Analyse Image Classification(Hand Drawn Shapes)

Mahdieh Pirmoradian
mahdieh.pirmoradian@stud.fra-uas.de

Omid Nikbakht
Omid.nikbakht@stud.fra-uas.de

***Abstract*— The purpose of this paper is to investigate a new approach to image classification based on open HTM version algorithm, a revolutionary biologically inspired model of the mammalian neocortex's structure, to maximize crucial influencing parameters of the HTM model predicting more precisely when applied to the categorized Hand Drawn Shapes.**

The primary goal was to investigate open-source implementation of the Hierarchical Temporal Memory in C#/.NET Core. By conducting more than 100 experiments for different HTM configurations, the behavior of system is observed and documented using diagrams to get a vivid overview of dependency of HTM to each of the modified parameters. Afterwards, a set of parameters, which have shown the better results of HTM functionality are discussed and proposed. Followed by, implementing a prediction code in the project, which enables HTM system to anticipate the category of an entered test input image.

Keywords—Hierarchical Temporal Memory (HTM), Sparse Distributed Representation (SDR), Spatial Pooler (SP), Temporal Memory (TM), Inhibition, Global Inhibition, Local Inhibition, Potential Radius, Local Area Density, Active Column, Inhibition Area, Neocortex, Visual Pattern Recognition.

I. INTRODUCTION

There are numerous sources of data in the world and in future the approach is instead of storing them in databases, to analyze them in the real time by online models and make predictions and detections from this data. The used model for this purpose should be fast enough to execute and decide at the same time. One of the effective factors on system speed is the amount of data which should be used for training. Therefore, previous models which worked based on large amount of dataset for training the models will not be useful. Besides, most of these models are used for controlling Sensors. So, another requirement is making precise prediction before that event happens. Previous used models didn't have prediction state; therefore, we need an unsupervised continuously learning algorithm.

Based on the believe that algorithms that use a more realistic neural model would produce stronger AI, Hierarchical Temporal Memory (HTM) is designed to mimic human NEOCORTEX. It turned computers more similar to the human brain and it's worth doing some analysis on it, as previous methods in deep learning, have significant limitations that is not intelligent. This model is introduced first by Hawkins and Blakeslee [1].

HTM works based on reverse engineering of the Neocortex of the human brain. Neocortex is part of the brain which controls many vital processes in our actions and behavior, thereby, making it the best suited for streaming data applications where the underlying patterns are continually changing [2].

HTM, learns from a stream of online data (online means in real time), That is how we acquire knowledge. It makes use of live, unsupervised data streams. The way we see, hear, and feel is based on the same framework. Furthermore, it can learn from a few examples as opposed to Convolutional Neural Network (CNN) as an example of a learning model, which requires millions of inputs.
If you take slices of your cortical brain from different places of your head, you will notice that the structure of this cells is almost identical, regardless of where this cell is taken from, visual processing or language generation are all the same. This means that the way of problem solving for different inputs are almost similar. They can only differ in the content of their internal memory, i.e., the information gained during the learning phase [3]. In general, it is accepted that different areas of the neocortex are interconnected to form hierarchical structure [4].

As information moves up the hierarchy, each area detects increasingly more general and invariant features of the input visual scene [5]. Another important feature of a human brain is that the brain of human consists of 20 to 30 billion neurons and if you look at the state of the brain you will notice that only 2 percent of this neurons are on at any time. Sparse Distributed Representation (SDR) which we use in HTM try to mimic the same behavior. It is a pattern representation approach in which just a limited number of cells are activated. The benefit of using it for a brain is that it only uses 2 percent of the capacity, so it saves a lot of memory. While, in most deep learning networks nowadays, dense representations are used. Data in computers are also represented densely. Which is in sharp contrast to the fact that our brains are quite sparse. Besides, it has been shown that SDRs can be significantly more noise-resistant than dense representations [6].

In this Project we looked over 100 diverse HTM network setups in total, to maximize the accuracy of prediction of model when applied to the categories of Hand Drawn Shapes Datasets from the Kaggle database [7].

Hierarchical Temporal Memory(HTM)

HTM, like the neocortex, consists of identical regions interconnected in hierarchy. While travelling up the hierarchy, information from several regions converges to one and vice versa. Also, intra-regional connections are possible. Such structure allows high-level representations to be formed from low-level sensorial data. Thus, objects inherit properties of their subcomponents [8].

ARCHITECTURE OF A HIERARCHICAL TEMPORAL MEMORY:

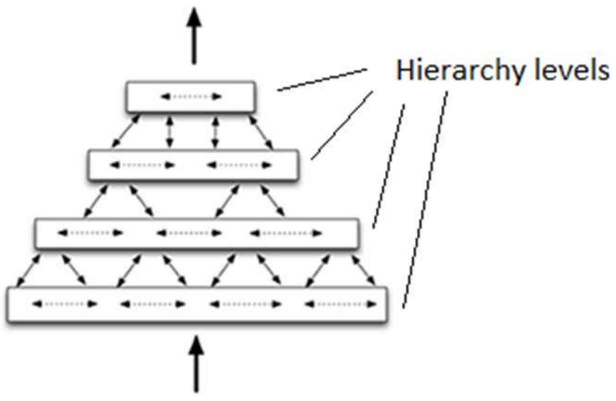


Figure 1 HTM hierarchy levels [9]

HTM consists of different hierarchy levels as its shown in figure 1. As the raw data is entered into the system in lower levels, it gets processed, and more specific information of the data is transferred to the higher layers of the HTM. This means the higher the layer of hierarchy, the more processed data and with more complex features are represented.

In other words, nodes in lower levels deal with simple events that change quickly and occupy smaller spatial areas, while nodes in higher levels deal with multiple events that were sensed by lower-level nodes, and thus are influenced by a greater range of data. Specifically, higher level nodes sense more complex causes, namely patterns of patterns, which evolve less quickly and exist in larger spatial areas [10].

MAIN COMPONENTS OF HTM:

The cells of the system are the Fundamentals of each HTM which play the same role which the neurons in neocortex play.

In the brain's neocortex layer, the cells are connected to each other by synapses and transfer the data upward or downward. In HTM inspiring from neocortex, the columns are made up of several cells that are sitting on top of each other and grouped together.

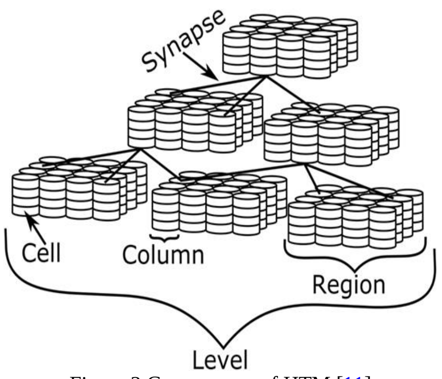


Figure 2 Components of HTM [11]

As shown in figure 2 the synapses are used to connect the cells to each other and make the data flow possible.

Six horizontal layers organized into vertical structures called cortical columns; these columns can be thought of as the basic repeating functional units of the neocortex [12].

A region is formed by combining a set of columns together and each level (layer) of hierarchy is consisted of numbers of regions.

HOW HTM WORKS:

Figure 3 gives us an overview of how HTM works from the beginning to the end of the process. The different parts of the system are described one by one as following. However, in this Special Project we did not use Temporal Memory.

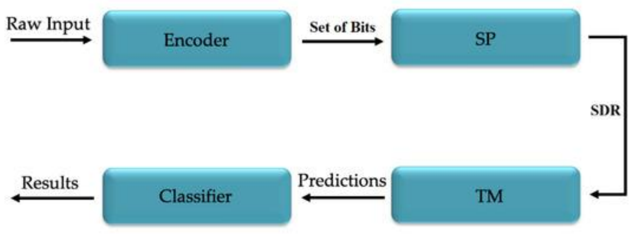
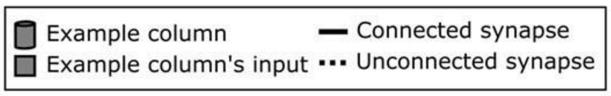


Figure 3 HTM network topology [13]

Input data, generated by a sensor or a data source which can also be continuous and temporal, is semantically encoded into a sparse array named sparse distributed representation (SDR). This sparse array of encoded data enters the system from the lowest layer of HTM.



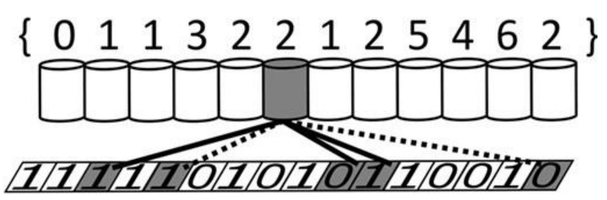


Figure 4 Synapse connection first layer of HTM and encoder [14]

As shown in figure 4 the synapses are responsible for connecting the encoded data bits to the columns so that the 1s and 0s can be transferred into the system. That is done by the spatial pooler which is in charge of handling the relationships between the columns of a region and the inputs bits [15].

Each column can be connected to several input bits, but not all the synapses that are connecting a cell to a column, are transferring the input data to the column. From input to the lowest layer or from each layer to its upper layer, the data is only transferred via the synapses which have the minimum permanence value of "SynPermConnected". Otherwise, the synapse does not transfer the data to the upper column. In figure 4 the unconnected and connected synapses are shown.

By entering the lower levels of hierarchy this encoded array of data gets standardised and normalised by spatial pooling. This standardisation and normalisation turn the input data generated in data source, into a sparse output vector with a specific size and a fixed sparsity [16].
This happens during the spatial pooling process, and according to the overlap of each column, where the columns of the lowest hierarchy level get connected to the bits of the input data. Overlap is the amount of 1s which are transferred to a column via a connected synapse which has a minimum amount of "SynPermConnected".

CONCEPT OF SPARSE DISTRIBUTED REPRESENTATIONS:

If we take some snapshots of the neocortex of the brain, this is very likely that we observe only about 2% of its neurons are in an active state. SDR is literally an array of 0's and 1's and as the HTM concept is trying to build a system on the same principles of brain's function, the SDRs are a mathematical representation of these sparse signals which will likely have around 2% of 1s. As it's shown in figure 5 the 1s are determined with red points and 0s are white.

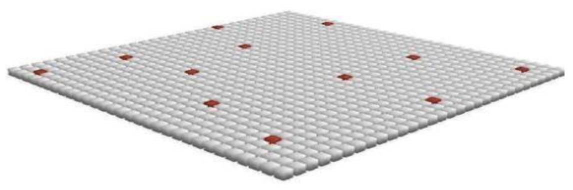


Figure 5 SDR with a sparsity of 2% [16]

To prevent the confusions, it should also be mentioned that the output SDR of the encoder is not necessarily 2% sparse, but after the spatial pooling process which uses the output SDR of the encoder, the output SDR of the special pooler must have around only 2% of sparsity.

After encoder provides a SDR of 0s and 1s, which as mentioned above is not necessarily 2% sparse, The Spatial Pooler takes this output SDR of encoder, and according to some parameters like inhibition area and Potential radius decides to set some of the columns in active mode. It actually is responsible for creating a 2% sparse distributed representation of the input SDR. For each input SDR (output of encoder) depending on Overlap value of columns it computes a set of sparse active columns.

SPATIAL POOLER AND ITS PARAMETERS:

Parameters of spatial pooler determine its functionality based on Hebbian Learning Rule.

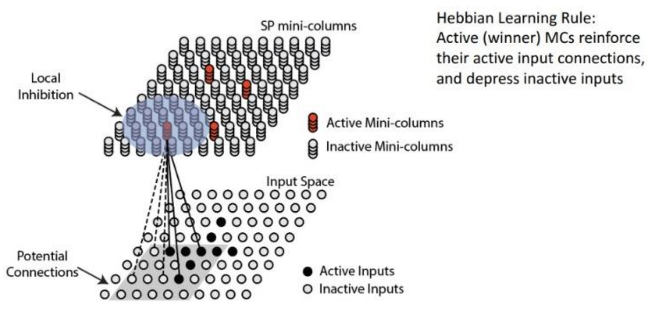


Figure 6 working illustration of spatial pooler [17]

Figure 6 is showing a general overview of how spatial pooler works.

On the input surface, which is the output SDR of encoder, the 1 bits are shown with black dots and 0 ones are white.

- **PotentialRadius:** The grey square on the input surface is determining a radius of entry bits to which a specific column can be connected and is called PotentialRadius. A short potential radius limits a column's receptive field, while a large potential radius allows the column to cover the whole input space.
- **SynPermConnected:** this parameter determines if a synapse has enough permanence to be able to transfer the data from the input bit to the column. As it can be seen in figure 4 there are two types of synapse connections. The synapse connections with complete lines are showing the synapses which their permanence is higher than SynPermConnected, this means they can transfer input bits to the column. But the synapses shown with incomplete lines are illustrating the ones which are not transferring data to the upper layer, because their synapse permanence is lower than the SynPermConnected. So only the synapses with a synapse permanence higher than SynPermConnected are eligible to transfer the bit data to a column to which they are connected.
- **InhibitionArea:** the purple area selected on the upper layer between the mini columns, is a group of mini columns in which only a specific number of columns can be set as active. This specific number

of active columns is chosen by either LocalAreaDensity or NumActiveColumnsPerInhArea.

- **LocalAreaDensity:** determines the density(percent) of active columns inside a local Inhibition Area. When LocalAreaDensity is used, a negative value should be assigned to NumActiveColumnsPerInhArea.
- **NumActiveColumnsPerInhArea:** this parameter determines the exact number of active columns in each Inhibition Radius. When this parameter is used the LocalAreaDensity should be set to a negative value.
- **Inhibition:** We have 2 types of inhibition in HTM. Global and Local Inhibition. Using global inhibition, the most active columns from the whole layer are selected. However, when global inhibition is disabled, column inhibition acts in local regions, in other word, the winning columns are determined by their own local neighborhoods. The potential radius parameter can partially govern the receptive field of columns; however, because the receptive fields of Spatial Pooling algorithm columns (and HTM cells in general) are dynamic, it cannot be manually modified. Using Global inhibition greatly improves performance.
- **GlobalInhibition:** when this parameter is set to true the InhibitionRadiuses are not considered anymore and the LocalAreaDensity or NumActiveColumnsPerInhArea is applied for the whole layer of mini columns and not for groups of mini columns with restricted number of columns in them.
- **Boosting:** It can aid in the competition for activation of columns. The ActiveDutyCycles and OverlapDutyCycles both measure boosting.

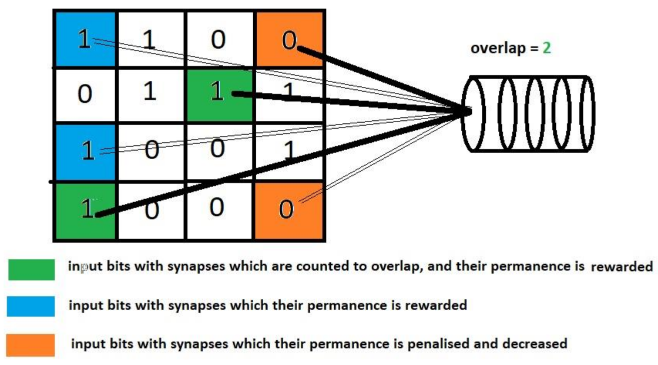
initializing of spatial pooler:

While the spatial pooler is initialized that uses the parameters mentioned above, the columns which are supposed to get connected to input bits are identified andsome synapses connect these columns to some of the input bits in PotencialRadius of each column. The permanence value of the connecting synapses is chosen random but will get updated later during each learning cycle.

overlap calculation:

After the spatial pooler is initialized, it calculates overlap for each mini column, to choose the active mini columns.

Overlap is the number of 1s that each column receives, through each of its synapses having a permanence of higher than SynPermConnected. So, the number of completely active synapses which can transfer data from input bits and are also transferring an input bit data with value of 1 is going to be the overlap value for each column. It should be also mentioned that the completely active synapses (having a permanence of higher than SynPermConnected, that are connected to input bits with a value of 0, also the synapses which are not completely connected, even though they are connected to an input bit of 1, are not considered calculating overlap of each column.



overlap = 2
🟩 input bits with synapses which are counted to overlap, and their permanence is rewarded
🟦 input bits with synapses which their permanence is rewarded
🟧 input bits with synapses which their permanence is penalised and decreased

Figure 7 overlaps and synapse connectivity’s

As illustrated in figure 7 each column has an overlap, and after the overlaps are calculated, the columns with highest overlaps are set as active. Number of active columns in each InhibitionArea is depended on either LocalAreaDensity or NumActiveColumnsPerInhArea. For example, in figure 6 only one column is allowed to be activated in the configured InhibitionArea (the purple circle), and that column is the one, with highest overlap among other columns in the same InhibitionArea.

Learning process of spacial pooler:

The active columns chosen, is a semantic representation of the input data which entered the HTM system by spatial pooler. As mentioned before the number of active columns should be around 2%, and that’s where the sparsity of the output SDR of spatial pooler maintains a 2% sparsity.

When observing the layer of mini columns illustrated in figure 6 from above, 4 active columns are observable. The view of mini columns layer from above is exactly the output SDR of spatial pooler, the unique semantic SDR representation of entry bits. This SDR is unique for each entry, but its also expected that for semantically similar input values, approximately similar output SDR of spatial pooler to be represented. And that’s where learning process plays its role, to help spatial pooler recognize if inputs are different or similar to each other.

The learning process of special pooler happens when it attempts to strengthen or weaken the connectivity (permanence) of the synapses which connect input bits to the mini columns. This strengthening and weakening follows the Hebbian Learning Rule, as mentioned before. For example, in figure 7 the synapses connected to the green and blue input bits are the bits which deliver a 1 value to system, so their permanence is going to be rewarded and increased by SynPermActiveInc value, because for the current output SDR of spatial pooler these synapses are the ones which deliver the semantic data of the input bits to system. The synapses connected to orange input bits are the ones which do not deliver any semantic information of input, and their permanence will be penalised and decreased by

SynPermInactiveDec. After some cycles of training, the spatial pooler updates its synapse connections so, that only the synapses which deliver 1s to the system, are connected to mini columns.

The higher the value of SynPermActiveInc is considered, the faster spatial pooler rewards and learns the synapses which represent input semantic information. Also, the higher the value of SynPermInactiveDec is considered, the faster the permanence of synapses which are not necessary for delivering the semantic data of input, are penalised, and forgotten. The two SynPermActiveInc and SynPermInactiveDec factors are also mentioned as learning and forgetting speed of the spatial pooler.

II. METHODOLOGY

How learning happens:

When two SDRs share active bits in the same place, the semantic properties represented by those bits are shared.

By finding the overlap between two SDRs(finding matching 1 bits) we can figure out how two representations are semantically similar and distinct.

```
listInputCorrelation.Add(temp, MathHelper
s.CalcArraySimilarity(binaries[filePathLi
st[i]].IndexWhere((el) => el == 1), binar
ies[filePathList2[j]].IndexWhere((el) =>
el == 1)));
```
(Experiment.cs L79)

Machine learning workflow:

This paper is discussing the influence of four parameters on behavior and efficiency of the implemented HTM system, regarding 100 experiments with different parameters values. The parameters which have been modified to observe the output results are as follows:

- Potential Radius (PotentialRadius)
- Local Area Density (localAreaDensity)
- Number of Active Columns per Inhibition Area (NumOfActiveColumnsPerInhArea)
- Global Inhibition (GlobalInhibition)

The data set for training is chosen from kagel [7] datacenter. And the chosen categories of shapes are Triangle, StraightCross, and Hexagon. For each category four images are provided for the HTM to be trained with.

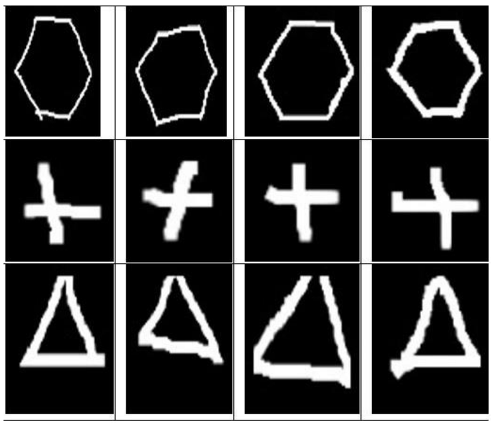


Figure 8 training dataset

The images and categories which are chosen for the system to be trained with is shown in Figure 9. But before the images are fed to the system they should get prepared for spatial pooler; this is done by the encoder. Because the entries of spatial pooler can be 0s and 1s, there is an encoder needed, which turns images into an array of 0s and 1s according to semantic specifications of the image.

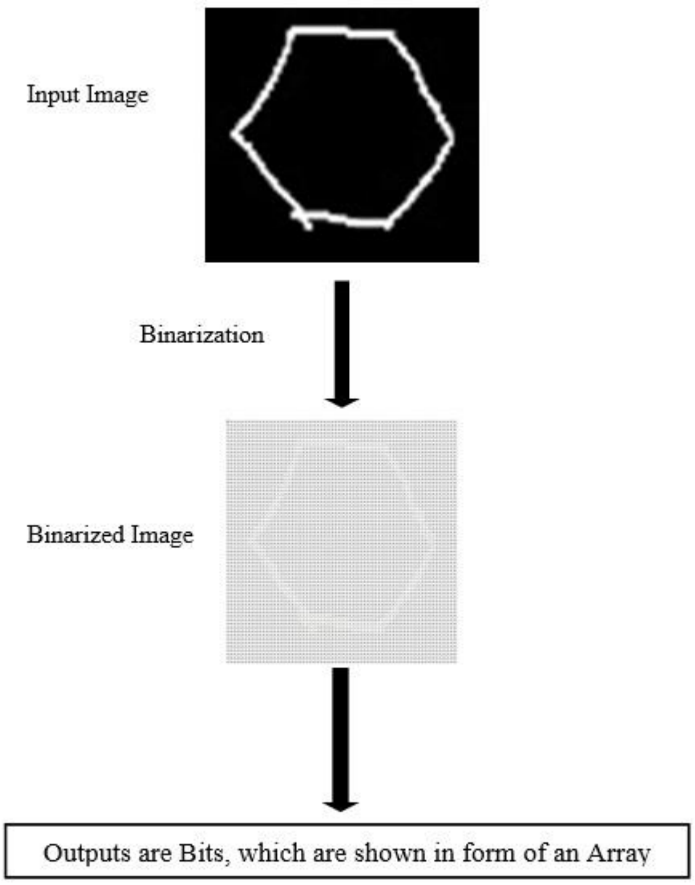


Figure 9 Binarization of dataset images

The Encoder uses a Binarize function which depending on the brightness of each pixel and comparing this brightness to a threshold, turns the brighter pixels into 1s and darker pixels into 0s (figure 9).

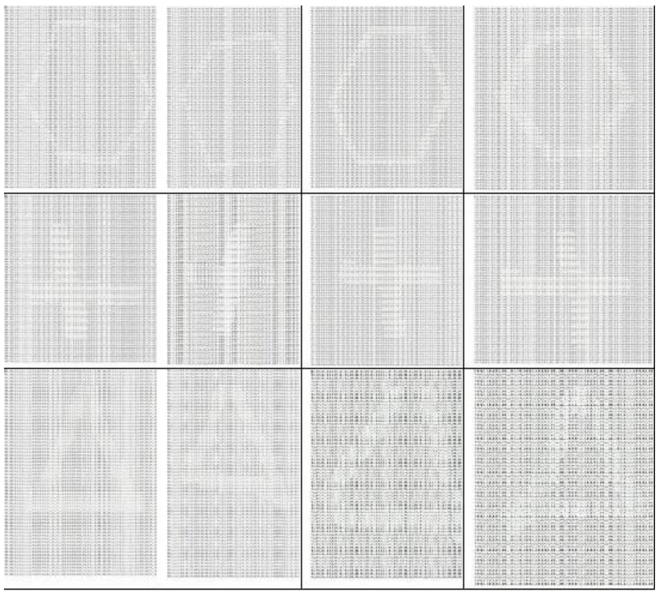


Figure 10 binarized output (SDR) of encoder for all categories

These 12 outputs of encoder shown in figure 10, are also mentioned as the output SDR of the encoder. In next step this SDR is fed to spatial pooler and enters the system.

After preparing the input data for spatial pooler, now the memorization process happens. This is where the HTM iterates through all input images, and by increasing or decreasing the connectivity of synapses, tries to learn the specifications of each category. The similarity of each category of images to itself (micro-similarity) and to other categories (macro-similarity) can also be calculated and can provide us with information which helps us to observe the functionality of HTM system even better. The target here, is to find the most efficient parameters of HTM, by modifying the four parameters mentioned above, regarding the macro and micro similarity results which can be seen in following. The parameters which lead to higher micro similarity and lower similarities, are desired here. The higher micro similarities mean that system can recognize images of same categories similar to each other, and lower macro similarities shows the ability of system to differentiate images of two different categories from each other.

III. Results

Results and discussing the experiments:

Depending on the parameters which are set for HTM the training cycles varied between 50 and over 300 cycles. First the similarity of two images of a same category is compared for several applied parameters of HTM, to see how similar or different are these images found by HTM. After that the similarity of images from different categories are discussed, followed by a comparison between the similarity results of images of a category to each other, and to images of another category.

Similarity between images of a same category:

Figure 12 show the compared images of the same category of hexagon, with their binarized representation.

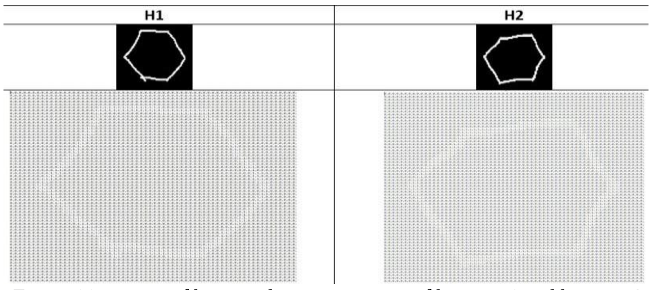


Figure 11 comparinf binarized representations of hexagon1 and hexagon2

1. Using NumActiveColumnsPerInhArea parameter, (value of localAreaDensity must be set to a negative value) Figure 13.

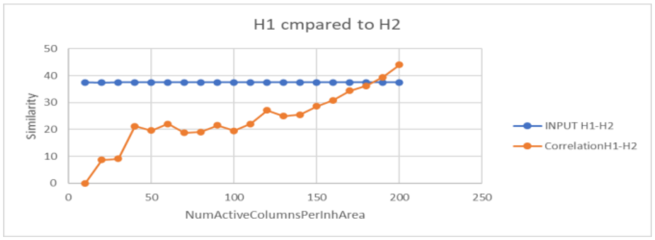


Figure 12 Comparing Hexagon1 to Hexagon2, varying the NumActiveColumnsPerInhArea from 10 to 200 (with Local Area Inhibition)

As shown in figure 13 The similarity is increased as the NumActiveColumnsPerInhArea is increasing. Increment of similarity of H1_H2 shows that the HTM sees the images of a same category more and more similar, as the NumActiveColumnsPerInhArea increases.

2. Using localAreaDensity (value of NumActiveColumnsPerInhArea must be set to a negative value) for different PotencialRadiuses Figure 14.

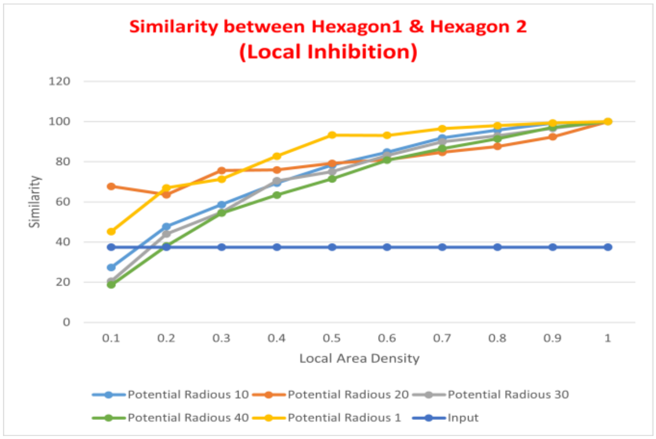


Figure 13 similarity between Hexagon1 and Hexagon2(with Local Area Inhibition)

As it can be seen, Because Hexagon1 and Hegaxon2 are both from the same category, we expect a high Similarity between both pictures' output SDR from the spatial pooler. Potential Radius 20 and Potential Radius 1 have the highest similarity from the beginning. And by increasing the Local Area Density the Similarity also increases.

3. Using localAreaDensity (value of NumActiveColumnsPerInhArea must be set to a negative value) and GlobalInhibition set to "True" Figure 15.

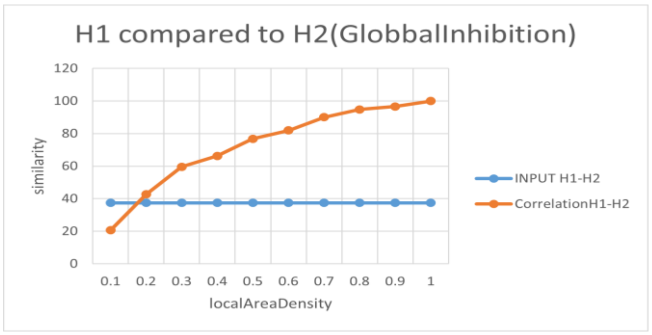


Figure 14 Comparing Hexagon1 to Triangle 1, varying the localAreaDensity from 10 to 100 (Global Inhibition)

By increasing the localAreaDensity while Global Inhibition is set to True, the H1 and H2 are looking more and more similar for the system, but even in the beginning of this increment the speed of similarity growth of these two is really high.

Similarity between images of different categories:

Figure 16 show the compared images of the two different categories of hexagon and triangle, with their binarized representation.

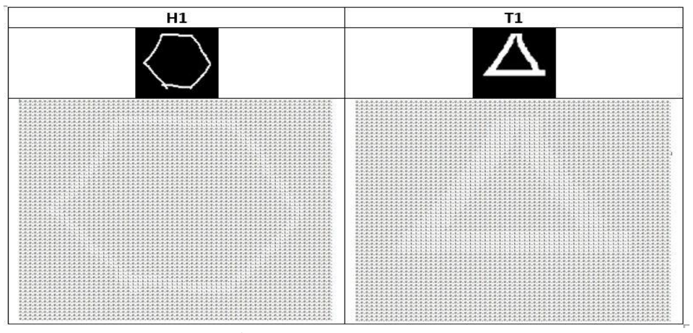


Figure 15 Input images of Hexagon1 and Triangle1 and their equivalent binarized format

1. Using NumActiveColumnsPerInhArea parameter, (value of localAreaDensity must be set to a negative value)

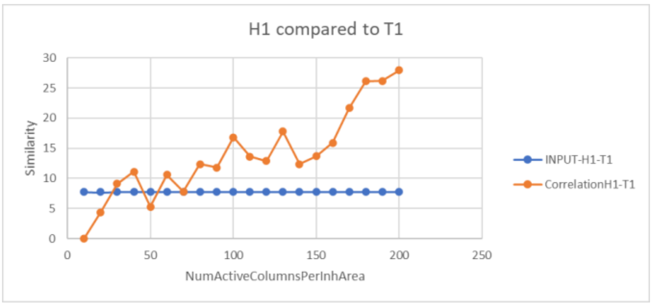


Figure 16 Comparing Hexagon1 to Triangle2, varying the NumActiveColumnsPerInhArea from 10 to 200 (with Local Area Inhibition)

In figure 17 When we compare Hexagon1 to Triangle1 as the NumActiveColumnsPerInhArea is increased, the similarity is also increased. This is not a good outcome. It means HTM considers pictures of different categories more and more similar to each other as NumActiveColumnsPerInhArea is increased, which is not desired. Specially between NumActiveColumnsPerInhArear of 150 and 190 that the similarity increases with a higher rate.

2. Using localAreaDensity (value of NumActiveColumnsPerInhArea must be set to a negative value) for different PotencialRadiuses Figure 18.

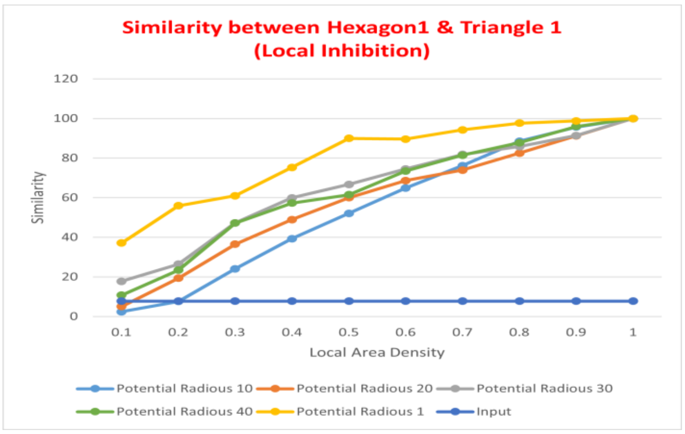


Figure 17 similarity between Hexagon 1 and Triangle1 (with Local Area Inhibition)

Here because the comparison is between 2 pictures of different categories, we expect a low SDR similarity after training the Spatial Pooler. Potential Radius 10 and Potential Radius 20 have the lowest similarity compared to other experiment, so they possibly can satisfy the desired expectation.
In this comparison by increasing the localAreaDensity, Similarity of images from different categories will increase.

3. Using localAreaDensity and GlobalInhibition (value of NumActiveColumnsPerInhArea has to be set to a negative value) and GlobalInhibition set to "True" Figure 19

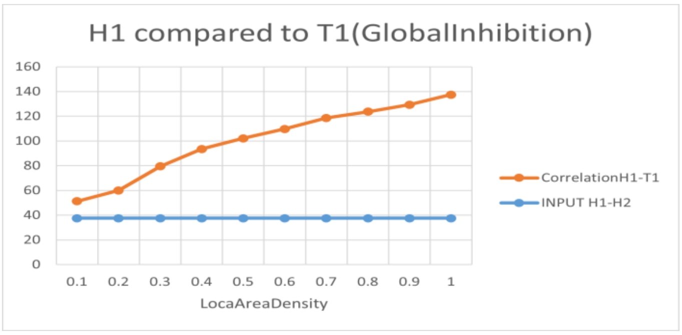


Figure 18 Comparing Hexagon1 to Triangle 1, varying the localAreaDensity from 10 to 100 (with Global Inhibition)

By increasing the localAeaDensity the H1 and T1 are looking more and more similar for the system, though the similarity growth in the beginning is not as high as the similarity growth in last part (comparing H1-H2) this means the higher localAeaDensity values couldn't be a good choice, as the system sees different shapes similar to each other.

Micro and Macro similarity of Hexagon category:

In this part the results of Macro and Micro similarities of Hexagon category are discussed and is tried to recognize the best parameters for an acceptable outcome, where the micro similarities are acceptably higher than macro ones.

1. Using NumActiveColumnsPerInhArea parameter, (value of localAreaDensity must be set to a negative value)

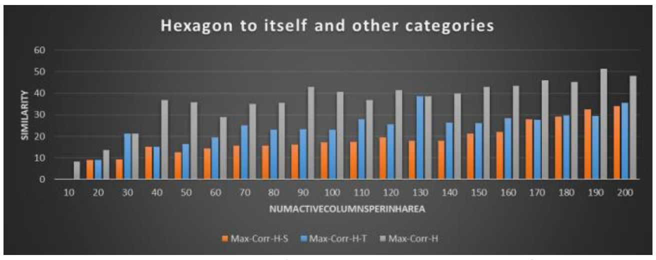


Figure 19 the Max-similarity of Hexagon category to itself compared to Max-similarity of Hexagon to other categories varying NumActiveColumnsPerInhArea from 10 to 200 (with Local Area Inhibition)

Figure 20 comparing the max-micro-similarity of Hexagon category to max-macro-similarity between Hexagon category and other categories, could be a good approach to choose the best NumActiveColumnsPerInhArea parameter among 10-200. Because here we can have a clearer sight on how HTM system is functioning. For example, for NumActiveColumnsPerInhArea=10 the similarity is not that high, but the safe margin (the difference of Max-micro-Corr-H compared to Max-macro-Corr-H) is 0. But for example, as we consider NumActiveColumnsPerInhArea value equal to 30, the Max-Correlation of H to itself is increased to 22% where Max Correlation of Hexagon to Triangle is also 22%, this means that due to a safe margin of 0, the system in this point, is not reliable at all.

So, the reliability factors could be a high similarity of a category members to themselves, and lowest possible similarity to other category members. As it can be seen in NumActiveColumnsPerInhArea=90,150,170 where Max-micro-Corr-H is interestingly high and Max-macro-Corr-H, is sufficiently low, the most sufficient value of NumActiveColumnsPerInhArea can be observed.

2. Using localAreaDensity (value of NumActiveColumnsPerInhArea must be set to a negative value).

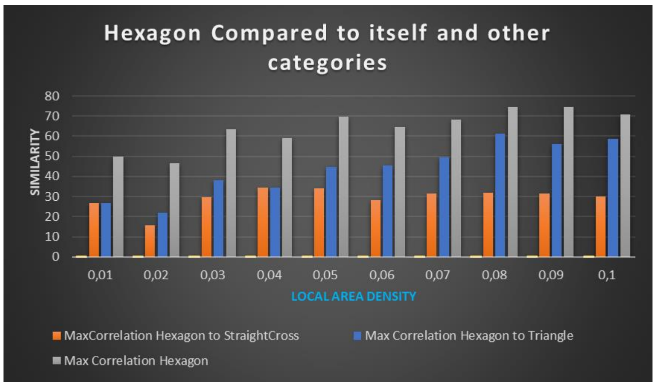


Figure 20 the Max-similarity of Hexagon category to itself compared to Max-similarity of Hexagon to other categories varying localAreaDensity from 0.01 to 0.1 (with Local Area Inhibition)

As can be seen in figure 21 this experiments were done comparing the max-micro-similarities of Hexagon category to max-macro-similarity between Hexagon category and other categories, by varrying localAreaDensities from 0.01 to 0.1 . The best results are the ones which have the highest similarity of Hexagon to itself (High Micro-similarity) and lowest similarity between Hexagon to others (low Macro-similarity). It seems Local area density = 0.02 can output the best result.

3. Using localAreaDensity (value of NumActiveColumnsPerInhArea must be set to a negative value) and GlobalInhibition set to "True" Figure 22.

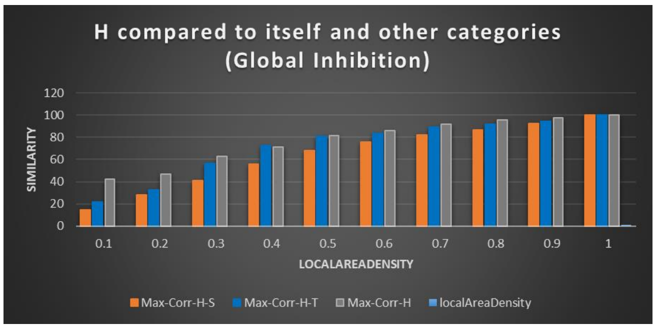


Figure 21 the Max-similarity of Hexagon category to itself compared to Max-similarity of Hexagon to other categories varying localAreaDensity from 0.1 to 1 (with Global Inhibition)

Though the increment of localAeaDensity leads to higher similarities of hexagon to itself, it’s also increasing similarity of other categories to hexagon too. This means a lower reliability in higher localAeaDensity values. In this case the best amounts of localAeaDensity are 0.1 and 0.2 where an almost good safe margin is observable.

choosing most sutible parameters according to Experiments:

considering the above experiments, the most efficient PotecialRadius can be considered as 20. The results of experiments where the GlobalInhibition is activated, are not satisfying, so it’s decided to take advantage of localAreaInhibition mode. Comparing the macro-micro similarities of Hexagon category which are observable in part macro micro vase numActive va localInhibition, the localAreaInhibition parameter set to 0.02 can be the best option to get the highest micro and lowest macro similarities.
The final chosen parameters are shown in a border below.

Parameters	**Default value**
InputDimensions	(64, 64)
ColumnDimensions	(32,32)
PotentialRadius	20
GlobalInhibition	False
LocalAreaDensity	0.02
NumActiveColumnsPerInhArea	-1

Border 1
final chosed parameters

Setting the parameters of border 1 to HTM system, and inputing the following test images in figure 1, the output results are shown in figure 25.

For choosing the mentioned parameters in the border 1 the training dataset wasmore restricted, where each category was containing four different images of that category.

Now to be able to observe the more generall functionality with more diverse training dataset, For training the HTM system we have chosen a wider range of images for each category, to be able to see how system functions when the traning dataset is expanded. Each category contains 10 images of that category.

The test input images which are expected to be classified by the HTM after it’s trained are shown in figure 23.

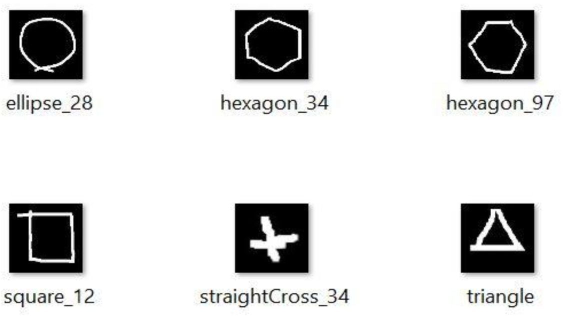
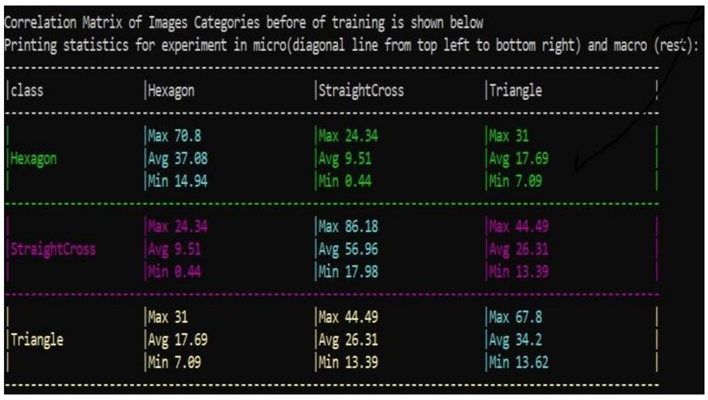


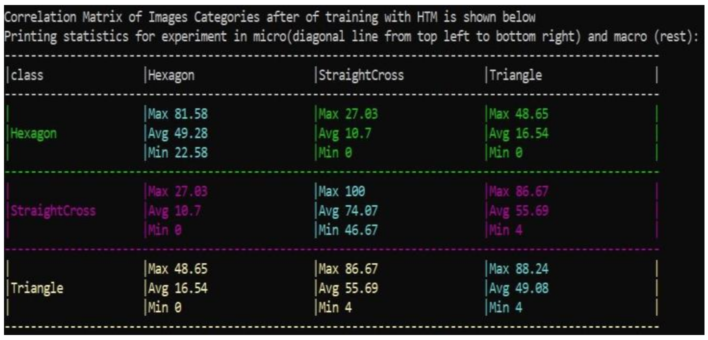
Figure 22 Input test images given to HTM



```
Correlation Matrix of Images Categories before of training is shown below
Printing statistics for experiment in micro(diagonal line from top left to bottom right) and macro (rest):
------------------------------------------------------------------------------------------
|class              |Hexagon                |StraightCross          |Triangle              |
------------------------------------------------------------------------------------------
|                   |Max 70.8               |Max 24.34              |Max 31                |
|Hexagon            |Avg 37.08              |Avg 9.51               |Avg 17.69             |
|                   |Min 14.94              |Min 0.44               |Min 7.09              |
------------------------------------------------------------------------------------------
|                   |Max 24.34              |Max 86.18              |Max 44.49             |
|StraightCross      |Avg 9.51               |Avg 56.96              |Avg 26.31             |
|                   |Min 0.44               |Min 17.98              |Min 13.39             |
------------------------------------------------------------------------------------------
|                   |Max 31                 |Max 44.49              |Max 67.8              |
|Triangle           |Avg 17.69              |Avg 26.31              |Avg 34.2              |
|                   |Min 7.09               |Min 13.39              |Min 13.62             |
------------------------------------------------------------------------------------------
```

Figure 23 the macro and micro similarity before training the system (Input Similarity)

In figure 24 the macro and micro similarities are shown, before training the system. Here we can see how similar the binarized outputs of dataset images look to each other.



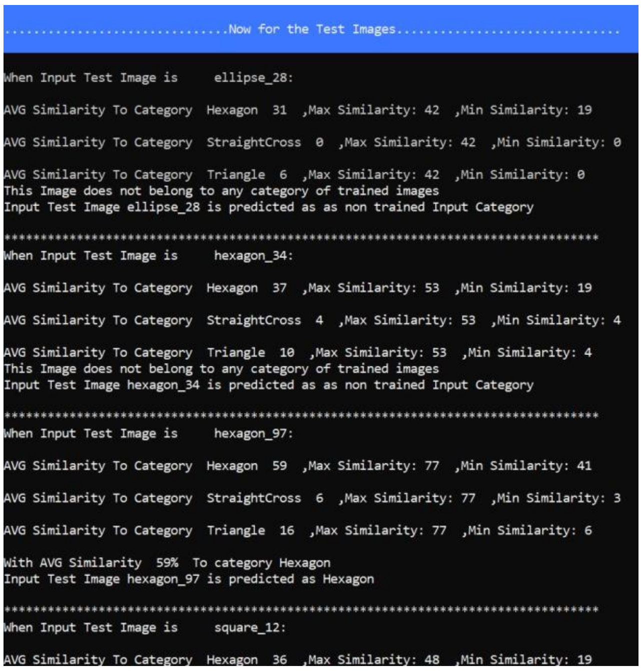
```
Correlation Matrix of Images Categories after of training with HTM is shown below
Printing statistics for experiment in micro(diagonal line from top left to bottom right) and macro (rest):
------------------------------------------------------------------------------------------
|class              |Hexagon                |StraightCross          |Triangle              |
------------------------------------------------------------------------------------------
|                   |Max 81.58              |Max 27.03              |Max 48.65             |
|Hexagon            |Avg 49.28              |Avg 10.7               |Avg 16.54             |
|                   |Min 22.58              |Min 0                  |Min 0                 |
------------------------------------------------------------------------------------------
|                   |Max 27.03              |Max 100                |Max 86.67             |
|StraightCross      |Avg 10.7               |Avg 74.07              |Avg 55.69             |
|                   |Min 0                  |Min 46.67              |Min 4                 |
------------------------------------------------------------------------------------------
|                   |Max 48.65              |Max 86.67              |Max 88.24             |
|Triangle           |Avg 16.54              |Avg 55.69              |Avg 49.08             |
|                   |Min 0                  |Min 4                  |Min 4                 |
------------------------------------------------------------------------------------------
```

Figure 24 the macro and micro similarity after training the system (Output Similarity)

In figure 25 the macro and micro similyrities of images are shown, after training the system. Micro similariies are acceptably high enough, which means the trained HTM finds images of same categories, similar to eachother. Macro similarities are also acceptable, except in the macro similarity of Triangle and straightCross. There is no overlaps happening here but this macro similarity of mentioned categories is unwantedly high.

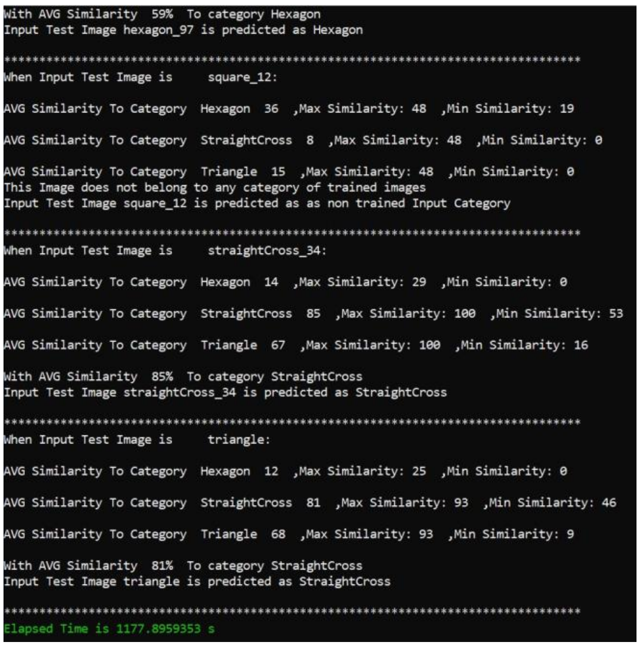
Prediction code which we have implemented takes the input test image, binarizes it then, without the system learns the properties of input test image, it is spatial pooled into the system. Afterwards the SDR of input test image will be compared to SDRs of images with wich the system has been trained.

The prediction results for each input test image is shown below in figure 26 and 27.



```
..............................Now for the Test Images..............................

When Input Test Image is     ellipse_28:

AVG Similarity To Category  Hexagon  31  ,Max Similarity: 42  ,Min Similarity: 19

AVG Similarity To Category  StraightCross  0  ,Max Similarity: 42  ,Min Similarity: 0

AVG Similarity To Category  Triangle  6  ,Max Similarity: 42  ,Min Similarity: 0
This Image does not belong to any category of trained images
Input Test Image ellipse_28 is predicted as as non trained Input Category

**********************************************************************************
When Input Test Image is     hexagon_34:

AVG Similarity To Category  Hexagon  37  ,Max Similarity: 53  ,Min Similarity: 19

AVG Similarity To Category  StraightCross  4  ,Max Similarity: 53  ,Min Similarity: 4

AVG Similarity To Category  Triangle  10  ,Max Similarity: 53  ,Min Similarity: 4
This Image does not belong to any category of trained images
Input Test Image hexagon_34 is predicted as as non trained Input Category

**********************************************************************************
When Input Test Image is     hexagon_97:

AVG Similarity To Category  Hexagon  59  ,Max Similarity: 77  ,Min Similarity: 41

AVG Similarity To Category  StraightCross  6  ,Max Similarity: 77  ,Min Similarity: 3

AVG Similarity To Category  Triangle  16  ,Max Similarity: 77  ,Min Similarity: 6

With AVG Similarity  59%  To category Hexagon
Input Test Image hexagon_97 is predicted as Hexagon

**********************************************************************************
When Input Test Image is     square_12:

AVG Similarity To Category  Hexagon  36  ,Max Similarity: 48  ,Min Similarity: 19
```

Figure 25 prediction results regarding HTM Paramters. For each test input image, it's Max, Min and Aerage similarity to each category is shown



```
With AVG Similarity  59%  To category Hexagon
Input Test Image hexagon_97 is predicted as Hexagon

**********************************************************************************
When Input Test Image is     square_12:

AVG Similarity To Category  Hexagon  36  ,Max Similarity: 48  ,Min Similarity: 19

AVG Similarity To Category  StraightCross  8  ,Max Similarity: 48  ,Min Similarity: 0

AVG Similarity To Category  Triangle  15  ,Max Similarity: 48  ,Min Similarity: 0
This Image does not belong to any category of trained images
Input Test Image square_12 is predicted as as non trained Input Category

**********************************************************************************
When Input Test Image is     straightCross_34:

AVG Similarity To Category  Hexagon  14  ,Max Similarity: 29  ,Min Similarity: 0

AVG Similarity To Category  StraightCross  85  ,Max Similarity: 100  ,Min Similarity: 53

AVG Similarity To Category  Triangle  67  ,Max Similarity: 100  ,Min Similarity: 16

With AVG Similarity  85%  To category StraightCross
Input Test Image straightCross_34 is predicted as StraightCross

**********************************************************************************
When Input Test Image is     triangle:

AVG Similarity To Category  Hexagon  12  ,Max Similarity: 25  ,Min Similarity: 0

AVG Similarity To Category  StraightCross  81  ,Max Similarity: 93  ,Min Similarity: 46

AVG Similarity To Category  Triangle  68  ,Max Similarity: 93  ,Min Similarity: 9

With AVG Similarity  81%  To category StraightCross
Input Test Image triangle is predicted as StraightCross

**********************************************************************************
Elapsed Time is 1177.8959353 s
```

Figure 26 prediction results regarding HTM Paramters. For each test input image, it's Max, Min and Aerage similarity to each category is shown

- The shape ellipse-28 which doesn't belong to any of the categories which the system is trained with is described as "does not belong to any categoriy of trained images"
- The shape hexagon-34 is found 41% similar to hexagon category. This similarity is actualy low, but we expected it, because the shape hexagon-34 compared to other hexagon images of traning dataset,is a little bit rotated. We didn't expect the system to despite this rotation, recognize, that this shape belongs to hexagon category with a high similarity percentage.
- The shape hexagon-97 is seen as hexaon, with a similarity of 60%, Which was expected.
- The shape quare-12 doesn't belong to any of the categories which the system is trained with is described az "does not belong to any categoriy of trained images".
- The shape StraightCross-34, as it's been expected, is recognized as a StraightCross category member, with a similarity of 76%.
- The shape triangle is also as expected, predicted correctly as a member of Triangle category whith an acceptable similarity of 75%.

IV. DISCUSSION

The goal of this project was investigating a new learning algorithm mimicking the neocortex of human for classifying labeled images. If we consider an infant from the beginning, they start to learn based on unlabeled data. Even in the future most of the input data from the world are uncategorized. But in this project for training the model we used a categorized set of images. Therefore, training the spatial pooler with uncategorized pictures will be a research topic for future works. Besides, in human brain only 2% of the cells are on at any given time. And the same was also observable in the experiments, where the best macro and micro similarities happen, when the localAreaDensity parameter is set to 0.02. Very high locaAreadensity values are not recommended, because as locaAreadensity is increased, meaning increase of active columns per inhibition area, system finds all types of inputs more and more similar to each other. This is not desired.

Generally, changing PotencialRadius and localAreaDensity values, cause more significant changes in the output result and that's the reason why in this paper it's tried to configure the HTM parameters, according to these two.

For future efforts, Same investigation about HTM parameters with a larger dataset of hand drawn input images of shapes, which also contains rotated shapes by not significantly high angels (15° - 45°) of rotation, could be interesting.

COMMITMENT

The commitments of each member in this experiment can be found at [19].

References

[1] J. H.-S. Blakeslee, On intelligence. Henry Holt and Company, New York, 2005.

[2] M. Taylor, "Numenta.com," Numenta, 17 may 2018. [Online]. Available: https://numenta.com/company/events/ai-singapore-meetup-numenta/.

[3] S. Š. a. I. Bajla, "On the Optimum Architecture of the Biologically Inspired Hierarchical Temporal Memory Model Applied to the Hand-Written Digit Recognition," *MEASUREMENT SCIENCE REVIEW*, vol. 10, 2010.

[4] D. v. E. D. Felleman, "Distributed hierarchical processing in the primate cerebral cortex. Cerebral Cortex," *Distributed hierarchical processing in the primate cerebral cortex. Cerebral Cortex* , p. 47, 1991.

[5] D. v. E. D. Felleman, "hierarchical processing in the primate cerebral cortex. Cerebral Cortex," *hierarchical processing in the primate cerebral cortex. Cerebral Cortex* , 1991.

[6] J. H. Subutai Ahmad, "Properties of sparse distributed representations and their application to hierarchical temporal memory," *Properties of sparse distributed representations and their application to hierarchical temporal memory,* 2015.

[7] A. NAZEER. [Online]. Available: https://www.kaggle.com/datasets/abdurrahumaannazeer/handdrawnshapes.

[8] E. N.-A. I. Panov, "Hierarchical Temporal Memory with Reinforcement Learning," *Hierarchical Temporal Memory with Reinforcement Learning,* 2020.

[9] J. Hawkins, "Hierarchical Temporal Memory (HTM) Whitepaper," *Hierarchical Temporal Memory (HTM) Whitepaper,* 2011.

[10] B. Bobier, "Handwritten Digit Recognition using Hierarchical Temporal Memory,," *Handwritten Digit Recognition using Hierarchical Temporal Memory,,* 2007.

[11] J. M.-E. F.-D. Kudithipudi, "A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler," *A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler,* 2017.

[12] G. M. Edelman and V. B. Mountcastle, The Mindful Brain: Cortical Organization and the Group-Selective Theory of Higher Brain Function, MIT Press, 198.

[13] R. S.-T. L.-A. A.-J. Machado, "Hierarchical Temporal Memory Theory Approach to Stock Market Time Series Forecasting," *Advances in Public Transport Platform for the Development of Sustainability Cities,* 2021.

[14] J. M.-E. F.-D. Kudithipudi, "A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler," *Frontiers,* 207.

[15] Numenta, "Numenta.org," Numenta, [Online]. Available: https://nupic.docs.numenta.org/stable/api/algorithms/spatial-pooling.html.

[16] Tavish, "www.analyticsvidhya.com," analyticsvidhya, 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/05/alternative-deep-learning-hierarchical-temporal-memory-htm-unsupervised-learning/.

[17] S. A.-Y. Cui, "The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding," *Frontiers in Computational Neuroscience,* 2017.

[18] F. W. F. R. A. K. Benedikt Feldotto, "Hebbian learning for online prediction, neural recall and classical conditioning of anthropomimetic robot arm motions," *Bioinspir Biomim,* 2018.

[19] M. Pirmoradian and O. Nikbakht, "Analyse Image Classification(Hand Drawn Shapes)," 2022. [Online]. Available: https://github.com/MahdiehPirmoradian/neocortexapi-classification/tree/main.