Investigating Spatial Similarity in Spatial Pooler

M.Engineering IT
Individual Project
Jayashree Regoti
1324798
jayashree.regoti@stud.fra-uas.de

Abstract— **Hierarchical temporal memory (HTM) renders a structure that models various fundamental computational principles of the Neocortex. In this experiment, we will work with a spatial pooler (SP), a vital component of HTM. Spatial pooler is referred to as a learning algorithm, which Neocortex inspires. Spatial Pooler is used to sparse encode spatial patterns as in later processing it is used as input inside Hierarchical Temporal Memory. The Spatial Pooler assort spatially alike inputs into the exact sparse distributed representation memorized as a set of active mini-columns during the learning process. Here, the functionality of sparse distributed representation (SDR) is to provide HTM with the learning of sequences generated during SP's learning process. In this paper, we try to find out the Spatial similarity of the Spatial pooler by changing few configuration parameters in the experiment and see how these changes can influence results. The spatial pooler, in this case, can make a difference of vectors more significant than it is. As it is suitable to recognize minor details. The output has recorded the change in percentage (which is represented in tabular form) and in graphical form, making further analysis much more effortless.**

Keywords— ***Hierarchical Temporal Memory (HTM), Spatial Pooler (SP), Temporal Memory (TM),* Sparse Distributed Representation (*SDR), Neocortex, Spatial Similarity***

I. INTRODUCTION

Hierarchical Temporal Memory (HTM) is a system that models a few of the algorithmic and structural characteristics of the neocortex. HTM carries out a biological model which is based on the theory of memory prediction of the brain. A standard HTM biological network has a vast population of HTM cells and neurons, usually organized as a single layer or two-dimensional array (Marcin Pietron).

Alongside the concept of cells, the HTM structure also uses the term HTM columns because the spatial-organizational feature in the analogy is present in the real neocortex. Other than a single layer and two dimensional, then HTM also comprises multiple, and layers of neurons aligned vertically; this alignment leads to virtual three-dimensional presentation of HTM cells which are arranged in columns. (Coding, 2017)

In HTM, one of the main components is Spatial Pooler (SP); the main functionality of Spatial Pooler in HTM is to continually encode the input stream as well as another input source of information.

HTM structure can have one, two or more cells per column; everything depends on the requirement and availability; this depends on distinctiveness that all cells in all the columns receive similar feed-forward data via the distributed proximal dendrites. In contrast, each cell in a column receives identical lateral input data from its adjacent cells through its own distal dendrites. The feed-forward input stream allows HTM neural columns to become active and allows individual cells in a column to change into the predictive stage when it is laterally simulated. This ability of this model is the main component to learn sequence. (Damir Dobric, Andreas Pech, Bogdan Ghita, Thomas Wennekers, 2020)

This paper shows how to experiment with Spatial similarity. It first starts the Spatial Pooler and waits until it enters the stable state. The SP is trained with a few artificial vectors. The primary goal of this paper is to change the input parameter like InhibitionRadius, PotentialRadius, Global and LocalInhibition and observe how they influence the result. Then compare the current result with the past result. As SP tolerates small changes, we have to see that output SDR similarities for all vectors increased. (Damir Dobric, Andreas Pech, Bogdan Ghita, Thomas Wennekers, 2020)

II. METHODOLOGY

A human brain continuously receives a large amount of data from the outer world through peripheral sensors that transform the input data into a vast amount of spike successions. Individual cortical neuron needs to sort out a surge of time-fluctuating contributions by shaping synaptic associations with a sub dataset of the pre-synaptic neurons. The activation pattern of these populations of neurons then contributes to our response and understanding. The main thing to know about neuroscience is to understand how each neuron learns to respond to specific input data spike patterns and how these neurons represent features of the input data more dynamically and flexibly. (Damir Dobric, Andreas Pech, Bogdan Ghita, Thomas Wennekers, 2020)

The HTM aims to work in a way that replicates the human brain (to perform with exceptional capabilities). To work in such a way, user input needs to be given to the HTM as then it can anticipate and can memorize. Here we can provide different types of inputs such as scalar data, date-time string or an image. Once input is provided, it is converted to SDR with the help of an encoder. The classifier then learns this converted data. (Coding, 2017)

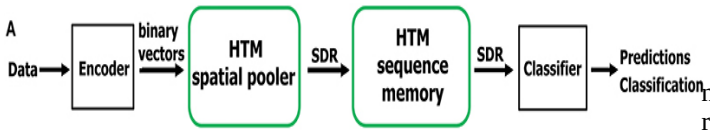

Figure 1: HTM Spatial Pooler

The above figure represents the HTM spatial pooler system with an encoder, an HTM spatial pooler, a temporal memory and an SDR classifier. (Coding, 2017)

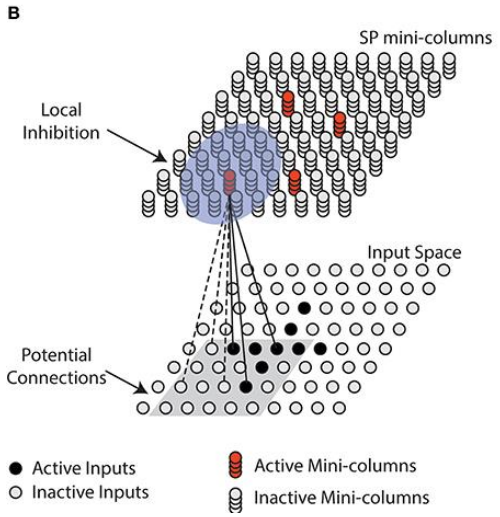

Figure 2: The above image represents the conversion of input to SDR

As HTM converts user inputs to SDR, an individual spatial pooler mini-column creates synaptic connections with the sub dataset of the input (such as potential connections and grey square). The local inhibition mechanism guarantees an inadequate fraction of the spatial pooler mini-columns. Here, most inputs are active inside the local inhibition radius, represented in a shaded blue circle. The Synaptic permanence are modified concerning the Hebbian rule: for every spatial pooler mini-column, active inputs (represented in black line) are augmented, and inactive inputs (represented in dashed lines) are corrected (Coding, 2017).

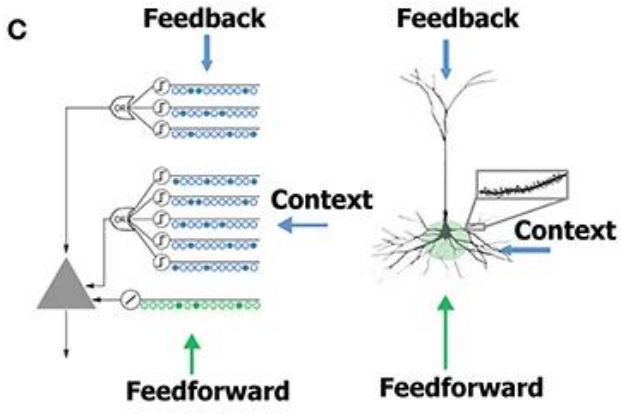

Figure 3: dendritic integration zones

In the above figure, on the left, there is an HTM neuron that has three discrete dendritic integration regions, resembling distinct sections of the dendritic tree (pyramidal neuron (right side)). The spatial pooler forms the feed-forward links onto the proximal dendrite.

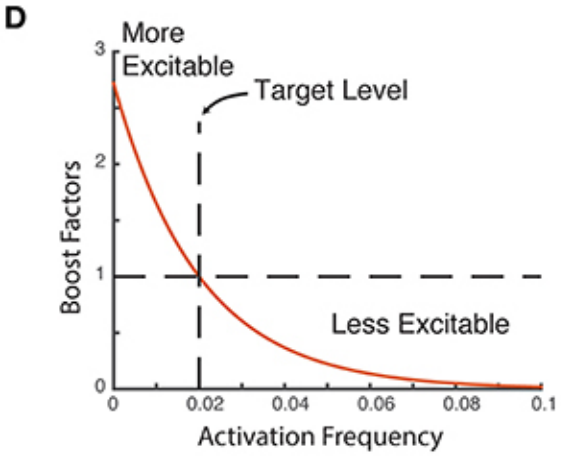

Figure 4: excitability of mini-column vs past activation frequency

The readability of a spatial pooler mini-column depends on its earlier activation frequency (Coding, 2017).

III. IMPLEMENTATION

The goal of this experiment is to check the spatial similarity in a spatial pooler. This experiment is carried out in C# language and using functionality of NeocortexApi packages also Microsoft.Net. The actual experiment is to find relationship between input sparsity and noise robustness in HTM spatial pooler. This experiment is divided into two parts, Spatial similarity, and noise robustness.
In this paper, we will discuss how spatial similarity is carried out. For this I was working on SpatialSimilarityExperiment.cs which takes image as an input.

Here is the github link to the experiment
https://github.com/JayashreeRegoti/neocortexapi/blob/master/NeoCortexApi/NeoCortexApi.Experiments/SpatialSimilarityExperiment.cs

The above experiment is carried out in the following stages:

- Giving image as an input and assigning the parameters like imageSize, localAreaDensityValue, potentialRadiusValue
- Then initializing the configuration parameters used in the experiment.
- Binarizing the input image
- Then create 1D training vector
- RunExperiment

1. Giving image as an input and assigning the parameters like imageSize, localAreaDensityValue, potentialRadiusValue

Assign of input images with parameters like image size, local area density value, potential radius value. In this experiment the specified image size is 15x15 pixels.

```
[DataTestMethod]

[DataRow(new string[] { "pixel1_1.png", "pixel1_2.png" },
                                              15, 0.2, 0.5)]
[DataRow(new string[] { "pixel1_1.png", "pixel1_2.png" },
                                              15, 0.4, 0.3)]
[DataRow(new string[] { "pixel1_1.png", "pixel1_2.png" },
                                              15, 0.2, 0.4)]
```

Figure 5: Assign of images and corresponding image size and configuration parameters

For this experiment, I have used following as an input. Created a box and gradually increasing the size of the box

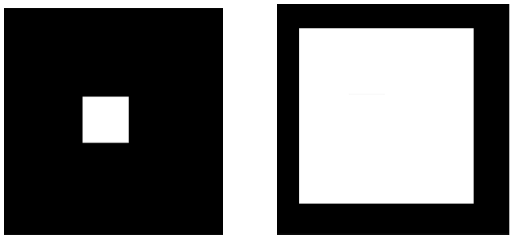

Figure 6: Input images with 10% (left) and 90 % (right) occupancy (non-zero bits)

2. Then initializing the configuration parameters used in the experiment

Here the potential radius and local area density has been changed gradually to check with the output of the input image.

```
//This is a set of configuration parameters used in the exper
HtmConfig cfg = new HtmConfig(new int[] { inputBits }, new int
{
    CellsPerColumn = 10,
    MaxBoost = maxBoost,
    DutyCyclePeriod = 100,
    MinPctOverlapDutyCycles = minOctOverlapCycles,
    StimulusThreshold = 5,
    GlobalInhibition = false,
    NumActiveColumnsPerInhArea = 0.02 * numColumns,
    PotentialRadius = (int)(potentialRadiusValue * inputBits),
    LocalAreaDensity = localAreaDensityValue,//  1,//0.5,
    ActivationThreshold = 10,
    MaxSynapsesPerSegment = (int)(0.01 * numColumns),
    Random = new ThreadSafeRandom(42)
};
```

Figure 7: Assigning configuration parameters

The change in these parameters determine the output of the spatial pooler.

3. Binarizing the input image

Once the images and parameters are assigned the next step is to run the test method. The primary thing the test does is to convert the input image to binary data. Below is the source code used for the same.

```
Binarizer binarizer = new Binarizer(200, 200, 200, ima
binarizer.CreateBinary(Path.Combine(TestDataFullPath,
    testImageFileName), binarizerFileName);
var lines = File.ReadAllLines(binarizerFileName);

var inputLine = new List<int>();
foreach (var line in lines)
{
    var lineValues = new List<int>();
    foreach (var character in line)
    {
        if (Int32.TryParse(character.ToString(),
            out int bitValue))
        {
            lineValues.Add(bitValue);
        }
        else
        {
            lineValues.Add(0);
        }
    }
    inputLine.AddRange(lineValues);
}
inputValues.Add(inputLine.ToArray());
```

Figure 8: Binarizing the input image

The resultant of this is as follows

```
000000000000000
000000000000000
000000000000000
000000000000000
000001110000000
000001111100000
000001111100000
000001111100000
000001111100000
000001111100000
000000000000000
000000000000000
000000000000000
000000000000000
```

Figure 9: Binarized input image

The 2D binary image is again converted into 1D training array.

4. **Get training vector**

Once the input image is converted to binary form then this binary is used to create a 1D training vector. For this we have considered experiment code to be 2, where it checks if the test image file name is null, or the image size is zero. If

this condition is satisfied, then it returns to the list and checks for the next position until whole array is read.

```
else if (experimentCode == 2)
{
    if (testImageFileNames == null || imageSize == 0)
    {
        return new List<int[]>();
    }

    return GetInputVectorsFromImages(testImageFileNames, imageSize);
}
else
    throw new ApplicationException("Invalid experimentCode");
```

Figure 10: Get training vector

5. **RunExperiment**

In this method the HTM is initialized then instance of spatial pooler is initialized. Then it learns the compute of spatial pooler object. We intern the following steps.

- Get overlap over every single column.
- Here we boost calculated Overlaps. This is called Homeostatic Plasticity Mechanism.
- Then we call inhibition column which performs the inhibition algorithm. This method calculates the density of the inhibition and executes either global or local inhibition algorithm.
- Then we adapt the permanence values of the synapses based on the input vector, and the chosen column after inhibition round. Permanence values are increased for synapses connected to input bits are turned on and decreased for synapses connected to input bits that are turned off.
- Next the overlap duty cycle is a moving average of the number of inputs which overlapped with each column. The activity duty cycles is moving average of the frequency of activation for each column.
- Then we increase the permanence values of synapses of columns whose overlap level has been too slow. Such columns are identified by having an overlap duty cycle that drops too much below those of their peers. The permanence values for such columns are increased.
- Then we update the boost factor which increases the overlap of inactive columns to improve their chances of becoming active, and hence encourage participation of more columns in the learning process.
- After certain number of rounds we disabled the boosting of the homeostatic plasticity new born effect.
- Once the enough rounds are passed it goes into stable state. Then we generate the result, which draws all the inputs are related SDR's. It also outputs the similarity matrix.

IV. RESULT

In this experiment we have used few artificial vectors to train the spatial pooler. The white section in this vector increases the space occupied as seen below. The shapes of size 5, 10%,15%, 25%,..50% respectively.

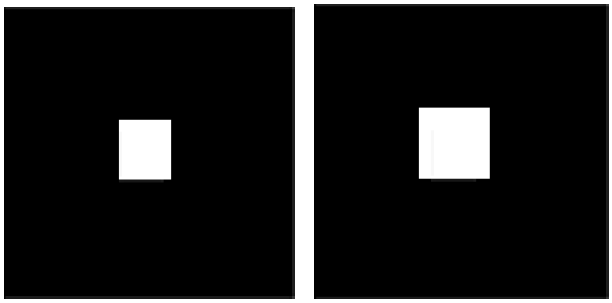

Figure 11: Input with 5% (left) and 10% (right) occupancy

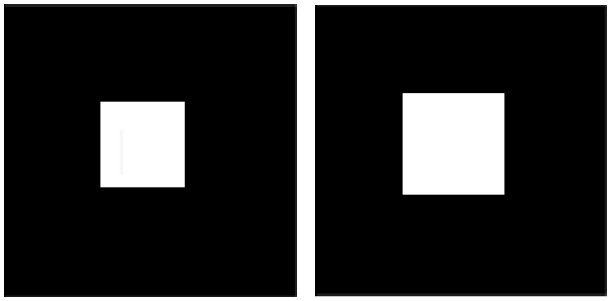

Figure 12: Input with 15% (left) and 25% (right) occupancy

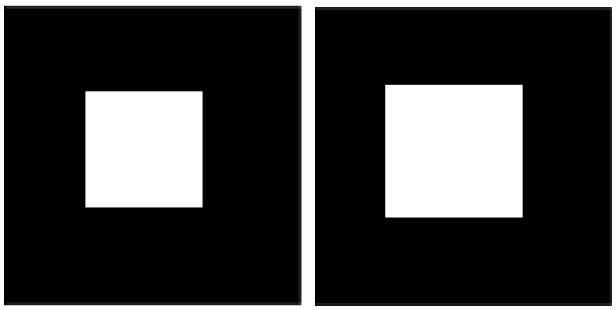

Figure 13: Input with 35% (left) and 45% (right) occupancy

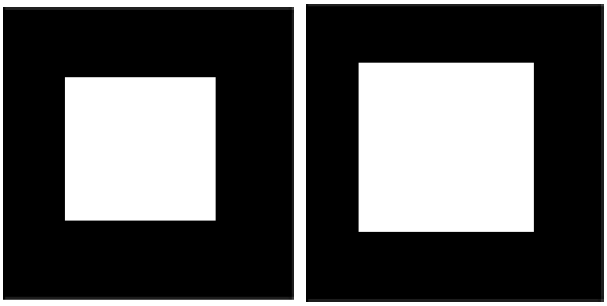

Figure 14: Input with 50% (left) and 65% (right) occupancy

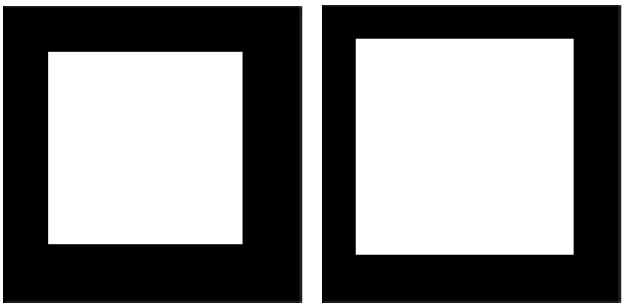

Figure 15: Input with 80% (left) and 90% (right) occupancy

After giving the above images as input and changing configuration parameters in the experiment, here is the image representations of non-zero input vector (left) and the resulting SDR (right).

Here, local area density for all the inputs images is 0.5 and the potential radius of the given input is change to 56, 67, 90 and 101 respectively.

LocalAreaDensity = 0.5, PotentialRadius= 56

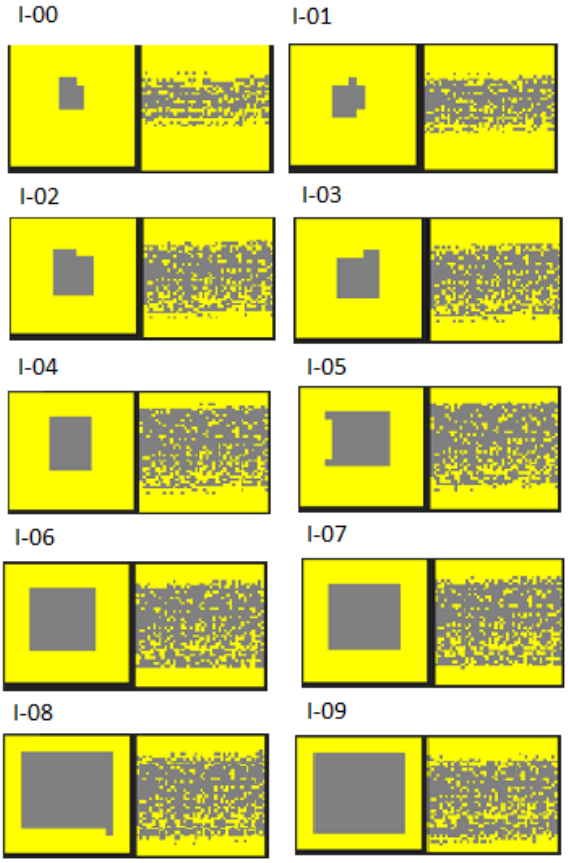

Figure 16: Image representations of the input (left) and the resulting SDR (right), where LocalAreaDensity = 0.5, PotentialRadius= 56

LocalAreaDensity = 0.5, PotentialRadius= 67

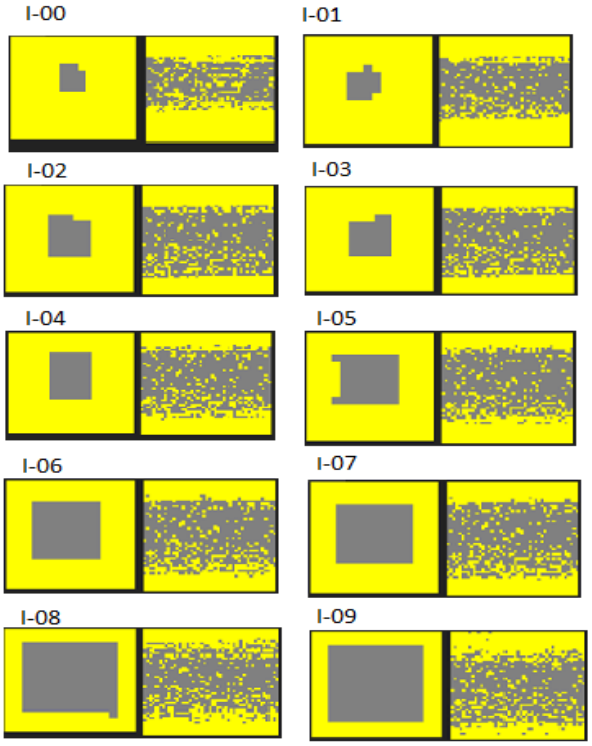

Figure 17: Image representations of the input (left) and the resulting SDR (right), where LocalAreaDensity = 0.5, PotentialRadius= 67

LocalAreaDensity = 0.5, PotentialRadius= 90

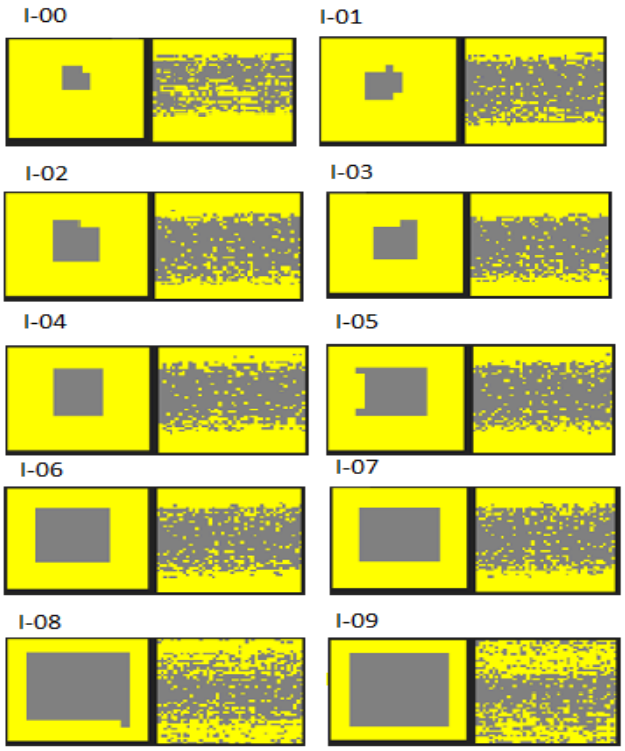

Figure 18: Image representations of the input (left) and the resulting SDR (right), where LocalAreaDensity = 0.5, PotentialRadius=90

LocalAreaDensity = 0.5, PotentialRadius= 101

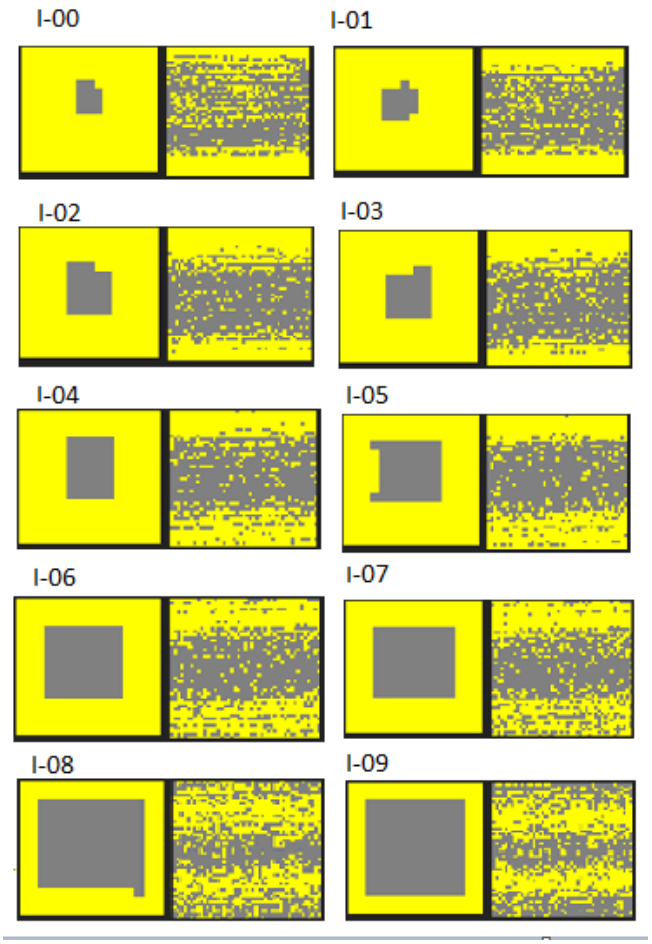

Figure 19: Image representations of the input (left) and the resulting SDR (right), where LocalAreaDensity = 0.5, PotentialRadius= 101

After this we are comparing every single image with corresponding other images like image 1(I-00) is compared with all the other image inputs (I-02, I-03….I-09), by this comparison a graph is drawn. The comparison table and a comparison graph for individual images are given below.

In the table, first column represents different input images, second column is the comparison of image I-00 with other input images and from column 3 to column 7 represents the comparison of I-00 with other input images when the potential radius is 45, 56, 67, 90, and 101 respectively.

I-00						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	100	100	100	100	100	100
I-01	62.5	73.93	77.08	76.17	82.23	80.08
I-02	39.29	53.78	64.55	71.48	78.03	72.46
I-03	33.33	54.71	64.26	70.9	76.46	67.77
I-04	31.43	44.43	64.84	74.61	74.02	68.16
I-05	21.57	44.43	64.06	72.56	69.04	57.62
I-06	17.19	44.34	63.87	70.21	62.4	51.76
I-07	15.28	44.24	63.48	69.43	59.67	47.46
I-08	9.91	44.24	60.45	59.57	42.09	35.35
I-09	9.09	43.55	58.01	51.07	36.72	30.76

Figure 20: Comparison table of I-00

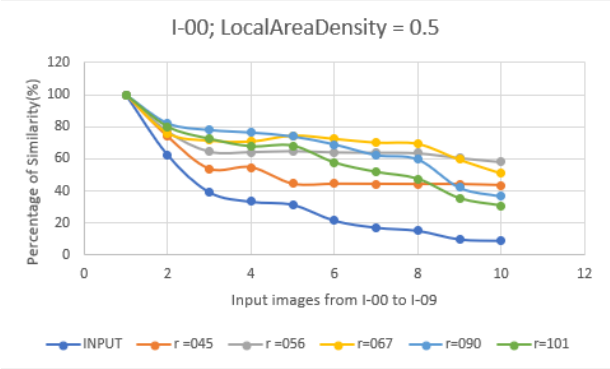

Figure 21: Comparison graph of I-00

I-01						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	62.5	73.93	77.08	76.17	82.23	80.08
I-01	100	100	100	100	100	100
I-02	57.14	72.1	82.81	91.31	84.77	76.27
I-03	55.56	73.43	82.71	91.31	83.4	73.24
I-04	45.71	59.57	82.52	90.82	80.86	68.95
I-05	31.37	59.57	82.03	88.09	76.56	56.84
I-06	25	59.47	82.13	88.28	68.75	49.51
I-07	22.22	59.47	81.54	87.79	66.5	45.02
I-08	14.41	59.47	78.22	76.46	45.51	34.28
I-09	13.22	58.79	75	68.65	38.67	30.18

Figure 22: Comparison table of I-01

Figure 23: Comparison graph of I-01

I-02						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	39.29	53.78	64.55	71.48	78.03	72.46
I-01	57.14	72.1	82.81	91.31	84.77	76.27
I-02	100	100	100	100	100	100
I-03	89.29	96.81	98.93	98.34	92.58	86.23
I-04	80	82.62	92.87	87.99	86.04	83.01
I-05	54.9	81.84	90.04	86.72	79.88	69.04
I-06	43.75	82.32	93.55	88.57	75.39	60.06
I-07	38.89	82.03	93.46	87.99	72.66	55.08
I-08	25.23	81.05	89.26	77.25	51.86	41.02
I-09	23.14	80.27	89.45	71.39	43.55	35.64

Figure 24: Comparison table of I-02

Figure 25: Comparison graph of I-02

I-03						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	33.33	54.71	55.56	70.9	76.46	67.77
I-01	55.56	73.43	82.71	91.31	83.4	73.24
I-02	89.29	96.81	98.93	98.34	92.58	86.23
I-03	100	100	100	100	100	100
I-04	77.14	80.86	92.87	88.09	85.84	82.13
I-05	52.94	80.18	90.14	87.21	78.71	68.65
I-06	42.19	80.66	93.65	89.16	74.32	59.38
I-07	37.5	80.47	93.65	88.57	71.78	54.69
I-08	24.32	79.69	89.26	77.64	51.46	41.7
I-09	22.31	78.91	89.36	71.88	43.46	36.52

Figure 26: Comparison table of I-03

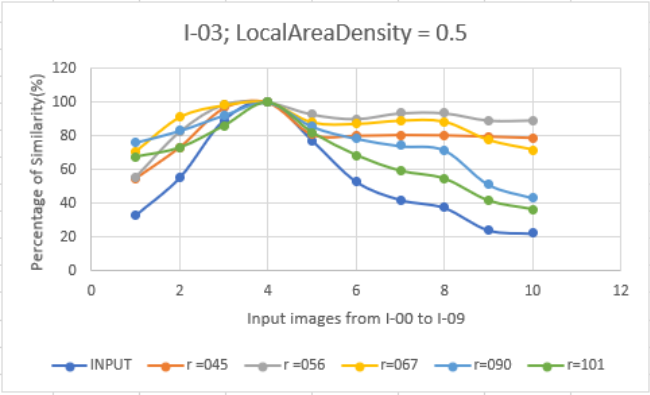

Figure 27: Comparison graph of I-03

I-04						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	31.43	44.43	64.84	74.61	74.02	68.16
I-01	45.71	59.57	82.52	90.82	80.86	68.95
I-02	80	82.62	92.87	87.99	86.04	83.01
I-03	77.14	80.86	92.87	88.09	85.84	82.13
I-04	100	100	100	100	100	100
I-05	68.63	97.56	95.02	92.87	85.84	76.07
I-06	54.69	92.19	93.75	89.65	79.69	66.21
I-07	48.61	91.99	92.87	88.96	76.17	60.16
I-08	31.53	90.33	88.87	78.61	55.18	44.73
I-09	28.93	85.35	85.06	67.68	46.68	39.06

Figure 28: Comparison table of I-04

Figure 29: Comparison graph of I-04

I-05						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	21.57	44.43	64.06	72.56	69.04	57.62
I-01	31.37	59.57	82.03	88.09	76.56	56.84
I-02	54.9	81.84	90.04	86.72	79.88	69.04
I-03	52.94	80.18	90.14	87.21	78.71	68.65
I-04	68.63	97.56	95.02	92.87	85.84	76.07
I-05	100	100	100	100	100	100
I-06	79.69	93.55	94.43	93.65	87.01	81.54
I-07	70.83	93.36	93.46	92.97	84.28	76.17
I-08	45.95	91.7	90.92	82.23	60.64	60.55
I-09	42.15	86.72	84.77	70.51	50.88	54.3

Figure 30: Comparison table of I-05

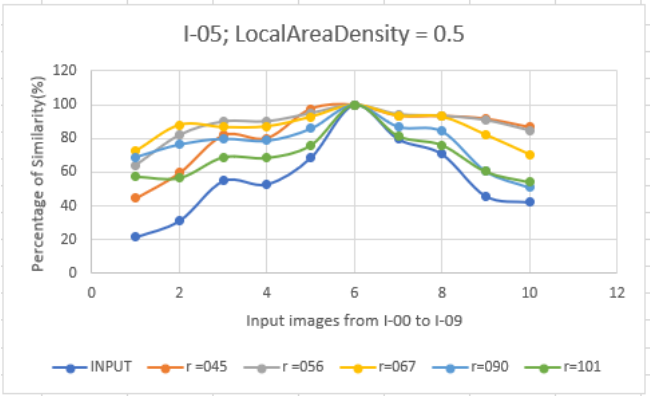

Figure 31: Comparison graph of I-05

I-06						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	17.19	44.34	63.87	70.21	62.4	51.76
I-01	25	59.47	82.13	88.28	68.75	49.51
I-02	43.75	82.32	93.55	88.57	75.39	60.06
I-03	42.19	80.66	93.65	89.16	74.32	59.38
I-04	54.69	92.19	93.75	89.65	79.69	66.21
I-05	79.69	93.55	94.43	93.65	87.01	81.54
I-06	100	100	100	100	100	100
I-07	88.89	98.63	97.46	97.36	92.97	87.3
I-08	57.66	95.61	91.11	85.06	69.04	70.31
I-09	52.89	92.58	88.67	74.8	59.38	65.14

Figure 32: Comparison table of I-06

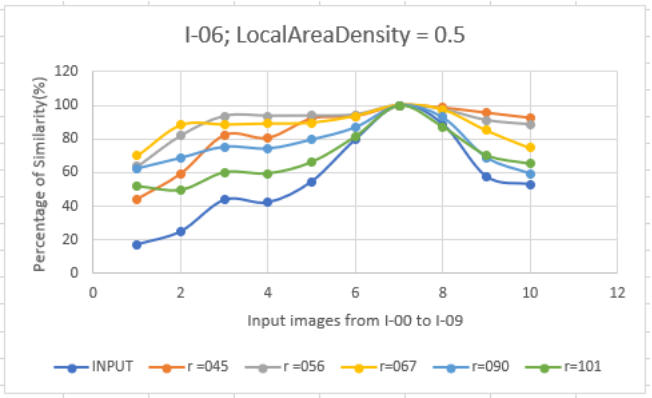

Figure 33: Comparison graph of I-06

I-07						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	15.28	44.24	63.48	69.43	59.67	47.46
I-01	22.22	59.47	81.54	87.79	66.5	45.02
I-02	38.89	82.03	93.46	87.99	72.66	55.08
I-03	37.5	80.47	93.65	88.57	71.78	54.69
I-04	48.61	91.99	92.87	88.96	76.17	60.16
I-05	70.83	93.36	93.46	92.97	84.28	76.17
I-06	88.89	98.63	97.46	97.36	92.97	87.3
I-07	100	100	100	100	100	100
I-08	64.86	95.9	92.19	86.23	72.66	77.15
I-09	59.5	92.68	89.75	75.98	62.79	70.9

Figure 34: Comparison table of I-07

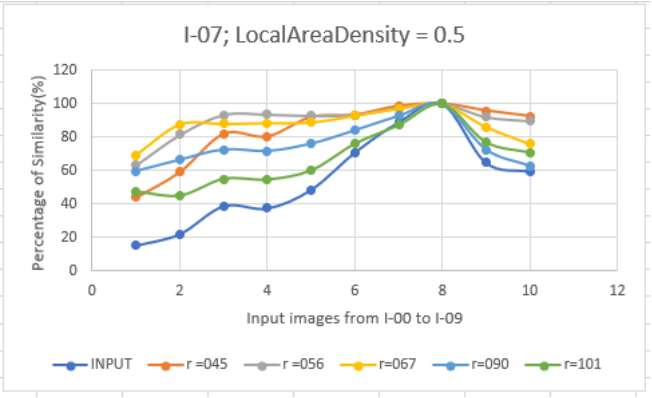

Figure 35: Comparison graph of I-07

I-08						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	9.91	44.24	60.45	59.57	42.09	35.35
I-01	14.41	59.47	78.22	76.46	45.51	34.28
I-02	25.23	81.05	89.26	77.25	51.86	41.02
I-03	24.32	79.69	89.26	77.64	51.46	41.7
I-04	31.53	90.33	88.87	78.61	55.18	44.73
I-05	45.95	91.7	90.92	82.23	60.64	60.55
I-06	57.66	95.61	91.11	85.06	69.04	70.31
I-07	64.86	95.9	92.19	86.23	72.66	77.15
I-08	100	100	100	100	100	100
I-09	91.74	94.34	91.21	85.74	88.09	91.31

Figure 36: Comparison table of I-08

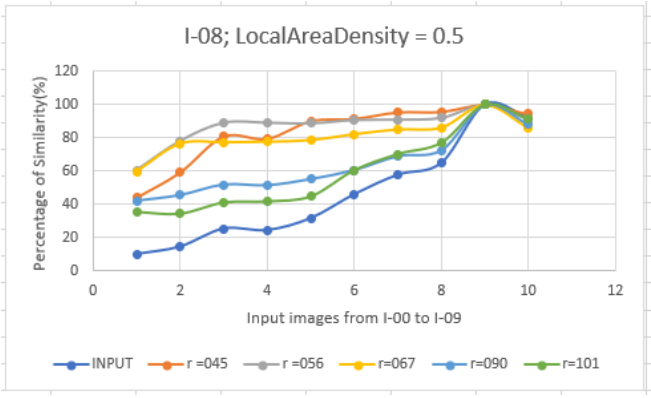

Figure 37: Comparison graph of I-08

I-09						
I01-I09	INPUT	r =045	r =056	r=067	r=090	r=101
I-00	9.09	43.55	58.01	51.07	36.72	30.76
I-01	13.22	58.79	75	68.65	38.67	30.18
I-02	23.14	80.27	89.45	71.39	43.55	35.64
I-03	22.31	78.91	89.36	71.88	43.46	36.52
I-04	28.93	85.35	85.06	67.68	46.68	39.06
I-05	42.15	86.72	84.77	70.51	50.88	54.3
I-06	52.89	92.58	88.67	74.8	59.38	65.14
I-07	59.5	92.68	89.75	75.98	62.79	70.9
I-08	91.74	94.34	91.21	85.74	88.09	91.31
I-09	100	100	100	100	100	100

Figure 38: Comparison table of I-09

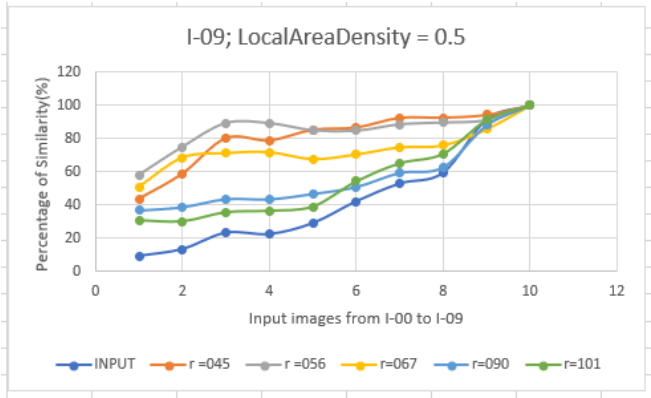

Figure 39: Comparison graph of I-09

After comparing all the graph, we can see that when the potential radius of the system is between 45 to 67, the similarity of images is more constant and being maximum similar. The efficiency of the system memorizing is high when the system falls under these parameters.

V. CONCLUSION

This paper concludes that the number of non-zero bits in the input vector and the configuration parameters enormous impact finding the similarity in Spatial Pooler. Keeping the configuration parameters with a specific range would be preferable to have higher memorization in the system. However, when we look at the comparison table, we can see

that the more the non-zero bits lesser the similarity. Finally, in this experiment, I have used images as an input, where I have gradually increased the number of non-zero bits in the images and compared every image with other images. The output of this comparison has been recorded in both tabular and graphical form.

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to Prof. Damir Dobric for helping me out to complete this project, as well as Prof. Dr Andreas Pech for your support and allowing me to do this project on the topic Investigating Spatial Similarity in Spatial Pooler, which also help me in doing much research. I have come to know so many new things, and it also helped me increase my knowledge and skills.

I. REFERENCES

[1] Dobric, D., Pech, A., Ghita, B. and Wennekers, T., 2020, November. On the Relationship Between Input Sparsity and Noise Robustness in Hierarchical Temporal Memory Spatial Pooler. In *Proceedings of the 2020 European Symposium on Software Engineering* (pp. 187-193).

[2] Cui, Y., Ahmad, S. and Hawkins, J., 2017. The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding. *Frontiers in computational neuroscience*, *11*, p.111.

[3] Pietron, M., Wielgosz, M. and Wiatr, K., 2016, February. Parallel Implementation of Spatial Pooler in Hierarchical Temporal Memory. In *ICAART (2)* (pp. 346-353).