# Analyse Image Classification of Fruits360 dataset

Anu Maria Varghese
anu.varghese@stud.fra-uas.de

Tiniya Vinod Puthanpurayil
tiniya.puthanpurayil@stud.fra-uas.de

Veena Alphonsa Jose
veena.alphonsajose@stud.fra-uas.de

***Abstract:*** **Image classification is the process of categorizing and labeling groups of pixels or vectors within an image based on specific rules. It is the most important part of digital image analysis. This paper aims to change various Hierarchical Temporal Memory (HTM) learning parameters and to find the best fit that shows the image classification and to demonstrate how these parameters influence learning. Our objective is to improve the input prediction and to implement the highest similarity of the input images. The HTM Cortical Learning Algorithm (HTM CLA) Spatial Pooler (SP) is a neocortical-inspired Cortical Learning Algorithm for learning. In this project, NeoCortex API focuses implementation of Hierarchical Temporal Memory Cortical Learning Algorithm in C#/.NET Core. Its purpose is to understand the spatial pattern by creating the input's Sparse Distributed Representation code (SDR). HTM is based on the biological functions of the brain as well as its learning mechanism. To achieve our goals, we have conducted various tests using Fruit360 dataset with the previously implemented image classification solution as a library and implemented the prediction code with highest similarity of input images.**

***Keywords: Spatial Pooler, Hierarchical Temporal Memory, Sparse Distributed Representation, neocortex.***

## I. Introduction

Over the years, and with the emergence of various technological innovations, the relevance of automatic learning methods has increased exponentially, and they now play a key role in society. This paper aims to change various learning parameters and to find the best fit that shows the image classification and to demonstrate how these parameters influence learning. Our objective is to improve the input prediction and to implement the highest similarity of the input images. HTM is based on the biological functions of the brain as well as its learning mechanism. HTM can be described as the theory that attempts to describe the functioning of the neocortex, as well as the methodology that intends to provide machines with the capacity to learn in a human way [1]. The neocortex is defined as the portion of the human cerebral cortex from which comes the highest cognitive functioning, occupying approximately half the volume of the human brain. The neocortex is understood by four main lobes with specific functions of attention, thought, perception, and memory. These four regions of the cortex are the frontal, parietal, occipital, and temporal lobes. The frontal lobe's responsibilities are the selection and coordination of behavior. The parietal lobe is qualified to make decisions in numerical cognition as well as in the processing of sensory information. The occipital lobe, in turn, has a visual function. Finally, the temporal lobe has the functions of sensory as well as emotional processing and dealing with all significant memory. Thus, the algorithm that is presented intends to create a transposition of this portion of the brain, creating a machine with true intelligence [2].

A HTM network is developed as a tree-shaped hierarchy. Although uncommon, multiple parents per node are allowed, therefore letting the receptive fields of the parents overlap. Its organization is like that of cortical mini columns, in which an HTM region is made up of numerous columns, each of which contains many cells. A level is made up of one or more regions. To build the whole network, levels are stacked hierarchically in a tree-like structure. Synapses are used to make connections in HTM, with both proximal and distal synapses being used to produce feedforward and adjacent connections, respectively as represented in Figure 1.

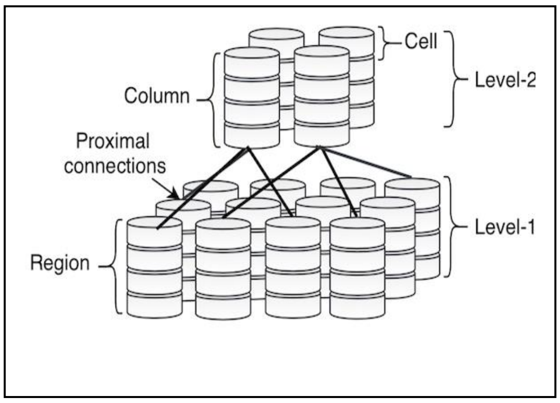


Figure 1:A depiction of two HTM regions arranged *hierarchically (two levels)*

Starting with the first region, the encoder deals with all the sensory components. This will receive the data in their raw form, converting them into a set of bits that will later be transformed into a Sparse

Distributed Representation (SDR). Transposing into the human organism, the SDRs correspond to the active neurons of the neocortex. Thus, a 1 bit represents an active neuron while a 0 bit represents an inactive neuron. This transformation is achieved by transforming the data into a set of bits while maintaining the semantic characteristics essential to the learning process. One of the characteristics that proved to be quite interesting is those similar data entries, when submitted to the encoding process, create overlapping SDRs; that is, with the active bits placed in the same positions. Another important characteristic is that all SDRs must have a similar dimensionality and sparsity (the ratio between the number of bits at 1 and the total number of bits) [3]. A certain percentage of sparsity will result in a system's ability to handle noise and under-sampling.

The second region, Spatial Pooler (SP), is responsible for assigning the columns according to a fixed number, where each column corresponds to a dendritic segment of the neuron that connects to the input space created by the region described above, the encoder. Each segment has a set of synapses that can be initialized at random, with a permanence value. Some of these synapses will be active (when connected to a bit with value 1) and consequently will be driven in such a way as to inhibit other columns in the vicinity. Therefore, the SP is responsible for creating an SDR of active columns. This transformation follows the Hebbian learning rule that for each input, the active synapses are driven by inhibiting the inactive synapses. The thresholds dictate whether a synapse is active or not.

Finally, the classifier is the region in which a decoder calculates the overlap of the predicted cells of the SDR obtained, selecting the one with more

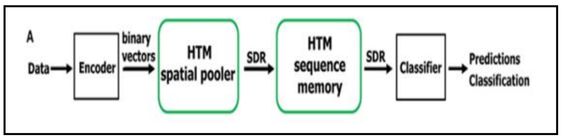


*Figure 2: HTM Pipeline*

overlaps and comparing it with the actual value (if known) [4] [5]. After initializing components, they are chained together in CortexLayer as HTM Module pipelines as shown in Figure 2Figure 2.

HTM is built on three main features of the neocortex: it is a memory system with temporal patterns and its regions are organized in a hierarchical structure. There are many biological details that the theory ignores in case they have no relevance for learning. In short, this approach includes Sparse Distributed Representation (SDR)s, its semantical and mathematical operations, and neurons along the neocortex capable of learning sequences and enabling predictions; these systems learn in a continuous way, with new inputs through time and with flows of information top-down and bottom-up between its hierarchical layers, making them efficient in detecting temporal anomalies. The theory relies on the fact that by mimicking the neocortex, through the encoding of data in a way that gives it a semantic meaning, activating neurons sparsely in an SDR through time will give these systems a power to generalize and learn, not achieved to date with other classic approaches of AI.

## II. Methods

### A) HTM

HTM is based on the biological functions of the brain as well as its learning mechanism. The results are of significant relevance and show a low percentage of errors in the predictions made over time. The Hierarchical Temporal Memory Cortical Learning Algorithm (HTM CLA) is a theory and machine learning technology that aims to capture the cortical algorithm of the neocortex [6].

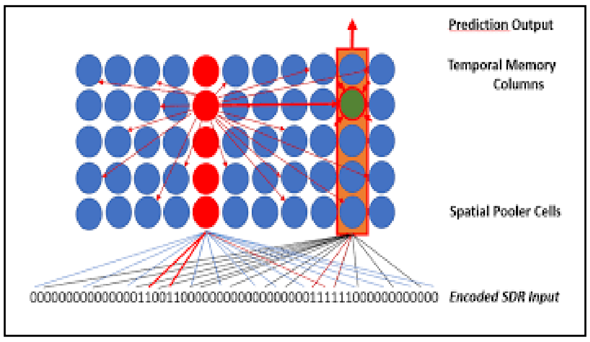


*Figure 3: Data Processing in HTM*

HTM consists of 2 different components: Spatial Pooler and Temporal Memory. Figure 3 depicts the principle network connectivity and data processing in the Hierarchical Temporal Memory. The HTM implements technical equivalents for sensory organs and learning of spatiotemporal sequences in the neocortex. Raw sensory data is processed by encoders and fed into the Cortical Learning algorithm consisting of Spatial Pooler and Temporal Memory cells.

### B) Sparse Distributed Representations (SDR)

An SDR is a vast array of bits with most of them turned off (0s) and only a few turned on (1s) (1s). Each SDR represents some meaning since two SDRs are judged to have equivalent meaning if they have several overlapping places on bits. The data is more comparable or the gap between two SDRs is smaller the more bits they share [6]. Figure 4 shows the SDR of the spatial temporal pooler. The blue cell represents the inactive cell while the yellow ones are the active cells that represent a particular input within a certain context.

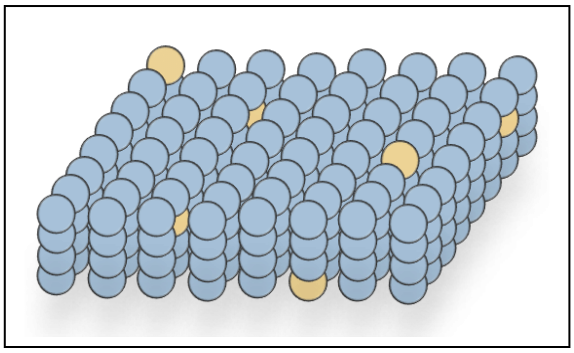


*Figure 4: SDR of the spatial temporal pooler*

## C) Spatial pooler

Spatial Pooler (SP) is a learning algorithm that is designed to replicate the neuron functionality of the human brain. Essentially, if a brain sees one thing multiple times, it is going to strengthen the synapses that react to the specific input result in the recognition of the object. Similarly, if several similar SDRs are presented to the SP algorithm, it will reinforce the columns that are active according to the on bits in the SDRs. If the number of training iterations is big enough, the SP will be able to identify the objects by producing different sets of active columns within the specified size of SDR for different objects [6].

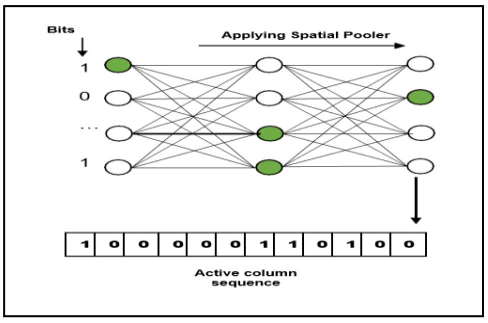


*Figure 5: Spatial Pooler Applying Process*

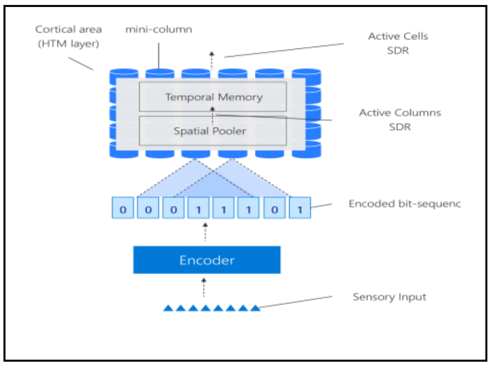


*Figure 6: Spatial Pooler and Temporal memory*

In the Figure 5 we can see after applying Spatial Pooler we get active column sequence. In the Figure 5 the green nodes indicate the active bits. We use spatial pooler to train our binarize data. Then we get an active column sequence and store it for further use. Figure 6 shows the columns in Spatial Pooler and Temporal memory.

## D) Encoder

Encoders are tools that convert data sources into SDRs without altering the semantic content. Each encoder will be assigned to a certain data type [5]. If the input data is blood pressure, for example, a blood pressure encoder is required. Encoder is chosen according to the type of the inputs. There are some encoders available for popular input type such as Scalar Encoder, Datetime Encoder, etc. A new encoder will be necessary if a new type of data is to be fed into the HTM system.

## E) Image Binarization

A binary image is a digital image that has only two uses for a binary image, black and white, through any two colors can be used. The color used for the objects in the image is the foreground color while the rest of the image is the background color [7].

- Get an image from a file.
- Create equivalent binary

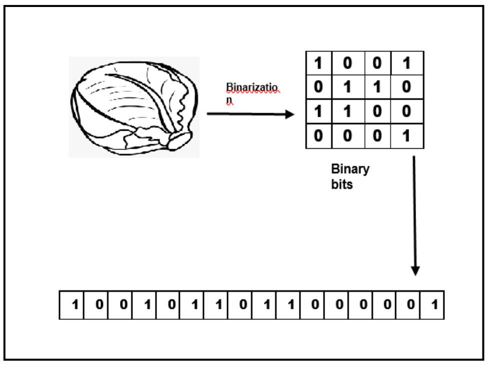


*Figure 7:Image Binarization Process*

Figure 7 demonstrates the process how an image converts into binary format. This is the process of taking a grayscale image and converting it to black and white, essentially reducing the information contained within the image from 256 shades of gray to 2: black and white, a binary image. This is sometimes known as image thresholding, although thresholding may produce images with more than 2 levels of gray. It is a form or segmentation, whereby an image is divided into constituent objects.

## F) Dataset

A dataset is a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes. The selection of a dataset as well as the features to be used may be determinant for the success of the research work. Datasets is not just a simple data repository. Each dataset is a community where discussion of data, discovery of public code and techniques, and creation of projects in notebooks. Dataset used for experiments are from Fruit 360 of Kaggle.

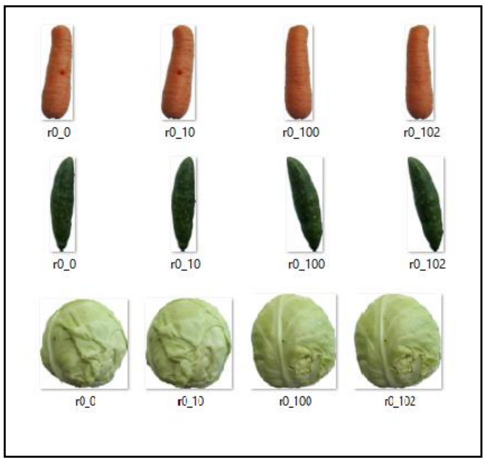


Figure 8: Dataset

Kaggle provides a vast container of datasets, sufficient for the enthusiast to the expert. Kaggle supports a variety of dataset publication formats, but dataset publishers encourage to share their data in an accessible, non-proprietary format if possible [8]. Not only are open, accessible data formats better supported on the platform, but they are also easier to work with for more people regardless of their tools. Figure 8 shows the datasets obtained from Kaggle.

## III. Results

In this section we have discussed the experiments we conducted with Fruit360 dataset.

| Parameter Name | Values |
|---|---|
| POTENTIAL_RADIUS | 30 |
| POTENTIAL_PCT | 0.5 |
| GLOBAL_INHIBITION | False |
| INHIBITION_RADIUS | 1 |
| LOCAL_AREA_DENSITY | 0.1 |
| NUM_ACTIVE_COLUMNS_PER_INH_AREA | -1 |
| STIMULUS_THRESHOLD | 0.0 |
| SYN_PERM_INACTIVE_DEC | 0.01 |
| SYN_PERM_ACTIVE_INC | 0.1 |
| SYN_PERM_CONNECTED | 0.1 |
| DUTY_CYCLE_PERIOD | 10 |
| MAX_BOOST | 10.0 |

*Table 1: HTM Input Parameters*

The experiments are performed to analyze the influence of HTM parameter in image classification and the given result shows the correlation validation between the input images. Also, we performed tests to check the image prediction after obtaining the best fit correlation matrix. New method called “PredictLabel” is implemented to predict the label of the image which is imputed by comparing the binarized values of the images. Figure 9 shows one of the binarized image of cabbage. For our experiments, we used the images of carrot, cucumber, and cabbage from Fruit360 dataset. Random images of each of these were taken as shown in Figure 8 and three folders were created with names same as carrot, cucumber, and cabbage inside the “Inputfolder”.

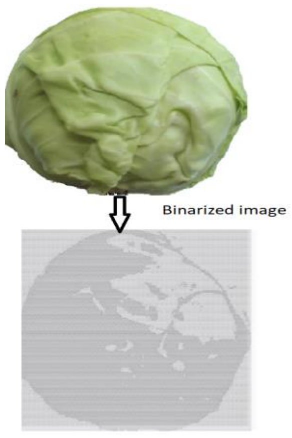


*Figure 9: Binarized image of cabbage*

### Case 1: By changing various HTM parameter to find the best fit correlation matrix

HTM setting of the project can be inputted to the program by means of a .json file htmconfig.json. Multiple experiments were therefore conducted by changing various HTM parameters in the json file. Every image is trained in several iteration steps specified by the argument iteration steps. The images are trained until the spatial pooler enters a stable state which is controlled by the class HomeostaticPlasticityController (HPC). The goal of HPC is to place the Spatial Pooler in a new-born state at the beginning 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 last set of Sparse Density Representations (SDRs), the output of Spatial Pooler (SP) for each binarized image were saved for correlation validation. There are 2 types of correlation which are defined as follows:

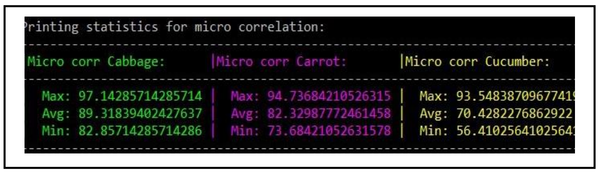
- Micro Correlation: Maximum/ Average/ Minimum correlation in similar bit percent of all images' SDRs which respect to each another in the same label.
- Macro Correlation: Maximum/ Average/ Minimum correlation in similar bit percent of all images' SDRs with images from 2 different labels. The results of the two correlations are printed in the command prompt when executing the code

During the learning process the number of cycles taken by spatial pooler to become stable were between 95 to 200 depending on the number of images in different learning classes and on the HTM parameters given in the htmconfig.json file. Most important learning parameters are: Global/Local Inhibition, Potential Radius, Local Area Density and NumofActiveColumnsPerInArea and we found how these parameters influenced learning. Table 1 shows the HTM parameters default values. After conducting various tests, we have been able to find the parameters at which we get the least overlapping in between Micro and Macro and thus the best correlation matrix. Local area density defines the number of active columns within a local inhibition area, we specify the desired density of active columns by changing LocalAreaDensity parameter in the htmconfig.json file. We have conducted experiments by varying the parameter PotentialRadius in the htmconfig.json file by selecting the values from among 10,20, 30 and 40. Later more experiments were also conducted by changing the InputDimensions from 100 to 500.

| Parameter Name | Values |
|---|---|
| InputDimensions | 200 |
| PotentialRadius | 40 |
| GLOBAL_INHIBITION | False |
| LocalAreaDensity | 0.3 |

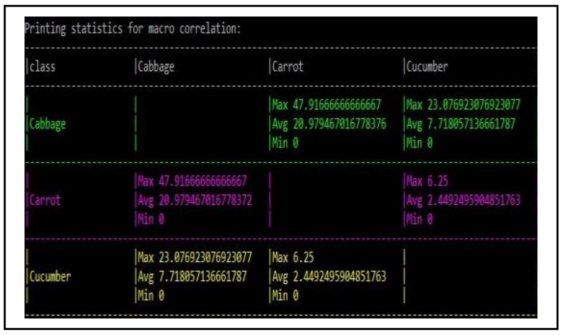
*Table 2: Parameters used for experiment*

Table 2 shows the parameter values which we used in the experiment where we obtained the output as shown in Figure 10 and Figure 11 which is the best fit correlation matrix.



```
rinting statistics for micro correlation:
------------------------------------------------------------------------------
Micro corr Cabbage:      |Micro corr Carrot:       |Micro corr Cucumber:
------------------------------------------------------------------------------
  Max: 97.14285714285714 |  Max: 94.73684210526315 |  Max: 93.5483870967741
  Avg: 89.31839402427637 |  Avg: 82.32987772461458 |  Avg: 70.4282276862922
  Min: 82.85714285714286 |  Min: 73.68421052631578 |  Min: 56.410256410256
------------------------------------------------------------------------------
```

*Figure 10: Output showing Micro Correlation*



```
Printing statistics for macro correlation:
------------------------------------------------------------------------------------------
|class      |Cabbage                  |Carrot                   |Cucumber
------------------------------------------------------------------------------------------
|           |                         |Max 47.9166666666667     |Max 23.076923076923077
|Cabbage    |                         |Avg 20.979467016778376   |Avg 7.718057136661787
|           |                         |Min 0                    |Min 0
------------------------------------------------------------------------------------------
|           |Max 47.91666666666667    |                         |Max 6.25
|Carrot     |Avg 20.979467016778372   |                         |Avg 2.4492495904851763
|           |Min 0                    |                         |Min 0
------------------------------------------------------------------------------------------
|           |Max 23.076923076923077   |Max 6.25                 |
|Cucumber   |Avg 7.718057136661787    |Avg 2.4492495904851763   |
|           |Min 0                    |Min 0                    |
------------------------------------------------------------------------------------------
```

*Figure 11: Output showing Macro Correlation*

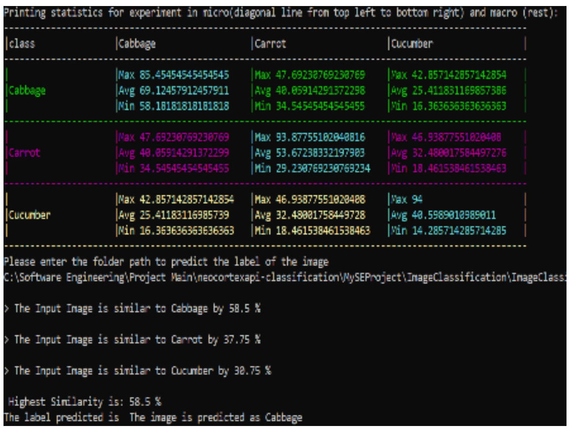
## Case 2: By modifying the prediction code to calculate the highest similarity of the input images

To test the quality of learning we have improved the prediction code to calculate the highest similarity of the input images. The prediction code provides a set of predicting results like: “Cabbage – 87%, Carrot 7%, Cucumber - 3%”. In Figure 12 the input label prediction and highest similarity matrix is displayed.

```
string PredictLabel(int[] sdrOfInputImage,
Dictionary<string, int[]> sdrs)

        {

                //Dictionary<string, List<string>>
inputsPath = new Dictionary<string,
List<string>>();

                string label = "Could not be able to
predict the label";

                double similarityWithEachSDR =
0;
```

Listing 1: Prediction code

Listing 1 shows the prediction code. Select one image path from input folder. The image is binarized into an integer array with ReadImageData. Using Compute method of API, the SDR of the image is calculated and is then compared with the SDR values of other images which is available in the input folder using the PredictLabel function. We can get the similarity of the images using CalcArraySimilarity.



```
Printing statistics for experiment in micro(diagonal line from top left to bottom right) and macro (rest):
-------------------------------------------------------------------------------------------
|class           |Cabbage                 |Carrot                  |Cucumber                |
-------------------------------------------------------------------------------------------
|                |Max 85.45454545454545   |Max 47.69230769230769   |Max 42.857142857142854  |
|Cabbage         |Avg 69.12457912457911   |Avg 40.05914291372298   |Avg 25.411831169857386  |
|                |Min 58.18181818181818   |Min 34.54545454545455   |Min 16.363636363636363  |
-------------------------------------------------------------------------------------------
|                |Max 47.69230769230769   |Max 93.87755102040816   |Max 46.93877551020408   |
|Carrot          |Avg 40.05914291372299   |Avg 53.67238332197903   |Avg 32.480017584497276  |
|                |Min 34.54545454545455   |Min 29.230769230769234  |Min 18.461538461538463  |
-------------------------------------------------------------------------------------------
|                |Max 42.857142857142854  |Max 46.93877551020408   |Max 94                  |
|Cucumber        |Avg 25.41183116985739   |Avg 32.48001758449728   |Avg 40.5989010989011    |
|                |Min 16.363636363636363  |Min 18.461538461538463  |Min 14.285714285714285  |
-------------------------------------------------------------------------------------------
Please enter the folder path to predict the label of the image
C:\Software Engineering\Project Main\neocortexapi-classification\MySEProject\ImageClassification\ImageClass

> The Input Image is similar to Cabbage by 58.5 %

> The Input Image is similar to Carrot by 37.75 %

> The Input Image is similar to Cucumber by 30.75 %

 Highest Similarity is: 58.5 %
The label predicted is  The image is predicted as Cabbage
```

*Figure 12: Correlation matrix with prediction result*

## IV. Discussion

We have used the process and properties of image classification based on Spatial pooler. The Spatial pooler has a flexible coding schema that can be used in real life machine learning applications. We did not use any encoder. We just binarize the image and give it to the spatial pooler. If we look in our image and the output table, we can see exactly matched images are showing 100%. Further, some images are quite similar but there may be some little dissimilarity.

In this paper, neocortex API is used for the analysis of image classification with Fruit360 dataset. The results show how the HTM parameters influence the similarity. The experiments were also helpful for predicting input images. If the input images to be predicted belong to the trained dataset, the prediction result was correct. If the image to be predicted does not belong to the training dataset, then the output will be "Couldn't predict the label". From this prediction code, we have incorporated a similarity threshold which prohibits false prediction. We improvised the prediction code to calculate the highest similarity of the input images.

Now in this era, there are many smart machines that use image recognition for the identification process. The advancements of how our brains work biologically may lead to new and revolutionary ways of achieving a true machine intelligence, the aim of the HTM theory.  All of this may not be 100% accurate. Though our project has some limitations, we can apply it to various real-life work. We can improve result accuracy by using more images.

## References

[1] C. Neto, M. Brito, H. Peixoto, V. Lopes, A. Abelha and J. Machado, "Prediction of Length of Stay for Stroke Patients Using Artificial Neural Networks," *Information Systems and Technologies*, vol. 1159, pp. 212-221, 2020.

[2] A. Ghazanfar and C. Schroeder, "Is neocortex essentially multisensory?," *Trends in Cognitive Sciences*, vol. 10, no. 6, pp. 278-285, 2006.

[3] Y. Cui, S. Ahmad and J. Hawkins, "Continuous Online Sequence Learning with an Unsupervised Neural Network Model," *Neural Computation*, vol. 28, no. 11, p. 2474–2504, 2016.

[4] D. Maltoni, "Pattern Recognition by Hierarchical Temporal Memory," SSRN, 2011. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3076121. [Accessed March 2022].

[5] S. Purdy, "Encoding Data for HTM Systems," *Neural and Evolutionary Computing*, no. arXiv:1602.05925, 2016.

[6] D. Dobric, "Neocortexapi," [Online]. Available: https://ddobric.github.io/neocortexapi/.

[7] M. Wirth, "Craftofcoding," WordPress, 17 2 2017. [Online]. Available: https://craftofcoding.wordpress.com/2017/02/13/image-binarization-1-introduction/. [Accessed 3 2022].

[8] "Kaggle," Kaggle, [Online]. Available: https://www.kaggle.com/. [Accessed 2021].