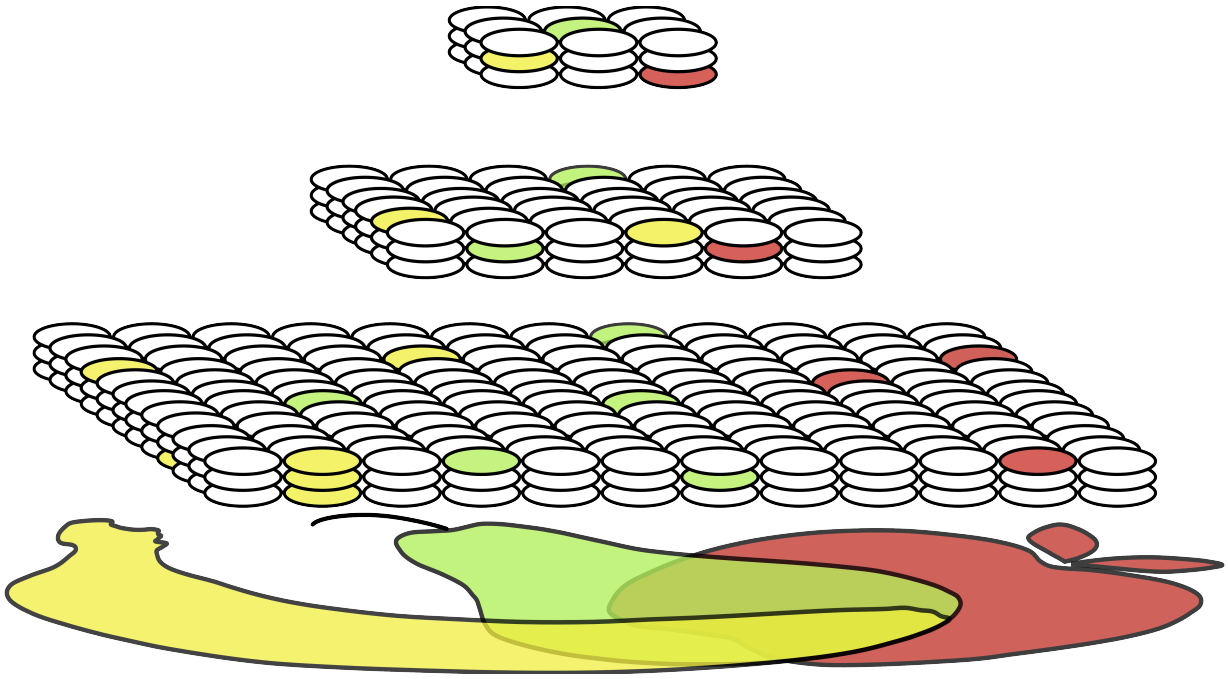
CHALMERS



Fruit Recognition by Hierarchical Temporal Memory

Master's Thesis in Complex Adaptive Systems

OLOV MATTSSON

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

MASTER'S THESIS

Fruit Recognition by Hierarchical Temporal Memory

Master's Thesis in Complex Adaptive Systems

OLOV MATTSSON

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2011

Fruit Recognition by Hierarchical
Temporal Memory
OLOV MATTSSON

Master's Thesis
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2011

Fruit Recognition by Hierarchical
Temporal Memory
Master's Thesis in Complex Adaptive Systems
OLOV MATTSSON
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

Food is a requirement for living, and traded in enormous amounts everyday. The globalization has led to optimization of the supermarkets and that a lot of stores have introduced self-scanning systems at check out and payment. When articles such as vegetables and fruits are traded the process becomes slower because the packages usually do not wear barcodes, which have to be added manually. This is a problem and the purpose with self-scanning drops out. In this thesis, a recognition system is built with the purpose to be used in self-scanning systems. The system thresholds the original image into a binary image. The binary image is sent to an advanced type of Neural Network called Hierarchical Temporal Memory. Such a network is independent of color, size, spatial space and rotations. These properties make it suitable for the given task. Two sorts of fruits were tested and the algorithm gave the accurate prediction in 97.5 % when tested on previously unseen images.

Keywords: Fruit recognition, Hierarchical temporal memory

Contents

Preface

The idea to this project was born when I was at the supermarket grocery shopping. I ended up at the self-scanning check-out and found it very time consuming to find specific vegetables in the self-scanning systems. After communicating the idea to people in my surroundings, I concluded that there is a need for a more effective scanning process at the grocery stores.

Aknowledgements

I would like to extend a special thanks to my supervisor Krister Wolff for his support throughout the project. I would also like to thank Viktor Jonsson and Per Ringqvist for good discussions throughout the project.

Göteborg October 2011
Olov Mattsson

1 Introduction

Almost all food stores have got new payment systems, where the costumer scans the articles at checkout. Articles that have barcodes attached to its package have been easy to scan in the self scan payment systems. When buying vegetables the costumer needs to enter each of every vegetable type separately to the system because they usually do not wear barcodes. This is a problem that customers have to become aware of and might lead to less usage of the self-scanning.

Image recognition problems often seem to be simple at first sight, but are often very hard to solve. Today there is no obvious way to handle these problems and many different techniques have been tested. To get a computer to recognize a single pattern is not hard. The hard part is to get a computer to recognize thousands of different patterns that belongs to the same object.

One technique that has been used is Neural Networks (NN). This type of network is suitable for solving complex problems such as image recognition tasks. An expansion of a such network is the Hierarchical Temporal Memory, (HTM), which has been used in this thesis. The outline to this algorithm is described in the book On Intelligence [1].

1.1 Objective

The primary goal for this thesis is to investigate if the HTM algorithm is able to differentiate between various positions of different fruits. A secondary goal is to study the HTM algorithm and understand how it works. The overall idea is finally to integrate the developed system into an existing self-scanning system.

1.2 Limitations

Due to lack of time only two types of groceries can be considered, namely apples and bananas. Furthermore, the classification will be probabilistic, which means that the system should determine which class an object belongs to, but not necessarily exactly which type it is. Another limitation is that spatial coordinates is not considered, i.e. the fruit is always centered in the image. When the customer buys fruit it often buys several pieces of the same type of fruit and places them in a transparent bag. Transparent bags are not considered and the recognition is carried out on a single fruit. Various sizes of the same product is not considered here.

1.3 Related work

Several fruit recognition techniques are using shape, color and size attributes to distinguish between different fruits [5], [7]. Texture features have also been proposed [6]. These techniques build on classic image recognition algorithms which creates databases that later are scanned and matched against unique patterns.

In the article [2], features such as color, shape, and roughness of fruit surface were extracted from the images and entered to the input layer in the NN. More complex models are using attractor, fractal dimension and NN to explain fruit shape quantitatively which are described in [3]. The HTM algorithm used in this thesis is described in [4] that was published in late 2010. One of the main authors to the HTM article [4], Jeff Hawkins, has also written the book On Intelligence [1] which describes the basic concepts of the HTM.

2 Theory

The follow section will give a short description of the HTM algorithm. For a complete description see [4].

2.1 Hierarchical temporal memory algorithm

The idea behind the HTM algorithm is to mimic the human brain. The algorithm is a type of NN which is built of regions. Each region consists of columns of neurons, see figure 2.3. which are arranged as in figure 2.1. The columns work similarly as the cells or neurons work in the human brain. These cells create links between other cells. The algorithm is divided into a spatial part and a temporal part. The spatial part is called Spatial Pooler (SP) and works in the spatial space during the temporal part called Temporal Pooler (TP) works over time. Each region receives inputs from the region below and pass processed data to the next region, see figure 2.1.

2.1.1 Spatial pooler

The SP will form a sparse distributed representation of the input. For example if the input consists of a $N \cdot N$ (N is the image size) pixel image the sparse distributed representation contains only a small amount of the $N \cdot N$ pixels.

Figure 2.2 shows how a column receives its input through it's spatial synapses. In the HTM framework, distributed means that activations of several cells are required to represent something useful and sparse means that a few number of the cells are active at a time.

How will the SP form the sparse distributed representation from its input? At any time a certain part of the input bits are active (1) or inactive (0). Each column has synapses, see figure 2.2, which create connections to a unique subset of input bits. Columns may have connections to the same input bit but are not allowed to connect to the exact same input pixels. The columns which connect to lot of input bits that are in state is (1) will be active and reduce the activity of weaker columns. This will create a stable number of active columns for different sets of input patterns and activate relatively stable sets of columns for similar input patterns. The method for this step will be described later in Chapter 3. The concept of sparse distributed representation is defined in paper [4] for further reading.

2.1.2 Temporal pooler

The TP get its input from the SP, i.e. the active columns. Each region will extract the context of the active columns.

This is done by predicting what will happen next and knowing what happened just before this time step. One can think of it as when the first tones of a melody is presented. If only one tone has been presented it is hard to know the context but if two or more tones have been presented it is possible to know the context, i.e. the melody. The predictions is based on the passed events in time. An example of how the temporal pooler works is given in Chapter 2.1.4.

2.1.3 HTM components

A region in the HTM consist of columns which consists of cells. Each cell have segments and synapses which they use to build connections to others cells. Figure 2.3 shows a column c_1 that has 3 cells i_1, i_2 and i_3 where $c_1.i_1$ has 3 segments s_1, s_2, s_3. $c_1.i_2$ has 2 segments and $c_1.i_3$ has one segment. Segment $c_1.i_1.s_2$ has a branch which is called a temporal synapse. It is through the synapses the cell make connections to other cells and create a network of cells as seen in figure 2.4. Each synapse has a certain strength called permanence in the HTM framework. The permanence value determines if the synapse is connected or disconnected. If the permanence value is great enough the synapse will be connected and the cell will be able to communicate and learn from other cells in the network.

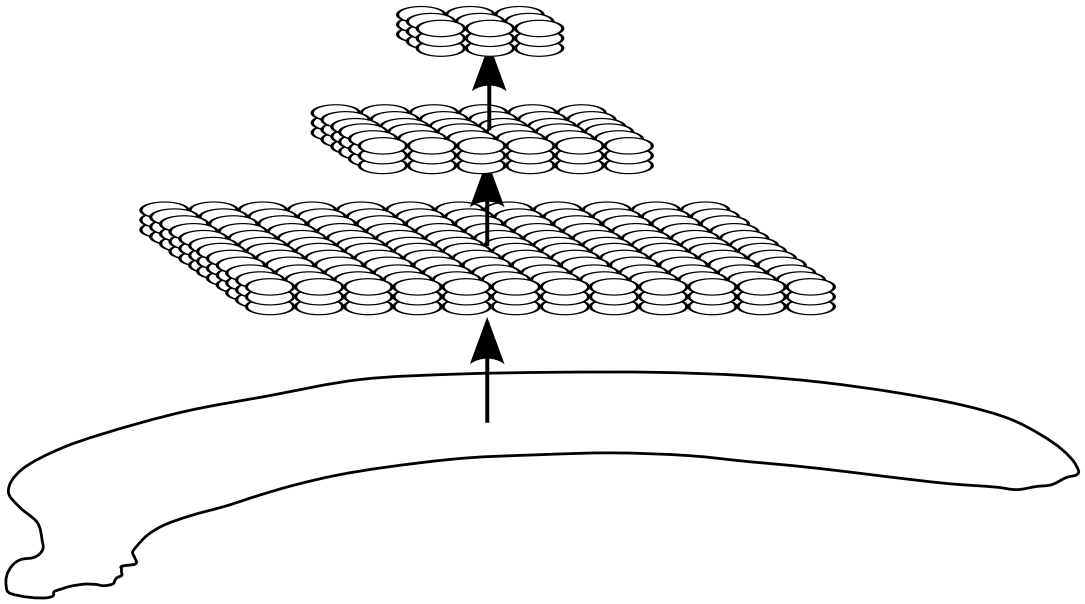


Figure 2.1: This figure shows three HTM regions with each 3 cells per column. The information is sent as the arrows show.

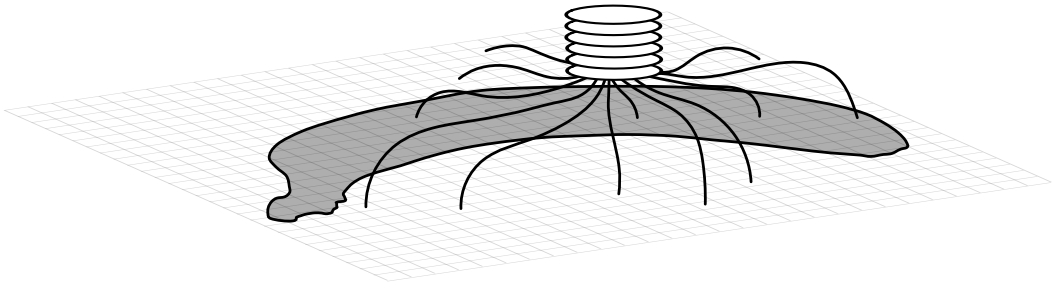


Figure 2.2: This figure shows a HTM column with its cells that connects to the spatial input.

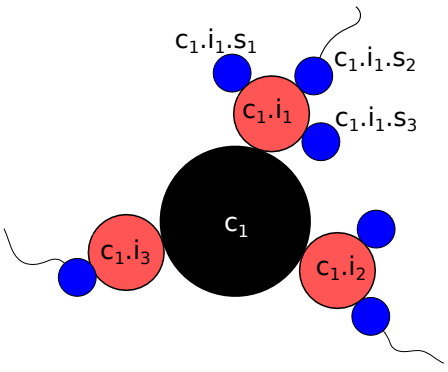


Figure 2.3: This figure shows a schematic picture of a HTM column, c_1, consisting of three cells $c_1.i_1$, $c_1.i_2$, and $c_1.i_3$. Each cell has segments. Cell $c_1.i_1$ has 3 segments where $c_1.i_1.s_2$ has a TP synapse. It's through this synaps the column send information in the TP. The spatial order of the cells is a schematic view. Actually, the cells is order on top of each other as in figure 2.2.

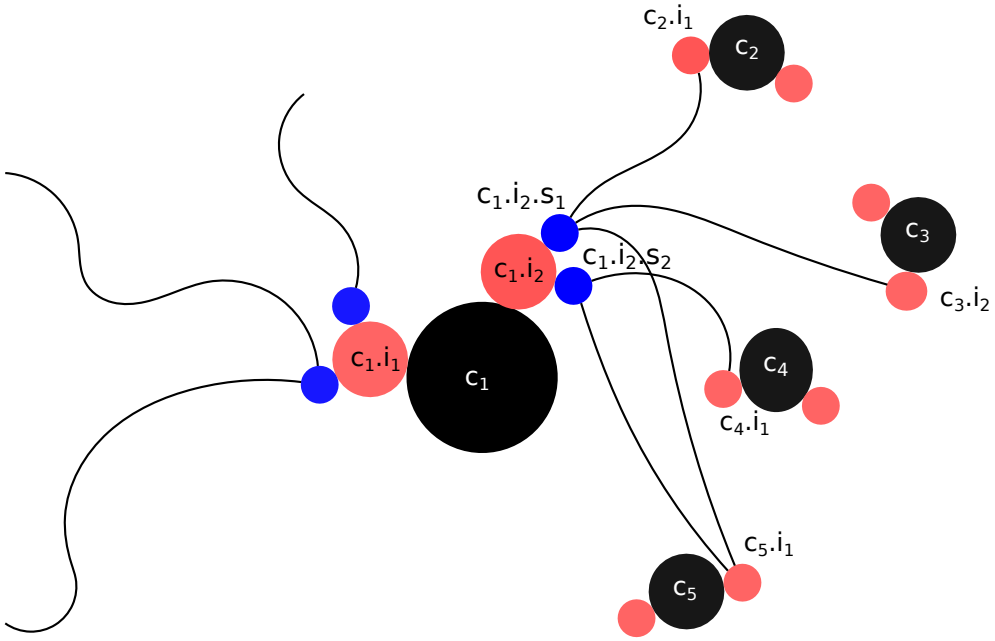


Figure 2.4: This figure shows 5 HTM columns which together create a HTM network.

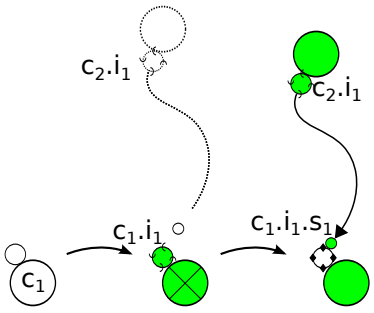


Figure 2.5: This figure shows how a cell creates a connection to another cell. A column becomes active (filled, green) and creates a segment, $c_1.i_1.s_1$, (see figure 2.5) which creates a synapse connection to $c_2.i_1$. The cross symbol shows if the column receives inputs from the SP. In the second step, cell $c_1.i_1$ is in learn state, hence the buckets around the cell edge. In the final step, $c_2.i_1$ is active which makes the segment $c_1.i_1.s_1$ active. When segment $c_1.i_1.s_1$ turns into an active state, cell $c_1.i_1$ turns into the predictive state (hence the diamonds around the edge). When a cell becomes predictive its column will be active. To depict whether a column is active or inactive the cell is filled (green) for active state and transparent for inactive state. The arrow tells in which direction the synapse acts.

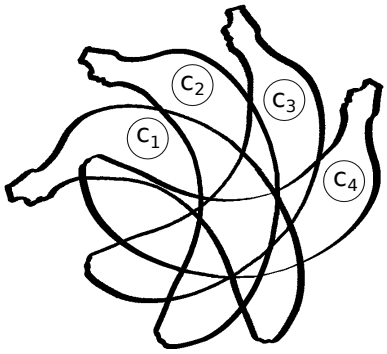


Figure 2.6: This figure shows four different patterns of a banana where each pattern overlap with a specific column.

2.1.4 An example of the temporal pooler

The following illustration will give a simple example how the TP works. Four columns are active in a repeating sequence i.e. the output from the SP is: c_1 in t_0, c_2 in t_1, c_3 in t_2, c_4 in t_3, c_1 in t_5 and so on.

Figure 2.6 shows 4 different patterns of a banana in a time-serie. In the first time step, t_0 ,the banana overlap with column c_1 and therefore SP sets column c_1 to active. This is simplification of the SP, it's not always necessary that the column is located over the pattern's area. The column could be outside the pattern's area but still be active. Figure 2.2 shows how a column gets input from its spatial synapses.

(a) Time step t_0. $c_1.i_1$ is in learnstate and c_1 is active.

(b) Time step t_1.

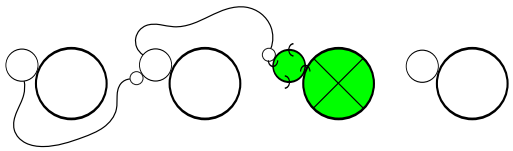
Figure 2.7: This figure shows 13 time steps in subfigure a-m.

Subfigure 2.7a shows the first time step when column c_1 receives input from SP (cross symbol). The cell $c_1.i_1$ turns into learn state but doesn't make any connections to the other cells since this is the first time step and nothing before has happened. In the next subfigure, 2.7b, where the cross symbolize that column c_2 gets input from the SP and its cell, $c_1.i_1$ creates a segment and connects with the cell that was active in the previous time step. The HTM has now created a segment with a synapse between two cells and has learnt one time step.

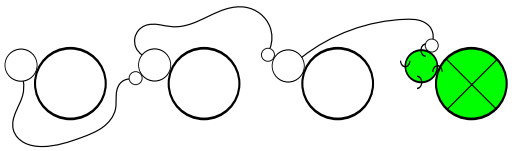
Subfigures 2.7c and 2.7d show how the cell ($c_3.i_1$) respectively ($c_4.i_1$) makes connections with the cells that were active in the previous time step.

In time step t_4 (see subfigure 2.7e) column c_1 receives input from the SP and becomes active. It also creates a segment with a synapse to cell $c_4.i_1$ which was active in the previous time step. Segment $c_2.i_1.s_1$ will be active because it has a synapse to cell $c_1.i_1$ and therefore cell $c_2.i_1$ turns into predictive mode. When a cell is in predictive mode its associated column will be active which can be seen in subfigure 2.7e.

The next subfigure 2.7f shows what happens when column c_2 receives input. Cell $c_2.i_1$ creates a new segment with a synapse to the cell that was active two previous time steps. Cell $c_3.i_1$ will also start to predict through its segment and therefore column c_3 becomes active.

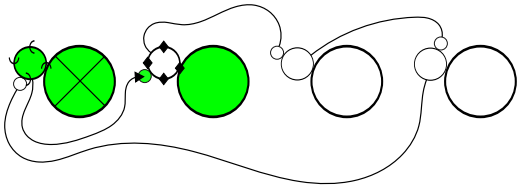


(c) Time step t_2.

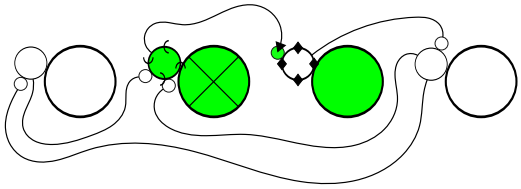


(d) Time step t_3.

Figure 2.7: Continued.



(e) Time step t_4.



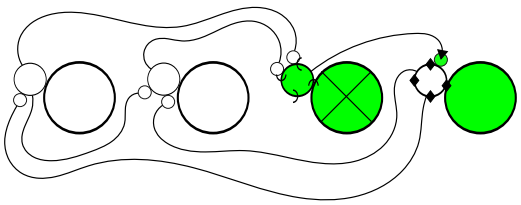
(f) Time step t_5.

Figure 2.7: Continued.

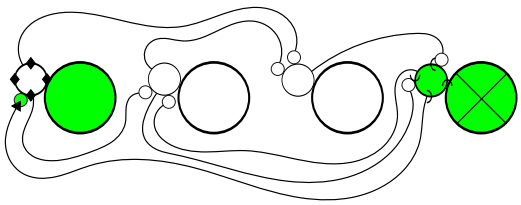
This is a repeating process where the HTM will learn to recognize the preceding state in each iteration. Subfigures 2.7g - 2.7l show how the network creates segments and synapses for each iteration.

In the last subfigure 2.7m cell $c_1.i_1$ will make a new segment with a synapse to $c_2.i_1$. All other cells in the chain will be predictive when column c_1 gets input from the SP hence they all have segments with connections to cell $c_1.i_1$. This will result in that all columns will be active. Now it does not matter which of the four bananas we send into the network, the network will give the same output. This is how the network become stable i.e. order common patterns into the same group.

If instead the network start to learn patterns of apples, where the patterns are described by columns c_6, c_7, c_8 and c_9, it will order them into one common group. The HTM have now created two different groups, where one describes the patterns of the bananas and the second the patterns of the apples. When an unknown pattern of a banana or an apple is shown to the HTM, it will either predict $c_1, ..., c_4$ or $c_6, ..., c_9$ which later is passed on to a classification algorithm, see Chapter 3.1.3.

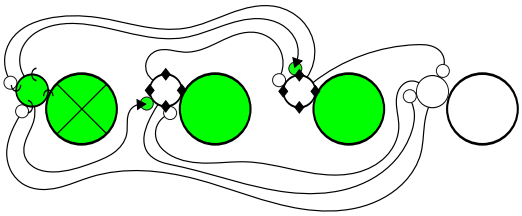


(g) Time step t_6.

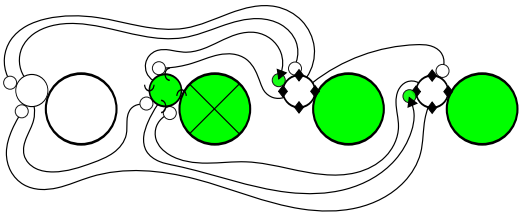


(h) Time step t_7.

Figure 2.7: Continued.

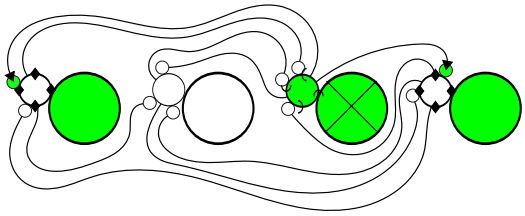


(i) Time step t_8.

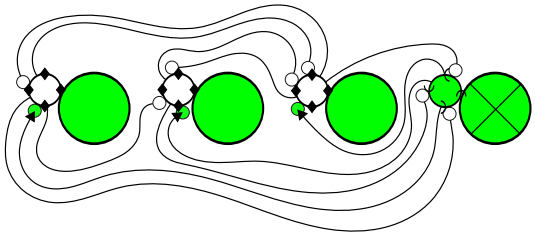


(j) Time step t_9.

Figure 2.7: Continued.

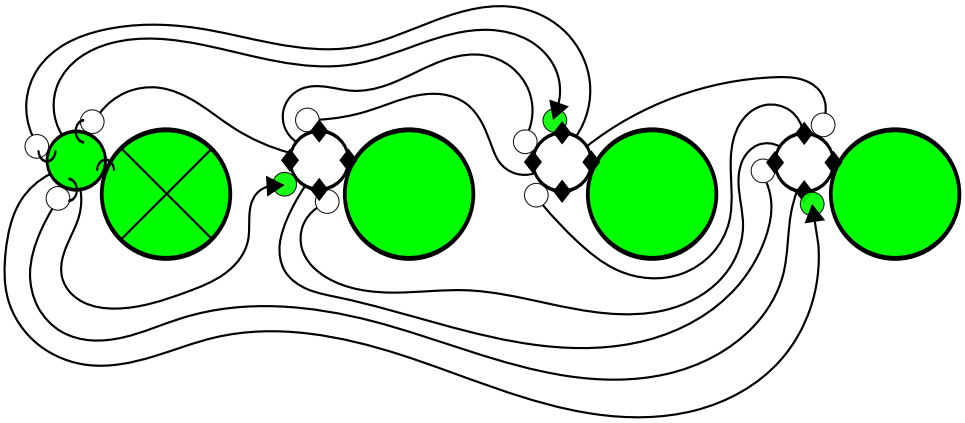


(k) Time step t_{10}.



(l) Time step t_{11}.

Figure 2.7: Continued.



(m) Time step t_{12}.

Figure 2.7: Continued.

3 Method and Implementation

This section describes the algorithm for fruit recognition and how it is implemented. The image processing of the input before it is sent to the HTM and the classification of the output from the HTM is also described in this chapter.

3.1 Algorithm Description

Photographs have been taken of fruits in different angles. Because of limitations only pictures of bananas and apples were selected. All images were taken with a white background which made it easy to separate them from the background. The image was thresholded into a binary image and then sent to the HTM. The output from the HTM was classified with a classification algorithm. Finally in the classification section a prediction was made of what type of fruit the original picture described.

3.1.1 Image processing

An RGB-image consist of 3 layer (red, green and blue). Each layer is represented by a matrix where the element is a value between 0 and 255. In this thesis a digital SLR Nikon D90 camera was used and each matrix had a size of $2848 \cdot 4288$ pixels. The image was converted into grayscale by the command rgb2gray in MatLab. The MatLab command uses E.q. 3.1 to convert the original pixels into grayscale pixels. Equation 3.1 sums up 29.89 % of the red i, j pixel, 58.7 % of the green and 11.4 % of the blue into one common pixel.

$$Gray(i,j) = 0.2989 \cdot R(i,j) + 0.5870 \cdot G(i,j) + 0.1140 \cdot B(i,j) \tag{3.1}$$

The image was cropped and resized in MatLab by the command imcrop and imresize. The new image consisted of $285 \cdot 285$ pixels and was finally binary thresholded with a parameter value equal to 130 for the apple and 120 for the banana. When a image is binary thresholded it means that all pixel values above a certain value is set to zero and the others to one. Figure 3.1 shows the image processing steps.

To get the full training data set the binary image has been rotated by the angle θ with the rotating matrix R, E.q. 3.2. The rotating angle, θ is created by dividing 360 degrees with the total number of subpatterns. Then the subpatterns will be uniformly distributed throughout 360 degrees.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.2}$$

Validation patterns have been created in the same way as the training set but with a small superimposed angular displacement. The angular displacement should be seen as noise in the input data. A schematic picture of the angular displacements for the banana is shown in figure 3.2.

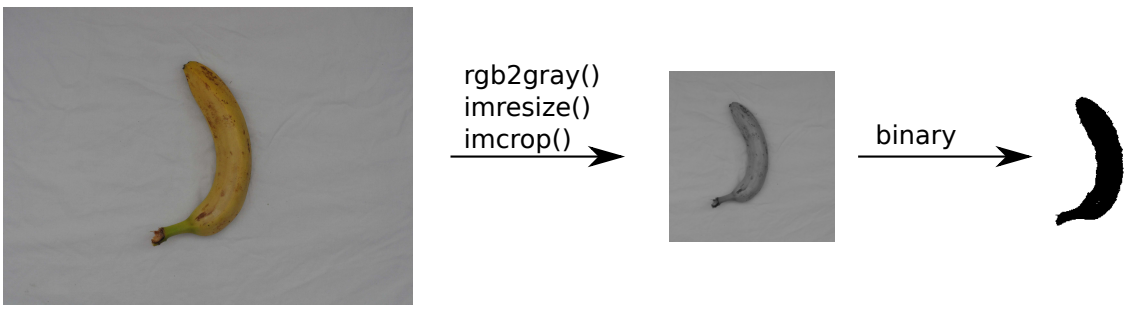


Figure 3.1: Image processing of the original image.

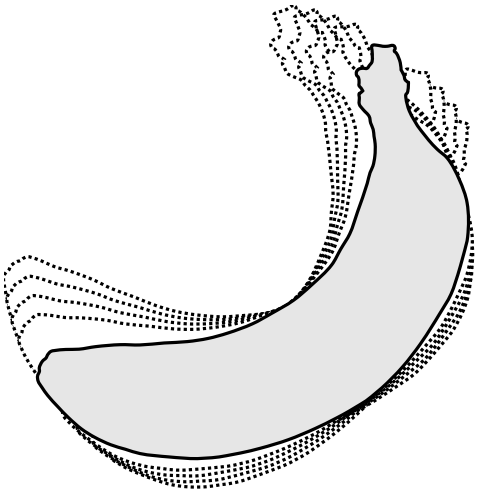


Figure 3.2: This figure shows the angular displacement for a single banana pattern.

3.1.2 HTM

The HTM has been implemented in C and the pseudocode for the main loop is given below. The graphic visualizations was made by the program platform OpenGL.

```
main() {
//SpatialPooler//
OverLap(...);
CalcNeighbors(...);
Inhibition(...);
Learning(...);

//TemporalPooler//
TemporalPoolerPhase1(...);
TemporalPoolerPhase2(...);
TemporalPoolerPhase3(...);

SetOutPutToNextRegion();
}
```

The program is built by structs and it allocates enough memory for a run at the beginning. An option could be to re-allocate memory each iteration, but it was realized that for this purpose it would have longer runtime. The SP contains of several functions, namely *Overlap*, *CalcNeighbors*, *Inhibition* and *Learning*. Below is a short description of the involved functions. A more complete description can be found in [4].

The *Overlap* function calculates how much overlap each column have with the input. The function *CalcNeighbors* computes which columns that are neighbors with a specific column. There is a specific radius (*inhibitionRadius*) that a column must be within to be neighbor to another column. By knowing the neighbors and the overlap score each column will either be active or inactive in the SP which is determined by the *Inhibition* function. The column must have higher overlap score (*desiredLocalActivity*) then its associated neighbors to become active. A active column from the *Inhibition* function is called a winning column. Finally the function *Learning* controls which column that will be learned to recognize specific patterns. For a winning column, its active spatial synapses values are increased, i.e. the connection strength to the input pixel is increased, otherwise it is decreased.

In *TemporalPoolerPhase1* the TP loops over the active columns from the SP. It is in this phase learning and active cells are selected. *TemporalPoolerPhase2* calculates if a cell will be in predict state or not. *TemporalPoolerPhase3* will update the selected cells from *TemporalPoolerPhase1*.

In this phase specific temporal connections are increased or decreased. New segments could also be created which have new temporal synapses connections. Individual synapses could also be added.

The last function in the loop, *SetOutPutToNextRegion*, is setting the output to the next region. It loops over all cells in the region and if a cell is active from *TemporalPoolerPhase1* or predictive from *TemporalPoolerPhase2* it sets its belonging column to active. The output from the HTM is a vector where each element describes if the column is active or inactive. If the network consist of one region with 100 columns the output vector contain 100 elements. A time average output vector is also calculated for each fruit. The time average vectors is stored and used when validation occurs.

3.1.3 Classification of output

This part of the program is coded in MatLab.

When a unknown pattern is sent to the network it is compared with the stored averaged vectors. An euclidean distance is calculated against each stored vector. The vector that corresponds to the smallest euclidean distance will be chosen with its belonging fruit as its prediction. The smallest euclidean distance will from now on be called column error. An error of the predictions, i.e. the fraction between right predictions to total number of predictions will be calculated. This error will from now be called validation error.

4 Results and discussion

Different simulations have been performed with the HTM. Both to ensure that the HTM behaves as it should, but also to see if it copes with its main purpose (recognize different fruits). A theoretical result for the HTM has also been found.

4.1 Sequence patterns

In Chapter 2.1.4 an example, see figure 2.7, about sequence patterns was given. That is one of the key features for the HTM and therefore it was simulated. In this simulation the SP was disconnected and instead a sequence of 100 unique patterns was given to the TP. Each pattern consisted of one active column.

The sequence looked like c_1, c_2, c_3,...,c_{99} and regardless of which one of the columns that was active from the simulated SP all other columns were predicted. The parameter setup for this simulation is shown in Appendix 6, Table 6.2.

4.2 Sequence patterns with noise

The parameter activationThreshold is crucial when the HTM is feed with noisy data. For this simulation a sequence, as in Chapter 4.1, was given to the TP. Each pattern consisted of 10 columns and the activationThreshold parameter was set to 7, the HTM was able to handle data that consisted of 30 % noise. That means that 3 out of 10 columns can be randomized for each pattern without loss of information in the output. If 4 columns are changed randomly and none of them belongs to the specific set of columns for a specific pattern the threshold for being predicted will not be reached and therefore no columns will be predicted. The HTM was also tested with data consisted 90 % noise. The activationThreshold parameter was set to 1 and the HTM was able to distinguish the different patterns. The parameter setup is shown in Appendix 6, Table 6.3.

4.3 Overlap patterns

The idea of simulating overlap patterns in the HTM is to see if it can distinguish between similar patterns, i.e. patterns that share common active columns. Three different groups of patterns have been exposed to the HTM. They are symbolized by the banana, apple and pear as the schematic figure 4.1 shows. The filled area in the schematic figure symbolizes the overlap between the three fruits. Each pattern consists of 10 active columns. For example subpattern p_0 in table 4.1 has the columns (c_0, c_{10}, c_{20}, c_{30}, c_{40}, c_{50}, c_{60}, c_{70}, c_{80}, c_{90}), as its active columns.

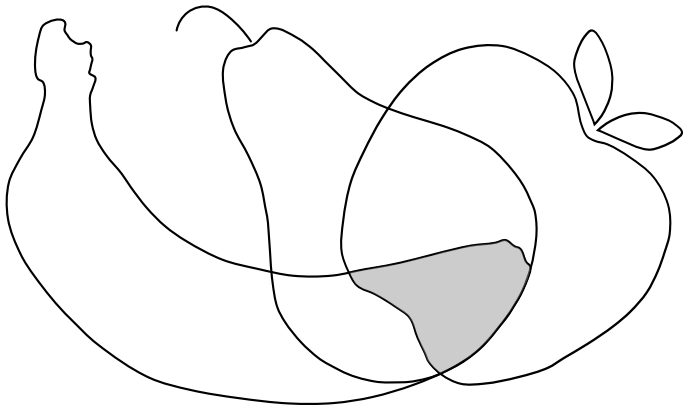


Figure 4.1: This figure shows the schematic overlap between an apple, a banana, and a pear.

Table 4.1: This table show the patterns which describes the banana.

Subpatterns	Active Columns
p_0	0 10 20 30 40 50 60 70 80 90
p_1	1 11 21 31 41 51 61 71 81 91
...	...
p_8	8 18 28 38 48 58 68 78 88 98
p_9	9 19 29 39 49 59 69 79 89 99

Table 4.2: This table show the patterns which describes the apple.

Subpatterns	Active Columns
p_0	0 10 20 130 140 150 160 170 180 190
p_1	1 11 21 131 141 151 161 171 181 191
...	...
p_8	8 18 28 138 148 158 168 178 188 198
p_9	9 19 29 139 149 159 169 179 189 199

Each fruit contains of 10 different subpattens which are enumerated in table (4.1), (4.2) and (4.3). If the banana covers the filled area in figure (4.1), some of columns c_0 ,..., c_{29} become active. For pattern p_8 columns c_8,c_{18} and c_{28} become active. These columns also describe patterns p_8 for the apple and the pear, which is then called the overlap for pattern p_8. The HTM should therefore be able to separate the fruits by the other active columns which are not included in the overlapped area.

This have been tested and the HTM separates the different groups of patterns very well. In the TP there is a parameter called activationThreshold that controls of how big overlap the HTM can manage. If the number of active connected synapses to a specific cell in a column is greater then the activationThreshold parameter the column will be active. That means that if an cell, $c_1.i_1$ has five synapses and the activationThreshold is set to 3 it needs 3 or more of the synapses to be active to activate the column. The conclusion is that the activationThreshold parameter must be greater then the number of sharing columns.

The overlap was increased to 9 out of 10 and the activationthreeshold parameter was set to 9. The HTM was still able to handle the task without errors. These results indicate that the HTM is good at distinguish common patterns and that the activationthreshold parameter is an important parameter when overlapping occurs. The parameter setup is shown in Appendix 6, Table 6.4

4.4 Training with noise

Three different subpatterns of bananas and apples were shown for the HTM. For each training iteration 20 validation iterations were made. In each validation a random noise was superimposed

Table 4.3: This table show the patterns which describes the pear.

Subpatterns	ActiveColumns
p_0	0 10 20 230 240 250 260 270 280 290
p_1	1 11 21 231 241 251 261 271 281 291
...	...
p_8	8 18 28 238 248 258 268 278 288 298
p_9	9 19 29 239 249 259 269 279 289 299

in the form of an angular displacement. The angular displacement had six different levels and at each level a small random factor was added.

The validations points in figure 4.2 have been averaged over time where each validation iteration consists of 50 training iterations, i.e. an average value of $50 \cdot 20 = 1000$ validation iterations.

The six different graphs in figure 4.2 shows how the column error for apple and banana decreases with increased training iterations. The column error is constant for the apple while for the banana it takes approximately 400 iterations before the error reaches steady state. The error increases for both banana and apple when the angular displacement is increased. An angular displacement of 6.3 degrees corresponds to a change of the original input pattern with 30 % noise for the banana and 5 % for the apple. The apple has a round shape and therefore angular displacement does not has large effect. The figure shows that the column error will stay low, although the noise is up to 30 %. The conclusion is that the angular displacement has almost no effect for the apple but some significance for the banana. This indicates a robustness in the system, i.e it is able to handle noisy input data.The parameter setup is shown in Appendix 6, Table 6.5

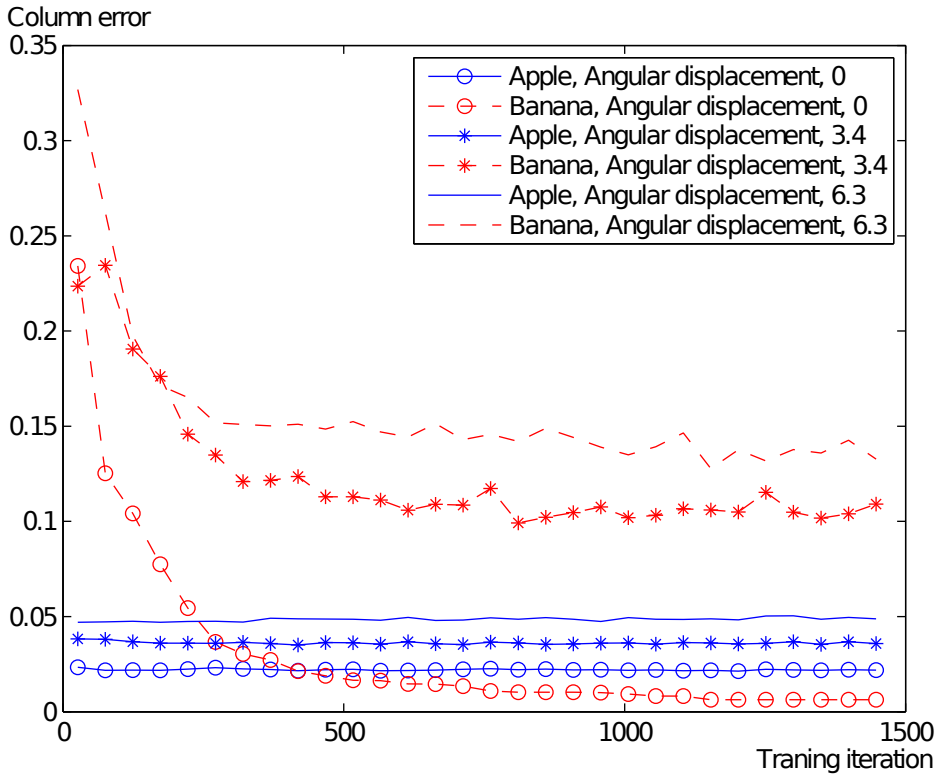


Figure 4.2: This figure shows how the column error changes over training iterations for different levels of angular displacement.

4.5 Increasing number of patterns

The idea of this simulation was to see if the training time changed with increasing number of subpatterns. Five simulations was performed and the parameter setup is in Appendix 6, Table 6.6. The validation data have been time averaged as in the previous example. Each validation pattern had a random angular displacement added. Figure 4.3 shows that the column error decreases for all setups with increasing number of training iterations. To achieve the same column error for different number of subpatterns, the HTM needs more training iterations with increasing number

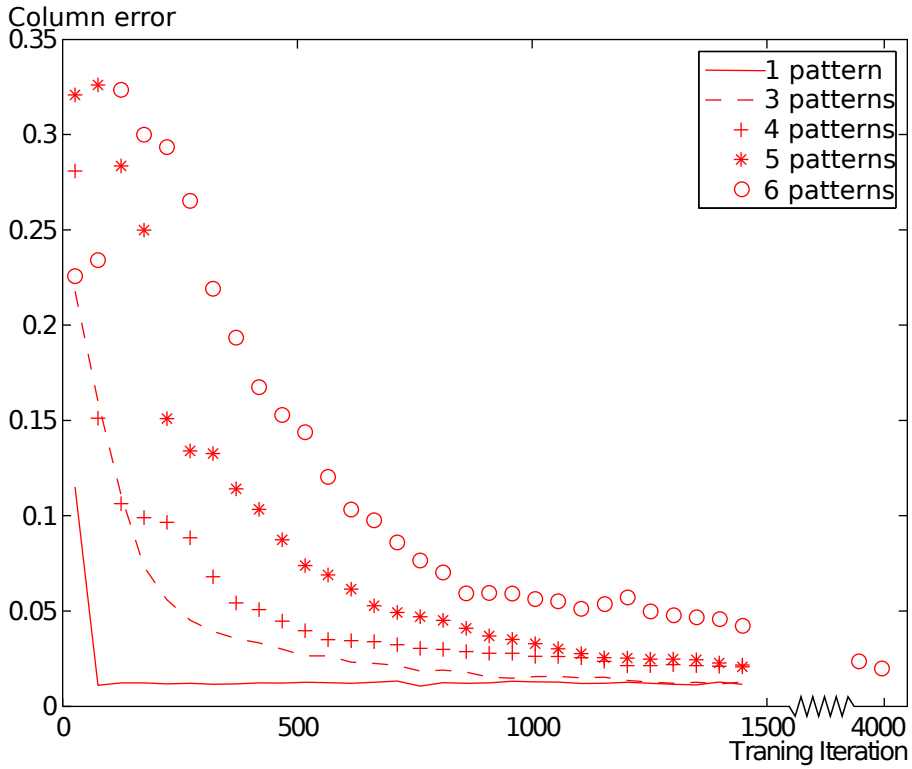


Figure 4.3: This figure shows how column error change against training for different number of input patterns. Note that the x-axis is compressed.

of subpatterns. Figure 4.3 also shows that the graph with 6 subpatterns needs 4000 iterations to reach the same column error as the graphs with lower number of subpatterns.

4.6 Validation error

The parameter setup for this simulation is shown in Appendix 6, Table 6.7. The validation set consisted of apples and bananas with an angular displacement. Figure 4.4 shows how the validation error changes over training iterations. For 3 and 6 subpatterns the validation error reaches zero before 100 training iterations, which means that the HTM after 100 training iterations make right predictions in every validation. If the number of subpatterns are increased to 13 the HTM will after 500 training iterations have a validation error around 2.5 %. A validation error of 2.5 % corresponds to a prediction accuracy of 97.5 for unseen patterns.

4.7 Theoretical result

From the given example in Chapter 2.1.4 calculations were made to calculate how many training iterations a network needs, with respect to number of input subpatterns, to be pre-trained. Table 4.4 shows the results from the calculations.

Based on the results in table 4.4 a relationship, E.q. 4.1, was found which describes how N_p (number of iterations) and p (number of subpatterns) are related to each other.

$$N_p = p(p-1) \tag{4.1}$$

$$p \geq 1$$

A induction proof for Eq. 4.1 is given below.

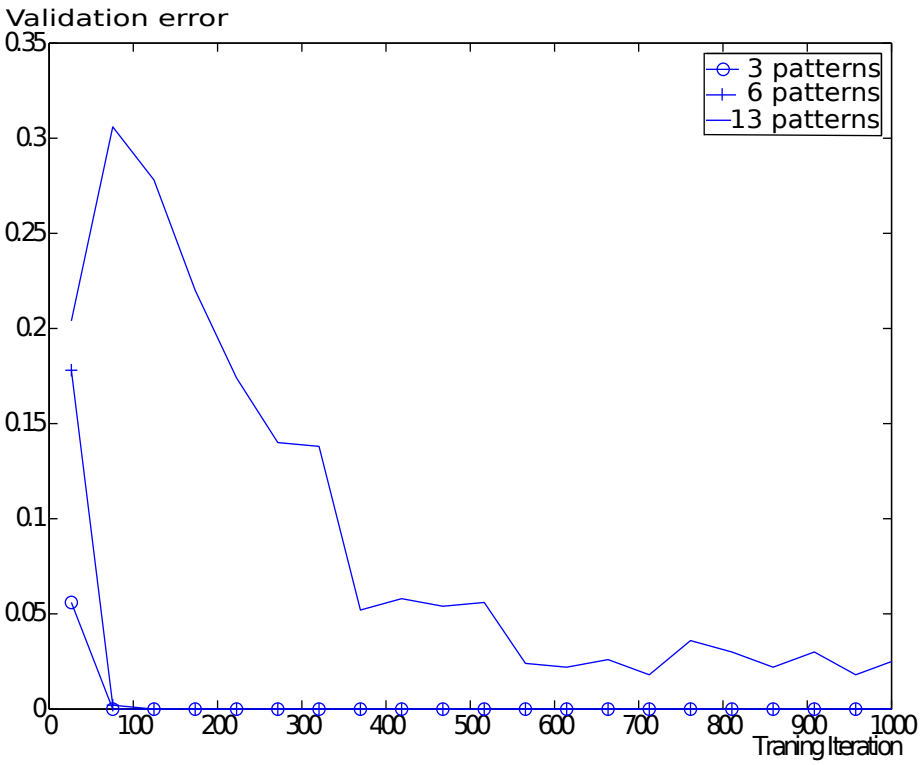


Figure 4.4: This figure shows validation error against training. Input patterns have been both patterns for banana and apple.

Table 4.4: This table show many iterations a network needs to be pre-trained.

Number of subpatterns, (p)	Iterations for be a pre-trained network (N_p)
2	2
3	6
4	12
5	20
6	30

When $p = 1$, i.e. only one input pattern, the network does not need to be trained and therefore $N_p = 0$. A network with p subpatterns requires N_p iterations to be pre-trained. A new network with $p+1$ patterns must separately establish connections to the new subpattern and the new subpattern must add a connection to each of the old subpatterns. It will then be required p iterations for the new subpattern to create connections to the old subpatterns and one iteration for each old pattern to connect with the new subpattern. In total, it will then be needed $p+p = 2p$ additional iterations for the network to be pre-trained, since only one connection can be created in each time step. The number of iterations for a network, to be pre-trained, with $p + 1$ subpatterns is then described by Eq. 4.2.

$$N_{p+1} = N_p + 2p \tag{4.2}$$

Assume, for $p = k$, that E.q. 4.1 holds, i.e. E.q. 4.3.

$$N_k = k(k-1) \tag{4.3}$$

Let $p = k + 1$, i.e. E.q. 4.4.

$$N_{k+1} = N_k + 2k \tag{4.4}$$

Insert E.q. 4.3 in E.q. 4.4 and get E.q. 4.5.

$$N_{k+1} = k(k-1) + 2k \tag{4.5}$$

Rewrite E.q. 4.5 into E.q. 4.6.

$$N_{k+1} = (k+1)(k+1-1) \tag{4.6}$$

Then E.q. 4.1 works for $p = k + 1$ and therefore is E.q. 4.1 valid for all integers, $p \geq 1$.

4.8 Stabilization of output

The main task for the HTM is to stabilize the output from each region i.e. to order patterns into common groups. Validation iterations were made every tenth training iteration. At each validation 30 random patterns from the training set have been sent as input to the network. The euclidean distances between all output vectors have been calculated and the median value of them is plotted in figure 4.5. The figure shows that with increasing number of patterns the network needs to be trained more before reaching a euclidean distance of zero.

From figure 4.5 has the point where the euclidean distance is zero been picked for each graph. These points are plotted in figure 4.6. The theory result, chapter 4.7, provides an E.q. 4.1 which also is plotted in figure 4.6. As the figure shows the simulation curve fits the theoretical curve very well. From the figure the simulation curve has lower N_p for higher number of p then the theoretical. This occurs because the number of iterations to reach pre-trained is median scored which results in a lower value then the theoretical. The parameter setup for this simulation is shown in Appendix 6, Table 6.8.

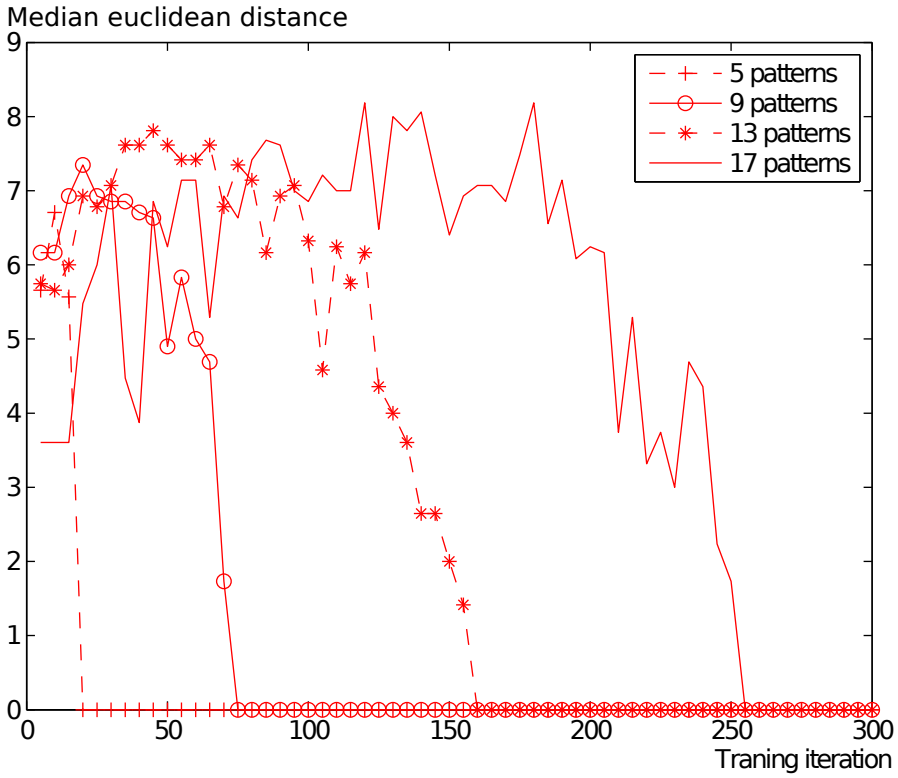


Figure 4.5: This figure shows how the output from the region stabilized against training iterations.

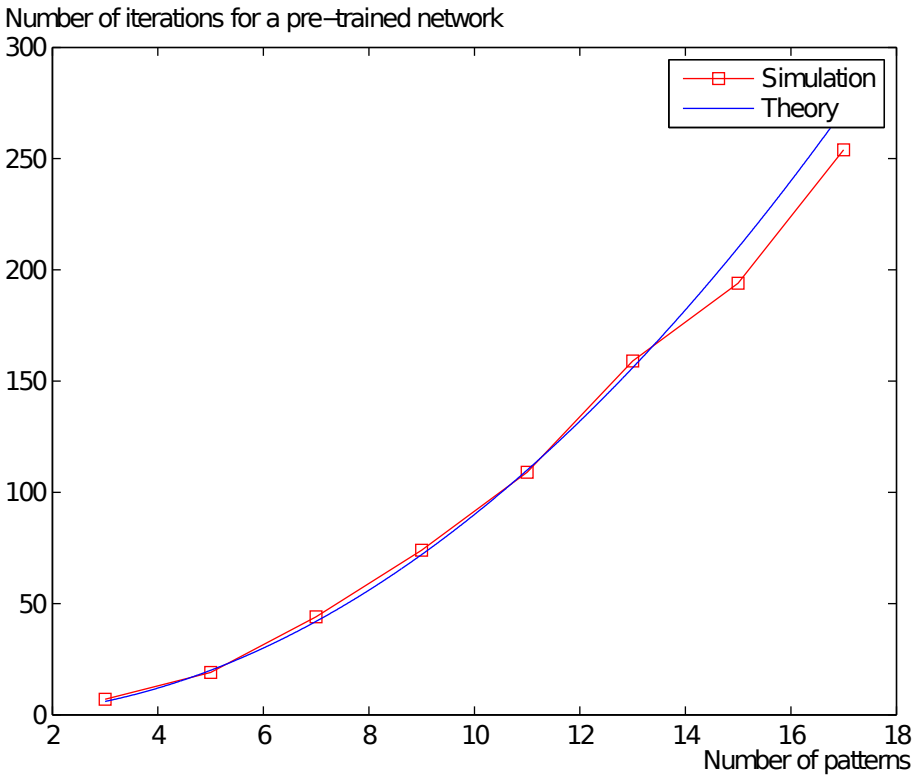


Figure 4.6: This figure shows total number of iterations for a pre-trained network against number of patterns. Note that the x-axis starts at 2.

5 Conclusion and Future Work

In this thesis, the HTM algorithm has been adapted and tested to the image recognition problem. A HTM network has been trained on segmented images. The system have been validated on unseen bananas and apples images with a accuracy of 97.5 %. By distinguishing different fruits the self scanning system is able to realize what the customer has placed in front of the cash desk.

In the SP the boost function was not implemented. The first version had it implemented but in later versions it was left out. The idea with the boost function is to let the columns that are not active often have a competition advantage against winning columns. When the system was tested a sets of columns were oscillating between being active and inactive. When columns are oscillating it causes the output being unstable which are not desirable. Therefore the boost function was left out. One possible reason that the columns were oscillating could be that the network was overtrained and that the boost function should be disconnected after a number of training iterations. The big difference without having the boost function is that the network would be more optimized with the boost function, i.e. fewer columns are needed.

The thesis is a preliminary study and only single objects was shown for the HTM. It would be interesting to test the performance with several fruits of the same sort. The HTM did only handled two sort of fruits in this thesis, in a store it must be able to deal with more groceries and therefore it would be interesting to test with several fruits and vegetables.

When buying fruits or vegetables they are often placed in a plastic bag. It was not investigated in this thesis. It can be assumed that it would not be very difficult to take that in account. Nowadays there are filters that are good at capturing specific colors. So as long the conditions for the input are the same and that the same plastic bags are used it will be possible.

Another thing is that HTM framework is hard to implement, for a novice person in the field it will take months to understand how the HTM works. The code should be quite optimized to achieve high speed performance, which causes the software engineer to be familiar with low level programming languages.

For future work it would be interesting to implement the boost function again and investigate if it works or to implement a new boost function. To test with different kind of fruit but also with many of the same kind would also be interesting. Another interesting thing would be to optimize the setup parameters and to allow the fruits to be in plastic bags.

6 Appendix A

This appendix shows the parameter settings for different simulations A short description of the parameters is given in in table 6.1. The large letter P at the end of a variable name, means that the variable is a pointer.

Table 6.1: This table explains the different parameters.

Parameter	Description
numberOfPatterns	number of group of patterns
numberOfSubPatterns	number of sub patterns
angularDisplacement	angular displacement
region	Specify the region
numberOfColumnsP	How many columns in the region
numberOfCellsP	How many cells in each column
numberOfSpatialSynapsesP	How many spatial connections per each column
minOverlapP	Minimum overlap required
newSynapsCountP	Maximum synapses added under learning
activationThresholdP	number of active connected synapses to activate a segment
minThresholdP	Minimum number of segment to learn
iterationAverageP	Number of iterations used for time averaging
connectedPermP	Permance value to be connected
permanenceDecP	Permance decrease
permanenceIncP	Permance increase
inhibitionRadiusP	intial value for the inhibitoon radius

The angular displacement in Table 6.3 was created by randomize a fraction of the columns in a specific pattern.

Table 6.2: This table shows the parameter values to the result for Sequence patterns.

Parameter	Value
numberOfPatterns	1
numberOfSubPatterns	100
angularDisplacement	0
region	0
numberOfColumnsP	100
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	16
newSynapsCountP	5
activationThresholdP	1
minThresholdP	1
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

Table 6.3: This table shows the parameter values to the result for Sequence patterns with noise.

Parameter	Value
numberOfPatterns	5
numberOfSubPatterns	10
angularDisplacement	30-90%
region	0
numberOfColumnsP	100
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	16
newSynapsCountP	10
activationThresholdP	1-7
minThresholdP	1
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

Table 6.4: This table shows the parameter values to the result for Overlap patterns

Parameter	Value
numberOfPatterns	3
numberOfSubPatterns	10
angularDisplacement	0
region	0
numberOfColumnsP	400
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	1
newSynapsCountP	10
activationThresholdP	1-9
minThresholdP	4
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

Table 6.5: This table shows the parameter values to the result for Training with noise

Parameter	Value
numberOfPatterns	2
numberOfSubPatterns	3
angularDisplacement	0-6.3
region	0
numberOfColumnsP	200
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	16
newSynapsCountP	5
activationThresholdP	2
minThresholdP	2
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

Table 6.6: This table shows the parameter values to the result for Increasing number of patterns

Parameter	Value
numberOfPatterns	1
numberOfSubPatterns	1-6
angularDisplacement	0.01+(randdouble()-0.5)/50
region	0
numberOfColumnsP	200
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	16
newSynapsCountP	5
activationThresholdP	2
minThresholdP	2
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

Table 6.7: This table shows the parameter values to the result for Validation error

Parameter	Value
numberOfPatterns	2
numberOfSubPatterns	3-13
angularDisplacement	0.01+(randdouble()-0.5)/50
region	0
numberOfColumnsP	200
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	16
newSynapsCountP	5
activationThresholdP	2
minThresholdP	2
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

Table 6.8: This table shows the parameter values to the result for Stabilization of output

Parameter	Value
numberOfPatterns	1
numberOfSubPatterns	1-17
angularDisplacement	0
region	0
numberOfColumnsP	200
numberOfCellsP	1
numberOfSpatialSynapsesP	400
minOverlapP	16
newSynapsCountP	5
activationThresholdP	2
minThresholdP	2
iterationAverageP	200
connectedPermP	0.2
permanenceDecP	0.02
permanenceIncP	0.05
inhibitionRadiusP	80

References

[1] J. Hawkins and S. Blakeslee. *On intelligence*. Owl Books, 2005.

[2] N. Kondo, U. Ahmad, M. Monta, and H. Murase. Machine vision based quality evaluation of iyokan orange fruit using neural networks. *Computers and Electronics in Agriculture*, 29(1-2):135–147, 2000.

[3] T. Morimoto, T. Takeuchi, H. Miyata, and Y. Hashimoto. Pattern recognition of fruit shape based on the concept of chaos and neural networks. *Computers and electronics in agriculture*, 26(2):171–186, 2000.

[4] Numenta. Hierarchical temporal memory including htm cortical learning algorithms. 2010. [http://www.numenta.com/htmcla.php, Online; accessed 17-10-2011].

[5] W.C. Seng and S.H. Mirisaee. A new method for fruits recognition system. In *Electrical Engineering and Informatics, 2009. ICEEI'09. International Conference on*, volume 1, pages 130–134. IEEE, 2009.

[6] R.N. Shebiah. Fruit recognition using color and texture features. *Journal of Emerging Trends in Computing and Information Sciences*, 1(2):80–89, 2010.

[7] Y.X. Sun, C.M. Zhang, P.Z. Liu, and H.M. Zhu. Shape feature extraction of fruit image based on chain code. In *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR'07. International Conference on*, volume 3, pages 1346–1349. IEEE, 2007.