Documentation of CrX architecture

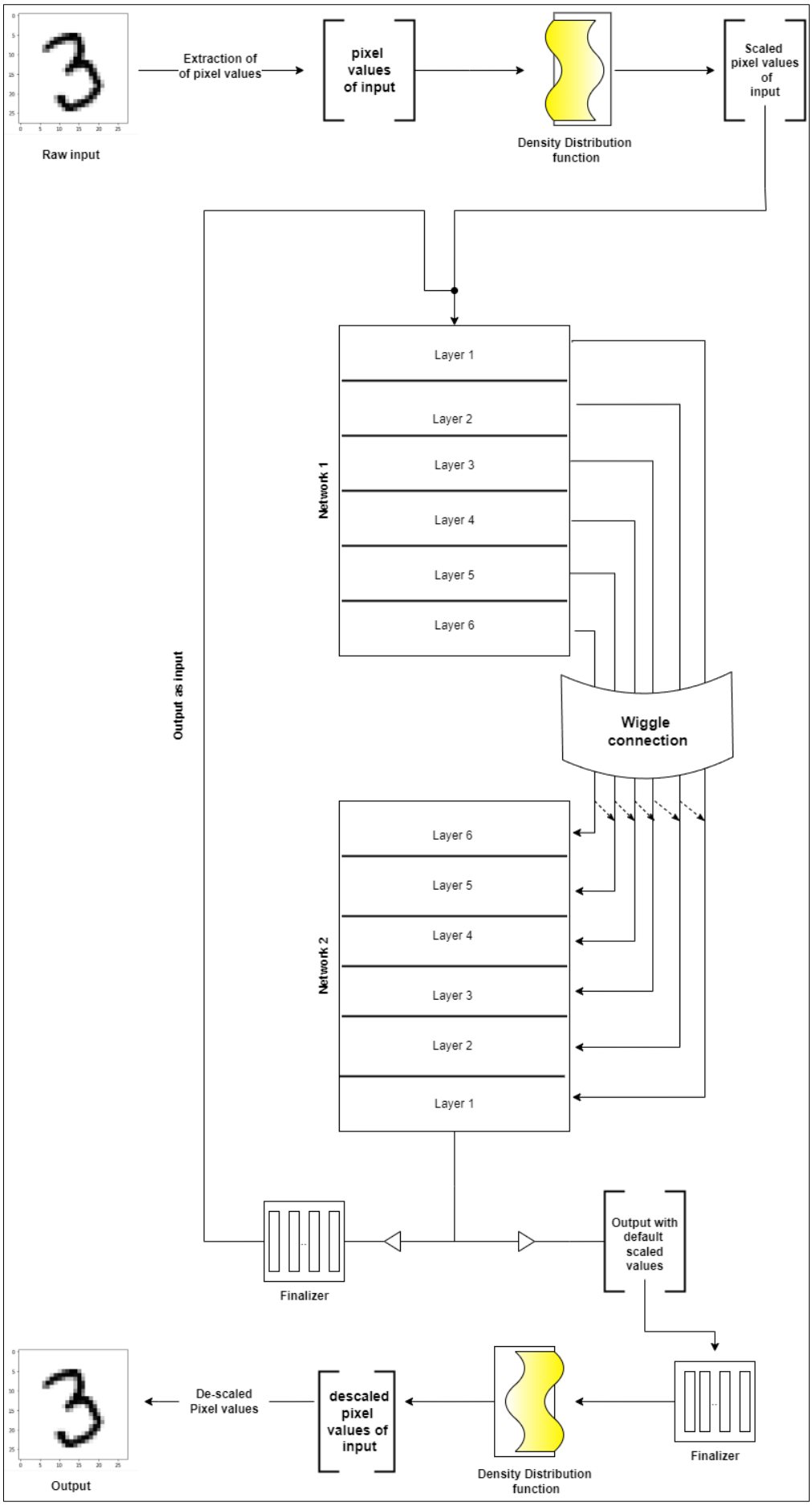
This Documentation talks about the CrX – Cognitive Revolution X architecture with a detailed step-by-step explanation, which was developed from BrainX theory. This architecture will initially function as a general problem solver/AGI.

- N – network
- P – partition gives specification
- C – cluster gives direction / dynamics
- R – returning of output

- BrainX theory into CrX architecture

The brain processes input and allows it to travel throughout its pathways, regardless of where it originated (it does not discriminate). Neurons serve as the paths for impulses to travel, with these paths branching in various directions, thereby altering the direction of the impulse at any given time. The impulse is not consciously choosing its direction; rather, it changes direction due to the laws of physics and biological assistance. Several factors determine the direction of the impulse, and a regularly attended pathway has a higher probability of impulse transmission (thus, a higher likelihood of the impulse traveling this route). When impulses from the sides converge into the main impulse, it alters the direction of the main impulse. Of course, there is considerable noise in the brain – a challenge addressed by the concept that a large sum of correctly functioning impulses is sufficient for output (60% of correct impulses offset 40% of noise). The neuron's diverse paths and their changing direction abilities are equivalent to clusters that alter directions. Unlike the brain, the direction-changing in this architecture depends on factors like the presence of clusters around the chained neurons. Even though the brain does not discern the type of impulse it receives, it processes impulses because they change the physiology of neurons. It is the input and output regions in the brain that give representation to the impulse. Following the same principle, denser regions in the input region prioritize receiving impulses as they receive more impulses than non-denser regions (more impulses sustain the propagation of impulses in the brain, aiding in noise reduction). This is akin to the encoder and decoder having a probability density distribution function that assists in the destabilizing mechanism – a mechanism aimed at reducing noise in the model. Finally, the wiggle connection, where outputs are merged to yield a combined final output, is equivalent to the brain's formation of shortcuts between input and output, thereby bypassing the computational part. The network replicates the property of neurons clumping together into layers. Partitioning and clustering replicate the plasticity of neuron properties.

- Complete picture of the architecture


Raw input
Extraction of of pixel values
pixel values of input
Density Distribution function
Scaled pixel values of input
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Network 1
Output as input
Wiggle connection
Layer 6
Layer 5
Layer 4
Layer 3
Layer 2
Layer 1
Network 2
Finalizer
Output with default scaled values
Finalizer
Density Distribution function
descaled pixel values of input
De-scaled Pixel values
Output

The network

- What is network

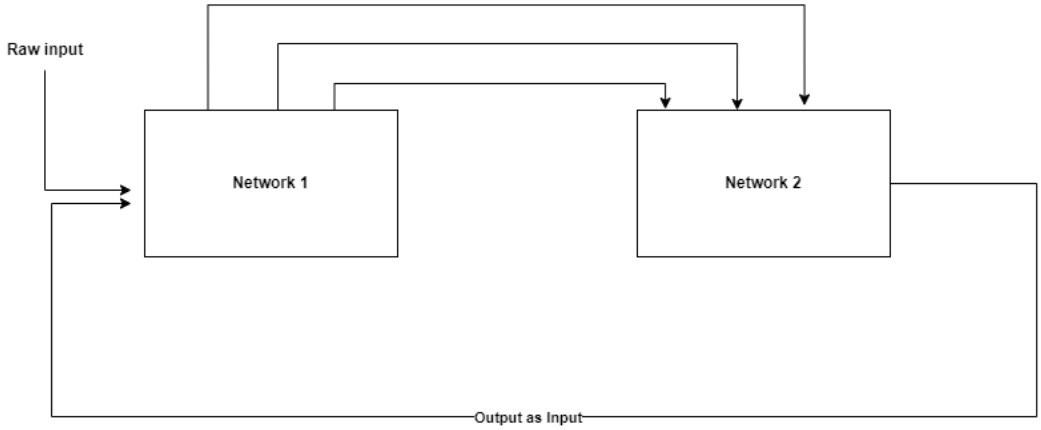
Networks are the millions of layers that are combined together. Initially, we are going to start with only two networks that contains 10 layers each (as a prototype). If it works well, then we can increase the number of networks. Increasing the number of networks will help in storing more different information, which will bring more understanding of the incoming information to the model. However, this will reduce the information reduction capacity (information will be reduced to its smallest truth due to over-consumption – same like brain) of the model, which will increase the correct output rate (other effects are unknown). [Smallest truth – it is reduced bits of information; each model can have its own smallest truth when given the same input. It largely depends on how the model is trained.]
In the layers, there are random neurons that get attracted to the chained neuron if it contains any values (similar to how the impulse of active neurons in the brain attracts other resting neurons via chemical cues). The random neurons are allowed to move randomly, but if they are within the clusters around the chained neuron, then they are fixed onto that position. This fixation of random neurons around the chained neuron helps us to activate the right set of chained neurons for every input. They do not reduce the information (not yet), but they just use the already-in-place information (cluster).

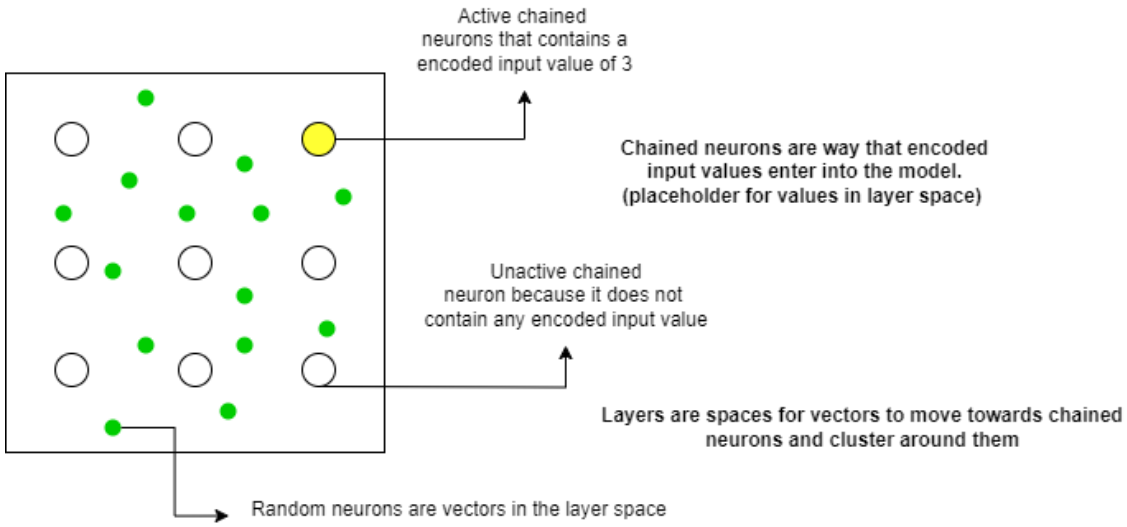
The brain exactly does the same thing by activating the right set of neuron clusters and outputting the reduced form of output. The type of reduced format is dependent on the type of information that the brain regularly stores. This has the capacity to induce curiosity and a problem-solving attitude in the human brain. Since it cannot be fully followed by this prototype, curiosity and giving up on the problem are not possible at the maximum level.

The first network processes inputs, and the second network receives combined clusters. The output is again fed as input to the first network to update the clusters in the first network and to put required clusters within the same layer, which will be in different layers and it will not be able to process input. Because of the wiggle connection, processing input is made possible, and the rate and accuracy are increased. This model should be continuously updated, and with the destabilizing mechanism in action, it can solve problems.

These networks, which consist of chained neurons in their layers, will act as input takers or input placeholders (in other words, the impulse generated by input in the brain is equivalent to chained neurons containing the input values in this model).

The layers contain vectors in their spaces, which are used to form clusters around the chained neurons. The clusters around the chained neuron will act as plasticity in the network. Impulses in the brain are transferred to other neurons only by plasticity (also known as connections). The same principles are applied here. The input values of chained neurons from one layer are passed to another layer only if there are clusters around the chained neuron. In this way, direction for input can be created, which is analogous to what the brain is doing with information, just changing the direction of impulses, and leading to activating a certain set of outputs.





Math behind the network

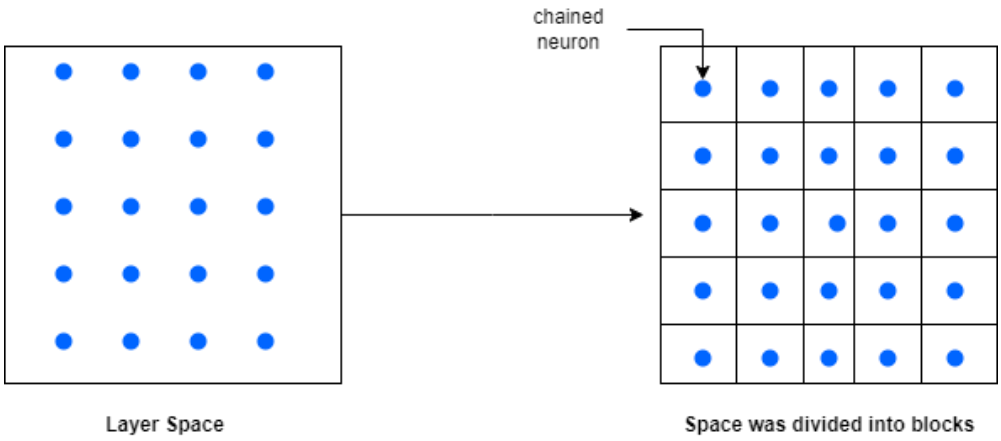
The network layers consist of empty spaces where chained neurons are fixed to each other at fixed intervals. These chained neurons take the scaled pixel values, with each layer containing 100 chained neurons receiving scaled pixel values. The random neurons serve as vectors, and they move randomly within the spaces. In each layer, the number of random neurons can be up to 500.
The input enters the model by chained neurons and the block diving function divides the block and partition of the clusters eliminates noise and clustering fixes the random neurons. These clustered active chained neurons value are transferred to second network respective layers via wiggle connection.

➢ Block dividing function

The layers spaces that consist of random neurons and chained neurons were first divided into equal parts to specify random neurons to specific locations in the layer. This idea of dividing into blocks was to reduce complexity and increase the activation of the right set of neuron clusters (to activate most of the chained neurons, which helps to create pattern), which leads to the right output. Dividing the space into blocks is a way of maintaining the equal dispersion of random neurons. Because of over-concentration, it takes away the ability

to find the smallest truth. This gives specialization to random neurons' activity in their respective blocks.

The brain provides spaces for each cluster to grow, and they won't clump all things together. Neurons are connected to the spaces more than to other neurons. So, to give their own spaces for the random neurons, we have to insert specifications for them to stay and connect to the chained neuron.

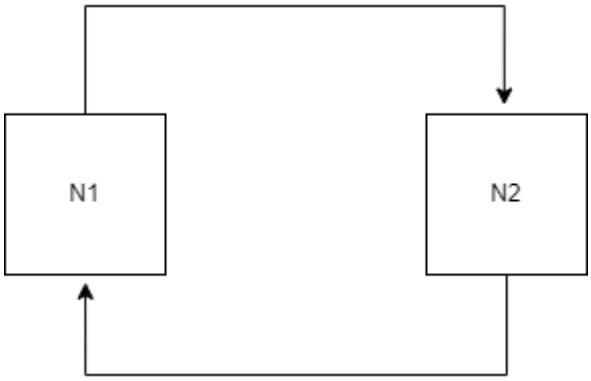


Math behind the block dividing function

This function divides the spaces of the layer to confine or limit the movement of random neurons within the blocks of this function. The spaces are first divided into equal blocks then the scaled pixel values are inserted into each chained neuron. And the chained neurons that contain value are active so they will attract the random neurons towards the chained neuron. Chained neurons that contain value of 0 are unactive and they will not attract the random neurons in the space. and only the random neurons within the blocks space are attracted to that particular blocks chained neuron active.

➢ Loop

This model follows a continuous closed loop to continuously update the clusters in the layers. The first network's layer output goes to the respective layers of the second network to store the required clusters nearby with the help of the wiggle connection. Each output from the second network is tested for similarities within each output. If the outputs are the same more than two times, then the output is returned back as input to the first network.



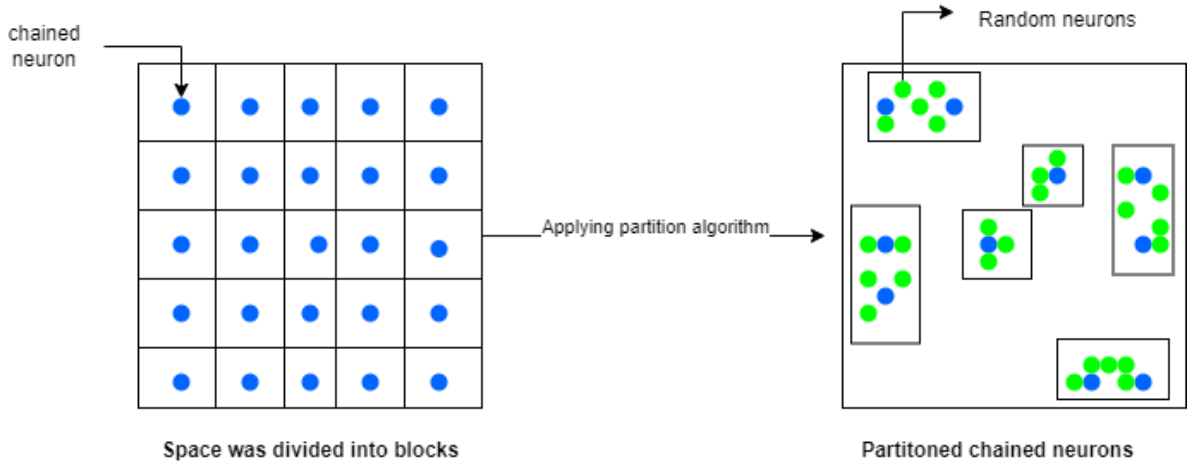
➢ Increase of Network effects

The network was the way to store and mix clusters by increasing their numbers. Both the mixing-up of clusters and storing can be increased. However, complexity will rise and optimization has to be adjusted. For now, going with simple two networks would be good.

➢ Biologically - inspired explanation

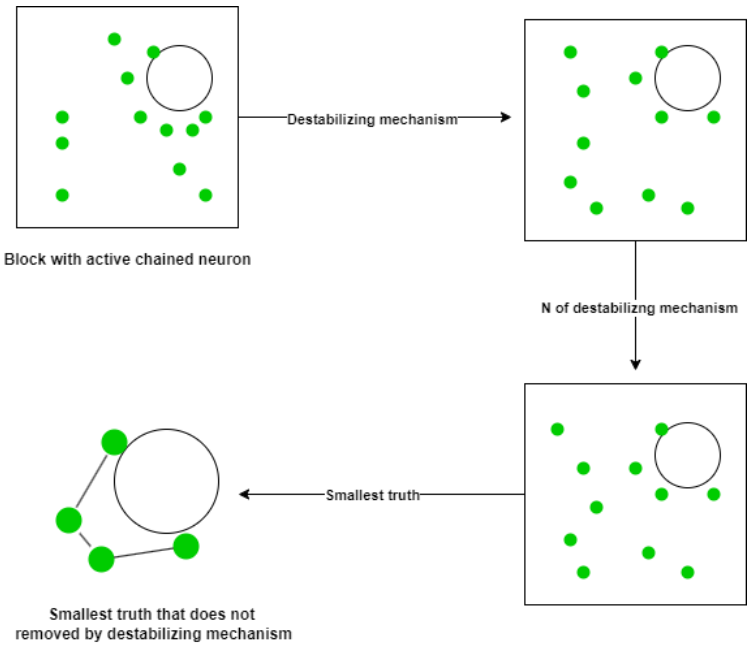
The network layers of this architecture mimic the property of clusters of neurons being condensed into thin layers. Not all properties were mimicked, but only the essential ones necessary for processing information in the brain.

The partition



➢ What is Smallest truth?

Random neurons are vectors in the layer, so after many iterations of pixel input values into the layer and with the work of the destabilizing mechanism, there may be only a few small clusters all over the layer. They do not get reduced but can increase with small increments, but stop after a certain limit, as the input region has an inbuilt predefined intensity limit. So, all these functions work together to extract the smallest truth from the input and help in problem-solving.



Math behind the smallest truth

When the model continuously receives information from the source, the clusters undergo significant changes, but smaller clusters that persist or remain unchanged even after numerous information inflows into the model are referred to as the smallest truth of reality. The attachment of these vectors (random neurons) to these chained neurons always indicates that these chained neurons are consistently active in response to most of the input information from the source.

- What is destabilizing mechanism?

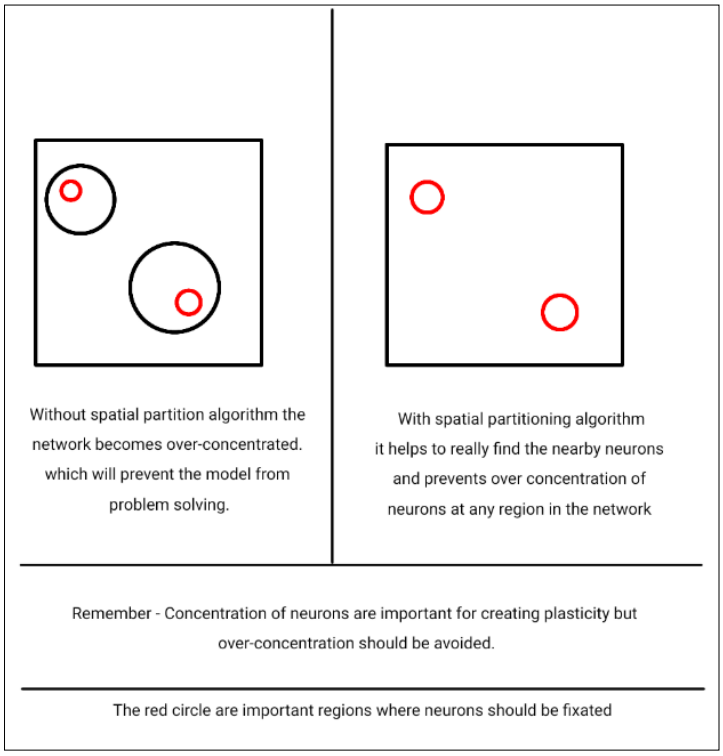
The destabilizing mechanism is equivalent to the activity of neurons not being connected to one neuron always; neurons are connected to locations, not to other neurons. This was the reason for introducing the concept of the destabilizing mechanism to this model.

The destabilizing mechanism makes the model continuous. That is, the input will continuously form and remove clusters, with the exception of true facts or small truths remaining always as clusters, as they are always getting activated.

There should be a preference for some chained neurons, and it should be possible by partitions, as where blocks get narrower, the random neurons should increase in those blocks. Think of each random neuron as an impulse in that chained neuron. As more impulses in that chained neuron, the higher the probability of that chained neuron's values being the output.

- Partition role in destabilizing mechanism

Partition is the first step towards the destabilizing mechanism. It reduces over-concentration by eliminating noise in the layer. This helps to find the smallest truth and eases the process of the destabilizing mechanism.



Math behind the partition

First, the spaces are divided by a block dividing function, and then random neurons are attracted to the active chained neuron. Next, a partition algorithm, like a quadtree, is applied to partition the layer space. The partition algorithm continues partitioning the space until only 4-5 random neurons remain inside each partitioned space. Then, this partitioned space containing 4-5 random neurons is subjected to a clustering algorithm.

- Partition role in finding the smallest truth

By filtering the noise and preventing the clustering of unwanted random neurons in the layer, sometimes there will be a loss of information by this partition mechanism. However, if that lost information is one of the smallest truths, then it will be stored eventually by other types of inputs.

- Biologically - inspired explanation

The assumption is that neurons are not limited to their connections in the brain but rather to spaces (this also contributes to the problem-solving abilities of the brain, by mixing similar information – connecting randomly with closer neurons). The block dividing function and partition algorithm were introduced to replicate this property of neurons in the brain.

The cluster

It is hoped that clustered neurons represent the smallest truth, while others are just noise. So, by activating the smallest truth, we will always get an output, even if it is the wrong output, similar to the brain. By increasing the smallest truth, we can improve the output to more accurately manifest reality. And by allowing only the smallest truth/cluster to activate and move forward in the layer, we can obtain the right set of outputs.

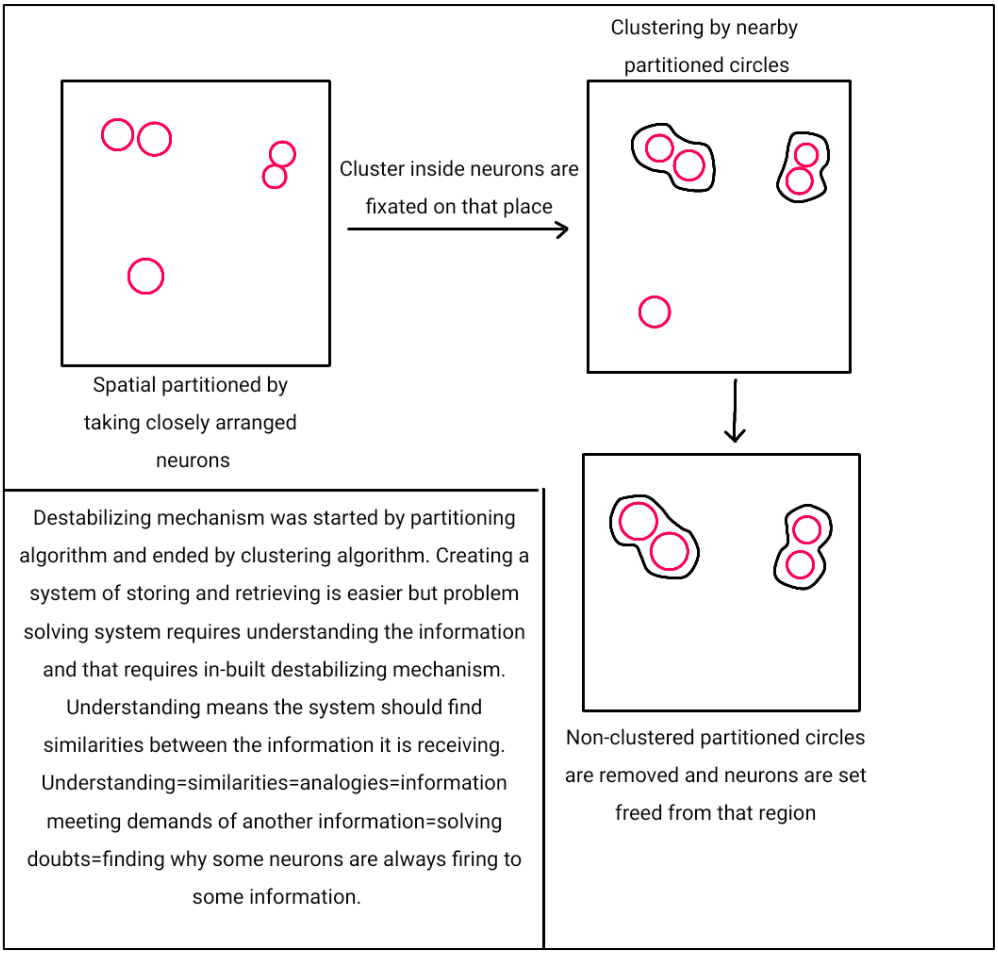
Math behind clustering algorithm

The partitioned blocks serve as input for clustering. Clustering occurs for random neurons within any partitioned blocks that are close to each other. By clustering the closely arranged random neurons, the algorithm fixes them into that position, while random neurons that do not meet this criterion are set to normal again (they are not fixed). Only the values (scaled pixel values) of active chained neurons that are surrounded by clustered random neurons are allowed to move to the next layers in the network. pixel values in the chained neurons that do not contain clustered random neurons are not moved to the next layers in the network.

- Cluster's role in destabilizing mechanism

Clustering is the final stage in the destabilizing mechanism. After clustering, except for the clustered neurons, every other neuron is set to normal. The only changes are made to the clustered neurons, which are fixed onto their positions. The clustered random neurons are not fixed permanently; instead, they follow a flexible fixing in that position. If after many iterations of different kinds of input into the model, the already clustered random neurons are not clustered again, then those random neurons in that cluster are dispersed into that space.

The clusters resemble the plasticity in our brain. If impulses do not activate a connection between two neurons for a long time, then any nearby impulse removes the connection (as without impulses, the connection weakens), and the neurons with the removed connection now connect to any nearby impulse neurons. Impulses are the food for the neuron; they are naturally attracted to it.



➢ Partition functions assistance in clustering

They filter the important clusters by partitioning only them. What is an important cluster? The cluster that is surrounding or nearby, close to the chained neuron, is an important random neuron. As they are directly attached to the chained neuron, these chained neuron activations, containing pixel values, represent a direct response to the pixel values of the input.

So, as the random neurons attach themselves directly to the chained neuron, they represent the smallest truth. The partitioning algorithm partitions only those closely arranged random neurons. By this, they are further clustered by an algorithm, and then the clustered random neurons direct the impulse (pixel values movement across layers) direction between layers.

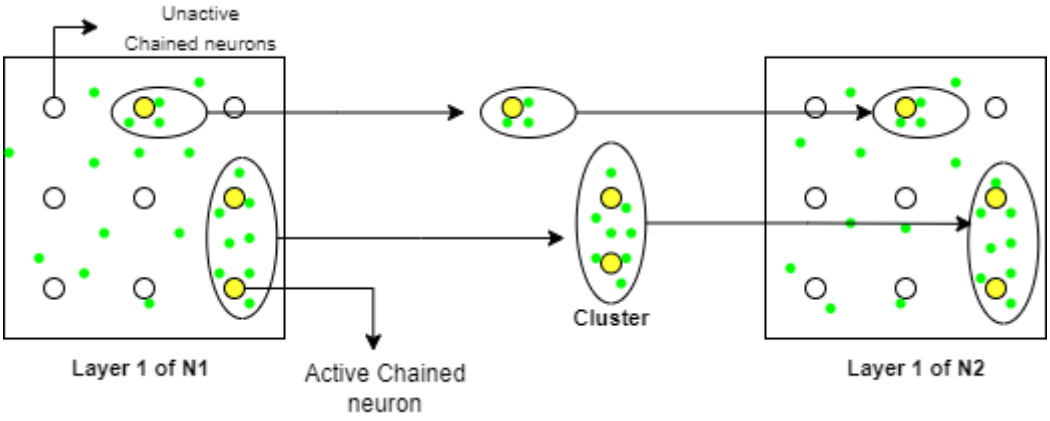
➢ What happens after clustering?

The clustering algorithm only clusters the random neurons that happen to be closer around the chained neurons. Once clustered, they are fixed onto that position, which is required to utilize the direction-changing abilities of the cluster. Only the chained neuron values that are surrounded by clustered random neurons are allowed to pass into the next layer within the same network. However, both clustered random neurons and chained neurons' pixel values are allowed to pass into the respective layers of the second network.

The random neurons that are not clustered are set to normal. These random neurons are again normally attracted to other chained neurons. Unclustered random neurons are not fixed onto that position.

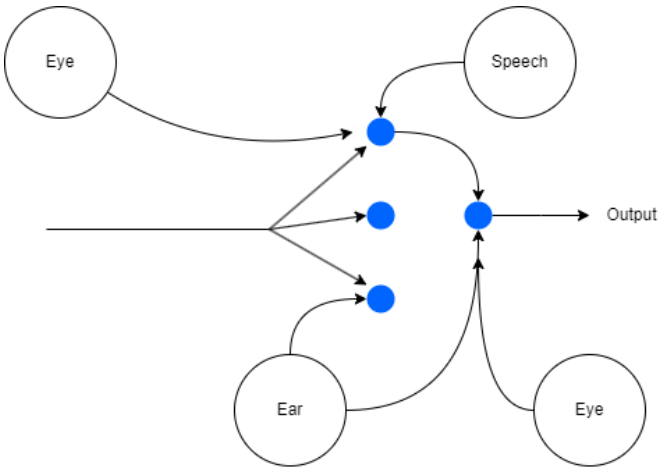
➢ After clustering, Layers output to next layers and next network layers –

After clustering the nearby random neurons, the clustered random neurons around the chained neuron become activated neurons. Only the activated chained neurons' pixel input values are transferred to the next layers of the same network. However, the transfer of output between layers of different networks will involve clustered random neurons and the pixel input values.



Impulse Direction changing abilities of cluster (important mechanism, no error should be made in this mechanism) –
(to better understand this concept, read Linearity of Thoughts from BrainX)

To enhance understanding, we will refer to the pixel input values in the chained neuron as impulses. If there are clusters around the chained neurons, they are active because the random neurons that cluster perform the role of impulses present in the brain. These clusters can change, create, or destroy the direction for impulse traveling in the network layers. These changing abilities map the correct set of clusters to the inputs. And these changing abilities are performed only when clusters are nearby.

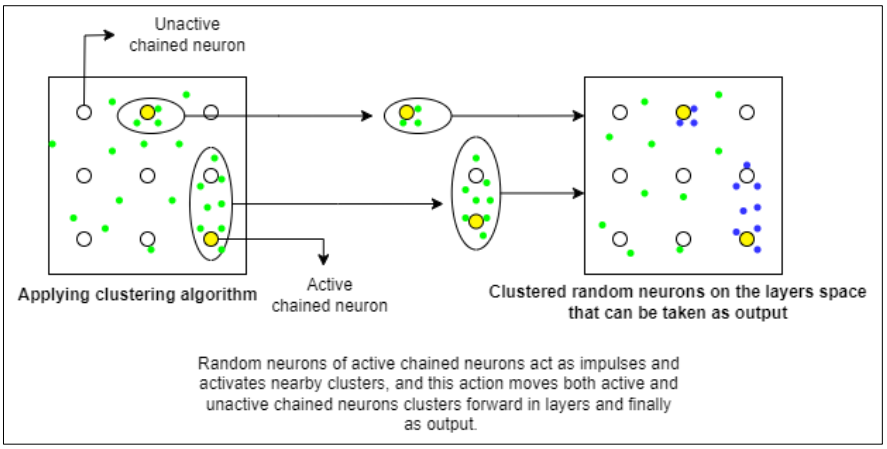


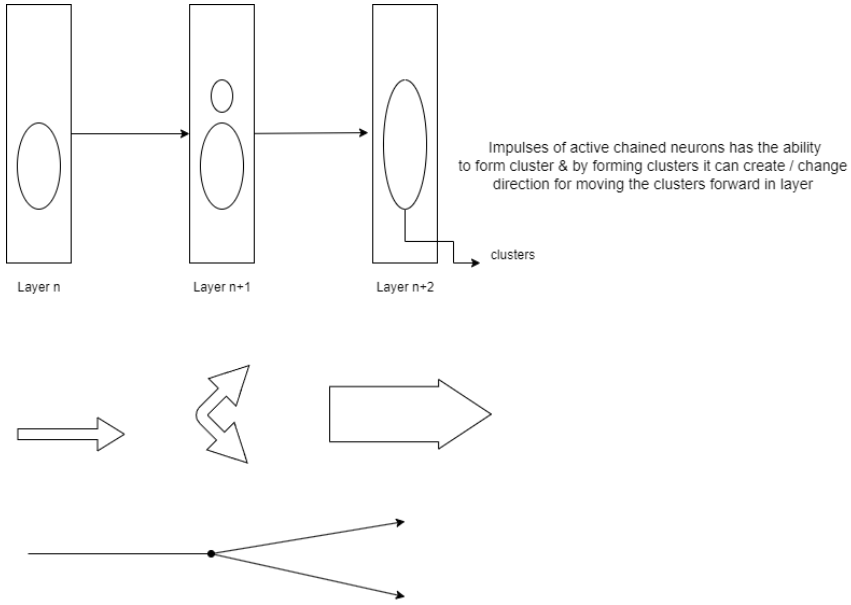
In our model/brain, the senses are what help to perform impulse manipulation. That is, each sensor collectively changes the direction of the impulse with the help of clustered chained neurons. Chained neuron paths are like multiple river paths, and each sensor's impulse is the incoming water force that changes the direction of the overall water flow. Direction will change only if there are potential nearby clusters to easily pass the impulses from multiple sources. If far-away, not every impulse will get to that neuron's pathway.

These nearby clusters hold significance because the inputs will store required clusters nearby, not far away. This statement is valid because the inputs will attract random neurons to form clusters if the input contains any value. There should be contrast with the background for this to happen, and for the object and background to be nearby, their values scaling by the distribution density function will have significant values nearby, not far away.

For example, if a tea cup sitting on a table is given as input, the significant values are vertical and horizontal continuous lines. This distinguishable continuity will set the distinct but important values to be nearby in positions inside the models layers. Therefore, the reason input will form clusters nearby is due to continuity in the input. If there is no continuity, then the input will not follow the laws of physics. The input will follow the order of nature, which contains continuity on both small and large scales. Whatever the input, there is definitely continuity in every input.

There is a another logic to be added into this mechanism, currently this function connects with nearby clusters that have high density. If this was the case, then everytime we have higher density clusters to get activated. No, we want the right cluster to get activated. So for that we have to include a connection or link after the source cluster finds any high density nearby cluster for a first round in any layer. This link will act as indicator to avoid the connected high-density cluster to get activated again, this second time. The second high density cluster will get activated. And also, we will include a refractory period type logic in this link, that is we set a weight on these links like (some number) and insert a decay rate (linear or exponential). In this way we can make the all clusters to get activated not just always the high-density clusters.
even the brain does this like – first activate a large cluster (question type? Why questions are related to primary emotions most of the time then the answers, so they will be of large clusters) this connection goes refractory period then nearby second large cluster gets activated (mostly an answer). At some time, we will get to the answer straight then in that case the intermediate impulse of questions is not got into input neurons, the answer cluster was so nearby it preceded and activated the answer cluster to give output. More like 5*5 = [4*5 (20) + 5] = 25 this intermediate step will not get into input as soon the output will beat the intermediate step impulse and gets into the frontline input neurons (or just output). Thus, making intermediate not happened.





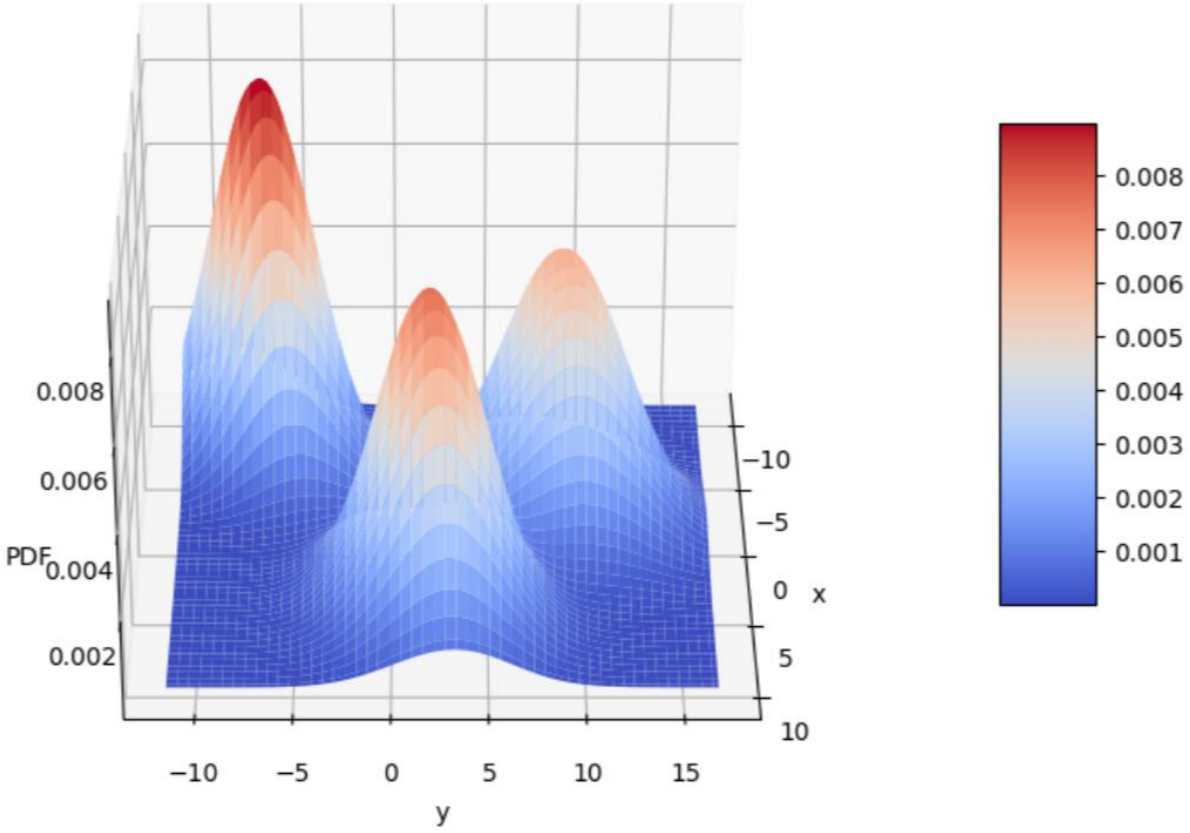
Math behind direction changing abilities of clusters

By allowing only the values of chained neurons that are surrounded by clustered random neurons to pass into the next layers in the same network, the ability to change direction can be achieved.

- Probability density function

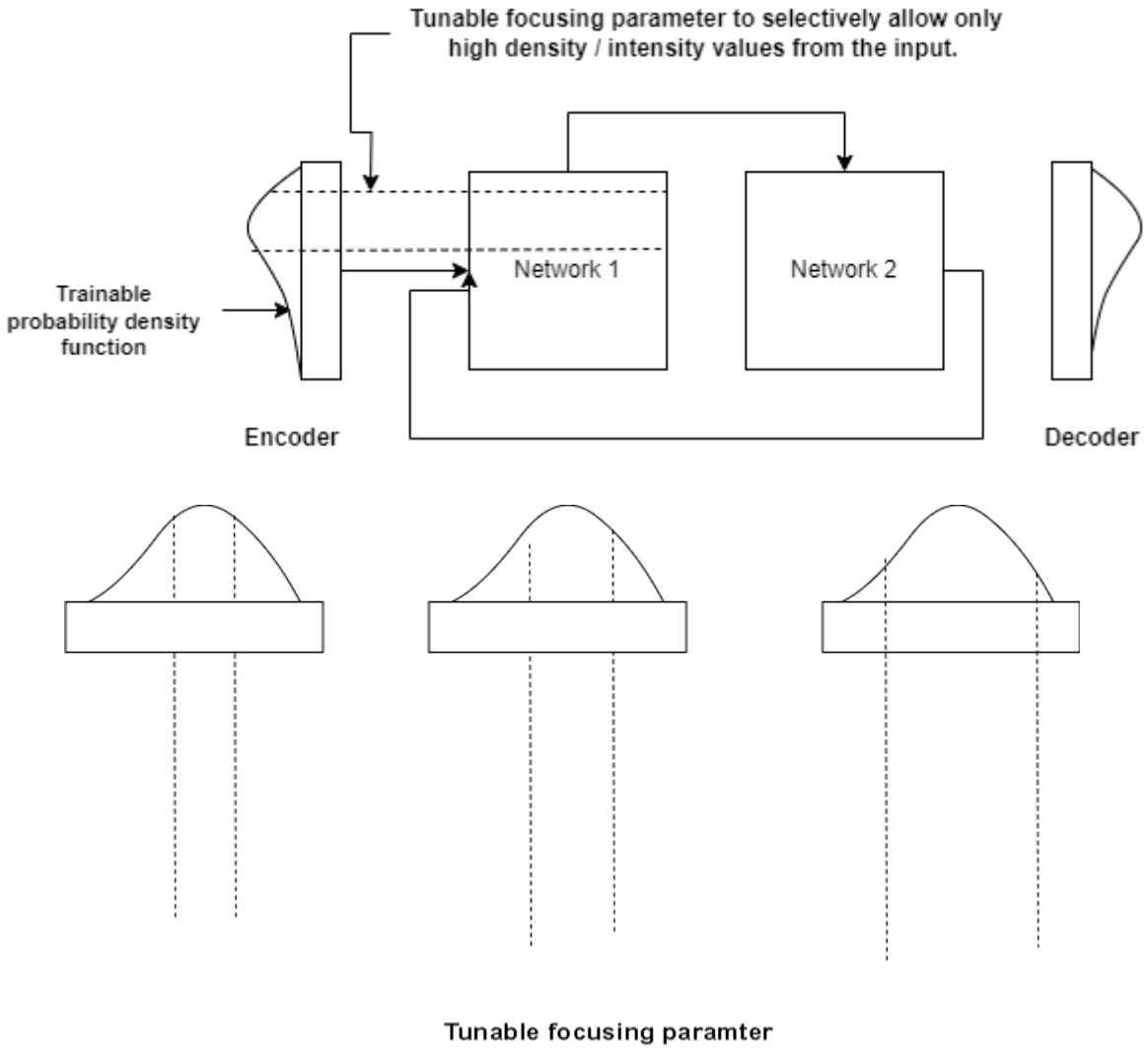
It was located after the encoder and before the decoder; the trainable PDF that was trained in the encoder was transferred to the decoder as trained to make the output correct. It was to mimic the neurons at the frontline of each sense. These neurons get crowded at the place where higher impulse was entering into the senses. So, importance was given to these impulses that enter via these crowded regions, as these impulses have more energy to fully propagate through the cycle. Higher valued regions get scaled by the arbitrarily trained value to give the values from that region more importance. The ability to change slows as more time passes, but it is not completely lost; it changes slower, similar to the neurons in the brain. (Also, drastic change is not needed as reality will not change drastically). Multiple PDF could also be used to get accurate output if demanded by the model. As more PDF will

collectively give more importance to particular set of inputs/outputs. Also, crowdness of neurons in the frontline input neuron will helps in increasing the intensity of the impulse and thus finally helps in propagation of the impulse. PDF will try to mimic this property, The goal is to learn a function that assigns a "probability" score to each location in this grid, such that locations with important/relevant patterns get higher scores, and locations with less useful patterns get lower scores.



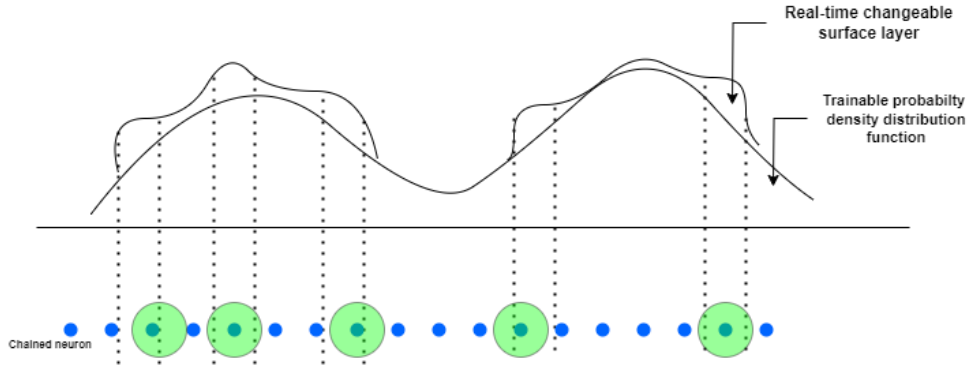
- Tuneable focusing parameter

There should be a function that focuses or scan and give out only values of the important regions (that is higher intensity regions) from the trainable probability density function into the first network layers. It was to eliminate noise from signal in the input. And it was tuneable parameter that can set to value when the input signal was correctly captured.



Tunable focusing paramter

And there should be a real-time changeable surface layer on top of each peaks of the trainable probability density function. It was to refine the focusing parameter to few important chained neurons. Because as more the function gets trained there is a higher probability that equal importance was given to many chained neurons. I think it will reduce the probability of correctly gathering the patterns for the input.



Refining the focusing parameter to activate certain chained neurons to connect with nearby clustered chained neurons

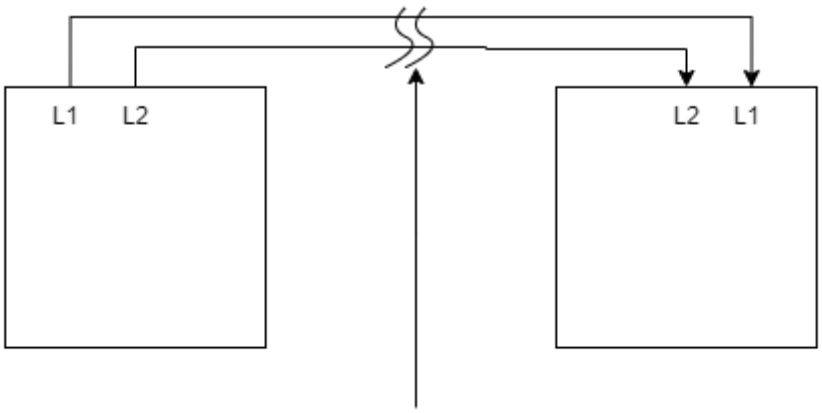
- Biologically - inspired explanation

Neurons in the brain are clustered together by the effect of impulses traveling in any neuron. These impulses promote attraction and make connections with nearby neurons by activating them to release chemicals and synthesizing receptors for accepting the synapse. After clustering nearby neurons together, this path of clustered neurons acts as an important contributor to impulse transmission & manipulation by attracting more impulses. If any input in the environment generates more impulses along these clustered pathways compared to other pathways, then that input is processed in the brain. If any input with more impulse that is not passing along these clustered pathways, then that impulse is less likely to get into output or this impulse converge to the regularly attended clustered pathways. These clusters change the direction of the impulse correctly, activating a certain set of input cells in the input region (refer to the BrainX theory for understanding the above concept).

Therefore, this process of clustering and the clustering pathways abilities of changing direction were incorporated into this model to process input and solve problems.

The Wiggle Connection

This is the method to combine the required clusters that are present in two different layers in the first network. The combined cluster serves as input to the second network layers, and these clusters will get fixed in the respective layers of the second network.



Wiggle Connection - current layer (L2) forms connection with previous layer (L1) and connection means merging of clusters from both those layers into one layer. These merged clusters as one layer's output were transferred to same current layer of next network.

Math behind the wiggle connection

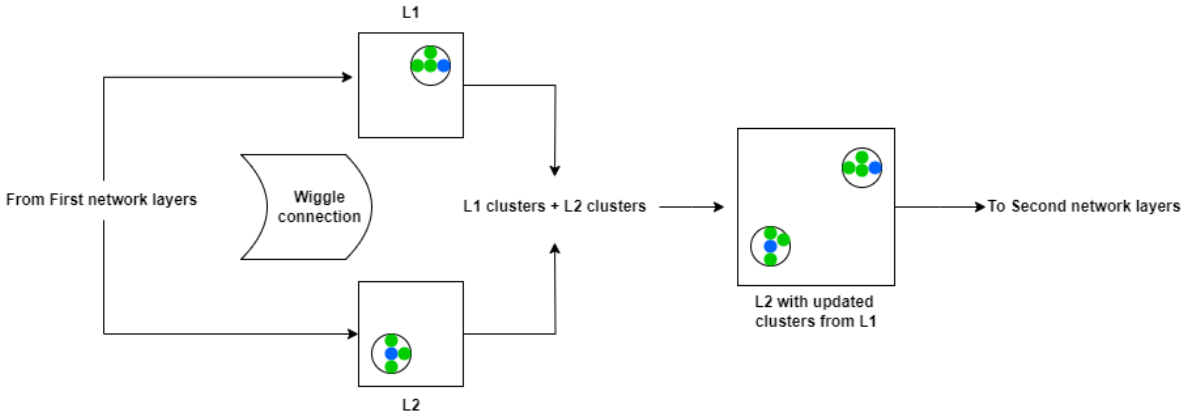
The function of the wiggle connection is to merge the clusters from two layers (the current and previous layer) into a single layer that contains all the clusters from these two layers, along with the pixel values of the clustered active chained neurons only. This newly created layer is then transferred as input to the current layer of the second network. In the second network layers, all the functions and algorithms of the first network layers occur, including block division, partitioning, clustering, and the passing of pixel values of clustered active chained neurons.

- Similarities of clusters in layer and in between layer

The layers will cluster only the random neurons that are closer together. This action will be helpful in the direction-changing abilities of the cluster to activate a certain set of clusters at each layer. However, there is a problem: clusters in a single layer will not be enough to solve a problem. Every layer consists of different clusters because there is continuous updating of clusters in both networks. This continuous update assists in problem-solving.

These different clusters in each layer will help to find the right set of clusters in the layer. Therefore, in order to combine these clusters, we need a wiggle connection. This wiggle connection combines the different clusters from two layers and combines them into a single layer in the second network. The output from the second network is then returned as input to the first network. In this way, the first network will store the right set of different clusters that can actually solve the problem, which is done by the mechanism of the wiggle connection.

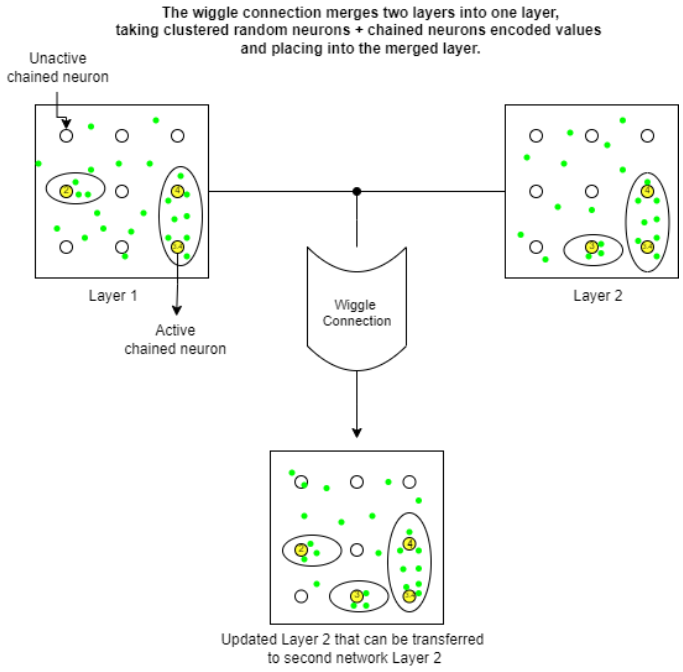
- How it will happen –



The wiggle connection only works for two-layer output. If there are five layers in the first network, each output goes into the wiggle connection. The wiggle connection takes each output from the layers. If the current layer contains any previous layer's output or the current layer is preceded by the output of previous layers, then both the output of the previous layer and the current layer are combined into a single clustered output and placed into the respective layers of the second network. For example, the output of the first layer is allowed to pass through without any modification by the wiggle connection and is transferred as-is to the respective first layer of the second network.

When the output of the second layer is passed through the wiggle connection, it modifies the output by taking the previous first layer's output and merging these two layers' outputs into one output, which is then transferred as output to the respective second layer of the second network. This process continues for all the layers of the first network.

This concept assumes that clusters are different in each layer, and they will be different after many iterations of training. The model is capable of finding the smallest truth from the input, which can be found by a suitable destabilizing mechanism.



- How wiggle connection changes the layers clusters?

There is a final output concept in this architecture. If the output from the second network does not change much, or there is not much difference in the two outputs, then the model should consider that output as the final output. It should then stop the loop of transferring the output between layers of the same network and different networks. Now, the model should halt all operations, such as looping and forming clusters, until the next input is presented to the model.

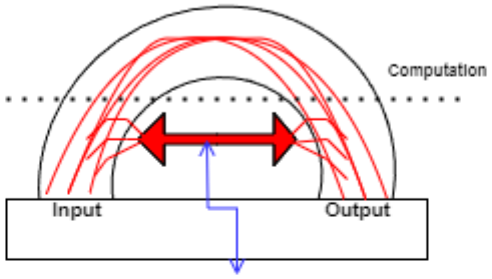
- How wiggle connection increases the correct output generation rate

By combining the saved clusters that represent the smallest truth of reality, the correct generation output rate can be increased. The major assumption is that each layer initially saves the same kind of clusters that represent the smallest truth, as few inputs were given to the model. However, after giving many inputs, the loop running for every input will change the clusters of each layer of both networks. This results in each layer saving different clusters that represent different smallest truths. By utilizing the property of changing direction ability of each cluster in the layers, the model can use both functions and retrieve the right set of clusters at each layer. This is achieved by leveraging the changing direction abilities of clusters and merging these retrieved clusters into one output by the wiggle connection. The output is then transferred into a second network, and the difference in each output is checked to finalize the output. This process increases the probability of generating the right output.

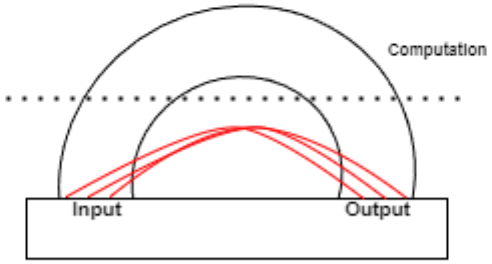
- Biologically - inspired explanation

The brain follows a simple mechanism that efficiently solves problems. Let me state the mechanism: impulses from the external environment enter the brain, and as these impulses cycle inside the brain, inevitable connections form between input impulse and output impulse neuron clusters. This allows the answer to appear quickly to the observer, making shortcuts to the answer by bypassing full computation and directly connecting to the output after undergoing full computation iterations. This process is followed for all inputs consistently. This process is a sure-to-happen mechanism given the neurons biological properties.

Slower processing of impulse. Generally applicable to all information. Connection lost is possible, but it is not that easy, as more connection prevent this action.



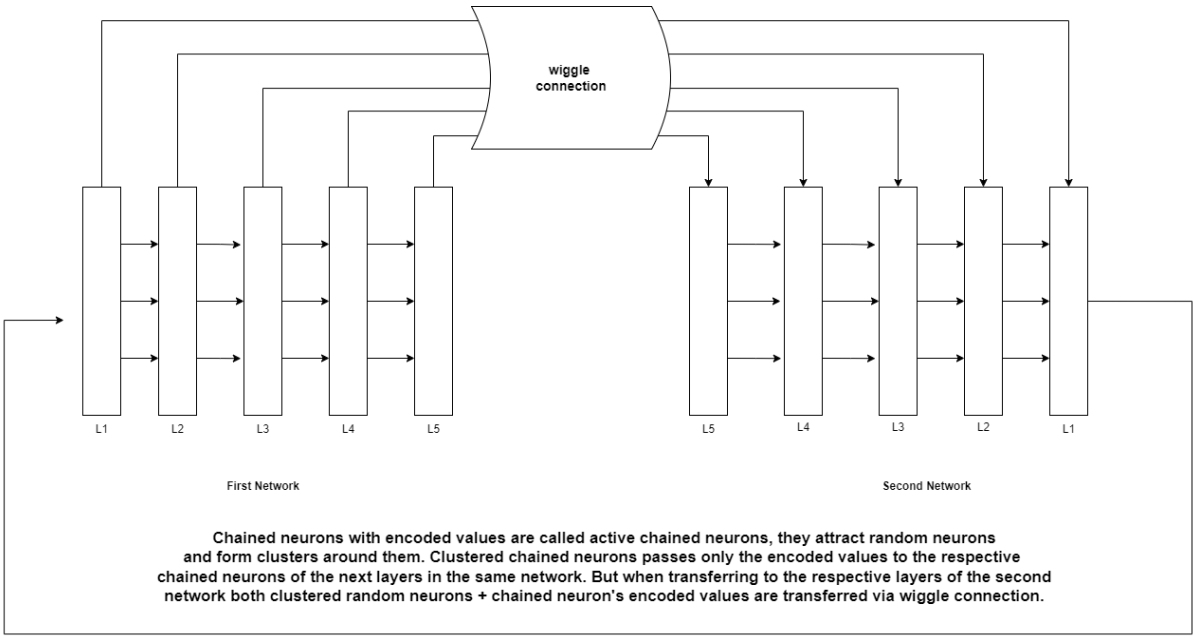
Wiggle connection - Shortcut formation by directly connection input path to output path.Target of connection is depend upon the intensity of impulse and timing of processing the imput.



Faster processing of impulse, applicable only to more familiarized inputs. connection easily lost by other strong impulses interference

The return

When the second network's output was returned as input to the first network, this was done to update the clusters in the first network. The next time when the input comes into the model, the updated clusters will correctly understand the input in a very short time.



Chained neurons with encoded values are called active chained neurons, they attract random neurons and form clusters around them. Clustered chained neurons passes only the encoded values to the respective chained neurons of the next layers in the same network. But when transferring to the respective layers of the second network both clustered random neurons + chained neuron's encoded values are transferred via wiggle connection.

Math behind the return

The model outputs the pixel values. The return of the output as input to the first network is to store the closely related clusters in the same layer of the first network, which would not be possible without returning the output. The returning-of-the-output stage contains the finalizer function where this function takes the output and finds similarities between the output for each output. If it checks with the previous accumulated outputs and if similarities are the same for at least 3 outputs from the second network, then this output is returned to the first network as input.

- Purpose of returning the output as input -

This concept was created to combine two related clusters in two different layers. The destabilizing mechanism will act as an algorithm that increases the smallest truth cluster in each layer. This makes each output from the model truly manifest reality, even if the output is totally irrelevant; the output will still manifest reality (good, but may not be correct). Additionally, the destabilizing mechanism will mix the smallest truths, enhancing the problem-solving abilities of this model.

The reason for returning the output as input to the first network is that the raw pixel input may or may not activate all the required clusters, as required clusters may exist in different layers which cannot be connected simultaneously because cluster connection is possible at one layer at a time. Therefore, there must be a mechanism to take the outputs from two layers and combine them into one (with 60% of the required cluster in one layer and 40% in another layer). A "Required cluster" is a group of clusters that provide output without any error. The "Wiggle Connection" is the mechanism that finds the required clusters in different layers and combines them into one.

The reason for returning output is to reduce noise and increase signal, because the output is a result of reduced-correct-important information (that was specific to the model). If we give this reduced information to the model again, this reduced information will specifically activate only the other correct clusters that are necessary to complete the full output. For example, if I give an input like D, A, E, R, B, then the model with alphabet knowledge outputs A. If this A output is sent back as input, the model now gives B. The outputs are the only correct questions that the model will understand to give a more clear answer. In this way, by many iterations, the complete output (at least what is known to the model) can be achieved. Once again, how the output will be handled as input in the network and their influence in the network.

When the finalized output enters the first network, it is treated as input again, and all processes like partitioning, clustering, and looping occur in both the first and second networks. It is assumed that the raw input will not activate a certain set of clusters present in each different layer. Therefore, this output, treated as input, collects all the necessary inputs that the model can understand in its own language, activating all the right clusters faster. Thus, this process serves as a way to update the clusters in the first network layers.

- How the return will help us reduce wrong output rate and increase right output rate, and how it will differentiate the right and wrong output?

The return will save the combined right cluster from the wiggle connection in the second network layers. The second network gives finalized output that has the right set of combined clusters collected from all the layers of the first network through the wiggle connection. This perfectly altered clustered randomness, as the final output, will be returned as input to the first network. This effect stores the right set of clustered vectors combined through many loops in the first network layers. The next time the same input is given, the first network produces output without many loop iterations. This reduces time and cost. The probability of the right output depends on how well the model stores the smallest truth. However, there is a little contribution from this return function: it eliminates much noise in the model by outputting only the not varying much output. That is, the outputs from the second network layer should not differ much; they should be the same for at least two outputs. This matching of output will eliminate much noise since it was stored in the first network layers. The errors are permanently being removed. That is how this function helps in generating a good output.

- Biologically - inspired explanation

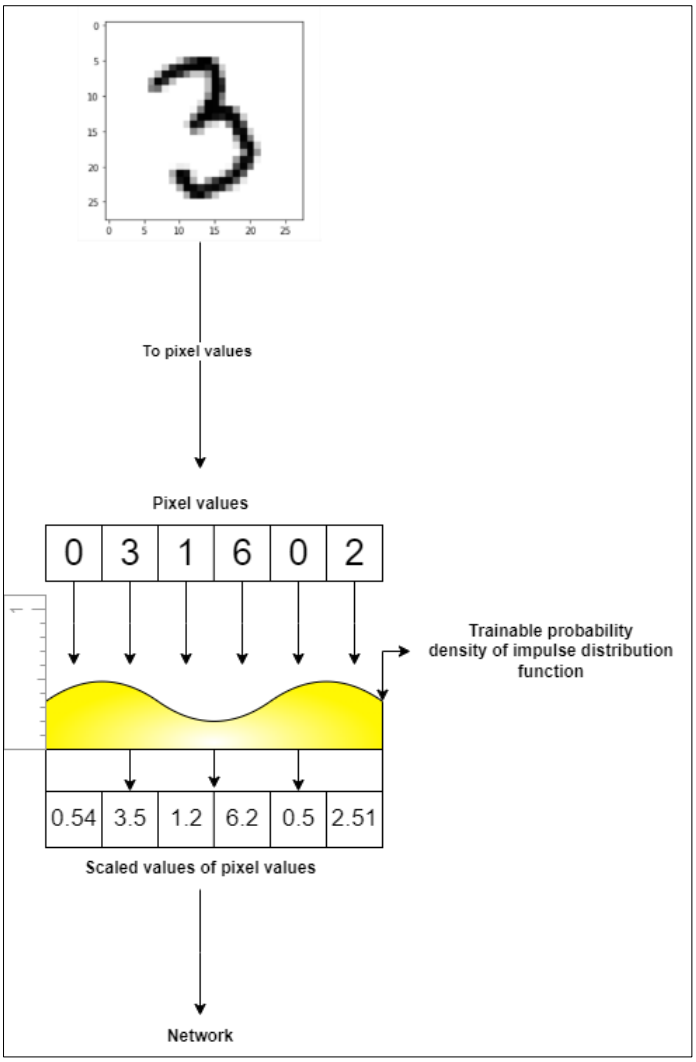
The returning impulse will only be of the highest intensity impulse that entered the brain. Therefore, only sustained impulses get as a returning impulse, equivalent to the model having the same output two or more times. This repetition of the same output two or more times is equivalent to a high-intensity impulse entering the brain. This is important for the brain to solve the problem.

The encoder and decoder

This model needs an encoder and decoder because the model processes input impulses by simply changing direction. The model does not know which or what those impulses represent within the model. The model just receives and delivers what those impulses demand, such as clustering and changing direction. The encoder and decoder understand what each impulse represents, and that's why this model needs an encoder and decoder.

- Encoder

For the prototype, simple images like MNIST will be used. The encoder will consist of a function that extracts pixel values from the raw image. This pixel values will then be fed into a trainable distribution density function, which will scale the resultant pixel values to give importance only to the important locations of the inputs. The distribution will take on a multi-modal distribution after training, where peaks or high-density regions correspond to the important regions in the input pixel values. These high-density regions will scale the feature maps in a way to give them more priority when processed inside the network, while low-density regions will scale the features to give them lower priority during processing. Since the distribution is trainable, it is adaptable to each kind of input category.

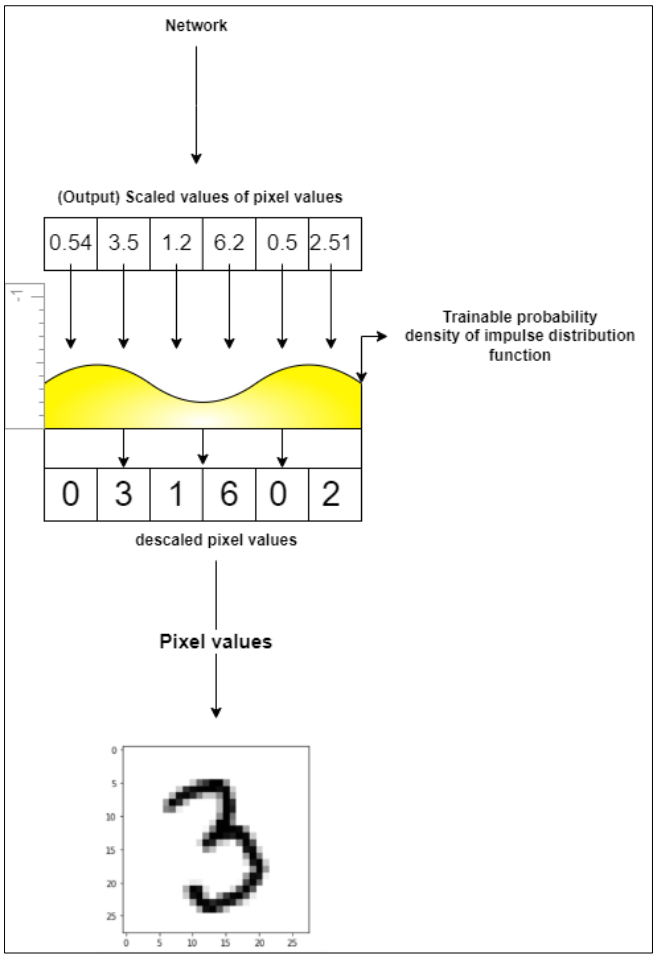


Math behind the encoder

The encoder will convert the image into a pixel values. A trainable density distribution function will scale the values of the pixel values.

- Decoder

The decoder will contain the same trained distribution density function (taking the already trained distribution density function from the encoder), and this function will de-scale the output scaled pixel values. This process is then subjected to a decoder function to obtain the deconstruction of the output for visualizing the input image.



Math behind the decoder

The output consists of the values of the scaled feature map, which are descaled by the density distribution function. Decoder to convert pixel values into an image.

- The sync

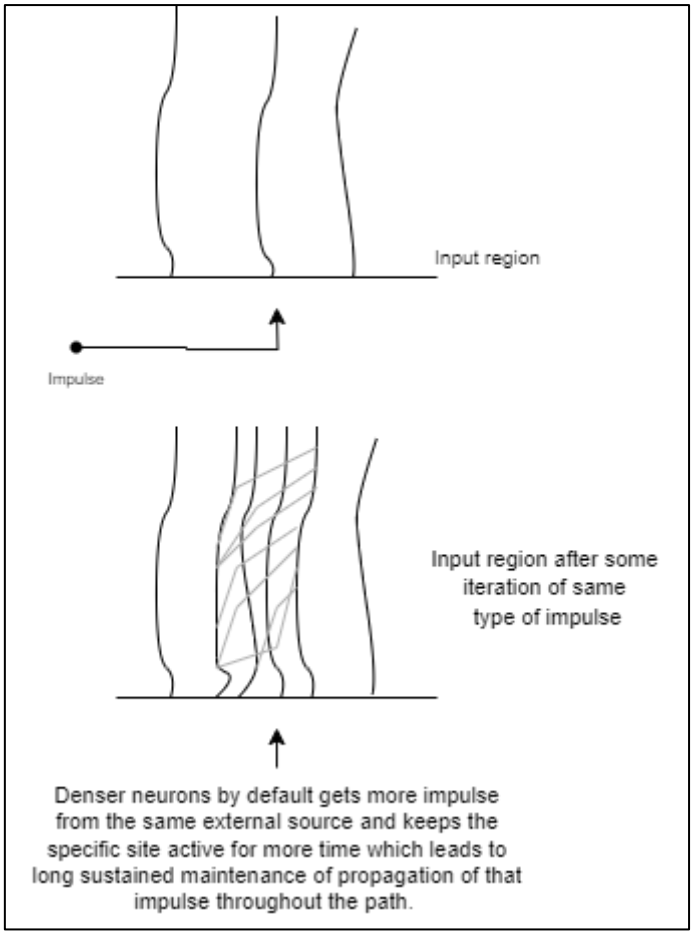
The encoder and decoder share the same continuously trainable distribution density function that correctly scales/descales the pixel values. Their synchronization is very important so that the destabilizing mechanism will work correctly, and the stored clusters will represent the smallest truth of reality.

- Biologically - inspired explanation

The brain receives impulses from the external environment by selectively prioritizing some pieces of input from the overall input. Over time, every piece becomes prioritized if we continuously receive that particular type of input. However, occasionally viewed types of inputs are selectively processed. The input regions, like the rods and cones in the eyes, and other receptors that receive input, contain dense neurons that can be attracted to the incoming impulses from the external environment. Naturally, impulses initiate the synthesis of neurochemicals that assist in making synapses or connections. This process leads to neurons becoming denser in specific locations on the input region. This denser region

collects more impulses from the same external environment (any particular spot on the overall image), which is useful for propelling the impulse a long way into the brain for computation. Additionally, the denser neurons come closer together so that they can connect with each other, allowing the receptor to activate itself after being activated by the external environment.

This setup is done to fully propagate the impulse and assist in the long journey it has to travel in the brain. The encoder and decoder are set up in a way to form these denser neurons at both the encoder and decoder regions in the form of distribution density function. Reactivation is not necessary for artificial models, and selectivity of specific pieces is important. Therefore, the concept of forming denser neurons was applied in this model. Selectivity is needed to form clusters in the network. (Refer to BrainX theory for further details).



We will include some kind of distribution to indicate the strength of the impulse. We will store the distribution of each pixel vector to assign specific priorities to each chained neuron. If one input neuron contains the highest distribution parameter, then its value will be higher. This is analogous to what happens in the brain: higher connections in the input neuron will have higher intensity and get activated to give a specific output. For each colour, there is a predefined intensity (predefined neurons accumulated because of that limited intensity) determined by our eye cells. Above and below that range is not suitable for

detecting the colour. Therefore, having a trainable distribution will act like that: more distribution means more neurons (as neurons are attracted to impulses, the amount of impulse determines the attraction of neurons) in that place which detects a particular specific colour.