figure 15) or very different (Figure 16)?

(6/42 = 14% difference) (16/42 = 38% difference)
Figure 15 Figure 16

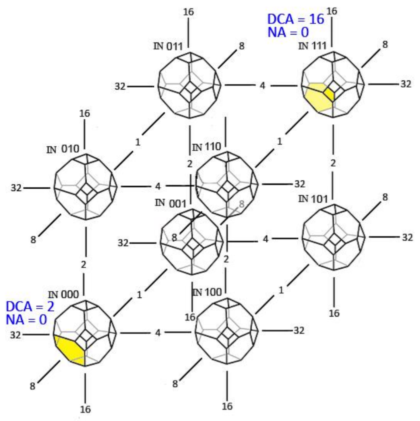
The resulting IN network for a <u>slightly different</u>pattern is identical to the original IN network with 0% difference, but the ON network is different, as shown in Figure 17.



(15/29 W_X, 52% difference)
Figure 17

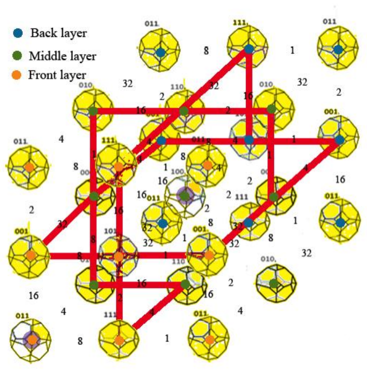
The resulting IN network for <u>very different</u> pattern (Figure 18) is very different from the original IN network.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.69 e-ISSN: 2251-7545



(0/5 W_X, 100% difference)
Figure 18

The ON network is also substantially different from the original as shown in Figure 19.



(20/29 Ws, 69% difference)
Figure 19

Output

Each module formed by input layer, hidden layer(s) and MU complex can compare subsequent inputs with the stored memory, without anything beyond itself. With further development, this system is ideal for learning 'meanings' of any given pattern.

This module can translate IN and ON networks into a new input layer for another module. As each IN or ON node has 6 neighbors, each node can be represented as a hexagon. As nodes have specific positions in MU complex, the hexagons can be placed in specific patch positions in the new node input/output layer (NIOL), where IN nodes are surrounded by 6/8 ON nodes for each IN node, and the missing nodes can be reconstructed from the given information. The inputs from NIOL focus the search, and can compare it to the simultaneous input from the input layer of the same module. With slightly adjusted software in different modules, this complex system of modules would allow for analysis of information processing, i.e. what a particular pattern at a particular level 'represents'.

Conclusion

How would this system explain the visual experience mentioned in the introduction? A given input would create a decentralized-'pixilated'-memory stored in IN and ON node networks, DCs and PCs. With learning from inputs of similar type, 'archetypal' IN and ON networks would be memorized that would expect 'something' in a given information slot.

A given input could stimulate many archetypal networks simultaneously, which would then compete, and only some 'archetypes' would stay active. They would fixate parts that fit together, and query for further fitting parts in expected positions. If those spaces did find valid choices from local memory, they would stabilize the whole IN/ON network complex, almost like chaotic attractors, thus 'recognizing' the meaning of that pattern.

As many patterns can be stored, and as asynchronous functioning is acceptable, it can form new unlearned patterns from many previously learned patterns when simultaneously activated by an input. That leads to a machine equivalent of "insight", and opens the door to an autonomous learning machine. "Self-reflection" which is necessary for formation of "meaning" (ability to answer an internal query about a given pattern by associating it with existing network memories) is inherent in this system.With extension of this system through other modules, ideas could be represented by patterns of activation, and a 'common sense' machine could be made.During development of this system new valuable insights could be made to the way brain networks work.

I am aware that many attempts have been made in AI to develop such a system (with limited success), but none of them (to my knowledge) were based on networks, distributed memory and distributed decision-making that was described in this article. I hope that some of the readers of this article will be willing to join me in my effort to fully develop this system. If you are interested, please contact me at the email provided for this article.

References

[1] Ben Goertzel, "The Hidden Pattern," ISBN 1-58112-989-0

[2] Douglas Hofstadter, "Fluid Concepts and Creative Analogies, computer Models of the Fundamental Mechanisms of Thought," Basic Books 1995, ISBN 0-465-02475-0

[3] Douglas Hofstadter, "Gödel, Escher, Bach: An Eternal Golden Braid," Penguin Books 1980

[4] "Proposal for Parallel Computer Architecture of a cellular type aimed at development of an autonomous learning machine," UKSim2012

[5] Pentti Kanerva, "Sparse Distributed Memory," MIT Press 1988

[6] Lee Middleton, Jayanthi Sivaswamy, "Hexagonal Image Processing, A Practical Approach," Springer 2005

[7] Dominic Widdows, "Geometry and Meaning," CSLI Publications 2004

[8] Stephen Wolfram, "A New Kind of Science," ISBN 1-57955-008-08