

Experimenting with the *Pictures* Demonstration Program

Overview

This easy-to-use demonstration software, called *Pictures*, will give a sense of what Hierarchical Temporal Memory (HTM) can do. Both engineers and non-engineers are encouraged to explore HTM with it. It is easy to set up and doesn't require the Numenta Platform for Intelligent Computing (NuPIC) to be downloaded.

Note that although *Pictures* demonstrates an image recognition task, HTM's are not restricted to this category. HTMs are well suited to non-visual domains (see <http://www.numenta.com/for-developers/education/evaluating-problem-data.php> for a discussion on suitable problems for HTMs).

What is this demonstration?

The *Pictures* demo is an image recognition application with a Windows front-end that can be installed on any PC running Windows XP or Vista. (Linux and MacOS versions of the demo are included as an example in the NuPIC Platform download.) The image recognition ("inference") is done remotely on the Numenta computing cluster in California. The HTM system that is running on the Numenta cluster has been pre-trained to recognize line drawings of 48 different kinds of objects. To explore *Pictures*' capabilities, you draw an approximation of one of the trained images in the window and see whether the system can correctly identify it.

Note that you must be connected to the Internet in order to do inference since the recognition capability is accessed remotely and is not embedded in the client software. Depending on the server load, you may experience a few seconds of delay.

How was it trained?

Pictures was trained using approximately six to twenty different line drawings of each object. The Numenta algorithms require a temporal component to do the training, so each image was shown to the NuPIC HTM moving across the screen. The specific training images are not kept in the system as templates, and no simple pattern-matching is done. Instead, as a result of the training, *Pictures* has built a hierarchical model of its world by keeping track of the inferred characteristics of common image properties at each level of a 4-tier hierarchy.

Note that the system was trained using multiple sizes of the line drawings, and versions facing left and right to the extent it makes sense (i.e. for images like "dog", but not for letters). The system was NOT trained on upside down images, or rotations and skews beyond a simple right-to-left flip. In addition, the system was not trained on any curved lines, only straight line objects.

How do I use the software?

Follow the installation instructions on the demo web page. Once the software has launched, you will notice a drawing window in the center, and a scrolling list of trained

object samples down the left. On the right, *Pictures* lists several possibilities of what it thinks the object in the drawing window is.

How do I best experiment with *Pictures*?

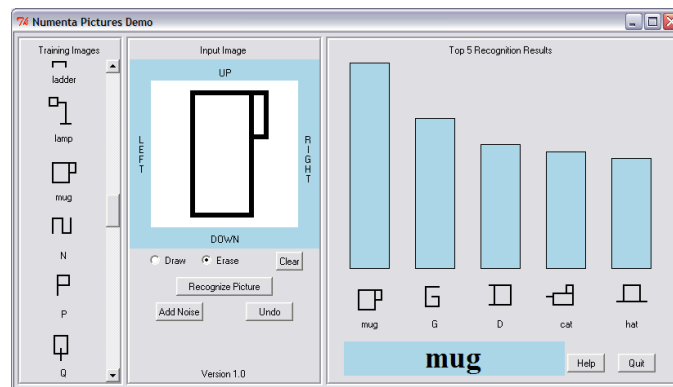
The point of the *Pictures* demo is to give you a feel for the strengths and weaknesses of the NuPIC HTM platform today. You will get a sense of where it succeeds -- and where it fails -- on this very hard problem. Bear in mind that there are a huge number of potential novel versions of each image, and the system only has seen a few.

We recommend trying the following experiments, described further below: 1) image manipulation, 2) morphing, 3) noise, and 4) untrained objects.

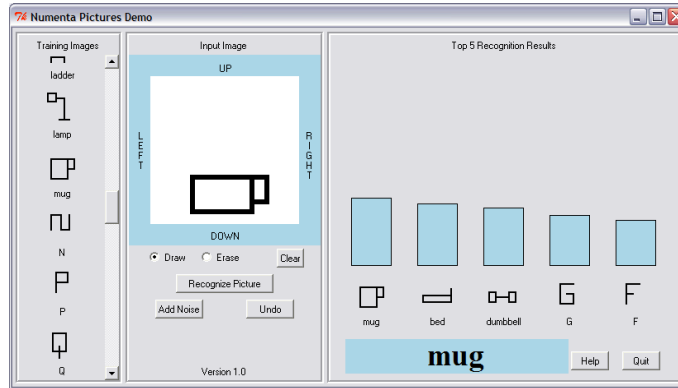
1. Image manipulation

Start with one object, and see how much variation the system can tolerate and still recognize the object. For example, start with the “mug” by clicking on the sample mug image on the left, which puts the prototype image in the center box, and then click on “Recognize Picture”. You will see that *Pictures* correctly identifies the image as a mug.

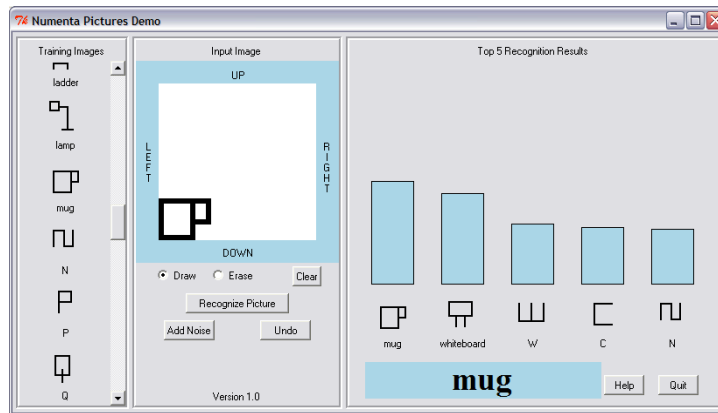
Change the proportions of the object by clicking on the draw/erase button to modify the image. In the below example, we have made the mug taller, and you will see that *Pictures* still believes it is a mug.



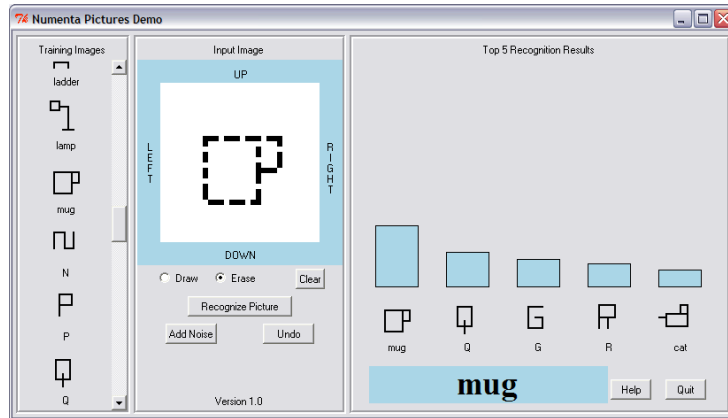
Now try a shorter, wider mug. Note below that *Pictures* still believes it is a mug, but it is less sure, as represented by a lower bar for mug, and a much closer bar for the alternate possibilities.



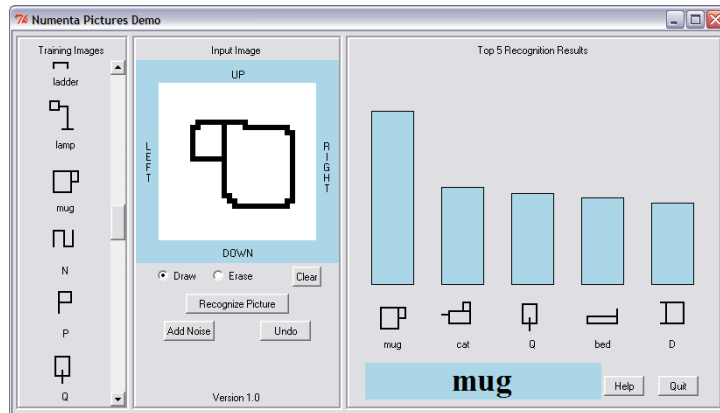
Next try placing the object in various locations on the screen. You can click on the edges of the box, labeled “left”, “right”, “up” and “down”, to move the image to a new location. Below we’ve made it smaller, and moved it into a corner, and it still works. Note that if you make the image too small, the system will ask you to draw it larger.



Now try removing parts of the mug, or punching holes in it, to see how the recognition holds up. *Pictures* will switch to guessing another object at some point when the original image becomes sufficiently degraded. However, you often will find that the recognition is extremely robust, as in the example below that contains only pieces of solid lines. Note that there are no longer any “features” of the mug, i.e. solid lines or corners.

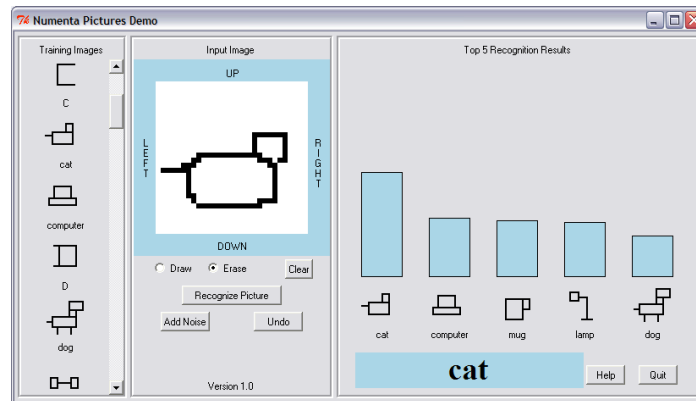


You can try an image flipped right-to-left, if it is logical to do so. This makes sense for a physical object, like a mug, but not for a letter, like “G”. You often can distort the image pretty significantly before losing recognition, as you can see in the following example.

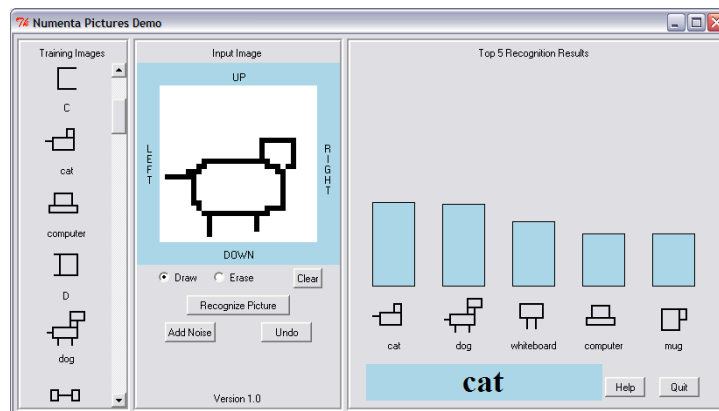


2. Image morphing

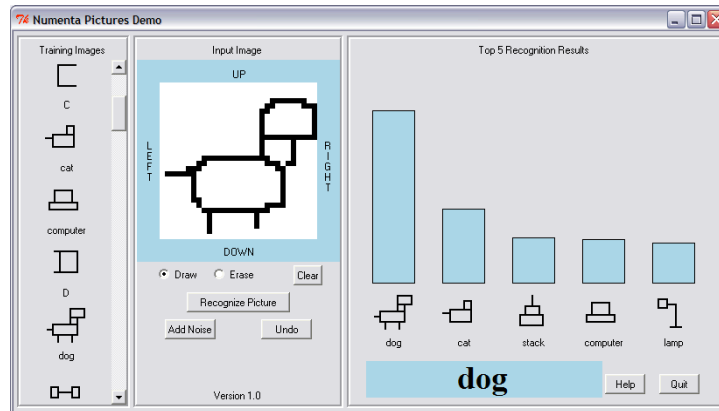
Try morphing one image into another, and see when *Pictures* switches its guess. The example below morphs a “cat” into a “dog”. In the world of *Pictures*, cats don’t have legs or necks, but dogs do. We start with a slightly distorted, but pretty clear cat. Note that “dog” is listed as a fifth choice, lower than several other choices.



Now we add short legs to the cat below. *Pictures* still believes it is a cat, but “dog” is now in a close second place. Note that if you draw longer legs, it may switch to “dog” more quickly.

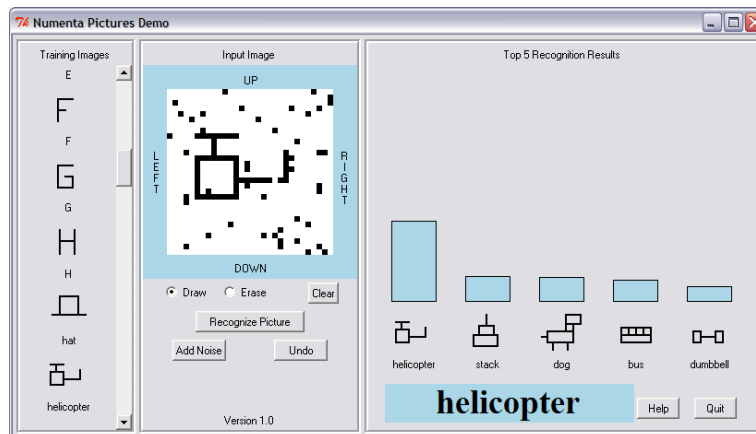


Finally, below we add a new head on top of the prior image, effectively making a “neck” out of the previous head. Now *Pictures* is quite positive that the image is a dog. Note that the entire cat image is included in dog. Note also that cat is the #2 choice.

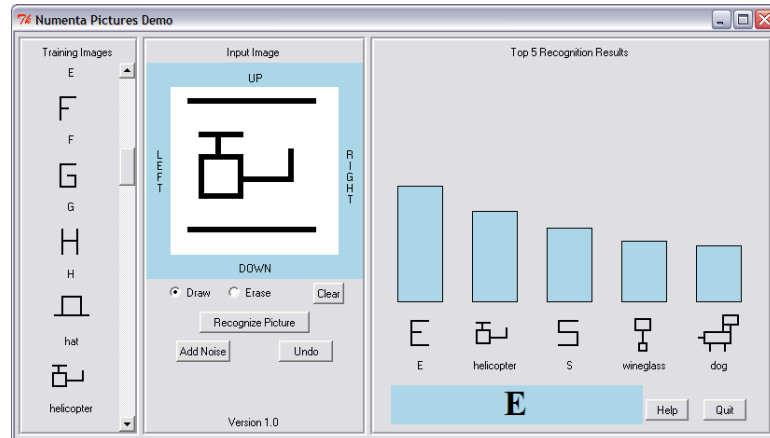


3. Adding noise

First, try adding unstructured noise. You can easily do this using the “Add Noise” button under the information box. Keep pressing the button to add more noise, and select “recognize picture” to see the results. You likely will notice that *Pictures* holds up well to unstructured noise. Below we’ve drawn a helicopter, then added unstructured noise.



Second, try adding structured noise, i.e. random lines or shapes rather than dots. You quickly will see that structured noise is more confusing to *Pictures* than was unstructured noise. This confusion is because the structured noise is more easily interpreted as legitimate potential lines in the drawing. Below we see that helicopter is no longer the top choice, although it is still second on the list.

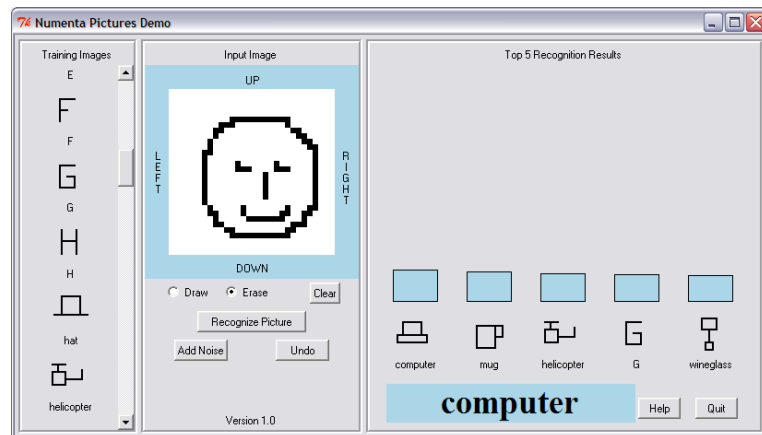


Pictures is trained to assume that there is only one image in the drawing box. The above noise is really three separate images, a helicopter and two lines, and as such, *Pictures* is unable to decode it.

This example also points to the problem of attention. The demo does not try to attend to part of the picture, only the whole picture. As a human, we can attend to part of the picture, ignoring the lines as irrelevant. Eventually, an HTM system will be able to attend to a part of the picture.

4. Trying new objects

You can try *Pictures* on an object it has never seen. Much as a child that sees a fox for the first time might think it is a dog, *Pictures* will be unable to guess a new object it hasn't been trained on, but will tell you which objects it believes are closest. Note that the bars showing the level of certainty are often lower, indicating that *Pictures* is admitting that it doesn't really have high confidence of the object's identification.



A note about the bar graphs

Although the bars on the right indicate to some extent the degree of confidence *Pictures* has about the object, they should not be too strictly interpreted. The bars arise from the complex Numenta algorithms embedded in NuPIC and an equally complex display calculation. They do not represent a normalized distribution of probabilities (i.e. they do not all add up to 100% probability), nor does the height of the bar signify anything quantitative, which is why we do not display a scale. The bars simply show relative confidence in the guesses. When the bars are very close in height, *Pictures* isn't quite sure whether the object is a "cat" or a "dog". When there is more difference between the bars, *Pictures* is more confident.

Summary

The *Pictures* demo allows you to experiment with HTM without requiring elaborate system set-up or extensive technical knowledge.

Pictures is not perfect. You will find areas of weakness; in particular you will find objects that you think it should recognize, but it doesn't. On the other hand, you will find positive surprises as well, where *Pictures* identifies an object in spite of severe noise or distortion. We have many enhancements that we will be implementing over the coming year to improve recognition. In addition, we are creating a version of *Pictures* that uses grey-scale images rather than line drawings.

Overall, we believe that the *Pictures* program demonstrates impressive capabilities on a problem that has vexed computer scientists for many years: using a general-purpose problem-solving system to identify objects irrespective of scale, proportion, placement and noise. The important achievement we are trying to demonstrate here is the ability to automatically extract the “invariances” of an object from the images. Can *Pictures* understand “dogness” by being exposed to multiple images of dogs, much as a child learns to identify dogs?

We believe that *Pictures* shows a degree of functionality that cannot be demonstrated with traditional computing techniques. It does not use algorithms designed specifically for this particular application, but instead demonstrates successful problem-solving using a generic platform (NuPIC) that can be applied to many applications.