A NOVEL MEMORY-BASED PATTERN RECOGNITION SYSTEM

A thesis presented to the faculty of the Graduate School of
Western Carolina University in partial fulfillment of the
requirements for the degree of Master of Science in Technology.

By

Andrew Kerfonta

Director: Dr. Peter Tay
Assistant Professor
Engineering and Technology

Committee Members: Dr. Brian Howell, Engineering and Technology
Mr. Paul Yanik, Engineering and Technology

April 2013

To Ray Kurzweil, that he may find his way to wisdom.

ACKNOWLEDGEMENTS

I would like to give a huge thanks to my committee for all their help and encouragement. In particular, thank you to Paul Yanik for countless hours of discussion, instruction, and revision. I would also like to thank Drew Dimmery, Wesley Honeycutt, and Hongda Shen for discussions, proofreading, and feedback. Last, but certainly not least, I would like to thank Rachel for always supporting me and my obsessions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Algorithms

ABSTRACT

A NOVEL MEMORY-BASED PATTERN RECOGNITION SYSTEM

Andrew Kerfonta, M.S.T.

Western Carolina University (April 2013)

Director: Dr. Peter Tay

This thesis proposes a novel method for learning and pattern recognition. The algorithm presented relies entirely on memory arranged in a custom hierarchical data structure which shifts the workload from the processor to memory. The structure and functionality draw on biology and neuroscience for inspiration while not losing sight of the inherent strengths and limitations of modern computers. A hierarchy of learned nodes is built, stored, and used for recognition without the need for complicated math or statistics. Recognition and prediction are inherent to the hierarchy and require little additional computation, even for matching of partial patterns. The experiments and results presented empirically demonstrate the robustness of memory-based recognition of images.

CHAPTER 1: INTRODUCTION

Traditional AI systems rely on symbols and representations of data which are often many levels removed from reality. In order for this to be achieved, incoming data must be carefully pruned of all excess noise and variation. Unfortunately, the real world is not noise-free and often by assuming it to be, AI systems become severely limited in their usefulness. This is acceptable for playing simple games where the outside world can be safely neglected, but it becomes a problem in more complicated situations such as driving a car or interacting with humans.

These classical AI approaches attempt to explicitly define an intelligence through rules, logic, statistics, or grammars. This demands that an expert knowledge of the situation or environment be known beforehand so that all potential situations can be accounted for. Humans have no such prior framework to depend on. Previously learned knowledge must be applied and new knowledge learned on the fly. This bottom-up approach is in stark contrast to the traditionally top-down philosophy of AI.

This paper shows that a framework consisting of a data structure for arranging and organizing memories can not only accomplish the same tasks as *narrow AI* solutions, but that it can do so without the previously mentioned drawbacks. As will be shown, the design of this data structure draws on neuroscience, cognitive science, and computer science in order to maintain flexibility and efficiency. By relying on memory and its organization, traditionally CPU-intensive tasks such as comparisons and predictions are made trivial.

The remainder of this thesis is divided into five chapters. Chapter 2 is a literature survey which discusses the existing research in related fields including AI, machine learning, and neuroscience. Chapter 3 details the design of the proposed algorithm while Chapter 4 discusses the implementation. Chapter 5 describes experiments performed and gives the results obtained. Future work, improvements, and conclusions are discussed in Chapter 6.

CHAPTER 2: LITERATURE SURVEY

The field of artificial intelligence began by promising computers able to think and reason like a human, but after progress stalled, more realistic and attainable goals were set. Rather than an *artificial general intelligence (AGI)*, task specific, or *narrow AI* solutions were created. These systems focuses on being able to accomplish a single facet of intelligence such as vision, pattern recognition, movement control, or reasoning for a certain situation. Narrow AI proponents maintain that by building a complete set of working modules, an entire functioning mind can then be assembled by simply combining them.

2.1 Existing Techniques

Some steps have been taken to remedy this problem, largely through Situated and Embodied Artificial Intelligence (SEAI). Brooks' subsumption architecture was able to implement simple robots that could function with apparent ease in the real world without an explicit representation of knowledge [2]. It did this by employing a layered system of action components. However, the complexities of the interactions between components of the subsumption architecture make it impractical for larger and more complex tasks. Since all potential actions must be known and planned beforehand, a complete knowledge of all possible situations is required. The subsumption architecture has no means of emergence or growth, and so cannot learn from its environment.

2.1.1 Template Matching

Template matching performs recognition by comparing a predetermined template to new images. The template is moved through each new image until a match is found. In its most basic form, an exact match is required. In order to recognize objects despite variation, convolution or distance metrics can be used to determine the similarity between an image section and a template. The simplest of these is to calculate the squared difference between pixel values in

the template and test image:

$$R(x,y) = \sum_{x,y}(T(x,y) - I(x,y))^2 \tag{2.1}$$

Where R is the result, T is the template, and I is the image being tested.

This method can work well in very constrained situations, but has several disadvantages. For instance, if the object in the image is distorted for any reason, if the viewpoint has changed, or if there are any variations in the object itself this method quickly loses accuracy [12]. The proposed method aims to overcome these obstacles by examining an image starting from the smallest details (pixels) and proceeding to larger and larger groups of pixels. This allows for small parts of an object to match what has been learned even if visual changes have occurred.

2.1.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are perhaps the most widely studied form of neurally-inspired pattern recognition systems. A layered network of artificial neurons interconnected by weighted links is constructed and then trained using various techniques. During training, the weights are adjusted to yield the desired output based on given input data. Each neuron's output is given by an activation function based on weighted inputs from other neurons or the input data itself if the neuron is in the input layer [8, 13].

ANNs come in many varieties which are typically divided into either single layer (Figure 2.1) or multilayer (Figure 2.2) networks depending on whether the implementation contains one or more layers of neurons. A single layer ANN is relatively simple to configure and train, but can only classify information based on a linear regression. Multilayer networks, by contrast, are more complex both in terms of configuration and training, but are able to perform nonlinear regression-based classifications. In either case, the topology of the network, including nodes, layers, and connections, must be determined before training and classification can take place [8, 13].

This need to have the details of the network determined prior to use is a major drawback for usage in the real world. In particular, lacking a mechanism for adding new nodes while online means truly flexible classification and recognition is simply out of reach of ANNs. For

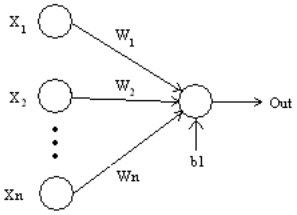


Figure 2.1: Single layer neural network [16]

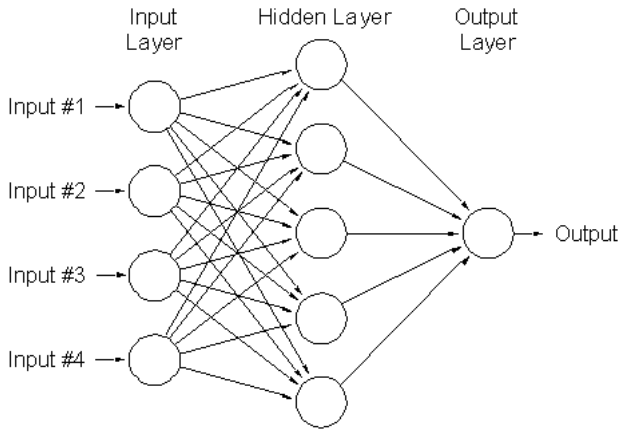


Figure 2.2: Multilayer neural network [17]

instance, a network could be configured and trained to recognize dogs, but if it was then desired that types of fruit be recognized, a complete reconfiguration would be necessary.

The artificial neurons that make up an ANN do not actually correspond to particular patterns. Instead, the overall function represented by the network is fit to data in order to achieve the desired outputs. This makes analysis of an existing network very difficult. Furthermore, as will be discussed below, neurons in the brain do correspond to specific input patterns.

2.1.3 Memory-based Techniques

Previous memory-based attempts have suffered from the inherent flaws of the sequential von Neumann architecture [21]. An exact match was often required, and searching the memory for the corresponding pattern could require examining every element of memory. This would be very time-consuming on a sequential processor. In order to cope with this problem, specialized architectures, such as the fully parallel Connection Machine, were used [21]. Reliance on a nonstandard architecture meant most techniques were not practical.

The human brain is able to perform a wide range of actions within as little as 200

milliseconds, implying that the total length of the longest path within the brain involves on the order of 100 neurons [7, 10, 19, 21]. The proposed data structure arranges learned patterns into a hierarchy. By doing so, recognition does not require checking every stored element, and therefore remains efficient on a sequential processor.

2.2 Neuroscience

Neuroscientists have discovered that in mammalian brains there is strong evidence to support a correlation between neural states and sensorimotor contexts [1, 3]. In other words, the brains of mammals create and store models of commonly perceived patterns. These patterns are classified as either declarative or procedural [20]. While declarative memories are snapshots of events or objects formed from a single instance, procedural memories are learned over time through multiple repetitions [5]. In this instance, the world as perceived through sensory input is its own best "representation." Attempting to preprocess the data only serves to abstract away reality.

The brain requires predictive models to function [14, 22]. Without them, reactions to real-world events would be far too slow due to the need to completely process new information before any conclusion can be made. By predicting an outcome from only partial information, mammals can react to situations even before they occur. These predictive models, built on procedural memories, have been theorized to be the basis of what may be termed "common sense [14, 20]." Prediction has even been referred to as the ultimate function of the brain [14].

Recent research has shown that the brain uses hierarchies of neurons to recognize images. It was demonstrated in [18] that when patients were shown pictures of various celebrities a single neuron would activate for each image. Converging towards this neuron is a hierarchy of thousands of interconnected neurons. The structure represents details from the smallest to the largest (the person recognized). This hierarchy, and more importantly the single high level neuron, changed when images of different celebrities were shown. Low level sections overlapped for some images, but the highest level was always unique.

Gintautas et al. detail a neurological foundation for lateral connections within hierarchies of neurons[10]. These allow one neuron, and those below it, to be a part of multiple

patterns that activate in parallel and help the brain recognize objects quickly. A feedforward architecture could also help account for the brain's ability to rapidly recognize objects [19].

Humans' ability to rapidly categorize objects and situations is not hard-wired at birth [20]. We learn common sense over many years, often with many errors along the way. This sort of emergent bottom-up hierarchy of knowledge has remained largely unexplored in machine learning and AI research. Systems that rely on a hierarchy of knowledge, such as the subsumption architecture, have no means of allowing new broader abilities to emerge [2]. Systems that can add to their existing knowledge often do so in a constrained or thoroughly abstract reality. Common sense, even in a limited way, remains elusive.

CHAPTER 3: DESIGN

In this chapter, the conceptual design of the proposed method is discussed. Implementation details are given in the following chapter.

3.1 Design Philosophy

The algorithm and associated data structure detailed in this paper were created to address many of the issues given above. To that end, the design centered around five principles:

1. Memory-based learning and recognition
2. Bottom-up learning
3. Built-in prediction
4. Expandability
5. Computational efficiency

The first point above is central to the entire system. By shifting away from computation-based learning and recognition, the inherent limitations of current methods may be significantly reduced or eliminated. This can be seen as the division between *concretely implemented (CI)* and *emergent* systems. CI systems involve rules, calculations, or architectures that are determined beforehand to match a specific problem or class of problems. For instance, the number of nodes and their connections within an ANN must be determined by the designer. This can lead, intentionally or unintentionally, to a bias in the system which can limit its effectiveness. To this end, the core of the algorithm is a custom data structure detailed conceptually below.

By using a bottom-up hierarchy for learning, large patterns emerge from the input data naturally. Since there are no rules or direction imposed during design, expanding is natural and effortless. This is vital for creating a flexible pattern recognition system.

Computational efficiency is often overlooked in the design of learning and recognition systems. Too often, a desire to create a neurally-inspired or optimal learning algorithm means that the limitations of available computers are ignored. The Von Neumann architecture is by nature sequential, so massively-parallel designs, including ANNs, will always be inefficient. Fully parallel computer architectures have been created [11], but none has become mainstream. The algorithm detailed in this paper is parallel in concept, but by tailoring operations to be as simple as possible, sequential speed is maintained. All comparisons use only integer values; there are no floating point operations in storage or recognition.

3.2 Data Structure Description

This algorithm relies almost entirely on a novel data structure we will term an Infinitely-Linked List (ILL) which was designed to accommodate learning and recognition based entirely on memory. The ILL can be viewed as a hierarchical web of interconnected nodes. At the lowest level of this hierarchy are the atomic pieces of raw input, i.e. pixels, characters, audio samples, etc. At progressively higher levels, these are combined into larger patterns to be used in future recognition. The components of the ILL, as well as its functionality, are detailed below.

3.2.1 Nodes

Nodes in the ILL represent patterns and link to one another. They can be seen as analogous to neurons in the brain. At the lowest level nodes will contain sensory information (e.g. image pixel color values, characters, etc.). Above this, pattern's are represented by linking multiple nodes together into a new node. The neurons in the human brain work in much the same way. Each neuron contains no explicit knowledge and is only a function of those that link to it. Because of this, neurons in the brain and nodes in the ILL need not have any knowledge of the global activity, and thus can be very simple. This also allows the ILL to be much more efficient in terms of memory and processor usage since no large pattern data is stored or compared, only memory addresses.

Nodes are individual objects which contain several important components. In the special case of lowest level nodes, the information is stored. Higher level nodes lack this structure.

All nodes contain expandable arrays of links to other nodes, as well as corresponding arrays of link weights. These links and weights will be discussed in more detail below.

The ILL contains layers of nodes based on the implicit size of the patterns represented. In a more concrete way, the size of patterns represented by nodes in a given level is determined by the type of input data. For instance, pixels are recognized and stored in matrices of increasing size (see Figures 3.1 and 3.2). While the size of patterns in a given level is fixed, there is no particular level at which full objects are represented. They do not suddenly form at a particular level, but instead will appear at whatever level corresponds to the size of pattern which they represent. Small objects will generally appear at much lower levels than large objects. There is also no explicit maximum number of levels. As long as new, larger patterns can still be created, the ILL may continue to expand.

In order to control expansion of the ILL, each node contains a counter that acts as a threshold. That is, only after a node in level n is seen k times will it create a node in level $n+1$. The threshold, k, can be varied to facilitate faster or slower creation of new patterns depending on the requirements of a certain task. For instance, dangerous situations may require learning from a single example.

3.2.2 Links

While nodes implicitly represent learned patterns in the ILL, the links between them are perhaps more central to its functionality. Every link has a weight associated with it that is incremented when that link is exercised. Since a node may potentially link to an infinite number of other nodes, this weight gives a likelihood to a particular connection relative to others. For instance, if node A has been recognized, and it links to nodes B, C, and D, it would be difficult to predict the next step without more information. However, if the link to node B has a much higher weight than the others, without more information it is reasonable to predict that B will be recognized next. Prediction will be discussed in more detail in section 4.

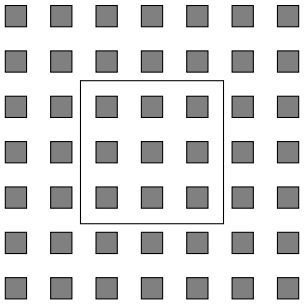


Figure 3.1: Building a single 3×3 node from constituent 1×1 nodes

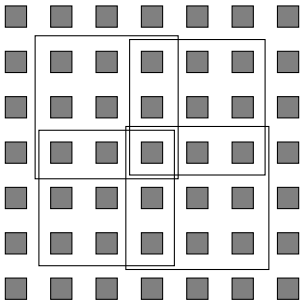


Figure 3.2: Building a single 5×5 node from constituent 3×3 nodes

3.2.3 Metanodes

The patterns represented by individual nodes increase in size in a set way, but real-world objects and other patterns are not so convenient. For this reason, a special type of node referred to as a metanode is used to contain multiple nodes of different sizes. When a group of standard nodes occurs together repeatedly, they are linked into a metanode. This is similar to the human brain, where neurons that often *fire* together *wire* together [5]. This process will be described in more detail later.

Metanodes also serve another similar purpose. When nodes, or metanodes, of multiple input types fire together, they are combined into a metanode. In this way, different "senses" are tied together to create context and allow for better prediction. For instance, the image of an apple would be connected to the word "apple", the smell of an apple, and any other information present pertaining to an apple (see Figure 3.3). It is intuitive that rich context and understanding might arise naturally at the metanode level.

Since metanodes represent context, they can be viewed as leading to the creation of the so-called *grandmother cell:* a single neuron which represents a specific complex object or concept. These individual neurons have been shown to exist at the top of a vast multi-sensory

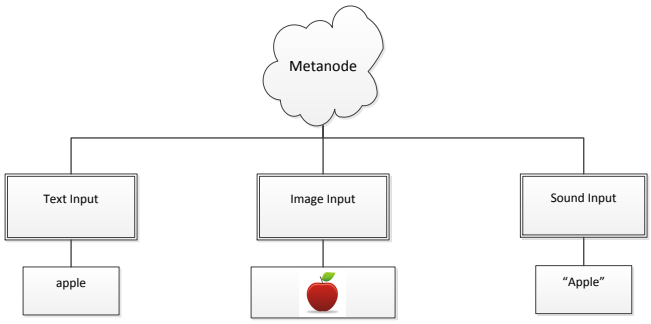


Figure 3.3: Example metanode

hierarchy [18].

3.2.4 Masterlist

The ILL has no means of directly accessing higher level nodes, so activation must start at the lowest level and propagate upwards. For this reason, a data structure referred to as the *masterlist* is used to hold links to all these first level nodes. For example, it would hold links to all observed characters or pixel-colors. The masterlist forms the connection between actual input data and the start of the ILL itself. In the human visual system, this is the role played by the optic nerve. Light hits photoreceptors creating the initial input data and triggering nerve cells in the optic nerve which connect to the brain and trigger vast hierarchies of neurons [19].

CHAPTER 4: METHODOLOGY

In this chapter, the operation and implementation details of the proposed algorithm are described.

4.1 Operation

The proposed algorithm centers around the ILL data structure, so its operation is mainly concerned with arranging patterns appropriately. Storing information in the ILL is carried out in three steps:

1. First level node creation (see Algorithm 1)
2. Higher level node creation (see Algorithm 2)
3. Metanode creation (see Algorithm 3)

Algorithm 1 First level node creation

```
repeat
  C = current pixel color value;
  if masterList[C] != NULL then
    return masterList[C]
  else
    Create newNode;
    masterList[C] = newNode;
    return masterList[C]
  end if
until Last pixel reached
```

Algorithm 4 shows how a recognition metric is generated.

Algorithm 2 Higher level node creation

```
for n < maxLevel do
  repeat
    n = currentNode.level
    create newNode
    newNode.level = n+1
    newNode.components = neighboring pixels of currentNode and currentNode
    if connection exists then
      strength++
    else
      link currentNode to newNode
    end if
  until Last pixel reached
end for
```

Algorithm 3 Metanode creation

```
level = maxLevel
repeat
  if currentNode.connectionToNeighbor ≥ threshold then
    Add both to metaNode
  end if
until Last pixel reached
```

Algorithm 4 Metanode recognition

```
for all metanodes linked from existing nodes do
  return constituent nodes present / total nodes
end for
```

4.2 Implementation

The ILL and its associated functions were implemented in C++[1]. The choice of language was based on the need for pointers as well as flexibility and control in constructing a custom data structure. The OpenCV library was used to access the camera and saved images as well as to read individual pixel values.

4.2.1 Nodes

Nodes were implemented as a custom class. The following list gives an overview of the member functions:

- `bool connectionExists(Node* n)` – Checks if a link exists between one node and another.
- `Node* connectionExistsMake(Node* n)` – Checks if link between nodes exists. If not, makes it.
- `Node* strengthen(Node* n)` – Strengthens a connection to another node.
- `int strengthenNeighbor(Node* n)` – Strengthens connection to neighboring node.
- `int getNeighborStrength(Node* n)` – Get strength of connection to neighboring node.
- `bool addConnection(Node* n)` – Add connection from this node to another.
- `bool addMetaConnection(Node* n)` – Add node to this metanode.
- `bool addTempMetaConnection(Node* n)` – Add temporary connection from this node to a metanode.
- `bool addUpwardMetaConnection(Node* n)` – Add connection from this node to a metanode.
- `bool compare(Node* n, Node* o)` – Check if nodes are identical.

[1] Source code available at http://akerfonta.com/ill.zip

- `bool compare(Node* n)` – Check if another node is identical to this one.
- `int expandMeta(Node* chrono, vector< vector<Node*> >& termNodes, masterListImage masterImg, int posx, int posy)` – Expand a metanode recursively.

Nodes also contain the following vectors:

- `vector<Node*> nodeAddress` – Addresses of nodes containing this node.
- `vector<int> strength` – Strengths of links to nodes containing this node.
- `vector< vector<Node*> > components` – Addresses of nodes contained in this node.
- `vector<Node*> metaParent` – Addresses of metanodes containing this node.
- `vector<int> metaStrength` – Strengths of links to metanodes containing this node.
- `vector<Node*> metaComponent` – Addresses of nodes contained in this metanode.

4.2.2 Masterlist

The masterlist was also implemented as a custom class. The following list gives an overview of the member functions:

- `int size()` – Returns current number of nodes in the masterlist.
- `bool nodeExists(Mat& current)` – Checks if a given node is in the masterlist.
- `bool writeToMaster(Mat& info, Node* address)` – Adds a given node to the masterlist.
- `Node* getAddress(Mat& info)` – Returns address to a node in the masterlist.
- `Node* createNode(Mat& info)` – Creates a new node and adds it to the masterlist.
- `void printMaster()` – Print masterlist stats to the screen.
- `bool compare(Node* n, Node* o)` – Check if nodes are identical.

The masterlist also contains one vector:

- `vector<Node*> list` – Addresses of nodes in the masterlist.

4.2.3 Links

As was shown above, links were implemented as vectors of pointers contained within the node and masterlist classes. C++ vectors were used rather than standard arrays in order to allow expansion whenever necessary during operation.

CHAPTER 5: EXPERIMENTATION

Multiple experiments with the ILL were carried out using both ASCII text and images as inputs. Text was used as a first test of the data structure and to demonstrate in a relatively simple manner how information is learned, organized, stored, and utilized. Since images are much more complex and contain far more information than text, they yield both a more challenging and more useful set of input data. Together, these two types of input show the inherent strengths and flexibility of the ILL data structure.

5.1 Text

Text is an ideal simple input for testing because there are a very limited number of atomic pieces: ASCII encoding only has 128 possible characters. Characters are fed in one at a time to the ILL which then organizes them into the hierarchy. It is important to note that all characters are treated with equal importance and no prior knowledge of the English language is assumed. Without pre-existing rules of any sort, the algorithm has no concept of spaces denoting words, punctuation, or capital letters beginning sentences.

For training, a selection of text was passed into the algorithm (see Appendix 1). The resulting patterns created in the ILL show common letter combinations of varying lengths that eventually lead to words. Figure 5.1 shows how following the strongest links starting at "a" leads to the word "and." If "a" is read in after training, it has links (in descending order by weight) to "an," "as," "at," "ar," and "a_." Since "an" is the strongest link, this is the prediction made. If "an" is then seen, the prediction will be "and" and so on. Actual links and strength values for these two tests are shown in Figure 5.1 and Figure 5.2, respectively. Predictions can be made to nodes in more distant levels by traversing subsequent strongest links. By following links to metanodes, prediction can be made to entire phrases, sentences, or even across input types.

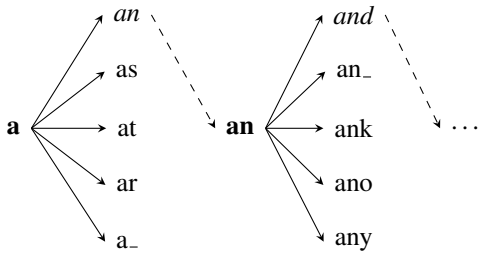


Figure 5.1: Text prediction

Link	Strength
an	22
as	21
at	20
ar	13
a	11
ab	10
ad	7
ai	6
al	5
ay	4
ak	3
af	3
ac	3
ap	3
av	2
ag	2
aw	1

Table 5.1: Text predictions from 'a'

Link	Strength
and	15
an	2
ank	1
ano	1
any	1
ana	1

Table 5.2: Text predictions from 'an'

Predicting small pieces of text is not terribly useful, outside of spell-checking, but it does verify and demonstrate the workings of the ILL.

5.2 Images

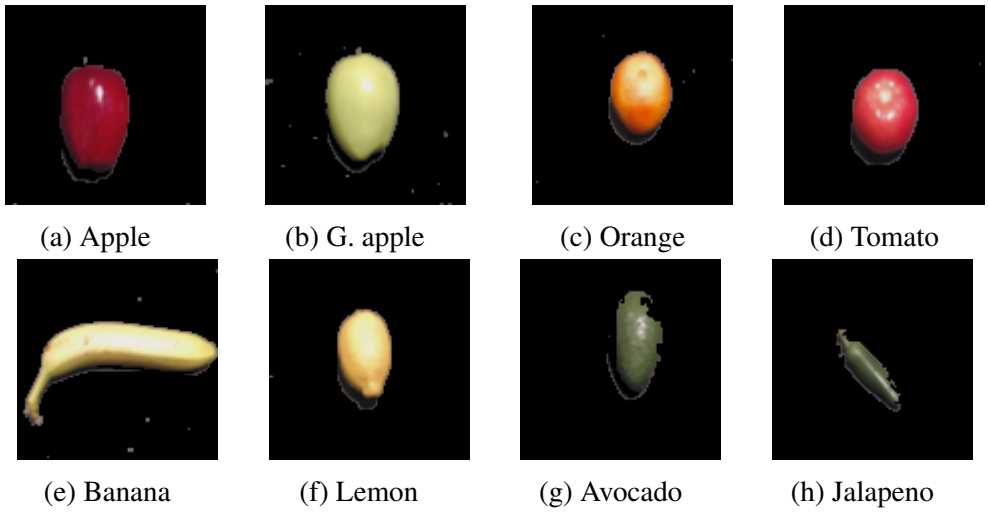
Images provide a useful input, but present a number of challenges. While textual data is inherently one-dimensional, images are two-dimensional. Because of this, nodes are no longer simple strings of information, but are, instead, square matrices. The first level in the ILL corresponds to single pixels read from the input while each progressively higher level's nodes are larger in all four directions by 1 pixel (see Figure 3.1). This can be seen as expanding the size of nodes outward evenly in all directions.

Images present more difficulties than increased dimensionality of data. Objects in an image can vary in both color and shape. The number of nodes increases rapidly. Colors can be reduced while still maintaining information integrity, but shape must remain unaltered. Otherwise, objects become unrecognizable. Clearly, square matrices will very rarely correspond directly to a single object since almost nothing in the real world is a perfect square. This is where metanodes become essential. Rather than being entire objects, nodes represent only parts of an object and are combined via metanodes to form the full entity.

Images can be read either from pre-existing files or from a camera. These images are then converted to a resolution of 100×100 pixels in order to ensure a consistent perspective. During training for the experiments in the this chapter, the background was ignored. This was done by making the area around the object a single color using OpenCV's floodfill function and then ignoring that color. While this is not necessary for functionality, since the training images used had the same solid background, the object and background would be learned together without this alteration.

5.2.1 Single Training Image

The first experiment using images as input involved four kinds of fruit (apple, green apple, orange, and banana) with only a single training image for each. After training, five new images of each fruit were presented to the algorithm. Results show the correct versus incorrect identifications (see Figure 5.3). Recognition took approximately 1 s. and achieved an accuracy of 85%.



(a) Apple (b) G. apple (c) Orange (d) Tomato

(e) Banana (f) Lemon (g) Avocado (h) Jalapeno

Figure 5.2: Fruit images

		Predicted			
		Orange	Apple	G. Apple	Banana
Actual	Orange	5	0	0	0
	Apple	0	5	0	0
	G. Apple	1	0	4	0
	Banana	0	1	1	3

Table 5.3: Single training image with 85% accuracy

5.2.2 Multiple Training Images

The second experiment using images as input used the same four types of fruit, but with three training images for each. After training, five new images of each fruit were presented to the algorithm. Results show the correct versus incorrect identifications (see Figure 5.4). Recognition took approximately 1 s. and achieved an accuracy of 90%.

		Predicted			
		Orange	Apple	G. Apple	Banana
Actual	Orange	5	0	0	0
	Apple	0	4	0	1
	G. Apple	0	0	4	1
	Banana	0	0	0	5

Table 5.4: Three training images with 90% accuracy

After the first two experiments, the algorithm was improved to handle 27 possible colors. All eight types of fruit were used to test the algorithm with three training images and five test images each. As is shown in Figure 5.5, this dramatically improved accuracy. Recognition took approximately 1 s. and achieved an accuracy of 97.5%.

		Predicted							
		Orange	Apple	G. Apple	Banana	Tomato	Lemon	Avocado	Jalapeno
Actual	Orange	8	0	0	0	0	0	0	0
	Apple	0	8	0	0	0	0	0	0
	G. Apple	0	0	8	0	0	0	0	0
	Banana	0	0	0	8	0	0	0	0
	Tomato	0	0	0	0	8	0	0	0
	Lemon	0	0	0	0	0	8	0	0
	Avocado	0	0	0	0	0	0	8	0
	Jalapeno	0	0	0	0	0	0	1	7

Table 5.5: Three training images with 97.5% accuracy

5.2.3 Facial Images

The final experiment used facial images of three individuals (see Figure 5.3). After three training images, three new images were presented for recognition. Figure 5.6 shows the results of this experiment as the percentage each image is recognized over the incorrect options. Recognition took approximately 1 s.

(a) Image A (b) Image B (c) Image C

Figure 5.3: Facial images

Face	Average Recognition Margin
A	33%
B	31%
C	5%

Table 5.6: Facial recognition results

5.2.4 Image Resolution

It was observed during testing that even though all images are converted to 100×100 pixels, the resolution of the original image had an effect on the algorithm's performance. Figure 5.4 shows the percentage of pixels recognized between subsequent images read from the camera at various resolutions. The object in the image was not moved or changed; variation was due only to noise from the camera. As can be seen, for this particular camera, 640×480 pixel resolution

(480p) yields the best recognition.

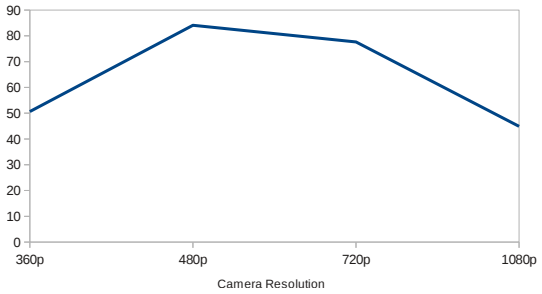


Figure 5.4: Camera resolution performance

5.3 Discussion

The results above show the strengths and abilities of the ILL data structure. The first set of experiments demonstrate that a purely memory-based pattern recognition system can not only recognize real world objects, but can do so with very little training despite object transformations. 85% of test images were recognized correctly, even with a single training image per fruit. The only case where this didn't happen for every test image was the banana. Due to its more complex shape, additional training is required. This was proven in the second set of experiments.

With multiple training images, the percentage of fruits recognized correctly rose to 90%. This illustrates clearly that, just like a person, the more an object is seen, the better recognition will be. In the case of recognizing objects from a live video stream, patterns will almost never appear instantaneously. More often, they will be seen from multiple angles or distances as they move closer to the camera.

When the possible number of colors was increased from 8 to 27, the results improved dramatically. These color depths were achieved by dividing the red, green, and blue channels into two or three parts, respectively. This was done because OpenCV has no means of reducing to below 256 colors. The third experiment gave a 97.5% recognition rate across eight different fruits. This dramatic increase in accuracy shows that greater color definition allows for better recognition.

The final experiment shows the flexibility and power of the algorithm by applying it to facial recognition. Two of the three faces were recognized with a high margin of around 30%

while the last had a much lower margin of 5%. This is thought to be due to the first two images having more distinctive features – hair, skin tone, etc. Another possibility is the presence of glasses makes the image more difficult to learn. Further experiments are necessary to confirm this.

For comparison, OpenCV's matchTemplate function was applied to the fruit dataset. This method was able to recognize the images of fruit with a confidence of approximately 70% if the object was not rotated. However, if it was rotated, confidences were roughly equal for multiple possible matches, showing that recognition in this situation was not possible. This highlights a major drawback of template matching for real world objects. This method took approximately 150 ms per image. While this seems fast, in order to replicate the third experiment above, 24 images would have to be tested which would give a total time of $(24 \text{ templates} \times 150 \text{ ms}) = 3600$ ms. The proposed method was able to do the same experiment in approximately 1 s.

This algorithm trades computational power for increased memory use. During experimentation, the maximum memory used was approximately 350 MB. Similarly, the human brain operates at a very slow speed, and instead relies on commodious memory to catalog and recognize patterns. With both long and short term storage being readily available in today's computers, emulation of the brain in this regard seems warranted.

Neurons are able to operate in parallel, but do so at speeds well below the KHz range [7, 10, 19, 21]. The von Neumann architecture is inherently sequential, although modern computers with multiple processing cores are now commonplace. Since clock speeds of up to 2 GHz and memory bandwidths of up to 5 GB/s are standard in even a modest personal computer, parallelism can be replaced with raw speed.

Even with the speed of today's computers, there are several optimizations which could significantly improve performance. The most beneficial of these would be sorting the arrays of links in each node by weight. Since the weight of a connection is implicitly a probability, it makes sense to arrange the links in order from strongest to weakest. This would speed up prediction in most cases. Sorting the masterlist was found to be extremely important to the overall speed of storage and recognition. By using hashes to jump directly to the required

starting node rather than searching through a list, the speed was increased by a factor of four.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

Future work will focus on rounding out the abilities of the algorithm to make real-world practical tasks possible.

6.1 Sequences

As it currently stands, the ILL is only capable of remembering and recognizing single patterns in text or images. Clearly, at least in vision systems, there is a need for learning of sequences of images or objects. When controlling a robot or other system which interacts with the world around it the change of patterns over time, for instance, a path through an obstacle course, are incredibly important.

Implementing temporal sequences within the framework of the ILL is thought to be viable. If one views the current hierarchy of patterns as two-dimensional, then time extends into the third dimension (see Figure 6.1). Patterns in time are built very much as current nodes are, except that the new temporal nodes are built from recognized standard nodes that occur adjacent to each other in time rather than space.

It is currently thought that the best approach for the handling of time would be to build sequences based on nodes occurring simply before or after one another, rather than at specific times relative to each other. By doing this, only the crucial steps in the pattern will be necessary to recognize the pattern while less important and potentially varying steps will be omitted. For example, when giving directions, only the turns are typically described. The parts of the route between them can vary so long as the critical turns are made. Even the exact timing between turns is generally irrelevant. Experiments will have to be carried out to verify whether this is the best method.

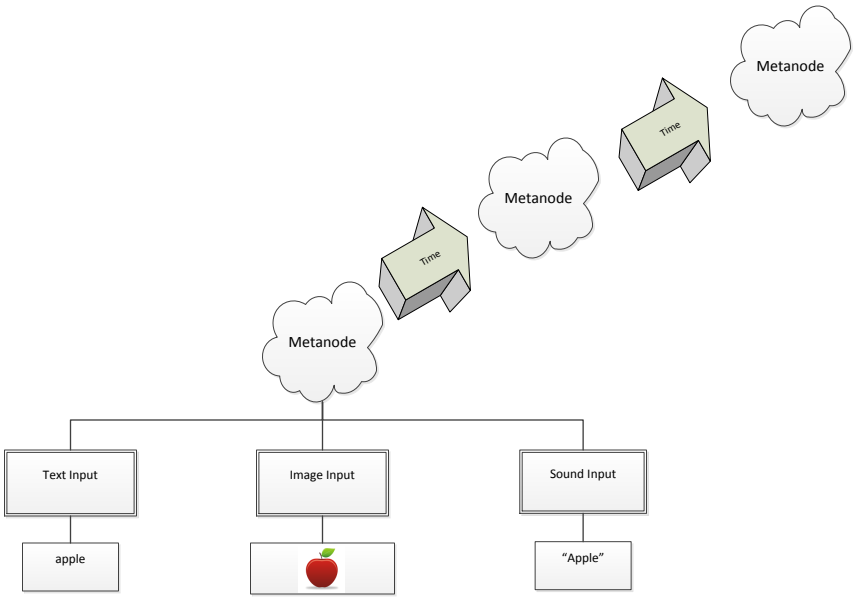


Figure 6.1: Example sequence

6.2 Inference

The ability to make connections based on inference is a useful tool for pattern recognition systems. Through inference, seemingly very different patterns can be related to each other and used. The appropriate response to a new set of inputs can be determined, allowing for much greater flexibility.

Like sequences of patterns, inference within the ILL data structure is conceptually simple. The act of making an inference can be seen as starting at a given node, and then moving upward until a metanode is reached that connects this node to another desired node. This may involve traversing a large number of levels, but just as in humans, some inferences are very obvious and some require many connections. For instance, if an apple is recognized for the first time, and is known to be a type of fruit, then by moving upwards to the ”fruit” metanode and back down to perhaps an orange, it is easy to determine that an apple should be eaten.

There are a number of things to consider when implementing inference in the ILL. A mechanism for finding the shortest, or perhaps the easiest, path between two nodes will be

essential. Also, it would be beneficial to have some means of learning common inferences so that shortcuts can be taken in future situations. This would greatly aid in allowing quick and intelligent responses to novel inputs.

6.3 Conclusion

The algorithm detailed in this paper shows the simplicity and inherent advantages of a hierarchical memory-based pattern recognition system. It deviates from the classical approaches to AI which rely on abstract symbols and representations. By doing so, it more closely follows many of the techniques used by humans to recognize and predict patterns. This allows the algorithm to remain simple, yet offer robust performance.

BIBLIOGRAPHY

[1] M. Bear, B. Conners, and M. Paradiso. *Neuroscience: Exploring the Brain*. Lippincott Williams and Wilkins, Baltimore, MD, 2nd edition, 2001.

[2] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[3] N. Burgess and J. O'Keefe. *Hippocampus: Spatial models, in The Handbook of Brain Theory and Neural Networks*. The MIT Press, Cambridge, MA, 2003.

[4] Lewis Carroll. *Alice's Adventures in Wonderland*. Project Gutenberg.

[5] Keith L. Downing. The predictive basis of situated and embodied artificial intelligence. *GECCO*, pages 43–49, 2005.

[6] Duda, Hart, and Stork. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2001.

[7] J.A. Feldman. Introduction to the special issue on connectionism. *Cognitive Science*, 9(1), 1985.

[8] Alexander I. Galushkin. *Neural Networks Theory*. Springer, 1st edition, 2007.

[9] Berry J. Gibb. *The Rough Guide to the Brain*. Penguin Books Ltd., 2nd edition, 2012.

[10] Vadas Gintautas, Michael I. Ham, Benjamin Kunsberg, Shawn Barr, Steven P. Brumby, Craig Rasmussen, John S. George, Ilya Nemenman, Luis M. A. Bettencourt, and Garret T. Kenyon. Model cortical association fields account for the time course and dependence on target complexity of human contour perception. *PLoS Computational Biology*, 7(10):1–16, 2011.

[11] W. Daniel Hillis. *The Connection Machine*. MIT Press, 1985.

[12] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao. Statistical pattern recognition: A review'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–36, 2000.

[13] Amit Konar. *Computational Intelligence*. Springer, 2005.

[14] R. R. Llinas. *i of the vortex*. The MIT Press, Cambridge, MA, 2001.

[15] Marvin Minsky. *The Society of Mind*. Simon and Schuster, 1985.

[16] NA. http://matlabgeeks.com/wp-content/uploads/2011/05/Perceptron.bmp.

[17] NA. http://www.cs.trincoll.edu/ ram/cpsc352/gifs/multilayer.gif.

[18] R. Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(23):1102–1107, 2005.

[19] Thomas Serre, Aude Oliva, and Tomaso Poggio. A feedforward architeccture accounts for rapid categorization. *PNAS*, 104(15):6424–6429, 2007.

[20] L. Squire and E. Kandel. *Memory: From Mind to Molecules*. Henry Holt and Company, New York, 1999.

[21] Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[22] Daniel M. Wolpert, R. Chris Miall, and Mitsuo Kawato. Internal modes in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, 1998.

Appendices

APPENDIX A: TEXT SELECTION

[4]Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and

book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled 'ORANGE MARMALADE', but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.