Information Technology Course
Master Thesis
by Damir Dobric / Andreas Pech /
Armin Lehmann

# Hierarchical Temporal Memory Image Classification

Dasu Sai Durga Sundari
1179354

**Abstract— Hierarchical temporal memory a unique approach to artificial intelligence that starts from the neuroscience of the neocortex. In this paper, we explore the feasibility of the Memory Prediction Theory, implemented in the form of a Hierarchical Temporal Memory (HTM), for Image classification. The primary goal of this thesis is to understand how image classification works and provide a simple image classification class in C# using Neocortex API. The secondary goal is to investigate how the similarity of images is effected with change in various configuration parameters.**

***Keywords— Hierarchical Temporal Memory (HTM), Spatial Pooler, Image classification, Neo Cortex API, Sparse Distributed Representation, Local Area Density, Potential Radius.***

## 1. INTRODUCTION

Hierarchical Temporal Memory (HTM) is a machine learning technique that simulates the functions of the human brain than many traditional ANN models such as Self- organizing Feature Mapping (SOFM) and Back-Propagation (BP). Hierarchical Temporal Memory (HTM) mimics the neocortex's structural and algorithmic properties. It can be thought of as a non-programmed memory system that is trained by exposing it to data flow. The training process is like how humans learn, which is essentially about identifying latent causes in newly acquired information. The HTM has no prior knowledge of the data stream causes it investigates at first, but it learns about them and incorporates them into its structure through a learning process. When all the data's latent causes have been captured and are stable, the training is considered complete. HTM has hierarchical structure, as shown in Figure 1 with a 3-level HTM structure.

Natural language processing (NLP), sensorimotor, anomaly detection, classification and prediction can all benefit from the neurally inspired HTM theory, which can be used to develop models with low computational complexity. It consists of two core parts: HTM Spatial Pooler (SP) for creating sparse distributed representations (SDR) of real-world sensory input or synthetic sensory-like data, as well as an HTM Temporal Pooler (TP) element for making predictions on the SDRs created by the HTM Spatial Pooler.

Image classification or Image recognition is a part of computer vision and a process to detect or identify an object or an attribute in a digital video or image. In general, image classification is one of the tasks that people excel at while computers struggle.

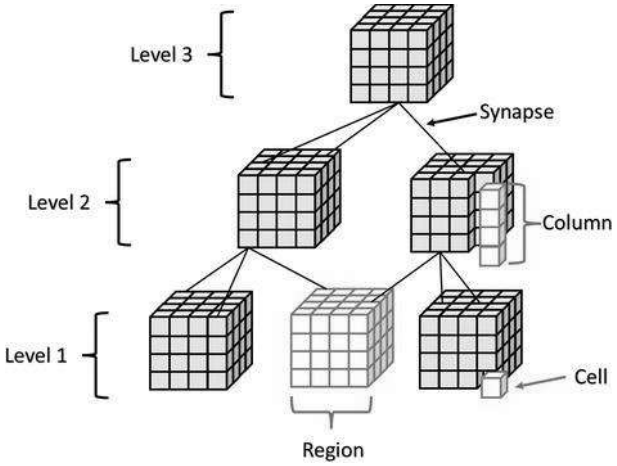



*Figure 1: 3-level hierarchical structure of HTM. Source* [1]

The feature extractor is an important part of an image classification system since it may discover relevant information that can be matched with another image. The focus of this thesis is to comprehend how Image classification functions using Hierarchical Temporal Memory which in turn uses Spatial Pooler for learning and classifying visual data.

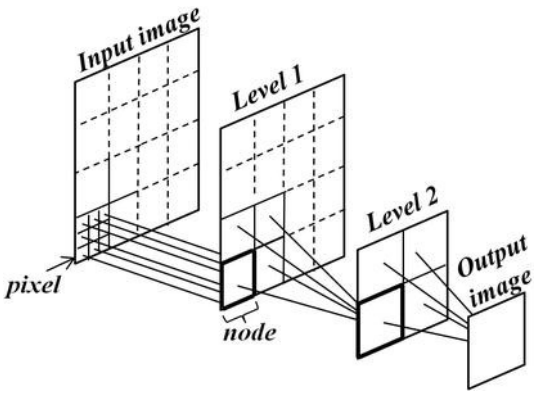



*Figure 2: HTM hierarchy based on the image recognition example. Source* [1]

Figure 2 shows how the hierarchy concept could be applied to an image recognition application. The single

level is represented by a single region made up of several nodes, with the number of nodes decreasing as we progress from bottom to top. The fact that we combine certain elements to form features as we go up could explain the lower number of nodes in the higher levels. Consider the input image, which depicts a human face.

By looking at the image and applying the top-down method, we can see that it portrays a human face, which has various distinguishing features such as a nose, eyes, and lips. We can then consider the iris and pupil, which are visible parts of the eye. Finally, we may zoom in on the image and notice that particular pixels contribute to the image's characteristics. As can be seen, we have access to varying levels of detail of the input information at different HTM levels [1].

## 2. SPATIAL POOLER

Spatial Pooler is a core component of HTM. The spatial pooler Learning Algorithm is a neocortical-inspired learning algorithm for learning spatial patterns. Spatial pooler is an unsupervised machine learning algorithm which takes the encoder's SDR input (binary input) and generates a set of active columns. Similar spatial patterns represented as active neurons are grouped together by Spatial Pooler into highly sparse representations of cortical mini columns. The number of active mini-columns in each HTM region is often limited to 2 to 4 percentage of the total mini-columns, resulting in sparse distributed representation (SDR).

Spatial pooler's primary function in HTM is to find spatial patterns in the input data. This process can be broken down into four distinct phases: (i) Initialization (ii) Overlap (iii) Inhibition (iv) Learning.

## 3. METHODOLOGY

Implementation of this project is done using C# .NET Core in Microsoft Visual Studio 2019 IDE (Integrated Development Environment). The goal of this project is to understand how image classification works and provide a image classification class which simplifies classification with HTM as shown in Figure 3. This section describes the algorithm for classification in detail and its implementation. binarize the input images before sending them to the HTM and the calculation of similarity are also described in this chapter.

In a typical HTM system, the Spatial Pooler takes input from an encoder and outputs to the TM. The primary objective is to transform the binary vectors into a semantically equivalent SDR with fixed sparsity. Following are the methods developed in this project and used in this project from Neocortex API.

- *ReadImageData*
- *HtmConfigSerializer*
- *CalculateSimilarity*

Image binarization is the process of turning a grayscale image into a black-and-white, thus reducing the image's information from 256 shades of gray to two: black and white, a binary image. The foreground color is applied to the image's objects, while the background color is applied to the rest of the image.

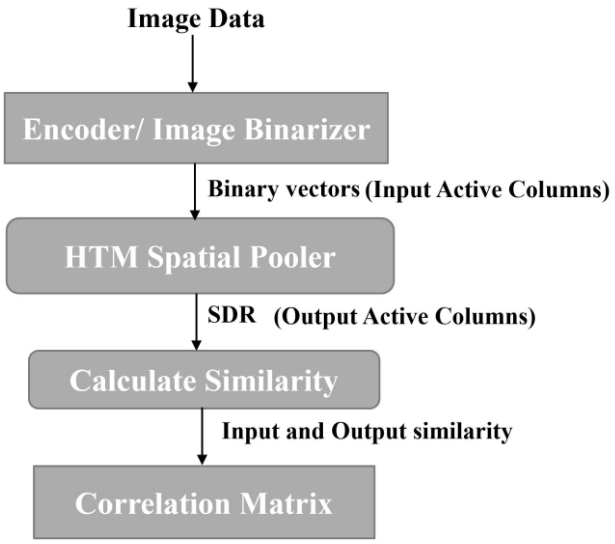



*Figure 3: HTM image classification working flow.*

### *A. Image Binarizer*

*ReadImageData* method reads the data of training image in binary form and returns binarized Image in integer array. Input vector having the data of images in binary form as shown in Figure 4.

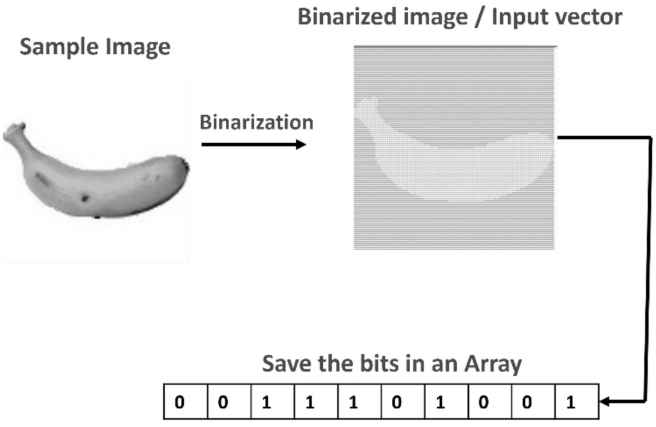



*Figure 4: Image Binarization Process*

### *B. Spatial Pooler*

After encoding the data into binary format, the next step is to send the encoded data to spatial pooling algorithm which requires two steps:

- Parameter configuration.
- Creating Spatial Pooler.

The HTM methods have several parameters that can be configured and choosing right parameters for experiments is very important. Below Table 1 contains a set of SP parameters

and their default values which are commonly used in most HTM experiments Separately, each of these variables have an impact on the performance of HTM, however, we will focus on the effect of potential radius (input variable PotentialRadius), local area density (input variable LocalAreaDensity) in this thesis work.

The first and most important step in using any algorithm is initialization of various parameters, which are defined by class HtmConfig (HTM configuration). After initializing, serialize the default configuration parameters to htmConfig.json file. Every time the user starts the HTM image classification program, the changed set of parameters are loaded. Table shows default parameters which are used in the htmconfig. json file.

| Parameters | Default value |
|---|---|
| InputDimensions | (100, 100) |
| ColumnDimensions | (64,64) |
| InhibitionRadius | 15 |
| PotentialRadius | 5 |
| PotentialPct | 0.5 |
| GlobalInhibition | False |
| LocalAreaDensity | -1 |
| NumActiveColumnsPerInhArea | 10 |
| StimulusThreshold | 0.0 |
| SynPermInactiveDec | 0.008 |
| SynPermActiveInc | 0.05 |
| SynPermConnected | 0.1 |
| DutyCyclePeriod | 100 |
| MaxBoost | 10 |

*Table 1: Default values of Spatial Pooler parameters.*

*1) Creating Spatial Pooler Instance*

After the parameters are initialized, the next step is to create the instance of SpatialPooler class. Currently, three versions of SP are implemented in this NeoCortex API [6].

- *SpatialPooler*: A single threaded original version without algorithm specific changes.
- *SpatialPoolerMT:* A multi-threaded version, which supports multiple cores on a single machine.
- *SpatialPoolerParallel:* Supports multicore and multimode calculus.

In this project, the single thread version of the spatial pooler is used for all the results presented in the Section results.

*2) Running the Spatial Pooler*

Once initiated, learning can be done by invoking of method *compute*. This is the primary public method of the *SpatialPooler* class. Each encoded row of data should be passed into the *compute()* method of the *SpatialPooler* instance. This function takes an input vector and outputs the indices of the active columns. If the flag 'learn' is set to True, this method also updates the permanences of the columns.

- **inputVector**: An array of 0's and 1's that comprises the input to the spatial pooler. The array will be treated as a one dimensional array, therefore the dimensions of the array do not have to match the exact dimensions specified in the class constructor.
- **activeArray**: An array whose size is equal to the number of columns. Before the function returns this array will be populated with 1's at the indices of the active columns, and 0's everywhere else.
- **learn**: A boolean value indicating whether learning should be performed. Learning entails updating the permanence values of the synapses, and hence modifying the 'state' of the model. Setting learning to 'off' freezes the spatial pooler and has many uses. For example, you might want to feed in various inputs and examine the resulting SDR's.
-

The argument *inputVector* which has input patterns that shows many data, grouped together into a common output representation that is *activeArray*.

*C. Calculate Similarity*

After training each image, the next step is to calculate the similarity, as a matrix, between active columns of input vectors and output SDRs of each image. This is done by using *CalculateSimilarityMatrix* method from *MathHelper* class in the API. The similarity matrix (correlation matrix) is calculated between the image files from the same image folder and between different image classes. For example, image11 vs. image 12 vs image 13 and image21 vs. image 22 vs image 23.

The correlation matrix inside of the same image class defines the micro-accuracy and the correlation matrix between different image classes is called macro-accuracy. Further, the similarity is saved in a CSV file (Correlation.csv) which can be read by excel.

## 4. EXPERIMENT

This section describes the performance of proposed architecture design. The overall flow of the HTM image classification system logic in this paper is shown in Figure 5 to validate HTMImageClassification method, different tests have been performed with a variety of images. In this thesis, tests are performed on datasets of fruits and shapes from Kaggle. A few hand drawn images are used to analyse how the similarity effects with the effect of parameters. The results of all the experiments presented in this work are done by using the local inhibition.

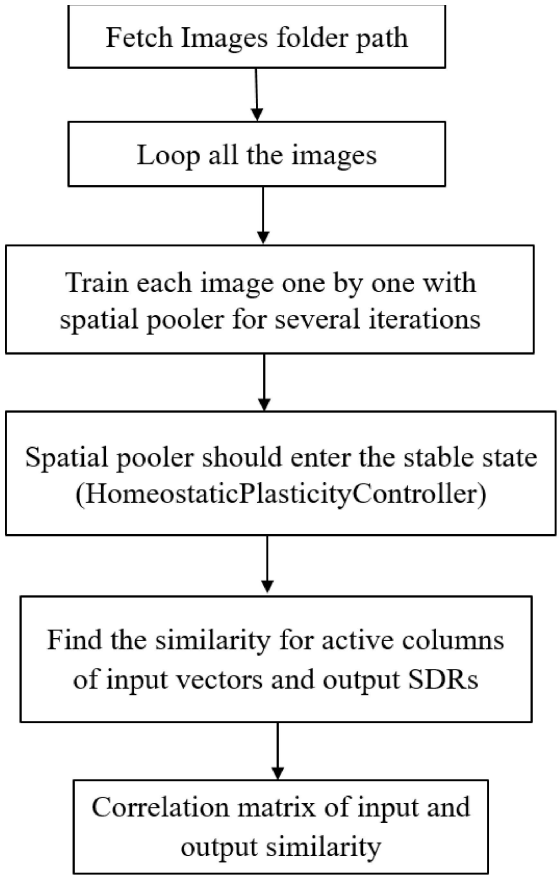



*Figure 5: HTM Program Logic*

### *D. Training*

For training process, images of 100x100 pixels have been taken from Fruits 360 dataset and images of 28x28 pixels from shapes dataset. During the training process, images are binarized into 0-1 vectors and used as input. In the initial phases of the experiments, the same instance of spatial pooler was used with default parameters from htmconfig.json file.

Every image is trained in several iteration steps specified by the argument iterationSteps. The images are trained until the spatial pooler enters a stable state which is controlled by the class *HomeostaticPlasticityController.* HomeostaticPlasticityController's (HPC) goal is to place the Spatial Pooler in a newborn state at the start of the learning process. The boosting is very active at this point, but the spatial pooler is instable. Once the SDR generated for each input becomes stable, the HPC will fire an event that notifies the code that spatial pooler is stable. The minimum number of cycles (iterationSteps) are automatically calculated by *trainingImagesPath.Length* × *newBornStageIterations.*

In the experiments conducted, the whole learning process took a minimum of 50 iterations depends on number of the images taken and the parameters in htmconfigjson file. Once the spatial pooler achieved stable state, the active columns of SDR are saved in a text file. The focus of this thesis is to investigate how spatial pooler parameters notably, ***local area density*** and ***potential radius*** effect the similarity with same image class and different image class. Images with less sparsity and images with different viewing angles of same object are also included in the experiments.

The first parameter, the experiments are made with is local area density, which controls the desired density of active columns within a local inhibition area. Alternatively, the ***numActiveColumnsPerInhArea*** parameter can be used to control the density. The ***localAreaDensity*** parameter must be less than 0 when using this method (*numActiveColumnsPerInhArea*). Within each local inhibition area, the inhibition logic will ensure that at most *numActiveColumnsPerInhArea* columns become active. We specify a density with the first parameter, so the actual number of active columns changes as the size of the receptive fields of the columns changes. The density will fluctuate as the receptive fields change with the latter parameter, but the maximum number of active columns will remain constant.

Next, the experiments are made with changing potential radius. Potential radius(***PotentialRadius***) parameter controls the input space to which a column might possibly connect—in other words, the column's receptive field. This number determines how far a column's effect spreads over the HTM layer.

### *E. Prediction*

Prediction follows the same procedure as training process with the same *compute* function, except now for prediction, the flag *learn* is set *false*. Prediction results (set of active columns) are compared with results collected during training process of spatial pooler. Further the best match is searched in correlation matrix and displays the image name with which it is matching including the percentage of similarity.

## 5. RESULTS AND DISCUSSION

The goal of the thesis is to investigate how similarity of the images (same image class and also different image class) varies with changing the parameters. Therefore, the first parameter, tests are performed is by changing ***local area density***. And also, further tests are performed to check the ***potential radius*** dependency on similarity. Throughout the experiments, the input dimensions are taken as 100 x 100 and columns dimensions as 64 x 64. Since, experiments are done with ***local area density, NumActiveColumnsPerInhArea*** parameter is taken as -1.

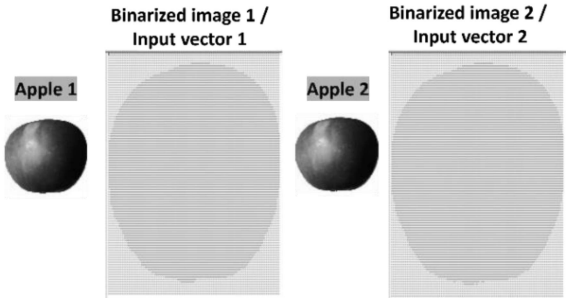
### *F. Same Image Class*

In the beginning, experiments are conducted on the same class of images with default Spatial Pooler parameters. Further the experiments are carried out by changing the parameters and investigated how it influences similarity.

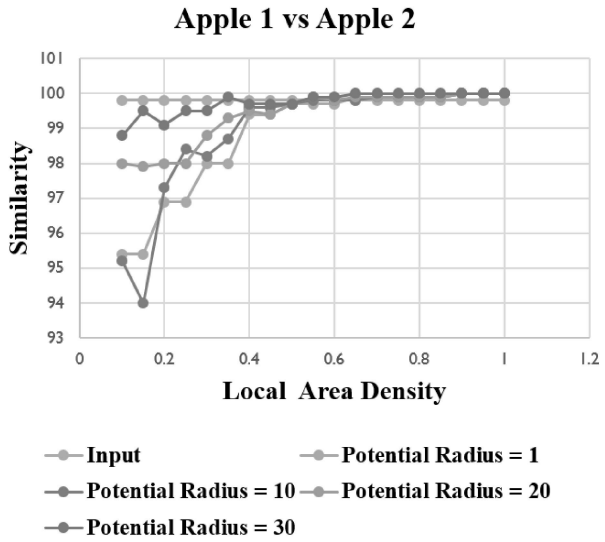
First four test cases show the images of same class with small changes and at different angles. Further test cases show similarity between images of different classes.

#### *1) Testcase 1*

Firstly, tests are performed using small data like two images of same class (two apples or two bananas). The Figure 5a shows two apple images with their respective binarized images.




*(a) Apple images and respective binarized images*




*(b) Input and Output similarity varying with local area density for potential radius of 1, 10, 20, 30*

***Figure 5**: Testcase 1 with same image class*

The first set of experiments are made with ***local area density***, which controls the desired density of active columns within a local inhibition area. Table 2 shows the parameter values which are used in Testcase 1.

| Parameters | Default value |
|---|---|
| InputDimensions | (100, 100) |
| ColumnDimensions | (64,64) |
| PotentialRadius | 1 |
| GlobalInhibition | False |
| LocalAreaDensity | 0.1 |
| NumActiveColumnsPerInhArea | -1 |

*Table 2: Parameters used in Testcase 1.*

Local area density value is varied from 0.1 to 1.0 in steps of 0.1 and the results are as follows. The similarity is always 100% irrespective of whatever the image is taken when local area density is above 1.0(e.g., 1, 2, 3, 10... so on) for any value of potential radius.

Table 2 shows how the similarity changes with the change in local area density, when the potential radius is 1.

| Local Area Density | Potential Radius | Input Similarity | Output Similarity |
|---|---|---|---|
| 0.1 | 1 | 99.8 | 95.4 |
| 0.15 | 1 | 99.8 | 95.4 |
| 0.2 | 1 | 99.8 | 96.9 |
| 0.25 | 1 | 99.8 | 96.9 |
| 0.3 | 1 | 99.8 | 98 |
| 0.35 | 1 | 99.8 | 98 |
| 0.4 | 1 | 99.8 | 99.4 |
| 0.45 | 1 | 99.8 | 99.4 |
| 0.5 | 1 | 99.8 | 99.7 |
| 0.55 | 1 | 99.8 | 99.7 |
| 0.6 | 1 | 99.8 | 99.7 |
| 0.65 | 1 | 99.8 | 99.9 |
| 0.7 | 1 | 99.8 | 99.9 |
| 0.8 | 1 | 99.8 | 99.9 |
| 0.85 | 1 | 99.8 | 99.9 |
| 0.9 | 1 | 99.8 | 100 |
| 0.95 | 1 | 99.8 | 100 |
| 1.0 | 1 | 99.8 | 100 |

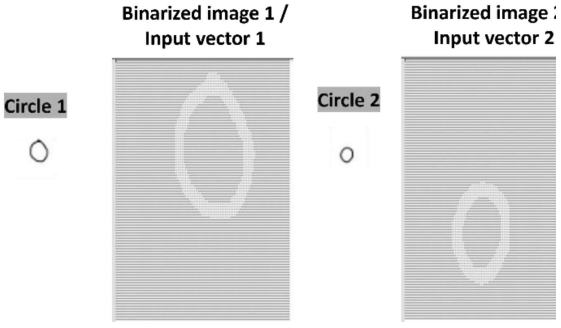
*Table 2: Input and output similarity with local area density for potential radius = 1*

Later, tests are performed with changing the potential radius from 10 to 30 in steps of 10. Figure 5(b) shows how similarity varies with respect to local area density and potential radius.

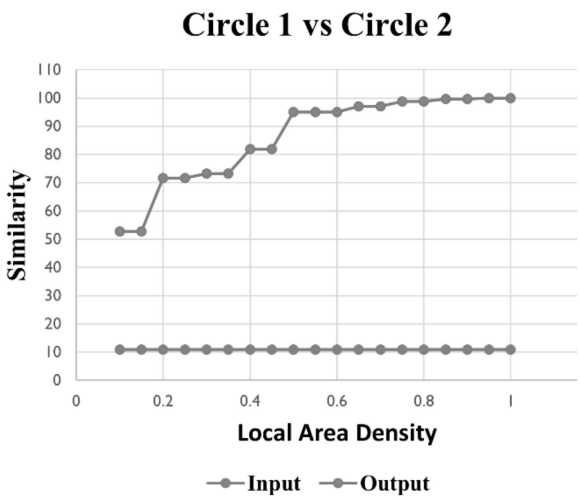
***Discussion***: As shown in the plot in Figure 5(b), as local area density increases, the similarity increases. In addition, we can observe that for potential radius 1, similarity is same for local area density value of 0.1 & 0.15, 0.2 & 0.25. 0.3 & 0.35 and so on, whereas the similarity changes for 0.1 and 0.15 for potential radius 10 and above. From the plot, we can conclude that for same class of images as shown in Figure 5(a) similarity is 100% when the local area density is 0.9 and above for potential radius 1, 10, 20. For a potential radius of 30 the similarity is 100% when local area density is above 0.65.

*2) Testcase 2*

In this testcase, experiments are done with images (28x28 pixels) that have less sparsity (less number of 1's) as shown in the Figure 6(a).



*(a) Testcase 2: Circle images with less sparsity*



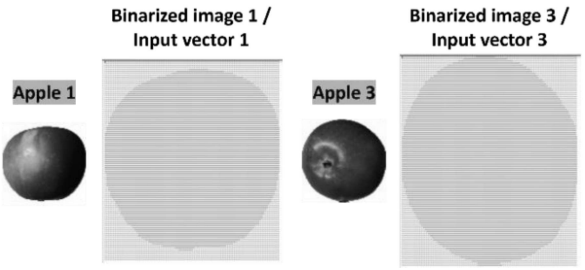

*(b) Output similarity with varying local area density for potential radius = 1*

***Figure 6**: Testcase 2 with less sparsity images*

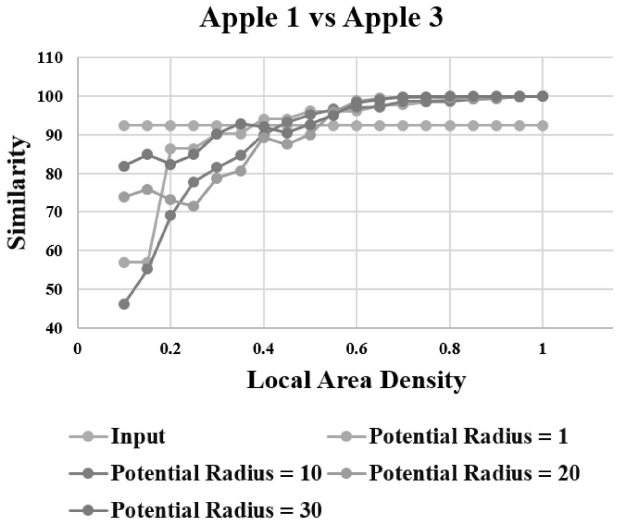
***Discussion*:** For the images with less sparsity as an example only one set of images are shown in Figure 6(a). From the Figure 6(b), we can see that the input similarity between Circle 1 and Circle 2 images is only 10.9%. Whereas the output similarity with local area density of 0.1, which is minimum value, is 52.8%. Even in this case both the images appear to be similar as local area density increases.

*3) Testcase 3*

In this testcase, images of the same apple with different viewing angles are taken. The Figure 7(a) shows apple images together with their binarized images. Figure 7b shows how similarity increases with increase in local area density. With a local area density value of 0.95 and above, the similarity is 100%, which means the images appear similar.




*(a) Same Image Class with different viewing angles*

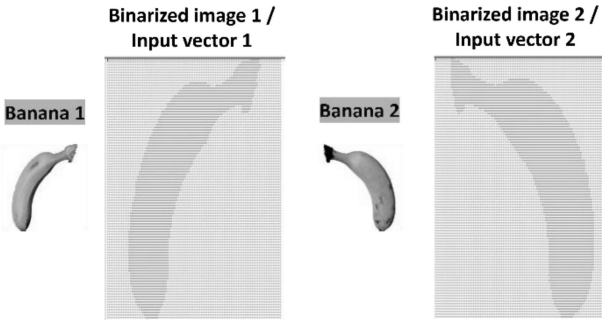



*(b) Input and Output similarity varying with local area density for potential radius of 1, 10, 20, 30*

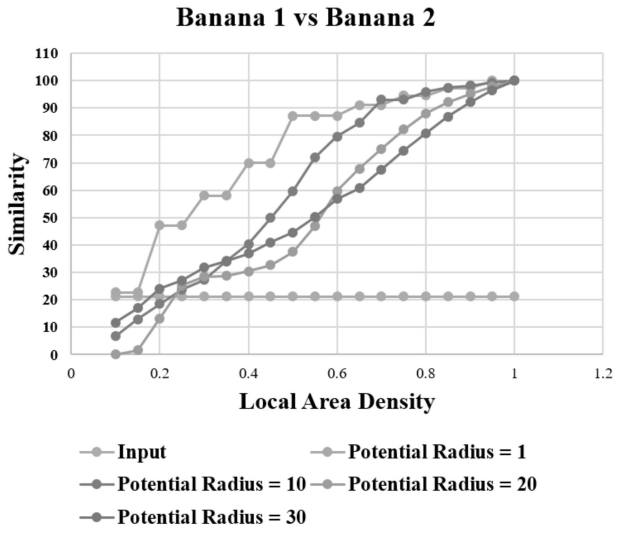
***Figure 7**: Testcase 3: Same Image Class with different viewing angles*

***Discussion***: As the potential radius is increased, the similarity also increased even for low local area density values. From Figure 7(b), we can observe that as potential radius is increased, similarity is more for less local area density. For local area density value of 1, Apple 1 and Apple 3 matches 57% for potential radius 1. Whereas similarity is 73.8% and 81.9% respectively for potential radius 20 and potential radius 30. For the same images with different viewing angles to appear similar, the local area density should be increased. The similarity is above 90% for the local area density value of 0.5 and above for all the values of potential radius.

*4) Testcase 4*




*(a) Same Image Class with different viewing angles*




*(b) Input and Output similarity varying with local area density for potential radius of 1, 10, 20, 30*

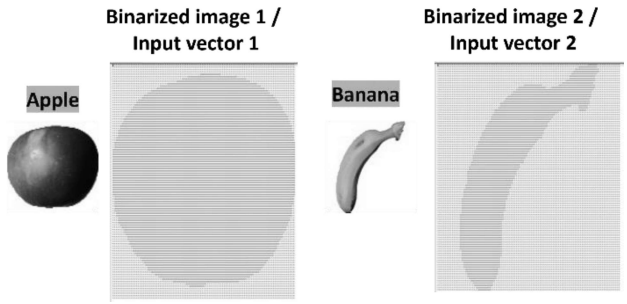
***Figure 8**: Testcase 4: Same Image Class with viewing angles*

***Discussion*:** Further, the images of bananas, which are mirror images are tested. Once the image is binarized, the similarity between two images is 21.2%. As the local area density increases, it can be seen from Figure 8(b) that, similarity between two images is above 87% for potential radius value of 0.5 and above. Similarity between two images is 0% when potential radius is 20 and as local area density increases, similarity between two images increases.

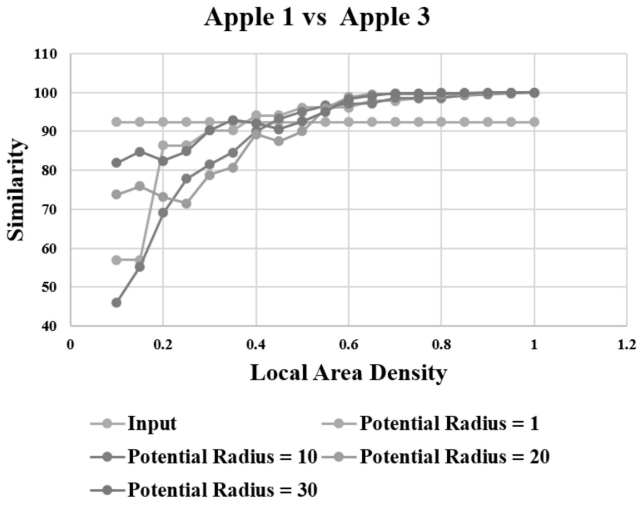
### *G. Different Image Class*

#### *1) Testcase 5*

Now as a final testcase, two images of different fruits, like Apple, Banana and Avocado are used for the experiments. In this test case, it shows how similar are apple and banana and how local area density and potential radius influences their similarity. The Figure 9(a) shows apple and banana image and their corresponding binarized images/input vectors.




*a) Testcase 5 : Fruits and respective binarized images*




*b) Input and Output similarity varying with local area density for potential radius of 1, 10, 20, 30*

***Figure 9:** Testcase 5 with different image class*

***Discussion:*** Images of different classes are tested during the work. However, as an illustration, results of only two image classes Apple vs Banana are mentioned in this paper. After the images are binarized, the number of active columns between Apple and Banana are 31.2%. After training, the number of active columns from SP for potential radius of 1 and local area density of 0.1 is 30%. For a potential radius value of 20 the similarity slightly increases and stands at 31.6%. It further increases for potential radius 30 to 32.2% When potential radius is 10, the output similarity is 3.4% for minimum value of local area density. As the density of active columns (local area density) increases, apple and banana appear to be similar.

### *H. Testing*

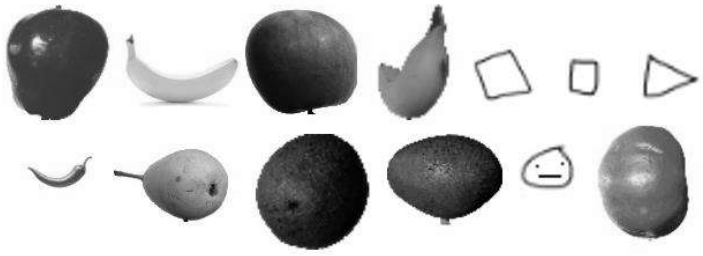


***Figure 10:** Prediction Images*

After training, the model is used to make a prediction of different images. The active columns of the prediction method are compared with the SDRs (active columns) saved in a text file during Spatial Pooler training process. For testing, several images like Apple, Avocado, Pear etc. are taken as shown in Figure 10.

The similarity of each image in prediction is calculated with active columns from the training. After searching for the best match in the correlation matrix it gives the matching image name and also similarity percentage.

Since, the images of fruits are used for training in this work, it predicts the images with the best match from training active columns. For the Apple images which are shown in Figure 10, with the local area density 0.1, potential radius as 1 and Apple and Banana images in the training folder, it predicts with the Apple 1 image in Figure 5(a) with an accuracy of 58.1%. Whereas, when the potential radius is 10, it predicts with Apple 3 image as shown in Figure 7(a) with a similarity of 63.2%.

Avocado images in the Figure 10, matches with the Apple image with an accuracy of 50% for local area density of 0.1 and potential radius 1. However, when the potential radius is 10, Avocado matches with Apples, with 17.5% similarity. For local area density 0.1 and potential radius of 1, Pear matches with Apple with an accuracy of 43.5% and for potential radius of 10, it matches only 11.5%.

Therefore, from the above prediction results, with the potential radius of 10 and local area density value of 0.1, the different class of images are less similar (similarity percentage is less) when compared to potential radius of 1.

## 6. CONCLUSION

The HTM theory aims to achieve true machine intelligence by advancing our understanding of how our brains work biologically. In this paper, Neocortex API is adapted and evaluated to the image classification problem with fruits dataset. In addition, it is also evaluated for the images with less sparsity. The results show how potential radius and local area density parameters influence the similarity. The similarity increases as the local area density is increased and images of same image class with different viewing angles and also different image class appears to be similar. It predicts the similarity of same images with viewing angles as similar with accuracy of 90% when local area density is increased. From the results, we can see that for local area density value of 1, the image matches 100% irrespective of whatever the image is taken.

From the discussion in the above section, we can conclude that if a large number of columns remain ON within a local area inhibition, the similarity between the images will be more and they appear to me similar. So, for the different class of images not to appear similar, the local area density must be as low as possible and for same class of images, even with different viewing angles, it should be as large as possible.
For future work, it would be rewarding to test with datasets containing colorful images. Since, colorful images cannot be trained using Image Binarizer which was used in this paper, we need an encoder which does this.

## 7. REFERENCES


[1] A. a. I. T. a. K. O. a. F. I. James, "Introduction to Memristive HTM Circuits," in *Memristor and Memristive Neural Networks*, 2018.

[2] D. Dobric, "Github," [Online]. Available: https://github.com/ddobric/neocortexapi.

[3] C. G. K. F. Z. Jahan Balasubramaniam, "Enhancement of Classifiers in HTM-CLA Using Similarity Evaluation Methods," *Procedia Computer Science,* 2015.

[4] W. a. C. Z.-G. a. Q. Y. a. Y. Z. a. X. Y. Zhuo, "Image classification using HTM cortical learning algorithms," *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012),* 2013.

[5] I. B. R. Škoviera, "Image Classification Based on Hierarchical Temporal Memory," *MEASUREMENT 2013, Proceedings of the 9th International Conference, Smolenice, Slovakia* , 2013.

[6] [Online]. Available: https://numenta.com/resources/biological-and-machine-intelligence/spatial-pooling-algorithm/.

[7] D. a. P. A. a. G. B. a. W. T. Dobric, "On the Relationship Between Input Sparsity and Noise Robustness in Hierarchical Temporal Memory Spatial Pooler," *ESSE 2020: Proceedings of the 2020 European Symposium on Software Engineering,* 2020.