# Pattern recognition with parts-based representations

Naveen Agnihotri and Walter Borden
Nexus Holdings Group

**Abstract**

There is an explosion of data on the Internet, and a paucity of means to search and analyze patterns in text and non-text data. We have developed a novel class of neural networks that use parts-based representations to analyze and search for patterns in databases of non-tagged data.

## 1. The Problem

The rapid growth of data storage capacity in modern computer systems creates new challenges for analyzing and extracting data. The task of finding patterns embedded in this data is exacerbated because the data exists either in a serially stored format (on individual computer directories) on in a completely unstructured format (on the Internet). These storage formats were created for their own separate purposes: to make it easy for the operating system to store and retrieve data (in the case of individual computer), and to facilitate the connectivity of large numbers of computers (for the Internet). These methods make it easy to answer questions about data storage history and geography ("When was this file modified?" "On which head/cylinder is this file located on disk?"); it is also easy to answer questions about data content ("Does this image file have a red pixel in it somewhere?"); but it is much more difficult to answer descriptive questions ("Is this an image of a human face?")

The human brain is set up in almost exactly the opposite way. Most people find it harder to answer questions about history and geography ("When did the US buy Alaska?" "What is the capital of Vermont?"); but find it easier to answer questions about patterns ("Is this an image of a human face?"). One of the reasons why this might be the case is that the brain does not store data serially, but in parallel, after breaking it down into component parts (1,2). This makes it easier to find patterns either as a sum of their parts or holistically, and harder to answer questions that involve combining the parts in a non-traditional way: for example, most people cannot tell if an upside-down face is normal or distorted.

As computer usage increases, this conflict in data representation and access between people and computers is creating an ever-widening gap between the way that people want to access data on computers and the access methods available. Let us say a person is looking for a Frequently Asked Questions (FAQ) file that they know is stored somewhere on their computer. They will find that it easy to answer *storage queries* ("find me all files in the directory 'FAQ' "), a little cumbersome to answer *search queries* ("find me all files that have the word 'FAQ' in them"), and very difficult to answer *pattern queries* ("find me all files that look like a FAQ"). The difficulty in answering the last question is simply a result of the flat-file based data organization on the computer.

The unstructured data format of the Internet does not affect this difficulty in any qualitative way: it simply increases the amount of data exponentially so any solution takes that much longer to run (or takes that much more computing power to run in the same amount of time).

## 2. The Current Solutions

Of the three kinds of queries described above, storage queries are trivial to answer, so we will not discuss them here. For search queries, particularly with text data, there has been much progress in the last thirty years: the advent of modern databases has led to many advances in indexing and organization of large quantities of text. The emergence and rapid expansion of the Internet has advanced the state-of-the-art in text searching even further. Google now indexes more then three billion text documents on the Internet, and can return answers to search queries in 2-3 seconds. There are similar applications that people can run on their individual computers, both by Google and others.

### 2.1 Solutions for Text Data: Bayesian Analysis and More

There are few good solutions for pattern analysis of text documents. One glaring application area for such analysis is detection of spam in email. Most people can tell if an email is spam in one quick look, yet it is a hard task for software. The most widely used spam filters use statistical techniques like Bayesian analysis. The user "trains" the spam filter by giving it examples of spam and non-spam mail, and the program tabulates the frequency of different words in the spam and non-spam contexts. When an unknown email is presented to the spam filter, it calculates a statistical likelihood that the new email is a spam based on the words in the email and the program's knowledge of word frequencies in the spam and non-spam context. This approach works pretty well for spam, as evidenced by its wide usage in the most popular spam filters. Other approaches, using natural language processing, are not yet commercially viable, and are used largely in academic settings.

For non-text data, meaningful search queries have intrinsic pattern queries embedded in them. Non-pattern search queries on non-text data ("find all images which have red pixels in them") generally do have much practical value. The useful search queries ("find all images of cousin Jim") generally involve some aspect of pattern analysis.

### 2.2 Solutions for Non-Text Data: Tags

One commonly used solution for non-text data (such as images) is to tag it with text indicating its content. Most common image formats allow adding such tags (called meta-tags) to the image. Now the search engine can basically treat the image as if it were just the text from it's meta-tag, and perform searches on that text. This approach is used by popular Internet search engines like Google and Yahoo to conduct image searches.

But this is a non-solution. Trusting that users will correctly add meta-tags to

their images is like trusting patients to walk into hospitals holding placards with the name of their disease. Most users don't know how to add meta-tags, and search engines can be easily deceived by simply meta-tagging an image different from it's content. The correct way of doing pattern matching on non-text data is to examine the content of the file, not a tag that someone else has attached.

### 2.3 Solutions for Non-Text Data: Neural Networks

Most of the current solutions for pattern analysis in non-text data involve some kind of artificial intelligence, and the bulk of these focus on using artificial neural networks to solve the problem. An artificial neural network (hereafter referred to just as a neural network) is composed of an interconnected group of artificial neurons, each one of them modeled after the actual biological neurons in the brain. Neural networks are designed to capture some properties of biological networks by virtue of their similarity in structure and function. Whereas the individual processing elements (neurons) are simple, the network is capable of complex global behavior, determined by the properties of the connections between the neurons (3).

Neural networks find wide applications because of the property of *learning*: Given the right setup parameters, and a large training data set, a neural network can be taught to differentiate one kind of pattern from another. A neural network is an adaptive model that can learn from the data and generalize things learned. It extracts the numerical characteristics from the numerical data instead of memorizing all of it. Because of their ability to learn and differentiate patterns, neural networks find applications is a wide variety of fields, including vehicle control, game-playing, object recognition, medical diagnosis, financial applications, data mining, spam filtering, among many others.

A well-known caveat of using neural networks is the training required upfront. Whereas most software solutions work right out of the box, a neural network needs to be trained, and without the training is not a very helpful tool. But with the appropriate training, it is able to outperform virtually all non-pattern-oriented algorithmic approaches (3).

One powerful feature of traditional neural networks (as described above) is that they require little or no *a priori* knowledge of the problem. This allows the network to tackle the hardest problems, even mathematically intractable and computationally hard problems. But this is also a limitation because they don't allow any way for the scientist or the programmer to specify any previously known facts about the relationships of the inputs or the input to the output. In that sense, they are **data-rich** and **theory-poor**. A neural network can build relationships from inputs to outputs but the precise behavior of the system is not understood: it is a kind of black box.

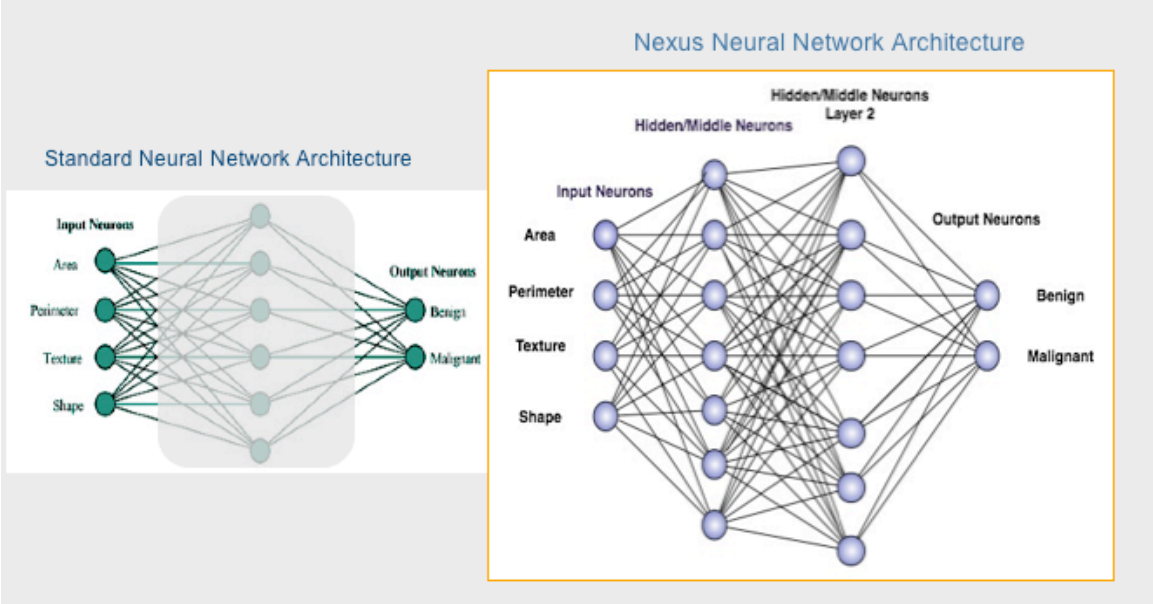
## 3. Our Solution

We have engineered neural networks that follow some of the same rules as the real neurons in the cerebral cortex. By virtue of our implementation of these rules, the neural networks embody a parts-based representation of the inputs: every neuron in the network represents a part of the solution. This means that we can now manipulate the individual elements of these networks to achieve much higher throughputs.

For an example, take the problem of object recognition. We would like the software to detect if a given image has a car in it. A traditional neural network can be trained to recognize cars by supplying it hundreds (often thousands) of examples of car and non-car images. Then, given an image that it has never seen before, it will classify it as being car or non-car. But how it determines that is never clear. And what if the neural network is not working properly? The only solution is to adjust the rules of the whole network, and hope that this new network with different properties will work better.

Our network is also trained in the same way, except that in the process of it's training, it develops an internal representation of cars: one neuron will correspond to a headlight, one to the rear door, one to the front fender, and so on. This internal representation is not dictated by the user (or by the programmer), but falls out naturally from the properties of the network itself. Now, if the network is not working as well as it should, the programmer (or the user) can change the performance of the network in a predictable, understandable way ("pay more attention to windows and less to headlights") by increasing the weights of the neurons that represent those elements of the input.

Our networks are no longer black boxes: we understand how the internal elements are transforming the inputs to the outputs, and we can manipulate them appropriately. They are now **data-rich** and **theory-rich**. They have the same applications as before, except we can "pre-load" them with known facts about input-output relationships. Interestingly, other than the output "result" of the network itself, examining the internal elements is also a new kind of "result", because it reveals the nature of the relationships of the inputs and the inputs to the output. Furthermore, this allows us to feed any new relationships that are revealed as the networks start to yield results right back into the network further improving its performance.




One additional feature of the parts-based representation in our neural networks is that they allow us to create a database of the images in a parts-centric way (in other words, store the data using an alphabet where every letter of the alphabet corresponds to a part of the input). This allows us to do searches in this database in the same way as traditional text searches, only the results are delivered on the basis of the parts-based patterns, thus becoming a wonderfully effective pattern matching tool. We have used this approach to generate databases of images (including commonly used face databases such as the ORL face database and CBCL database), and the software performs extremely well in recognizing faces and images. We have also had great success with pattern analysis of stock prices over time. An adaptation of this technique to pattern analysis in VoIP data is in the works.

By using rules from the activity of biological neurons, we have developed neural networks that have the ability to perform sophisticated pattern matching in real time. Given any appropriate set of data, the software can make a database that can then be used to search and retrieve pattern-based matches. We predict that such networks will find utility over a wide spectrum of data.

## References


1. Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque. E. Wachsmuth, M. W. Oram, & D. I. Perrett, *Cereb. Cortex,* 4, 509–522, 1994.
2. Hierarchical structure in perceptual representation. S. E. Palmer, *Cogn. Psychol.* 9, 441–474. 1977.
3. Neural Networks for Pattern Recognition. C. M. Bishop, Oxford University Press, 1996.