Open Source Project

Neocortex

User Guide

Neocortex User Guide

Summary	Neocortex version 1.4 User Guide	
Document Control	The document is identified by the version and date/time of saving.	
Status	Issued	
Author	David Green	Version Date 14/12/2008
Reviewed by	David Green	Date 14/12008
File Version	NeoV1.4 User Guide 1_0.docx	
Main Changes at this version	General responses to comments.	
Change forecast	N/A	

Table of Contents

Subject and Audience 3
- Acknowledgements 3

Introduction 4

The Model 5
- Basics of the Memory-Prediction Theory 5
- Architecture of the Bayesian Model 6
- Learning 6
- Recognition 7

Running the Model 8
- Learning results 9
- Recognition 9

Setting Parameters 12
- Setting the Number of Regions 12
- Forgetting Memories 13
- Constraining Memory Size 13
  - Performance 15
- Further work 15
- Constraining Saccades during Learning – More work 15

References 16
- Abbreviations and Glossary 16
- Document Cross References 16

Table of Figures

Figure 1 Tree-shaped hierarchy. 6

Subject and Audience

This document is a user guide to the Neocortex program developed under the Source Forge Neocortex project [12].
The intended audience for this document is people who have a general interest in AI and those who wish to study specifically the use of Bayesian inference for hierarchical memory models of the mammalian brain.

It is assumed only that the reader is generally conversant with AI concepts. No prior knowledge of the model is assumed.

Acknowledgements

The core model, along with the original interface using Borland C++ Builder, was released by Saulius Garalevicius in 2005.

David Green has contributed a new partial framework of Neocortex, a pixel editor based on an example in [13], the multi-platform user interface and more options for the model itself. Also he has written a Developer's Guide [9].

The user interface uses the open source (GPL) version of the cross platform toolkit Qt (Owned by Nokia Corporation).

The Neocortex development team thanks Dileep George for publishing the Pictionary project (used to be available at http://www.stanford.edu/~dil/invariance/), which inspired several important ideas used in the Neocortex.

We also thank Numenta for their many interesting and useful white papers on Numenta web site [11].

Introduction

Artificial Intelligence has a strong element of pattern recognition.

The two basic methods of pattern recognition are:

- Explicit analysis of the patterns using pre-defined specific algorithms.
- Learning from sets of patterns presented to the system using a general algorithm.

It is the latter approach that Neocortex and similar systems use to recognise patterns.

The aim of using the learning approach is that the AI system can act autonomously and learn like an animal might learn.

The name 'Neocortex' is meant to indicate a biologically inspired model although this is to be taken as a metaphor and not literally. Neocortex is not a direct analogue of the brain, but an approximation of the unified cortical algorithm described by Hawkins and Blakeslee [2].

Neocortex models the function of some of the key large scale structures of the real neocortex of mammals.

In other words we do not model individual neurons or cortical columns, but rather work on the level of, say, the V1, V2, and V4 regions of the visual cortex as well as hippocampus.

There is nothing restrictive about this approach. It provides a superstructure into which any detailed processing methods can be placed.

The current status of the model is of a training and research tool. It allows you to explore the effect of varying certain important aspects of the model such as the number of levels (e.g. thinking of V1, V2 etc. regions), the size of each level and how one level maps onto another (the model is a hierarchy among levels).

The Model

Basics of the Memory-Prediction Theory

[1]The memory-prediction theory[1] focuses on the functioning of the mammalian neocortex - a thin multi layered sheet of cells (many of them neurons) that covers much of the mammalian brain. Despite the thinness and apparently two-dimensional physical structure, the cells are connected and functionally arranged in a tree-shaped hierarchical network structure. We can identify layers in the hierarchy that represent functional regions of the neocortex. . Each region is further subdivided into a large number of subregions that operate in parallel.

All subregions of any region in the hierarchy are believed to perform the same basic operation (V. B. Mountcastle quoted in Bibliography of [2]). The inputs to the regions at the lowest levels of the cortical hierarchy come from our senses and are represented by spatial and temporal patterns. However, a system running the neocortical algorithm will be able to learn based on any kind of patterns that we choose to give it.

The neocortex learns sequences of patterns by storing them in an invariant form in the hierarchical structure. It recalls the patterns auto-associatively when given only partial or distorted inputs. The structure of stored invariant representations captures the important relationships in the world, independent of the details. The primary function of the neocortex is its ability to make predictions by combining knowledge of the invariant structure with the most recent observed details.

The regions in the hierarchical network are connected by multiple feedforward and feedback connections. Prediction requires a comparison between what is happening (feedforward) and what you expect to happen (feedback).

Each region is a collection of many smaller subregions that are connected to their neighbours indirectly, through regions higher up the hierarchy.

Each region learns patterns (or sequences of patterns), develops “names” (invariant representations) for the sequences it knows and passes these names to the next region higher in the cortical hierarchy.

As a region of cortex learns sequences, the input to the next region changes and the higher region can now learn sequences of these higher order objects.

These are the basics of the memory-prediction theory first published in 2004 by Jeff Hawkins and Blakeslee [2]. Since the theory is novel, few theoretical or practical studies of the memory-prediction framework exist today. There are two articles about modelling the framework by George and Hawkins [3] and [4]. Jeff Hawkins has founded a company called Numenta that is developing a new type of computer memory system modelled after the human neocortex.

We recommend you have a look at the white papers on the Numenta web site as they cover more in depth the way in which Numenta’s similar HTM model works [11].

In general, invariant pattern recognition has been an area of active research for a long time. These efforts were based on artificial neural networks as well as other technologies and usually used only the spatial image information to develop invariant representations [7, 8]. However, the performance of these models was limited and generalization questionable.

[1] Ref [1] is quoted here with minor changes

Architecture of the Bayesian Model

[2]The Neocortex program recognizes visual patterns in black and-white images. It uses a Bayesian model similar to the one described by George and Hawkins [3] as well as prototype materials and sample images posted online by the book authors.

The Bayesian model mirrors the large-scale biological structure of cortical regions and subregions connected in a functional hierarchy, but implements the functions of a subregion using Bayesian inference algorithms, not as a collection of columns and neurons. However, a fine mapping of these algorithms to the cortical anatomy has been found [3].

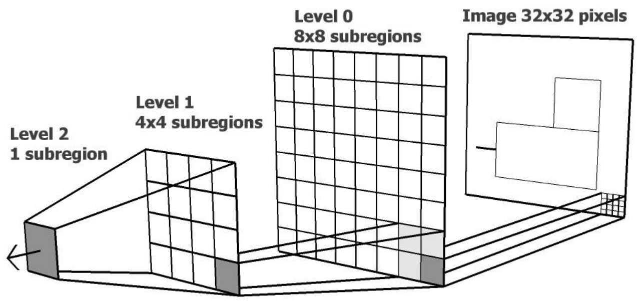


Figure 1 Tree-shaped hierarchy.
Each subregion in region 0 (level 0 in the diagram) receives image fragment of size 4x4 pixels. Each subregion in level 1 receives input from 4 children in level 0. A single subregion in level 2 receives input from all level 1 subregions

The hierarchical tree-shaped structure (Figure 1) contains several subregions in each region, with the subregions at the bottom of the hierarchy tiling the visual field and the single top subregion outputting the result of image recognition.

Learning

Each subregion in region 0 remembers in a set **B** all input patterns with their percentage of occurrences in the training data greater than a threshold **t**.

Then every pattern in **B** can be uniquely represented by its index **k**, which is called the "name" of the pattern. This name is sent as an input to the parent subregion in region 1.

Thus a subregion in region 1 is able to learn the most frequent simultaneously occurring combinations of patterns observed in its child subregions in region 0.

The name (index **k**) of the (currently active) pattern in the region 1 is fed back to its child subregions in region 0. This allows the subregions in region 0 to form a conditional probability distribution matrix (table) that encodes the probability of observing known patterns given the patterns observed by the parent subregion.

By the same process, subregions in region 1 memorize observed patterns coming from child subregions in region 0. Then they form a probability distribution matrix based on feedback from region 2. The process is repeated for all hierarchy levels.

The top region has a fixed number of learned image categories.

[2] Text on this page is quoted in part from [10]

Recognition

When we come to recognition, a pattern is presented to region 0.

Each subregion in region 0 infers the most likely explanation of the presented pattern based on its memories and the conditional probability table as described above. It then passes this explanation (belief) as a probability distribution up to region 1. So each subregion in region 1 receives beliefs from its child subregions in region 0 that best represent the input based on the previously learned data in region 0.

Incoming patterns to subregions in region 0 are compared to the previously learned patterns using the Hamming distance to determine the best match.

Notice that the pattern memory is common for all subregions of the same region. Subregions do not have their own pattern memory. That enables all   patterns to be shared across the entire region.

The process is repeated up the hierarchy until, at the top, we get a probability distribution over the known image categories. The category with the highest probability is recorded as the result of the recognition, while we can give reasonable alternative answers based on the confidence in the beliefs from the rest of the distribution.

Therefore, recognition of any given image can be viewed as finding the set of known patterns at each subregion that best explains the image in a probabilistic sense.

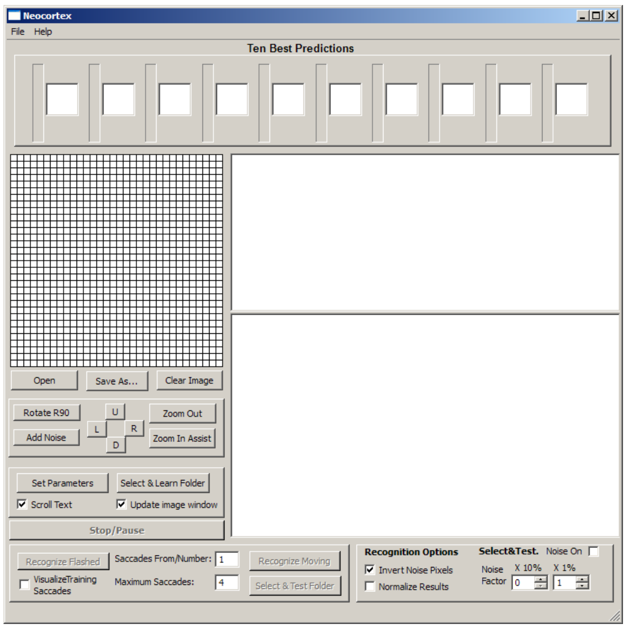
The conditional probability tables are constructed using Pearl’s Bayesian belief revision algorithm [6].

Running the Model

The program is supplied as a Windows executable file in a folder that you will have copied to your PC. Double click on the file Neo.exe to run the program.

You may not see "Neo.exe" but rather just "Neo". But you can still double click the file.

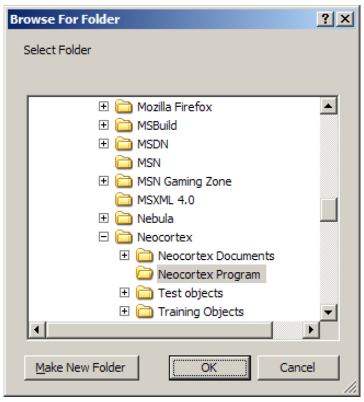
When the program starts you will see the following screen:



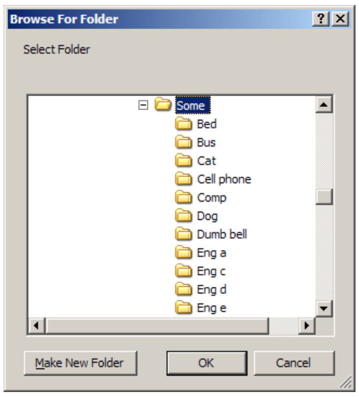
The main window of Neocortex

This is slightly formidable, but let's run through a simple example to get started.

Click the 'Select & Learn Folder' button (towards the bottom left of the window) and you will see a folder selection window something like this:



You may need to scroll down a bit to see the Training Objects folder.
Just click the '+' next to the Training Objects folder and continue navigating until you reach a folder called 'Some':

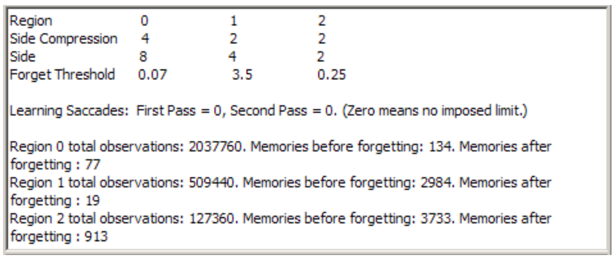


Then, with 'Some' highlighted as shown here, click 'OK' to start the learning process.

(Actual folders may differ slightly from these screenshots).

Learning results

When learning is finished you will see the results in the top right pane as follows:



```
Region              0            1            2
Side Compression    4            2            2
Side                8            4            2
Forget Threshold    0.07         3.5          0.25

Learning Saccades:  First Pass = 0, Second Pass = 0. (Zero means no imposed limit.)

Region 0 total observations: 2037760. Memories before forgetting: 134. Memories after
forgetting : 77
Region 1 total observations: 509440. Memories before forgetting: 2984. Memories after
forgetting : 19
Region 2 total observations: 127360. Memories before forgetting: 3733. Memories after
forgetting : 913
```

Things to note are:

- Three regions are in use, from region 0 at the bottom to region 2 at the top. The training images were presented on the 32X32 grid on the left.
- There are 64 (8X8) subregions in region 0, so every one of them accepts an image patch of 4x4 pixels ($32^2 / 8^2 = 4^2$).
- Similarly, going from region 0 to region 1 the data is compressed again to a 4X4 grid and then from region 1 to region 2 to a 2X2 grid.
- The top ‘region’ (not shown) is known, for want of a better term, as the ‘Hippocampus’.
- Moving down we come to the ‘Forget Threshold’. This prevents the memory structure from becoming too large. The setting is explained in more detail below. In summary, at each phase of learning, memories of rare patterns are erased. At region 0 the memories that have been observed less than 0.07 times the average memory for the region are forgotten and so on. The numbers have been set by trial and error to give good results. Theorising as to why a given set of numbers gives good results is part of working with this model. Later we see how to change these, and other, parameters of the model.
- ‘Learning Saccade’ limits are to do with how many eye movements over each example are used during learning. This will be explained later.
- Finally, for each region you see the total number of observations, how many memories were formed and the number retained after forgetting (see above).

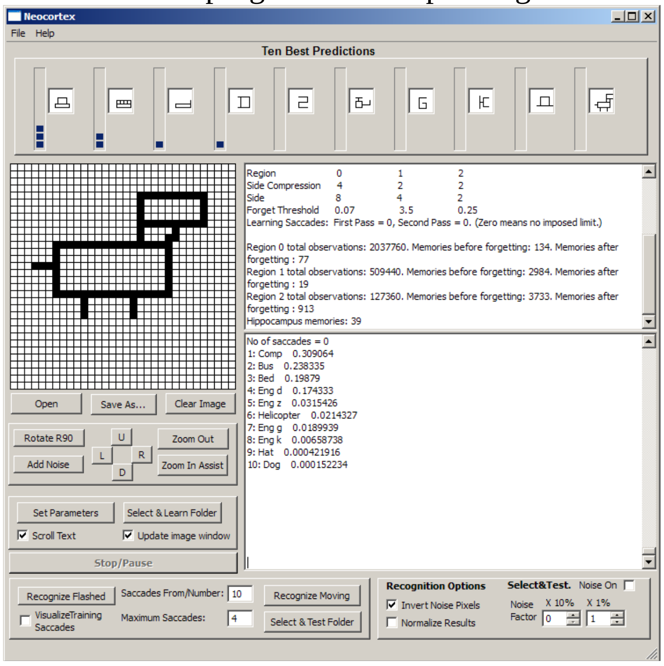
Now let’s see how good it is at recognising images.

Recognition

After learning, just for fun try drawing an image of a dog in the top left window.
It works much like any other ‘paint’ program as follows:

- Hold down the left mouse button continuously whilst moving the mouse over the grid (you have to start with the mouse pointer on the grid before pressing the button). This will draw a line wherever the pointer goes.
- You can, similarly, erase points by means of the right mouse button.
- There are some control buttons under the grid that you can play with if you wish – it’s fairly self explanatory what they do.
- Notice that you can use the ‘Open’ button to select, by means of the usual file dialog, an image from suitable ones on the disc. In this case you can easily get to the training data in the “Objects” folder.
- Once you have done this, press the ‘Recognize Flashed’ button at bottom left of the window.

In the drawing window below, is a very beautiful drawing of a dog.
I pressed the 'Recognise Flashed' button.
This caused the program to attempt recognition of the static image.



```
Region              0           1           2
Side Compression    4           2           2
Side                8           4           2
Forget Threshold    0.07        3.5         0.25
Learning Saccades:  First Pass = 0, Second Pass = 0. (Zero means no imposed limit.)

Region 0 total observations: 2037760. Memories before forgetting: 134. Memories after
forgetting : 77
Region 1 total observations: 509440. Memories before forgetting: 2984. Memories after
forgetting : 19
Region 2 total observations: 127360. Memories before forgetting: 3733. Memories after
forgetting : 913
Hippocampus memories: 39
```

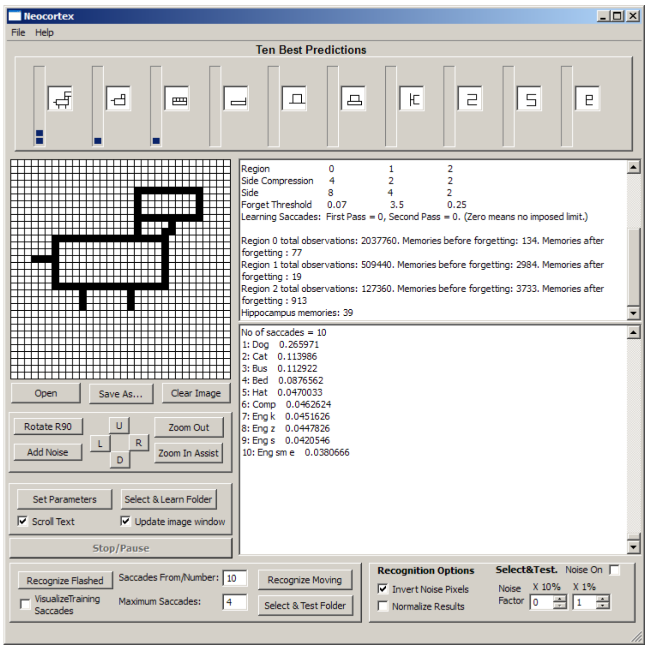
```
No of saccades = 0
1: Comp   0.309064
2: Bus   0.238335
3: Bed   0.19879
4: Eng d   0.174333
5: Eng z   0.0315426
6: Helicopter   0.0214327
7: Eng g   0.0189939
8: Eng k   0.00658738
9: Hat   0.000421916
10: Dog   0.000152234
```

Saccades From/Number: 10
Maximum Saccades: 4
Noise X 10% X 1%
Factor 0 1

Things to note:

- Yes the program has little clue!

Here is a run using the same parameters but with a moving image.



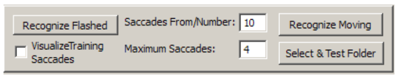
```
Region              0           1           2
Side Compression    4           2           2
Side                8           4           2
Forget Threshold    0.07        3.5         0.25
Learning Saccades:  First Pass = 0, Second Pass = 0. (Zero means no imposed limit.)

Region 0 total observations: 2037760. Memories before forgetting: 134. Memories after
forgetting : 77
Region 1 total observations: 509440. Memories before forgetting: 2984. Memories after
forgetting : 19
Region 2 total observations: 127360. Memories before forgetting: 3733. Memories after
forgetting : 913
Hippocampus memories: 39
```

```
No of saccades = 10
1: Dog   0.265971
2: Cat   0.113986
3: Bus   0.112922
4: Bed   0.0876562
5: Hat   0.0470033
6: Comp   0.0462624
7: Eng k   0.0451626
8: Eng z   0.0447826
9: Eng s   0.0420546
10: Eng sm e   0.0380666
```

Saccades From/Number: 10
Maximum Saccades: 4
Noise X 10% X 1%
Factor 0 1

To the right of the 'Recognize Flashed' button is a small text entry box labelled 'Saccades From/Number'.



Saccades From/Number: 10
Maximum Saccades: 4

This can be set to a number, say 10, and if you then click the 'Recognize Moving' button to its right a wonderful thing happens. You see the image moving by 10 steps and the results are often, as in this case, a lot better.

This movement is what we mean by 'saccades'. Saccades are used in learning to ensure that the model sees many possible positions of the image and during retrieval to improve recognition accuracy. If you check the 'Visualize Training Saccades' checkbox you can see the saccades operating when training is in progress.

Finally, we can test a whole set of images with a range of saccades.
If you are running the model and have trained it as above, set the Saccades From/Number back to 1, and the 'Maximum Saccades' to 4.

Then click the 'Select & Test Folder' button.

You see the folder selection dialog with the 'Some' folder already selected (clever eh?). Click OK.

Starting with a single saccade, each image in turn is tested and recognised (or not) then the program quotes, in the upper right pane, the recognition accuracy (the fraction of images correctly classified). The process repeats for 2, 3 and 4 saccades: Results are:

```
Testing 1 saccades: accuracy = 0.205128
Testing 2 saccades: accuracy = 0.333333
Testing 3 saccades: accuracy = 0.410256
Testing 4 saccades: accuracy = 0.589744
```

With 10 saccades the accuracy reaches 89% and with 20 saccades it is 92% for the 'Some' folder.

Better results can be obtained with other settings but this gives you an idea of how to work the model with the basic settings.

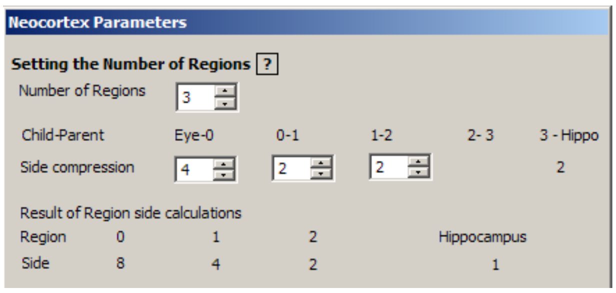
This may seem a slightly odd test – after all the system was trained with this very same data. However, what we are really doing is using an ideal case to determine the accuracy with which features have been memorized.

Test data is available (in the Test Objects folder) that contains the same categories. Results will not be as good with the distorted images that are supplied in the test data.

Setting Parameters

Neocortex allows you to vary key elements of the simulation and this gives a lot of scope for experiment with the model.
If you click on the ‘Set Parameters’ button you will see a rather complicated looking dialog. It is not really all that complex once you break it down into sections. So here are the four parts of the dialog explained.
Note that if you place the mouse pointer over the question mark that appears next to some of the labels this will show some help text for the section.

Setting the Number of Regions



Neocortex Parameters

Setting the Number of Regions ?

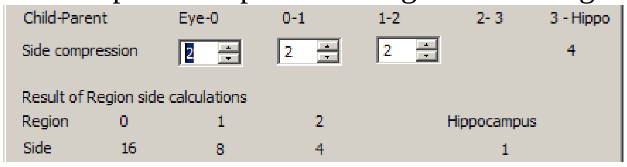
Number of Regions 3

Child-Parent	Eye-0	0-1	1-2	2- 3	3 - Hippo
Side compression	4	2	2		2

Result of Region side calculations

Region	0	1	2	Hippocampus
Side	8	4	2	1

You can change the number of regions and that will cause the region sides to change to reasonable defaults.
But you can change the defaults if you want to by changing the Side Compression parameter. It defines how many times the side of the child region (measured in subregions) is larger than the side of the parent region. So, for example, setting the side compression spinner for region 0 to two gives the following result:

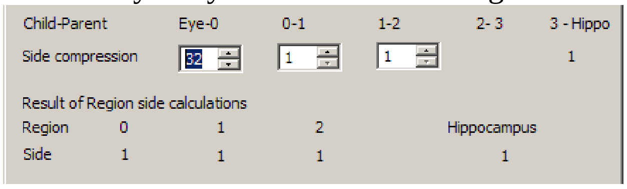


Child-Parent	Eye-0	0-1	1-2	2- 3	3 - Hippo
Side compression	2	2	2		4

Result of Region side calculations

Region	0	1	2	Hippocampus
Side	16	8	4	1

Some fairly ‘silly’ values can be set e.g.



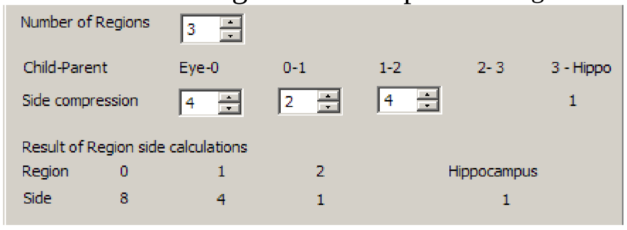
Child-Parent	Eye-0	0-1	1-2	2- 3	3 - Hippo
Side compression	32	1	1		1

Result of Region side calculations

Region	0	1	2	Hippocampus
Side	1	1	1	1

Note that, in this case, when the left hand spinner was set to 32 the other two spinners adjusted automatically.

Here are the settings that correspond to Figure 1:



Number of Regions 3

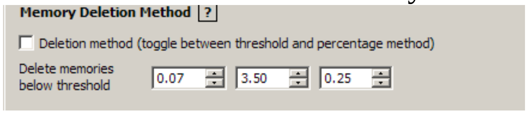
Child-Parent	Eye-0	0-1	1-2	2- 3	3 - Hippo
Side compression	4	2	4		1

Result of Region side calculations

Region	0	1	2	Hippocampus
Side	8	4	1	1

Forgetting Memories

During the first pass a potentially large number of patterns may be learned.
There is a maximum limit of 30,000 memories per region and if we exceed this level memories will be lost (new data will be ignored).
So we forget some of the memories in a more controlled fashion and the default way of doing this is to simply delete memories of patterns that have been seen less than a certain number of times (i.e. of below a certain 'frequency') after the first pass at each level (the feed-forward pass in which the patterns are memorized).
The number of times below which memories are deleted is the average frequency of the observed memories scaled by a factor that is set on the spinners shown here:

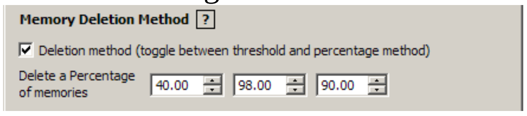


Memory Deletion Method ?

☐ Deletion method (toggle between threshold and percentage method)

Delete memories below threshold	0.07	3.50	0.25

For this example, at region 0 the memories that have been observed less than 0.07 times the average pattern frequency for the region are forgotten and so on.
Another way is to forget a percentage of memories. If you check the 'Deletion Method' check box you will be given a set of reasonable default percentages of memories to forget.



Memory Deletion Method ?

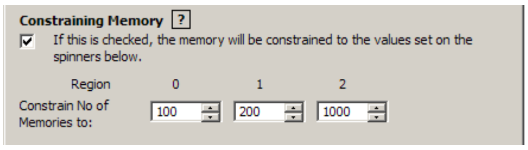
☑ Deletion method (toggle between threshold and percentage method)

Delete a Percentage of memories	40.00	98.00	90.00

In either case the method is very efficient but uses a lot of memory because the memory must be large enough to accommodate all the patterns else some will be arbitrarily lost.

Constraining Memory Size

Another solution to the problem of the memory filling up is to have a tight upper bound on the memory at each region (this actually does save memory in the program) and start discarding less important memories as soon as it gets full.
If you click the 'Constraining Memory' checkbox the 'Forgetting check boxes are greyed out and the upper limits of 30,000 on memory size are reduced to some reasonable defaults:

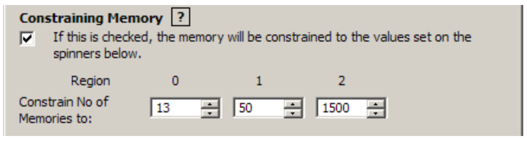


Constraining Memory ?

☑ If this is checked, the memory will be constrained to the values set on the spinners below.

Region	0	1	2
Constrain No of Memories to:	100	200	1000

For this test I set the Side compression to 2, 2, 2 and changed the default memory limits as follows:

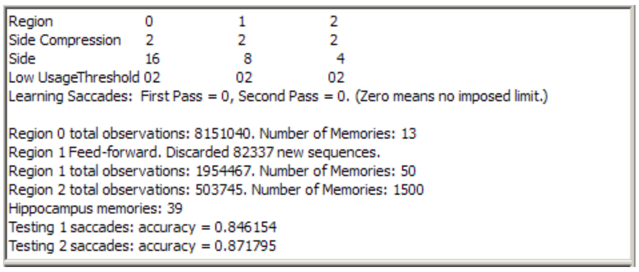


Constraining Memory ?

☑ If this is checked, the memory will be constrained to the values set on the spinners below.

Region	0	1	2
Constrain No of Memories to:	13	50	1500

I then ran the test using 'Select and Test Folder' on the training images (as previously) for 1 and 2 saccades.



```
Region              0            1            2
Side Compression    2            2            2
Side                16           8            4
Low UsageThreshold 02            02           02
Learning Saccades:  First Pass = 0, Second Pass = 0. (Zero means no imposed limit.)

Region 0 total observations: 8151040. Number of Memories: 13
Region 1 Feed-forward. Discarded 82337 new sequences.
Region 1 total observations: 1954467. Number of Memories: 50
Region 2 total observations: 503745. Number of Memories: 1500
Hippocampus memories: 39
Testing 1 saccades: accuracy = 0.846154
Testing 2 saccades: accuracy = 0.871795
```

Notice that because we have constrained memory we do not see the lines for forgetting at the first pass for each region.

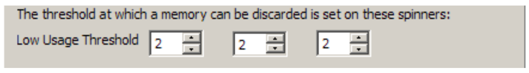
Instead we are told that the program has been unable to discard memory for new patterns during the first pass at region zero.

The results are, nevertheless, good:

For 6 and 7 saccades 100% accuracy is obtained (not shown – you can try it!)

When memory is too constrained, we are losing new and potentially useful patterns so we can rectify this either by increasing the Low Usage Threshold or increasing the amount of memory.

The threshold at which low usage memories are discarded is defaulted to 2:

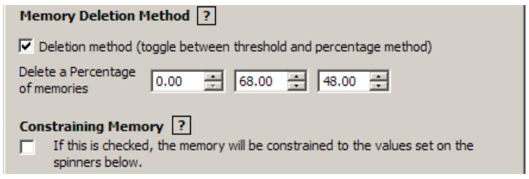


The threshold at which a memory can be discarded is set on these spinners:

Low Usage Threshold 2 2 2

And this can be increased to make it easier to find space.

Slightly better results are obtained from the default forgetting method with the following settings:



Memory Deletion Method

[✓] Deletion method (toggle between threshold and percentage method)

Delete a Percentage of memories 0.00 68.00 48.00

Constraining Memory

[ ] If this is checked, the memory will be constrained to the values set on the spinners below.

```
Region                        0         1         2
Side Compression              2         2         2
Side                          16        8         4
Forget Threshold              0         68        48
Learning Saccades: First Pass = 0, Second Pass = 0. (Zero means no imposed limit.)
Region 0 total observations: 8151040. Memories before forgetting: 13. Memories after forgetting : 13
Region 1 total observations: 2037760. Memories before forgetting: 134. Memories after forgetting : 43
Region 2 total observations: 509440. Memories before forgetting: 2304. Memories after forgetting : 1339
Hippocampus memories: 39
Testing 1 saccades: accuracy = 0.897436
Testing 2 saccades: accuracy = 0.897436
Testing 6 saccades: accuracy = 0.974359
Testing 7 saccades: accuracy = 0.974359
```

Performance

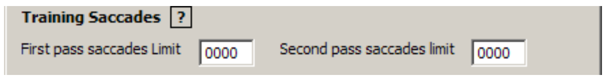
Performance is another factor. Often the more memories we use, the better the accuracy but the poorer the performance. In practical applications the trade-off is most important. Less accurate but faster recognition allows a machine to react to its environment and, as the results are sensed, further fast reactions can correct the previous results. Slow but highly accurate recognition could spell death to an animal that is being stalked by a predator!

Further work

The above results show that the approach of forgetting memories in bulk based on some fraction of the mean (the default method) can be better as far as the above tests are concerned. However, the advantage of discarding memories as we go along is that this may make it easier to add new learning later.
So more work is needed to run the model with various constraints and see if memory management can be improved.

Constraining Saccades during Learning – More work

This feature is the final one on the Parameters dialog.


Training Saccades ?
First pass saccades Limit 0000 Second pass saccades limit 0000

It allows you to set an upper limit on the number of saccades used for the first pass (pattern learning) and the second pass (feedback) respectively.

No results have yet been obtained as this is a new feature so there is plenty of room for experiment

References

1	Saulius Garalevicius, “Using Memory-Prediction Framework For Invariant Pattern Recognition”, Final Report, Dec 2005.
2	Jeff Hawkins, Sandra Blakeslee “On Intelligence” Henry Holt and Company, 2004
3	George, D., and Hawkins, J. 2005. A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex. In Proceedings of the International Joint Conference on Neural Networks 2005. Montreal, Canada: International Neural Network Society
4	Dileep George, Jeff Hawkins, “Invariant Pattern Recognition using Bayesian Inference on Hierarchical Sequences“, 2004
5	Discussion group on implementing cortex like learning systems: http://www.onintelligence.org/forum/viewforum.php?f=3
6	Judea Pearl, “Probabilistic Reasoning in Intelligent Systems“, Morgan Kaufmann Publishers, Inc., 1988
7	Kunihiko Fukushima ,Neocognitron: “A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position“. Biological Cybernetics, 36(4): 193-202, 1980
8	Maximilian Riesenhuber, Tomaso Poggio “Hierarchical models of object recognition in cortex”. Nature Neuroscience, 2(11): 1019-1025, November 1999.
9	Neocortex Developer’s Guide for Version 1.4 at [12]. http://neocortex.cvs.sourceforge.net/viewvc/neocortex/Design/
10	Saulius J. Garalevicius “Memory–Prediction Framework for Pattern Recognition: Performance and Suitability of the Bayesian Model of Visual Cortex”, American Association of Artificial Intelligence, 2007.
11	Numenta web site http://www.numenta.com/for-developers/education/general-overview-htm.php
12	The Neocortex project. Hosted by Source Forge at http://neocortex.sourceforge.net/
13	C++ GUI Programming with Qt 4, Jasmin Blanchette and Mark Summerfield, Prentice Hall 2006.

Abbreviations and Glossary

GPL	General Public License
HTM	Hierarchical Temporal Memory

Document Cross References

Ref	Name Version and Date	Title