Neural Computation and Industry: Real-World Uses

Leon Barrett

December 11, 2006

1 Introduction

Artificial Intelligence has had disappointments before, but now there are again new and promising efforts arising.

In the early days of computers, it was believed that they would soon do everything that a human could. Researchers promised great breakthroughs, such as fluent natural language translation, that have still not occurred. In part, this failure stemmed from a reliance on logic, certainty, and computer-oriented data structures. Of course, it turns out that human intelligence is not nearly so based on logic as was once believed. In particular, humans must deal with uncertainty and confusion must at every moment.

Recently, a new revolution has been going on in the field of Artificial Intelligence. Based on results coming out of statistics and neuroscience, new models are seeing much more promising results. On the statistics side, Bayes nets and Support Vector Machines, among others, have given new and principled ways to learn from and interpret data. In the area of neuroscience, fresh insights into the massively connected, massively parallel human brain have given rise to models such as connectionist models and Boltzmann machines, which perform computation by combining many simple elements.

Of course, it is one thing to have clever new models that give principled results in a laboratory, and quite another thing to perform real-world tasks. In this paper, I take a look at "real-world" (non-academic) work in this area. I focus on how four new companies use neurally and cognitively inspired models to accomplish real-world tasks. These companies are Sightech, Numenta, SpikeNet Technology, and Imagine Engines Inc.

2 Sightech

Sightech, founded by Art Gaffin in 1983, is a company that makes a defect-detection product called PC Eyebot. Their product analyzes video to detect aberrations in its input; they manage this in realtime by virtue of an extremely simple algorithm. This algorithm works essentially by counting surprising features present in each image and triggering an alarm if there are enough such features.

Sightech's algorithm is based on insights derived from Art Gaffin's studies of yellow jacket wasps. Insects have crude eyes and even cruder brains, possessing vastly fewer neurons and synapses than humans do. Nonetheless, they are able to detect important features and even navigate by sight. Gaffin found that wasps seem to follow consistent paths to their destinations. When released in an unknown location, they fly in wide arcs until they encounter one of these paths, at which point they rejoin the remembered path.

Based on this behavior and the constraints on the wasps' brain size, Gaffin concluded that yellow-jacket wasps must use a simple feature mapping to determine their course of action. His idea, as I understand it, is approximately as follows: In response to photons exciting the wasp's eye, a pattern of input neurons becomes active. This activates the neurons in an intermediate layer that have enough

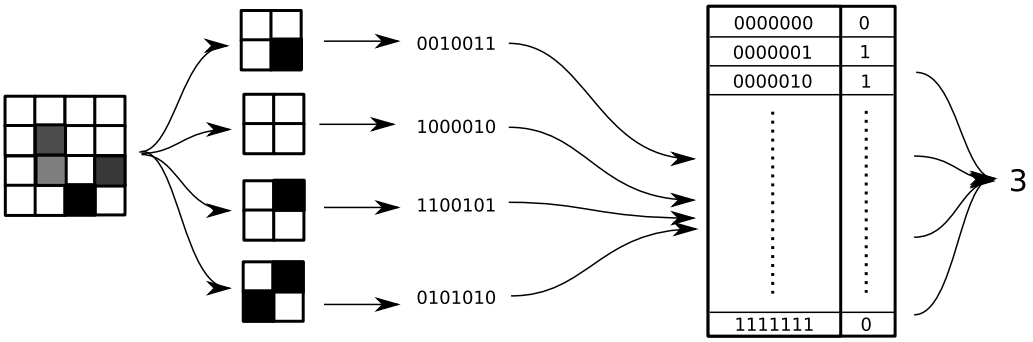

Figure 1: Diagram of how Sightech's algorithm uses features as memory addresses to determine whether features are novel.

synapses linked from the active inputs. These intermediate neurons thus represent features by encoding patterns in the input. Finally, the active intermediate neurons stimulate the output neurons to which they have synaptic links, triggering a behavioral response. The synaptic links act as an address, so the outputs are activated only by intermediate neurons with sufficient synapses to them. Essentially, visual features act as indexes into an array of actions.

2.1 Algorithm

Sightech's anomaly-detection algorithm is based on the insect vision model, and works as follows. First, they take the image and extract a set of "features" (more on this in a moment). They then perform a simple hashing operation on each feature, mapping the feature to a memory address. The memory at this location contains a marker: either a "0", indicating that no such hash has been seen before, or a "1", indicating that this hash has been observed. If there are enough "0" values, then that indicates that this image is uncharacteristic of the training data, so the system signals an anomaly. This algorithm is shown in Figure 1.

Publicly available information on Sightech's visual feature extraction is deliberately scanty, since this is a critical part of their algorithm. Their patent suggests thresholding 5-by-5 pixel regions of grayscale images, resulting in a 25-bit vector that can be used as a feature. However, from my conversations with Gaffin, it became clear that Sightech uses a much more complex set of features; for instance, the presence of two features near each other may be encoded as a third feature. Clearly, the performance of the anomaly-detection algorithm will be in large part determined by the usefulness of the chosen feature set.

2.2 Analysis

One point to consider is that the feature-hashing mechanism may not map each feature to its own memory address; there may actually be many features that point to the same location. We would like the feature's addressed memory slot to be a strong indicator of whether the feature was seen before. Of course, if the value in the memory slot is a "0," we can be certain that the feature was not seen before. However, if it is a "1," all we know is that one of the features that hashes to that address has been seen. Thus, we want to prevent this occurrence by keeping the density of observed features in the memory space low. This can be accomplished in two ways: by having a peaked distribution of features, so that many features are

never seen, and by having a large memory space. These constraints should be taken into consideration when designing a hash-lookup system like Sightech's.

While this algorithm seems perhaps too simplistic to be useful, it is its very simplicity that makes it useful for real-world problems. Admittedly, the system detects very crude patterns in the world. However, this is made up for by the fact that the system operates so quickly. For many tasks, crude patterns are sufficient to detect important information. For instance, in a bottle-manufacturing plant, a broken bottle will produce very different image patches than a normal bottle, and so this method will catch it. Then, the ability to scan many bottles in quick succession, at low cost, becomes a critical factor, making this technique very attractive in practice.

Sightech is definitely a place to look for real-world, functioning artificial intelligence. Its PC Eyebot seems to be an established, fully functional product, used in many manufacturing plants. Admittedly, its algorithm is limited to operate on the small domain of visual anomaly detection, but it is highly useful within that domain.

3 Numenta

Numenta, founded in 2005 by Jeff Hawkins and Dileep George, is the newest company of those discussed in this paper. It is developing a learning system based on Jeff Hawkins's model of the human neocortex. This model is derived from several key insights: First, the human cortex seems to have the same general structure throughout, suggesting that it runs a single algorithm. Second, it has a hierarchical linking structure (seen most clearly in the visual areas), with smaller "high-level" areas receiving sensory input from larger "low-level" areas. Moreover, there are at least as many "downward" connections in the hierarchy as there are "upward" ones. Finally, the real world is organized into temporal and spatial groups in a way that makes clustering inputs easy. Hawkins considered these and concluded that the cortex does hierarchical pattern matching and uses these patterns to predict its future inputs.

As Hawkins admits, these insights are not terribly controversial. The cortex is known to have a repetitive structure throughout, suggesting a common algorithm, with cells organized laterally into layers and vertically into columns. It is also known that there is a hierarchy in at least the visual regions of the cortex, with cells in low-level areas like V1 responding best to simple stimuli, and cells in higher-level areas like FFA (fusiform face area) responding to extremely complex stimuli, such as faces (see an experimental hierarchy diagram in Figure 2). Also, natural images are well-known to have a "1/f" correlation structure, with the strength of correlation between two pixels decreasing as the reciprocal of the distance between the pixels. In general, pixels near each other in space and time will tend to belong to the same object, and thus be related, while more distant pixels will tend to be unrelated; making this assumption will aid in automatic clustering. The most unusual claim that Hawkins makes is the assertion that the cortex is critically involved in prediction. This could explain many things, such as the high number of top-down cortical connections, and the way humans seem to have a natural tendency to discern causes in the world.

3.1 Model

Numenta's general approach is to use a hierarchical network of pattern-detecting nodes (see Figure 3). Each node gets inputs from the nodes below it (its children) and sends data to the one or more nodes above it (its parents). Because real-world data is clustered in space, and because the node's children are close in space, the inputs from the children will tend to be related. Similarly, because real-world data is clustered in time, the node's inputs will tend to be related to the inputs it got previously. So, assuming that it will observe these correlations, the node attempts to learn patterns and sequences in its inputs. Then, it summarizes its inputs into a pattern or sequence label and sends this up to its parent. So, each node works as a temporal-spatial pattern recognizer. Ideally, the node will learn patterns that represent

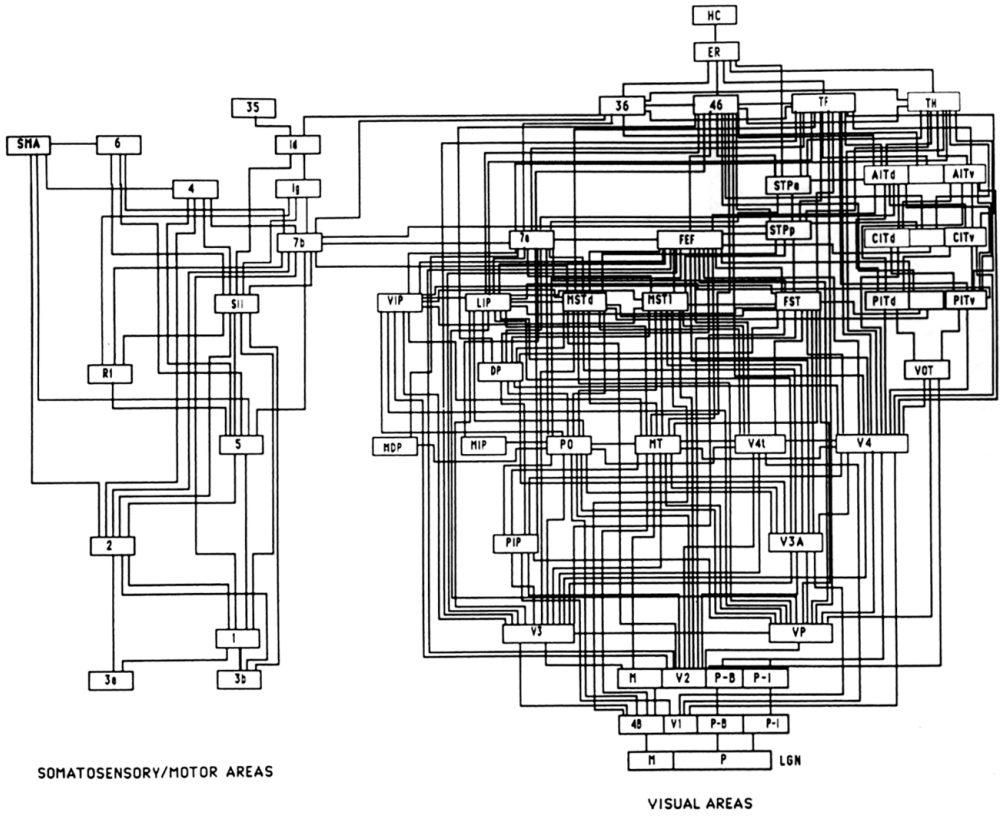

Figure 2: A hierarchy of the visual system, as observed in the macaque monkey.

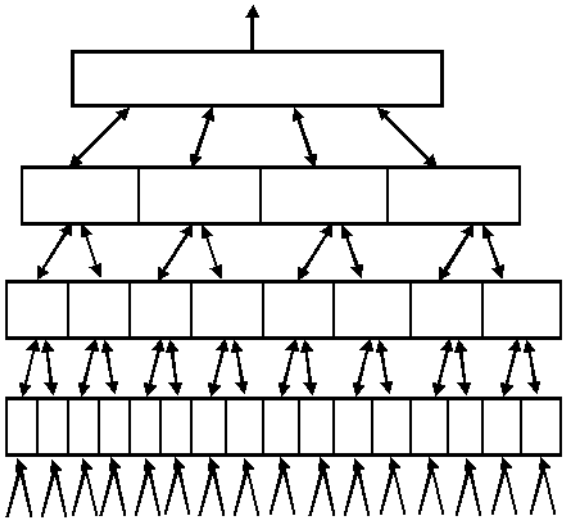

Figure 3: A hierarchy of statistical pattern-generating and -recognizing nodes (represented by boxes). Each node acts as a cause for its children, and is caused by its parent.

invariants on its input (e.g. an object remains the same, though the inputs may change as the view of the object changes), thus simplifying the input and summarizing it for the parent node.

3.2 Generative model

In Numenta's model, the collection of nodes may be viewed as a hierarchical, statistical, generative structure. The labels of each node are viewed as *causes* for its children below it. The model is called "generative" because one could use it to generate or predict its inputs. To generate inputs, one would start from the top of the hierarchy, having each node randomly choose a pattern based on the pattern of its parent. Statistically, then, each node chooses its value n from its parent's value p using the distribution $P(n|p)$. When this sampling is performed on every node, this results in a distribution on inputs I given the top node of the hierarchy, T. So, we can sample from $P(I|T)$, and, to do this well, we would like our model to correctly learn the distribution of inputs in the world.

In addition to modeling input generation, this model can also guess causes. When inputs are moving up the tree, each node calculates a distribution over its possible values given its children's values c_i by way of Bayes' Rule: $P(n|c) = \frac{1}{Z}\prod_i P(c_i|n)$, where Z is a normalization constant. Thus, each node calculates how likely it is to be in each of the patterns it has learned. To include temporal information, just expand the distribution of the node's value n to also depend on the node's previous value, n_{-1}. Then, we can calculate $P(n|c, n_{-1}) = \frac{1}{Z}P(n|n_{-1})\prod_i P(c_i|n)$. So, in general, this hierarchical model is used by passing inputs up from the bottom layer and calculating likely causes at each step (considering past states), thereby calculating what sort of top-level causes might have brought these inputs about.

It should be noted that this statistical framework corresponds in great detail to Hawkins's model of the neocortex. There is not space here to analyze this fully, but, for example, Hawkins proposes how specific functions (pattern detection, pattern inference) are performed by various cells and connections in the cortex. See Figure 5 for a few of these mappings, and see Chapter 6 of Hawkins and Blakeslee [2004] for more.

Although in theory, simply applying the Expectation Maximization (EM) algorithm to this probabilistic network should cause nodes to learn appropriate patterns, in practice it may not be so simple.

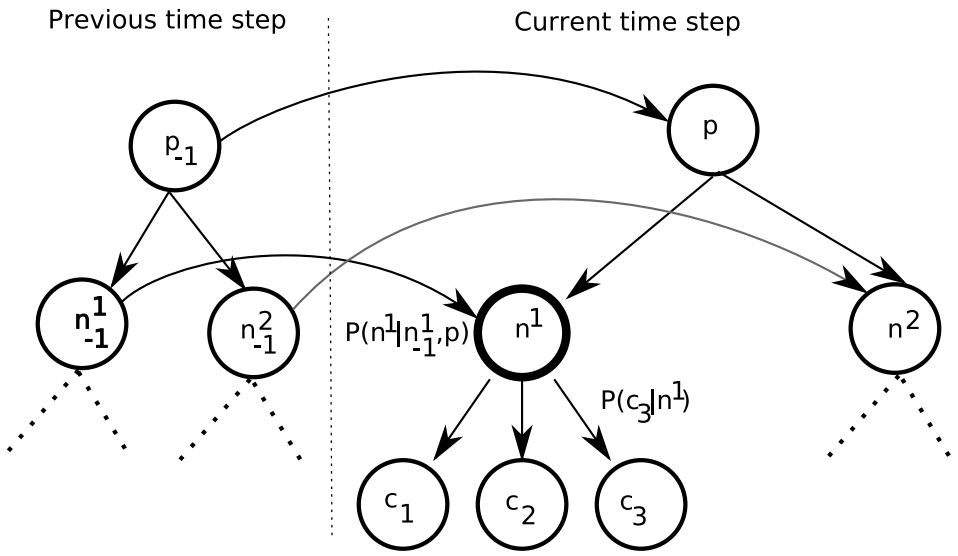

Figure 4: Numenta's hierarchical and temporal probabilistic model, focusing on node n^1, including its causes (parent p, previous state n^1_{-1}), sibling (n^2), and effects (child nodes c_i).

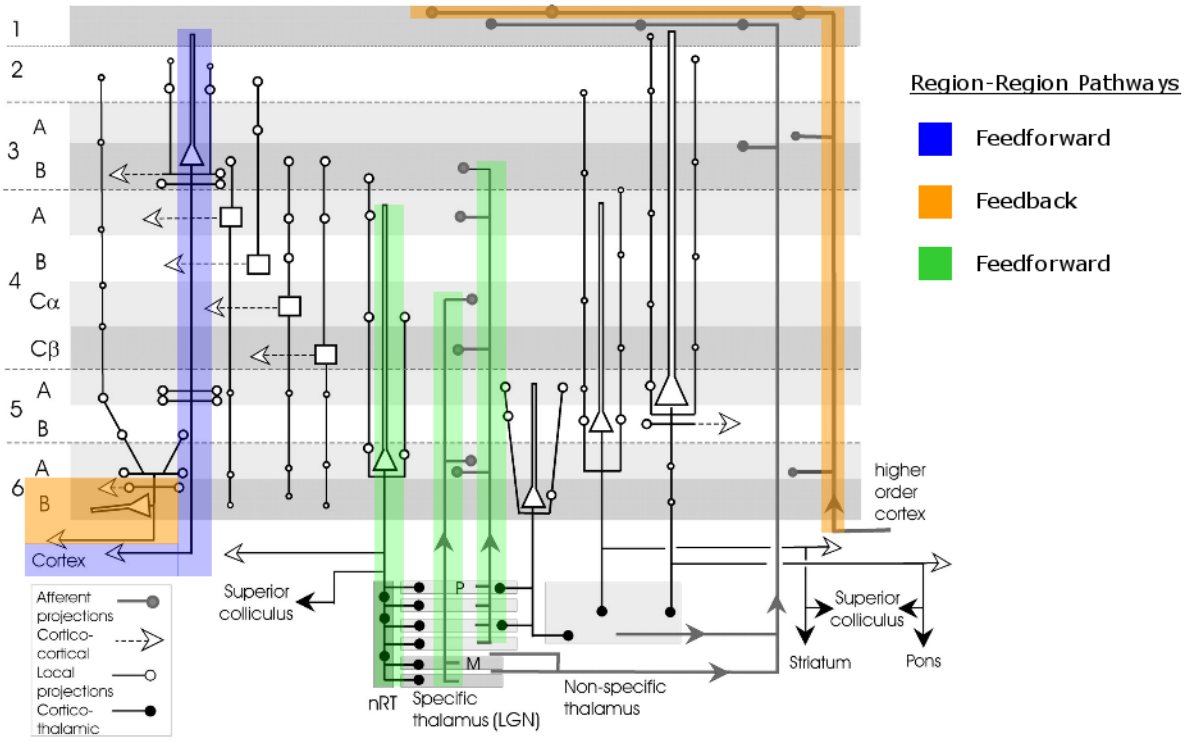

Figure 5: Part of how Numenta's model maps onto the layers and connections of the neocortex

In my experiments with EM, it is rare that the desired outcome simply pops out. Instead, it may be necessary to impose some sort of structure on the random variables. Thus, one of Numenta's main efforts has been to find ways to extract appropriate temporal-spatial patterns. [George and Hawkins, 2005] However, their current methods for tackling this problem are not publicly available.

3.3 Model progress

Numenta will be releasing this system for free, including source code for some components, in early 2007. However, for now, this only loosely qualifies as "real-world" artificial intelligence. Although it is believed to be practical enough to receive both investment and income from partner companies eager to see pre-release products, Numenta's model has not been seriously tested on anything more complicated than a simple vision task.

They trained a network on a series of moving binary-pixeled sketches, including images such as letters and line drawings. [George and Hawkins, 2005] As hoped, it learned temporal-spatial groups; for instance, bottom-level nodes learned groups of moving lines and corners. Then, when tested on distorted sketches, the system correctly recognized a fair fraction. Since this success, they and their partner companies have begun applying their algorithm to a variety of tasks, such as grayscale images and sounds, but groundbreaking results on complex data remain a work in progress. Getting useful accuracies on real-world tasks will be Numenta's primary challenge.

4 SpikeNet Technology

SpikeNet Technology SARL was spun off from the French laboratory The Brain and Cognition Center. It is based on the following insights of The Brain and Cognition Center's Simon Thorpe. First, activation arrives in high-level vision regions (such as the face-recognition regions) as quickly as 100 ms after the onset of the stimulus. These regions are separated from the stimulus by approximately 10 layers of neurons. Also, in the vision regions, the maximum neuron firing rate is about once every 10 milliseconds. Therefore, each layer only has time to fire once during computation! [Thorpe et al., 2001]

If the neurons in each layer can only fire once, then this completely rules out "rate coding," in which a neuron encodes a floating-point number via its spiking rate. Instead, Thorpe proposes that neurons use rank order coding, so that the order in which neurons fire in response to a stimulus determines the encoded information. This rank order coding has a few useful properties. First, if a layer contains n neurons, then we can represent $n!$ different values. This is even more than the 2^n we could get if we used a binary code, but the rank order code is still quite robust. Also, it is vastly more than the $(k+1)^{\frac{n}{k}}$ values we could get if divided the n neurons into groups of $k \gg 1$ neurons and used each group to represent a single number (i.e. population coding).

In particular, Thorpe suggests that rank order coding is done by attenuating the firing strength of later neurons. The first few neurons fire strongly, but after that the signals become weaker and weaker. Thus, a neuron is most activated when the strengths of its synapses to its inputs and the order in which its inputs fire are identical. This order specificity suggests a simple learning algorithm. When a neuron successfully fires, it most strengthens its weights to the inputs that fired first, and decreases its weights to the inputs that fired last. [Rullen et al., 1998]

4.1 Algorithm

Of course, the above insights into neural coding do not translate readily into an inference algorithm. The SpikeNet algorithm is more or less like a neural net, with layers of neurons with weighted links. Its primary advantage over neural nets comes in the simplicity of its coding. With a neural net, many floating-point numbers need to be stored and transmitted between neurons, but with rank order coding,

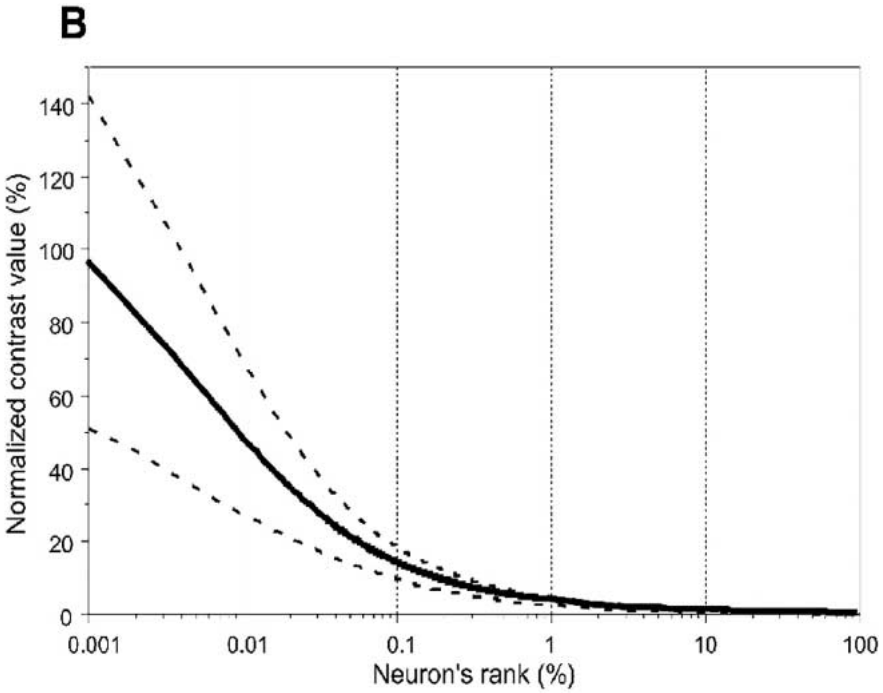

Figure 6: The rank order code falloff function, as shown in Thorpe et al. [2001]

only an ordering of neural firing must be transmitted. This allows their algorithm to work quickly even on ancient 300-MHz computers, and also lends itself to parallelism, since only small amounts of information need be transmitted between layers.

In the SpikeNet model, each neuron has a set of input neurons, a weight for each input neuron, and a threshold. Once the threshold is reached, the neuron fires; if the threshold is never reached during the simulation, the neuron does not fire. Neurons are organized into layers, which are processed in turn. When a layer is simulated, its set of inputs is read through in order in which the input neurons fired. Each input neuron has its firing strength multiplied by a function of its order, so that the first neuron fires at 100% strength, but the later neurons are reduced to a small fraction of their original strength. The desired function from neuron order to firing strength is shown approximately in Figure 6. When any neuron in the current layer reaches its threshold, it is recorded in an ordered list, and this list of neural spikes is then sent on to the next layer.

According to the description of the SpikeNet face-recognition system in Rullen et al. [1998], it works as follows (also shown in Figure 7). First, in Layer 1, they process the image with two sets of neurons; some that respond to on-center "Mexican hat" functions, and some that respond to the negative of this function. These neurons fire in the order of the strength of their input signals, so the most strongly stimulated neurons fire first, with the rest proceeding in descending order. These simple-receptive-field neurons then feed into a set of orientation-detection neurons with Gabor-filter-like receptive fields in Level 2. In Level 3, neurons representing high-level features (eyes and mouths) receive inputs from the oriented neurons. Finally, Level 3 neurons feed their spikes into Level 4, which detects whole faces. Each neuron's receptive fields are, of course, determined by their weights to the neurons in the previous layers. These weights are determined either by hand, as in the case of Level 2's oriented receptive fields, or are trained on a hand-labelled set of images. Thus, each level is trained in a supervised manner to

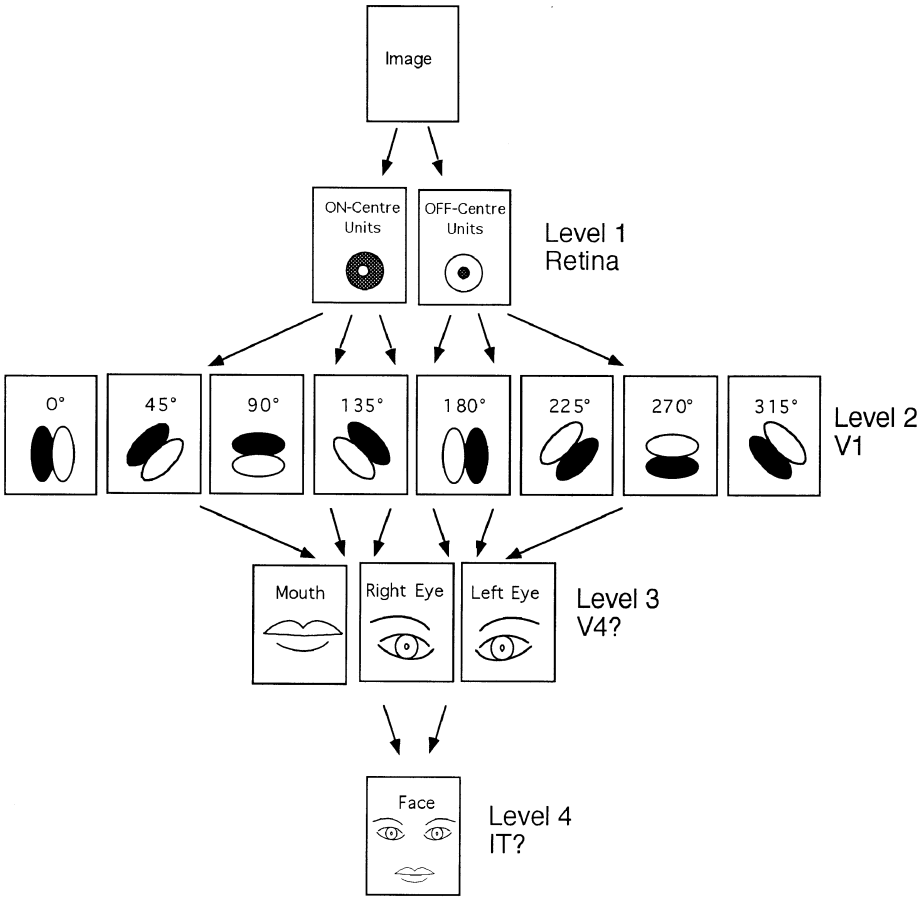

Figure 7: The SpikeNet face recognition model, as shown in Rullen et al. [1998]

recognize human-chosen features; this contrasts with how an ordinary neural net is trained all at once via backpropagation.

SpikeNet's SNVision tool provides the ability to create template patterns to match and perform recognition. Their GUI allows one to create a template pattern to find matches for. Presumably, this creates a layer of neurons approximately corresponding to Level 3. It receives inputs from oriented filters and has highest weights for those that would fire first when shown the template pattern. Then, one may define a threshold and set the system loose on an image or video, detecting every matching instance of this pattern. In addition to providing this simple pattern-detection tool, they also provide a programming toolkit wrapping this algorithm, allowing one to create multilayer networks like the face recognition network.

4.2 Analysis

SpikeNet's insights do not intrinsically define or suggest a particular algorithm; instead, they define a coding that neurons might plausibly use. However, there is one key feature of the resulting coding that might make their algorithm more powerful than backpropagation neural nets. That feature is that instead of allowing neurons to take any floating-point value (as in a traditional neural net), the weighting is set

by the spikes' order, so that only the first 1% of spikes have a value above 20% activation. This enforces a sort of sparsity that ordinary neural nets do not, and could provide additional power.

5 Imagination Engines

Imagination Engines Incorporated (IEI), founded in 1995 by Dr. Stephen Thaler, is by far the most curious of all the companies I examine here. It is based on traditional backpropagation neural nets, but it uses them in a creative way. Thaler's key insight is that neural networks, which are usually used to learn a mapping between a set of inputs and a set of outputs, can be used in a completely orthogonal manner. If a well-trained neural network is perturbed in some way, for instance by altering the network's weights or activations, then it can output values it has not been trained to produce. Thaler labels this generation of novel outputs as "creativity," and asserts that this algorithm is so groundbreaking that IEI is "intrinsically worth gross national products." [Imagination Engines Inc., 2006a]

5.1 Algorithm

IEI's main algorithm is as follows. First, they train a standard neural network to produce outputs in a given domain. For instance, it might produce chemical compounds, musical melodies, or even English words. (As input to the net, they give arbitrary values, such as a code representing which melody it should produce.) Then, they take the trained network and repeatedly modify its connection weights by a small random amount, while keeping its input constant to provide context. The result will be a string of loosely related outputs, roughly in the given domain, that may or may not have been seen before. By filtering out the previously-seen outputs, IEI then obtains a set of novel concepts. [Thaler, 1997] This neural network is called an Invention Engine.

Of course, not all of the newly-generated concepts will be useful or interesting. To automatically sort out the valuable concepts, IEI relies on a second neural network, which I will call an evaluation network. They train this second neural network to discriminate between high-quality concepts (e.g. the same ones the original network was trained on) and useless concepts. IEI then uses this second network to score the concepts. Then, it thresholds the scores of the novel concepts, selecting only the ones the evaluation network deemed worthwhile.

The Invention Engine can produce outputs of varying degrees of originality. If the connection weights are not disturbed much, then it will produce outputs much like those it was trained on. If the weights are modified heavily, then its outputs will be wildly different and often of little value. Somewhere in the middle is a critical region where outputs are different enough to be interesting but not wild enough to be useless.

5.2 Other Ideas

IEI has also received other patents, though at least one is of questionable value. In particular, IEI received a patent for the "self-training artificial neural network object," or STANNO. This patent describes the implementation of a backpropagation neural network in a "non-algorithmic manner." As far as I can ascertain from IEI's patent, this means implementing the neural network as a computer spreadsheet (the patent suggests the use of Microsoft Excel). IEI claims that this network is self-training because it "no human-invented mathematics are involved;" [Imagination Engines Inc., 2005, Thaler, 1998] however, as far as I can tell, all the mathematical calculations are still performed. Because they are performed in a spreadsheet, rather than in an ordered, flowchart-like fashion, IEI believes that something is somehow different. Personally, my reading of the patent is that IEI fails to demonstrate its claim that this type of learning is different from traditional backpropagation. Still, I must give IEI and the patent office the benefit of the doubt, and admit to the possibility that either I have somehow failed to understand the

technique described in their patent, or that I do not fully understand what IEI means by the phrase "self-training."

6 Connections—How Are These Concepts Related?

Having studied and analyzed the techniques used by these four companies, I am in a unique position to analyze their relationships. There are many similarities in their algorithms, and many ideas are not limited to the ways that these companies applied them. Overall, I believe that each of these companies can learn something valuable from the others.

6.1 Sightech and Numenta

Sightech and Numenta have similar ideas about detecting patterns of features, but have one important difference. Sightech uses a discriminative model to map from features to classifications, while Numenta uses a full-blown hierarchical generative model describing how the classifications might produce the features. This mirrors the standard debate between using discriminative models, such as Support Vector Machines, or generative models, such as Bayes Nets. In general, if one can accurately model the generative statistics, then a generative model will be at least as good. However, for many domains, a good generative model is hard to produce, so a discriminative model may be more effective.

6.2 SpikeNet Technology

SpikeNet's methods have two benefits. On the one hand, their rank order coding scheme is quite efficient at encoding responses of a layer of neurons. The second, and possibly more important benefit is that using a spike rank weighting curve like that shown in Figure 6 biases their learning algorithms toward sparse representations. This requirement of sparsity may tend to emphasize the sort of causal learning that humans find so useful. A whole set of sensory inputs might correspond to the presence of a single sort of object, and a representation object would be a useful sparse representation of these inputs. Thus, other algorithm designers might also benefit from the incorporation of sparsity constraints.

6.3 Imagination Engines

Imagination Engines describes their creative algorithms in terms of neural networks, but the fact is that their methods should apply to any sort of generative model. Any generative model that produces a distribution of outputs like that observed in the real world can be perturbed, and the resulting distribution will be slightly different than that observed in the real world. If this distribution is over a class of concepts, then sampling from it will result in concepts, some of which may be novel. Of course, the better a model represents intermediate constraints and invariants, the more likely it will be to generate ideas in a useful part of the concept space. Neural networks tend to learn useful intermediate levels, so they should be relatively good candidates. However, Numenta and SpikeNet both possess generative models that could also operate in this mode. Of course, any sort of classification system, not just a neural net, could be used to evaluate the quality of the resulting concepts. Thus, Imagination Engines's algorithms may be needlessly limited to neural nets.

7 A Word of Caution

In writing this paper, I have not made any significant attempt to determine or label which concepts are patented. Many of the algorithmic details I describe are published in a public forum, but that does

not necessarily mean that they are not patented. Regardless of whether software patents are a boon to society, before implementing any of these algorithms you are legally responsible for checking the relevant patents and determining whether your implementation infringes upon them.

References

A. Delorme, J. Gautrais, R. Van Rullen, and S. Thorpe. Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, (26–27):989–996, 1999.

Dileep George and Jeff Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 05)*, 2005.

Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. Times Books, 2004.

B. Holmes. The creativity machine: It writes music, invents soft drinks and dreams up hard materials. the man who built it thinks it points the way to immortality. *New Scientist*, (2013), 1996.

Imagination Engines Inc. IEI's patented self-training artificial neural network objects (STANNO). `http://www.imagination-engines.com/stanno.htm`, 2005.

Imagination Engines Inc. website. `http://www.imagination-engines.com/about.htm`, 2006a.

Imagination Engines Inc. website. `http://www.imagination-engines.com/`, 2006b.

Numenta. website. `http://www.numenta.com/`, 2006.

R. Van Rullen, J. Gautrais, A. Delorme, and S. Thorpe. Face processing using one spike per neurone. *BioSystems*, (48):229–239, 1998.

Sightech. website. `http://www.sightech.com/`, 2006.

SpikeNet Technology. website. `http://www.spikenet-technology.com/`, 2006.

S. L. Thaler. Is neuronal chaos the source of stream of consciousness? In *Proceedings of the World Congress on Neural Networks (WCNN96)*, Mawah, NJ, 1996a. Lawrence Erlbaum.

S. L. Thaler. A proposed symbolism for network-implemented discovery processes. In *Proceedings of the World Congress on Neural Networks (WCNN96)*, Mawah, NJ, 1996b. Lawrence Erlbaum.

Stephen L. Thaler. Device for the autonomous generation of useful information. US Patent 05659666, 1997.

Stephen L. Thaler. Non-algorithmically implemented artificial neural networks and components thereof. US Patent 05845271, 1998.

S. Thorpe, A. Delorme, and R. Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, (special issue 14):715–725, 2001.