

## Searching a string in a data base, method and speed performance

---

Courtesy of General Vision Inc, [www.general-vision.com](http://www.general-vision.com)

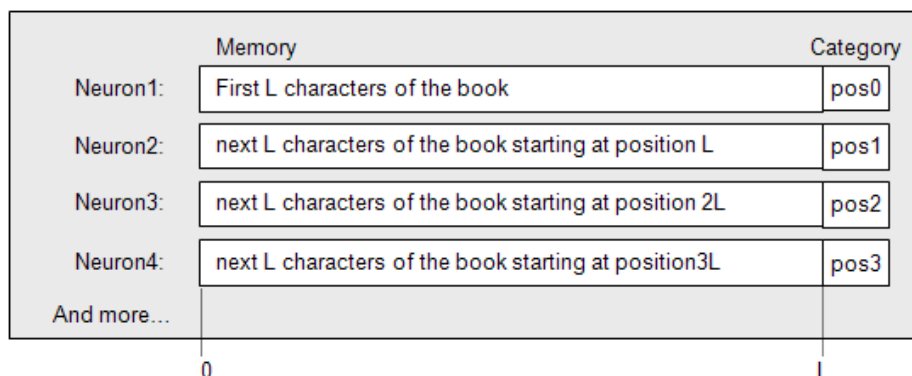
Subject The parallel architecture of the CogniMem neural network makes it possible to find a word or a combination of words in a body text in a number of clock cycles independent from the length of the body text.

Usage Text recognition, DNA sequence finding, bioinformatics, internet address matching, etc

### Step1: Learn a Body text

---

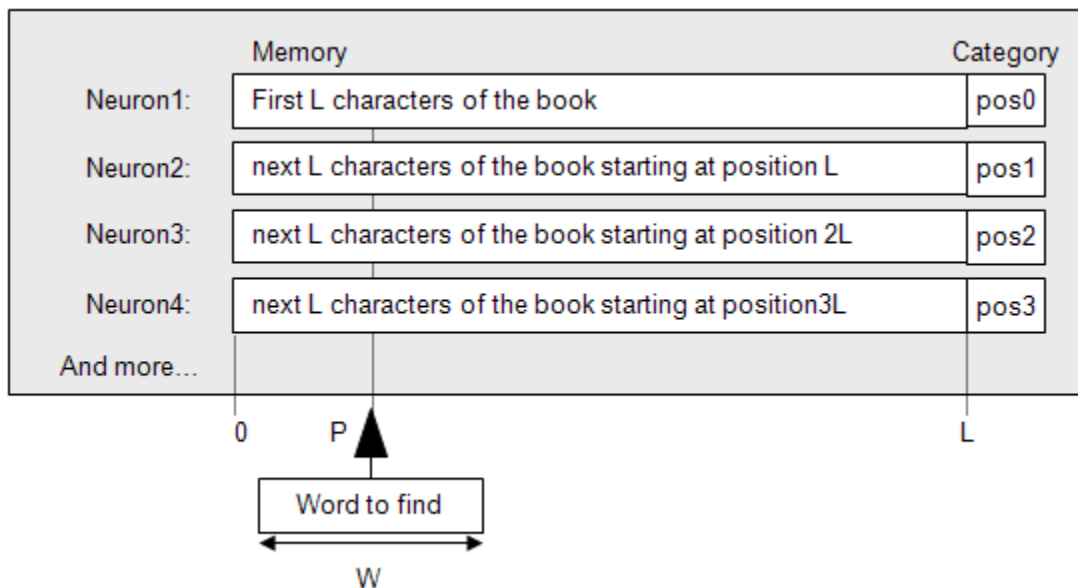
- 1) The body text of total length M is segmented into strings of L characters, with optional cropping and padding to avoid that a segment contains a split word.
  - a. The neurons of the CM1K have a memory of 256 bytes, so L must be less than or equal 256.
  - b. The CM1K chip features 1024 neurons, so a single CM1K chip can store up to 262,144 characters.
  - c. Multiple CM1K chips can be cascaded to extend the total capacity of the network per increment of 1024 neurons. A text composed of 1 million characters can be stored in 977 CM1K chips in parallel.
- 2) The segments are loaded in the neurons using the CM1K chip in Save and Restore mode along with a category equal to the number of the segment in the body text, or the position of the first character of the segment in the body text.
  - a. Loading a segment of L characters into the neurons takes only L+1 clock cycle in Save and Restore mode.
  - b. The loading time can be optimized by designing an architecture where a chain of CM1K chips can be loaded in parallel.
  - c. The category register is coded on 15 bits and can have a value between 0 and 32767. If the body text is composed of more than 32767 segments, the usage of the context register of the CM1K will allow entering 127 times more segments. In this case the position of the segment in the body text will be decoded by reading its context register and category register.
  - d. Changing the context will take one extra clock cycle every 32767 segments.



Register Transfer level loading sequence	cycles
Write NSR to set the network in Save/Restore mode	1
Write RESETCHAIN	1
Loop ( $M \div L$ ) times	$(M \div L) * ($
Write (L-1) COMP	(L-1)
Write LCOMP	3
Write CAT	1
End loop	)
Write NSR to set the network to Learn/Recognition mode	1
Total cycles =	$(M \div L) * (L + 3) + 3$
Body text of 1 million characters $M=1,000,000, L=256$	1,004,000 cycles
@27 Mhz clock	37 seconds

## Step2: find a word in the text

The recognition of a word of length  $W$  is made by sliding it over the  $(L-W)$  possible positions ( $P$ ) in a segment (i.e. neuron memory). At each position  $P$ , the  $W$  characters are broadcasted to all the neurons and the response of the firing neurons, if any, can be read per increment of 36 clock cycles to read their category and distance registers.



Remark #1: An important feature of the CM1K for this application is the ability to disable the distance evaluation unit of the neurons while moving in their memory cells. This allows starting the recognition at a given position  $P$  and also discarding some wild characters (\*, ?, etc) inserted in the word to recognize. This feature is the INDEXCOMP register.

Remark #2: The recognition time of a word does not depend on the length of the body text, but on the number of positions of the word in the neurons' memory and the number of firing neurons per position.

Remark #3: The response of the neurons is read per increasing order of distance and not position. This distance refers to the difference between the ascii codes of the word to recognize and the W characters in the neuron memory at the position P.

Register Transfer level recognition sequence	cycles
Loop from P=0 to L-W	$(L-W) * ($
<i>Move to the memory cell P in all the neurons at once</i>	
Write INDEXCOMP	1
<i>Broadcast the word of length W</i>	
Write (W-1) COMP	W-1
Write LCOMP	3
Read results of all k firing neurons	$k * ($
Read DIST	18
Read CAT	19
End of Loop	)
End of Loop	)
Total cycles =	$(L-W) * (W+3+(k*37))$
W=8, L=256, k=0	2,728 cycles
W=8, L=256, k=1 (assuming one firing for each position P)	11,904 cycles
@27 Mhz clock	101 $\mu$ sec
	440 $\mu$ sec

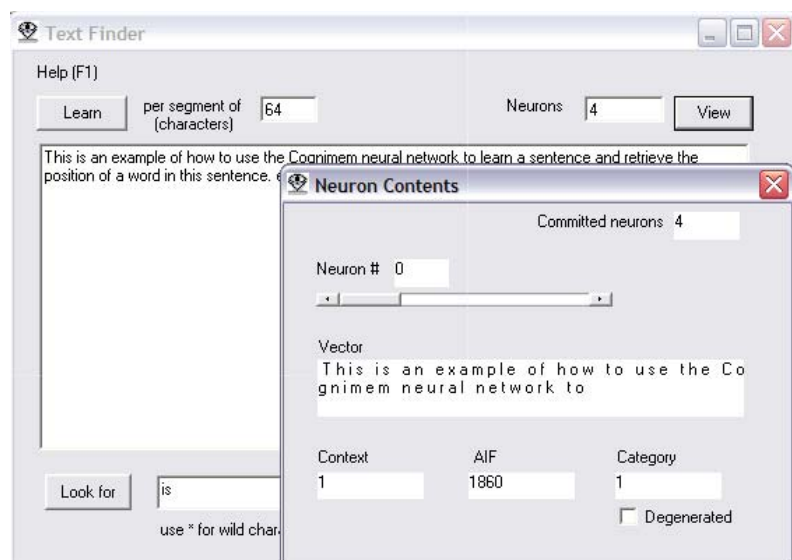
## Example of code

The code provided below is an extract of a Text Recognition example developed by General Vision. It is written in Visual Basic, but easy enough to understand and convert to other programming languages.

The Text Recognition example displays the following sentence as default:

“This is an example of how to use the CogniMem neural network to learn a sentence and retrieve the position of a word in this sentence. edit a new sentence and look for a word of your choice.”

You can build different knowledge bases by changing the number of characters per segment and clicking Learn. You can then review the portion of text held by each neuron. The category is the



---

position of the first character in the input text.

#### Knowledge 1 with L=256

Creates a single neuron containing the entire sentence "This is an example of how to use the CogniMem neural network to learn a sentence and retrieve the position of a word in this sentence. edit a new sentence and look for a word of your choice."

#### Knowledge 2 with L=64

Neuron1 contains "This is an example of how to use the CogniMem neural network to"  
Neuron2 contains "learn a sentence and retrieve the position of a word in this"  
Neuron3 contains "sentence. Edit a new sentence and look for a word of your"  
Neuron4 contains "choice"

---

## 1) learning the text of reference

```
'Clear the neurons before building a new knowledge base
CM_Init
TextRemaining = TextToLearn
Pos = 1
'switch to the save and restore mode
CM_Write CM_NSR, 16
CM_Write CM_RESETECHAIN, 0
While Len(TextRemaining) <> 0
'find position of the last characters of the last word in the next segment
  If Len(TextRemaining) <= L Then TempLength = Len(TextRemaining)
  Else
    For j = L To 1 Step -1
      If Mid(TextRemaining, j, 1) = " " Then Exit For
    Next j
    TempLength = j
  End If
'fill the vector and pad with spaces if needed
  For j = 0 To TempLength - 1
    Vector(j) = Asc(Mid(TextRemaining, j + 1, 1))
  Next j
  For j = TempLength To L - 1: Vector(j) = 0: Next j
'learn the vector and assign a category equal to the position of the first character in the sentence
  For j = 0 To L - 1: CM_Write CM_COMP, Vector(j): Next j
  CM_Write CM_CAT, Pos
  Pos = Pos + TempLength
  TextRemaining = Mid(TextRemaining, TempLength + 1, Len(TextRemaining))
Wend
'cancel the save and restore mode
CM_Write CM_NSR, 16
```

---

## 2) Finding a word in the text of reference

```
Hit = 0: TxtHit = ""
For Pos1 = 0 To L - WordLength - 1

'At each new position, reset the distance registers of all neurons without 'erasing their memory. Select the KNN classifier at the same time
  CM_Write CM_NSR, 32

'point to the component Pos1 and broadcast the characters of the word to the 'neurons. If this character is the wild character, simply skip to the neuron 'next component without updating the distance register.
  CM_Write CM_INDEXCOMP, Pos1
  For j = 0 To WordLength - 2:
    'Test for the wild component (code 42)
```

---

```

        If VectorToReco(j) <> 42 Then CM_Write CM_COMP, VectorToReco(j) Else CM_Write
        CM_INDEXCOMP, Pos1+j+1
    Next j
    CM_Write CM_LCOMP, VectorToReco(WordLength - 1)

    `look for the exact matches and convert the category into an absolute `position in the
    sentence.
    If CM_Read(CM_DIST) = 0 Then
        Do
            Pos2 = CM_Read(CM_CAT) And &H7FFF
            If Pos2 = &H7FFF Then Exit Do
            Matches(Hit) = Pos1 + Pos2 - 1
            Hit = Hit + 1
        Loop
    End If

    Next Pos1

```

Remark: The matches are listed per order of occurrence with respect to Pos1 which is not the position within the input sentence but the relative position within the neuron memories.