

Experiments with Image Content Analysis¹

by

Svein Nordbotten, Professor Emeritus,

University of Bergen

Contents

1. Introduction.....	3
2. Research problem.....	3
3. System outline.....	4
3.1 Feature extraction.....	5
3.2 Training.....	8
3.3 Testing.....	12
3.4 Indexing.....	12
3.5 Retrieval.....	13
4. List of scripts/programs used.....	13
5. Experiments.....	14
5.1 Experiments with TC.....	15
5.2 Experiment with Shapes.....	21

¹ Presented to a CAIM project seminar in Bergen, June 4, 2010

5.3 Experiments with Icons	26
6. Indexing and retrieval	40
6.1 Indexing.....	40
6.2 Text query retrieval	41
6.3 Image query retrieval.....	44
7. Conclusions and future tasks.....	46
8. Tables	48
Table 1: Results from Experiment: tc40-8_b	48
Table 2: Results from Experiment: tc40-8_s.....	50
Table 3: Results 1 from Experiment: shapes80-8_b.....	52
Table 4: Results 2 from Experiment: shapes80-8_b.....	53
Table 5: Results from Experiment: icons80-8_s_vector.....	54
Table 6: Results from Experiment a: icons80-80_s_matrix1	56
Table 7: Results from Experiment b: icons80-80_s_matrix2.....	57
8. References.....	59

1. Introduction

This note is a preliminary report on part of a project on content analysis of images. The present report is limited to experiments with a prototype system applied on images reflecting objects prepared by computer tools.

The work is related to the objectives of the CAIM project in the sense that given a collection of images, it can be useful to be able to perform an automatic mass indexing of images indicating the categories/types (airplanes, animals, boats, bridges, buildings, cars, churches, people, ships, statues, etc.) of objects reflected in each image. Content-based indices can, in addition to context-based indices, improve retrieval of requested images.

2. Research problem

The question investigated: Is it possible to design an automatic system which recognizes different categories of objects reflected in an image independent of the location, scale and rotation of the objects within the image?

A human being observes a picture or scenery by means of her eyes in which the picture is reflected at the retina. The retina is composed of more than 130 million retinal receptors. These are sending the signal through approximately 12 million axons to the brain with a speed higher than 1 Mb/sec. These figures indicate that an extensive preprocessing must take place in the retina before a compact result is forwarded for further processing in the brain based on stored knowledge and experience.

The aim of the preprocessing and compression of the stimuli seems to be selecting important features and discarding unimportant signals, while the task of the brain is to learn, analyze, store, retrieve and present results generated from the received feature data. The capacity of the brain is enormous with a number of brain cells

assumed to be of size order 10^{14} , and knowledge about its working is still in the beginning.

The prototype system reported on in this note is designed and implemented with the human eyes and brain in mind. The prototype used is, however, very simple and slow compared with the human system it simulates.

3. System outline

The prototype system consists of 2 preprocessing versions, denoted *Vector* and *Matrix*. Each version consists of partly separate and partly shared components for the following tasks:

- Pre-processing (Feature extraction)
- Training
- Testing
- Indexing
- Retrieval

and a

- Database as a back end for the Indexing and Retrieval components.

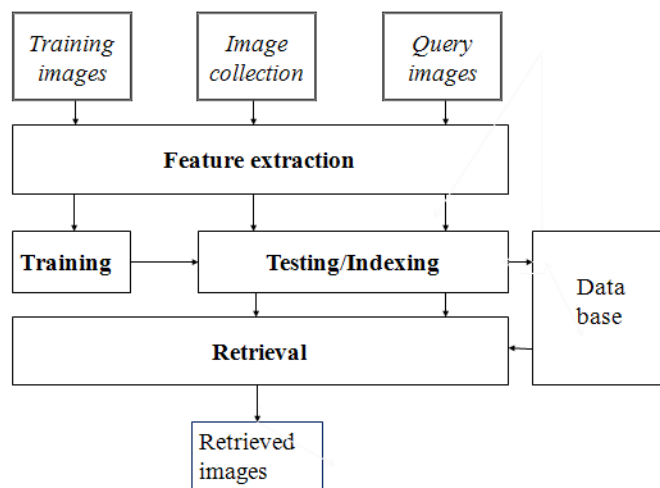


Figure 1: System outline

Vector has a more complex Preprocessing than *Matrix*. The advantage of *Vector* compared with *Matrix* is that the former pretends to generate representations which are invariant for object rotations in the 2 dimensional planes, i.e. the representation of a clown will be the same for an image in which he is standing on his feet, and in another in which he is standing on his head. In *Matrix* the two images must both be available and the system has to be trained to assign both to the category *persons*.

3.1 Feature extraction

The preprocessing component reads an image reflection of a picture in format *.jpg* and carries out a sequence of processes related to feature extraction.

The component comprises a number of steps and delivers an encoded set of vectors as its final output (*Figure 2*). The set of steps corresponds to the processing of the retina of a reflected scene. The encoded set of vectors is ideally a representation which is invariant for scaling, location and 2-dimensional rotation of objects reflected in the images.

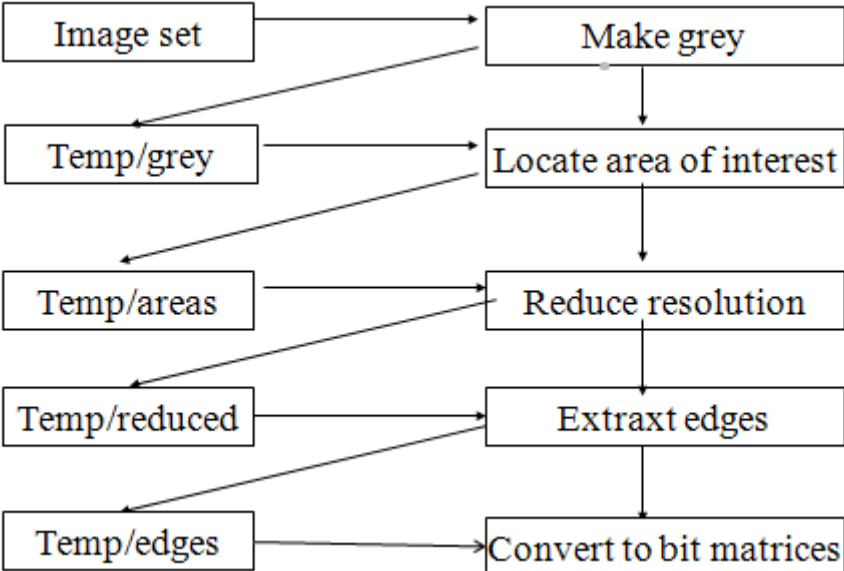


Figure 2: Steps through the pre-processing

In contrast to many current image systems, the present prototype considers the colors of less importance for object identification and starts by converting each pixel of the images to grey scale. The gray scale has 256 grades from value 0 (black) to value 255 (white). However, the component can be modified to work with colors in applications coloring seems important.

The next step aims at identifying areas of interests. The approach used is scanning for major pixel value changes row by row and column by column and marking by horizontal and vertical lines, respectively. The definition of major change can be set. By this process, the image is subdivided into rectangles of varying size. The rectangles are named *fragments*. Each fragment can contain the reflection of a single object. Fragmentation removes to a large extent the *location* problem, i.e. where in the picture the object reflection appears.

After empty or nearly empty fragments are removed, each fragment assumed to contain an object reflection is resized to a predefined standard size. In the experiments carried out, 2 sizes, 40*40 or 80*80 pixels, are used. By transforming to a standard size, the much of the *scaling* problem is resolved. This version of the fragments is named *reduced images*.

The object representations in the fragments are further simplified by retaining as black pixel only those assumed to be contours of objects. In this prototype, the contours are extracted by scanning the image by rows and by columns marking pixels as black only if their values differ significantly from the previous horizontal or vertical neighbors' pixel values of the reduced image. The significant difference can be set. Single black pixels surrounded by white neighbors are considered noise and erased. Finally, fragments with a number of black pixels less than a specified *cut-off limit* are considered non-important, are removed. The remaining fragment images are referred to as *contour images*.

To facilitate numerical processing, the contour images are converted to *binary matrices* in which the white and black pixel values of the contour images are converted to '0' and '1' values in elements of corresponding binary matrices. This

ends the preprocessing for *Matrix* providing a *binary matrix* for each image processed.

For *Vector*, the number of elements, 1600 elements, even of a 40*40 element matrix is considerable for the following steps. To be able to process a matrix with satisfactory accuracy without exceeding the moderate resources available, the matrix can be transformed to a so-called *coarse representation* with a less number of elements. [Spirkovska and Reid 1992].

The transformation can be illustrated by an 8*8 element *grid* covering the whole 40*40 element matrix. The first element of the 8*8 grid covers the column elements 1 to 5 of row 1 to 5 of the 40*40 matrix, the second grid element covers column elements 6 to 10 of row 1 to 5, and so on. If one or more elements of the 40*40 matrix covered by a grid element have value 1, the corresponding element of the first element of the first coarse matrix is assigned value 1. The second coarse matrix is developed in the same way after the grid has been moved one place down along the diagonal of the 40*40 matrix. The third to the fifth coarse matrices are developed similarly. By a wrap around procedure, each original element of the 40*40 matrix will be represented 5 times in the coarse representation. Instead of the 40*40=1600 elements of the original matrix, we get $5*(8*8) = 320$ coarse elements. For an 80*80 matrix which must be represented by 10 8*8 coarse matrices, the gain will be even higher, $80*80 = 6400$ elements compared with $10*(8*8) = 640$ coarse elements. It can be demonstrated that the coarse matrix representation provides an acceptable representation of the original bit-matrix if it is not too densely populated with 1's.

This last preprocessing step used in *Vector* version of the preprocessing is the most complicated and the reasons for the above explained coarse representation. The main purpose is to generate a final representation invariant for object rotation. The assumption is that a shape contour matrix can be represented by the *triangles* formed among all triple value 1 points. This collection of triangles will remain the same if the shape is rotated. The encoding process requires an ordered predetermined list of all possible triple points mapped to triangles for a given angle precision. Enumeration of all appearing triple points is the critical process. The result of this step will be a binary vector corresponding to the list of the possible triangles and marked with value 1 in

elements for triangles occurring in the matrix and 0 in the other elements representing triangles not present. The vector is here referred to as the *encoded* representation. The size of the vector is determined by the angle precision selected. The angle precision can be set for the preprocessing.

These steps are repeated for each fragment. The pre-processing results are also appended with some meta-data such as image and fragment numbering with fragment-to-image associations, as well as a catalog of categories to which the images should be assigned.

In addition to the matrix or vector output representation, a link list of all processed images, a list connecting fragments to images and a catalog of all categories (image types) appearing in the training set are prepared during the preprocessing.

3.2 Training

While the tasks of the *eyes/retina* are focusing on the main features of the images and transmitting the extracts to the brain, the first task of the *brain* is to learn to recognize the received feature representations. In our application, the task is to learn to associate certain vector/matrix pattern to categories. The approach is to present the learner with pairs of patterns/categories repeatedly until the pairs are remembered.

The training component of the prototype is implemented as a feed forward neural network which can be perceived as a simplified explanation of how a small part of the brain may work. Some neural networks have the ability to be trained on a set of training image representations with associated categories, by so-called supervised training, and, when properly trained, to compute similarities between the learned image features and features of new and, for the network, unknown images presented to the network [Rumelhart a.o.1988].

The most simple neural network type is a *Single-Layer Network, SLN*. The inputs are the encoded binary vectors from the Preprocessing component, and the outputs from the network are indicators of the presence of objects belonging to each object category. It is used in all experiment based on *Vector*. In *Version Matrix*, a multi-

layer networks are also used with binary matrices as inputs. Both networks types are trained by so-called supervised training requiring the real object category for each image input.

Figure 3 gives a graphical illustration of a simplified single layer network with 3 input elements and 2 image categories.

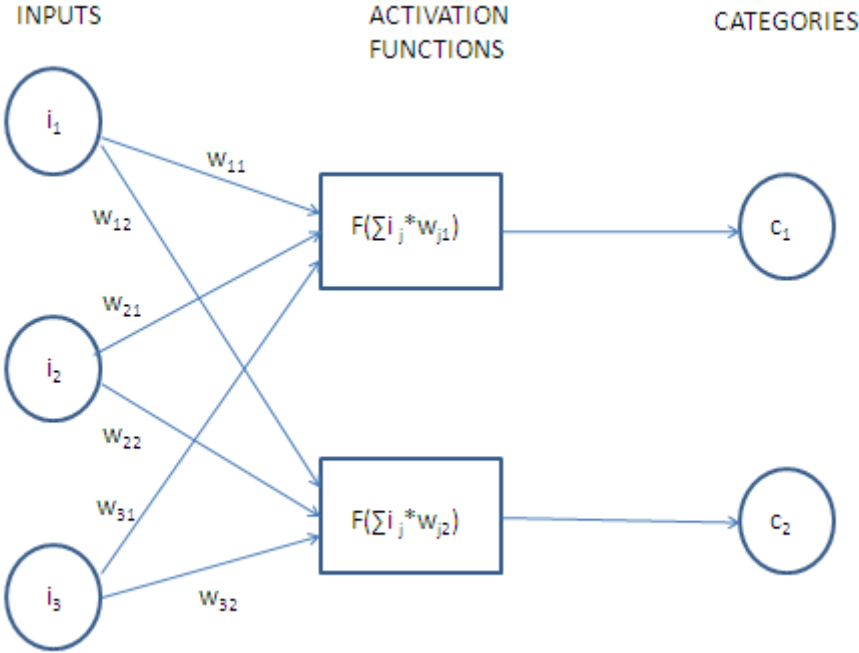


Figure 3: Single layer feed-forward neural network model

The central parts of a network are the set of *weights*, w_{ij} , and the set of *activation functions* F corresponding to each output category. The weight values represent the learned knowledge of the system, while the activation functions correspond to the interpretation of inputs based on the knowledge.

Each input element value is sent to each activation function after being multiplied by its respective weight w_{ij} . The sum of all weighted inputs received by an activation function is its input for computing the indicator of its attached category output o_j .

Two different activation functions are used in our experiments.

1. The *binary threshold function*:

$$o_j = 0 \text{ if sum of weighted inputs } < 0, \text{ and } 1 \text{ if sum is } \geq 0.$$

An output value 1 indicate that an object of category \mathbf{j} is present in the currently processed image, while value 0 gives no such indication.

2. The *sigmoid function* transforms the weighted sum of inputs according to:

$$\mathbf{o}_j = 1/(1 + \exp(-\sum \mathbf{w}_{ij} * \mathbf{i}_i)) \quad \text{where } 0 \leq \mathbf{o}_j \leq 1.$$

Each category output value can subject to certain assumptions be interpreted as a prediction of the probability that an object of the category type is present in the image processed [Bishop 1995]. Note that the sum of all outputs is not 1 , because the system may ‘see’ several alternative objects in the image.

To train the network to recognize the different categories of objects means to compute values for all weights, \mathbf{w}_{ij} , which will generate acceptable category indicators in responses to a set of input vectors from images independent of those applied in training the network.

A set of training images each representing an object with its target category and a training process is required for determining weight values. When the image representation is invariant for rotations in a 2 dimensional space one training image per object may be all that is needed. More training images are, however, needed to take care of rotations in 3 dimensions, for example of a car driving to the left and to the right direction, a front and a back-side view of a person, etc.

The training algorithm used in the applied learning process is iterative and can be described by the following steps:

1. Compute category outputs \mathbf{o}_j for the training set. Initially, a random set of small weight values is used.

$$\mathbf{o}_j = \mathbf{F}(\sum \mathbf{w}_{ij} * \mathbf{i}_i)$$

2. Compare all \mathbf{o}_i with the corresponding target categories, \mathbf{t}_i , for all fragments in training images, and stop if a satisfactory accuracy *tolerance* is obtained.. The target \mathbf{t} 's must be pre-assigned values 1 for categories of objects present in the image, and 0 for those not.
3. Adjust each weight according to the formula

$$w_{ij}' = w_{ij} + r * (o_j - t_j) * i_i$$

where r is a pre-determined constant called *training rate*, o 's and t 's are predicted and target outputs, respectively.

4. Repeat step 1-3 with adjusted weights until required tolerance is satisfied.

We use the *Mean Square Error*, **MSE**, to indicate the accuracy of the network being trained:

$$MSE = \sum_k \sum_j (t_j - o_j)^2 / N$$

where k is an index for the images represented in the training set and N is the size of this set. An upper limit for an acceptable **MSE** can be specified as tolerance and used as stop criteria for the training.

The *Multi-Layer Network*, *MLN*, has the same type of elements, but a more complex architecture with one or more hidden layers of neurons as shown in *Figure 4*,

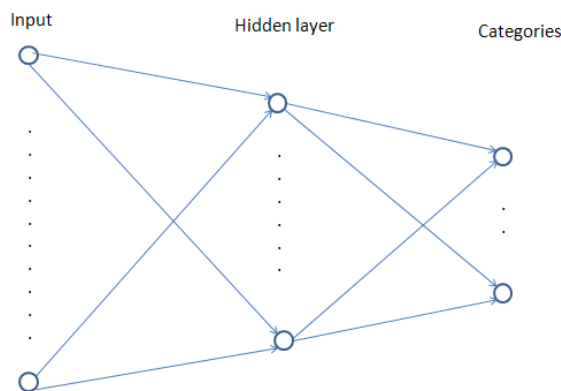


Figure 4: 2-layer MLN

The first layer is now called the hidden layer because it cannot be observed from the outside. A 2-layer network has 2 weight matrices; a 3-layer network has 3 weight matrices, and so on. *MLN* have the ability to learn more complex shapes, discard irrelevant noise, etc. The price to be paid for use of *MLN* is a more intricate learning algorithm, more training images for learning and usually more time to learn than a *SLN*.

3.3 Testing

Evaluation of the performance of a trained system requires an independent set of images associated with the target categories for the objects reflected in the respective images. The images should be representative for the system's intended application domain.

The testing component uses a test set of preprocessed images and the weights from a training session, predicts the object categories for the fragments, and displays the results compared with the targets for each fragment. The computation of predicted category indicators is the same as used by the training algorithm.

When using binary step activation functions, the category indications, I or O , are presented with the associated weighted sums giving the relative strength of the indications, i.e. a high sum value is a stronger indication of the existence of an object belonging to the category, than a smaller value. If the sigmoid activation function is used, the category predictions are estimates of the probabilities for the presence of an object of the respective categories in the fragment.

In case there are more fragments per image, consolidation to image level is done by selecting the highest output category value for each fragment of the image. In this way, each image will be described with the category outputs reflecting the contents of each image fragments.

3.4 Indexing

While training and testing aim at system development, *indexing* and *retrieval* represent the system application. Both components assume that the system is developed with sigmoid activation functions.

Indexing predicts, as the testing component, the category probabilities for each read preprocessed fragment/image, but does require no associated target

categories because these are unknown in application of the system. The index component builds and maintains an index file with 2 elements for each image loaded an entry with category and probability prediction and a link to the respective fragment/image. Since there for each read image can be predicted probabilities for several or all categories even though some of the prediction values are near 0 and can be ignored, the index file can comprise up to $\mathbf{N}^*\mathbf{K}$ records where \mathbf{N} is the number of images/fragments in the collection and \mathbf{K} the number of types of objects wanted.

3.5 Retrieval

The retrieval component contains a simple text-based acting on a query specifying the type of image wanted and a minimum predicted probability for relevant responses. A process searches the index for images satisfying the query, and returns links to the images found. The references in the list are ordered by falling values of the predicted probabilities.

The system performance for a specific image collection and a given query can be expressed by the query *precision* (relevant references as percent of all listed references) and the query *recall* (relevant references in the response as percentage of all relevant images in the collection) and evaluated. More useful evaluation metrics are the average precision and recall based on queries representative for the user group at which the system aims at.

4. List of scripts/programs used

With exceptions of the BrainMaker software, the prototype components are implemented by PHP. The following scripts/programs were used:

Preprocessing

Training SLN

Testing SLN

Converting preprocessing outputs for BrainMaker [California Scientific Software 1998]

MLN training and testing (BrainMaker Professional).

Converting output from BrainMaker for Indexing

Indexing an preprocessed and testing batch

Retrieving image references

The task components are installed on a server to be easily available in the future for experiments based on the system. The present Preprocessing for Vector is slow, and suited only for demonstration.

5. Experiments

To test the performance of the prototype, a number of experiments have been carried out on different sets of images. A major distinction can be made between images from *photographs* and images from simple *drawings, printed symbols, etc.* The former type of images usually has a more detailed and complex background, i.e. more variations, and less distinct object edges than the latter type. From a recognition point of view, the images from photographs can be seen as images inflicted by more noise than those drawn or printed. Images from photographs require more complex neural networks, i.e. large multi-layer networks which can take care of details and at the same time eliminate the noise. All experiments reported in this note are limited to recognition of the second type of images.

To try only a limited number of options offered by the system implemented, a large number of experiments had to be carried out. Before starting the experiments, a number of tuning runs were carried out to identify system settings which seem to work properly. Based on the experience from the tuning, a limited set of parameter values were used. They will be reported in connection with the individual experiments.

The experiments performed were based on 3 sets types of images:

- *TC images.* The purpose was to investigate the system's ability to distinguish between 2 capital letters when they appear *scaled* and *located* in different parts of the image, *rotated* in the 2-dimensional plane, and *combined* in the same images.
- *Shape images.* These images include one or more of the following colored shapes: *circle*, *square*, *star* and *triangle*, and the experiment purpose was to investigate the system capabilities to *recognize* shapes with 2 or more shapes in each image.
- *Icon images.* The objective of these experiments were to investigate how well the system could *distinguish* of 4 types of objects, the system's ability to use its knowledge to correctly *classify* unknown images in the 4 categories and to recognize and classify *overlapping* images correctly. Thirty two icons were downloaded from Internet, and for each a mirror icon was generated. The 32 icon pairs were divided randomly into a set for training the system and another set for testing the trained system.

The test set was also used to demonstrate the Indexing and Retrieval prototype components.

5.1 Experiments with TC

Distinction between the 2 letters T and C is a classical test in pattern recognition, and was probably used for the first time about 50 years ago. In an extended form it serves the purpose for testing a system's ability to recognize different shapes rotated.

The size of the original images varied from 100.000 to 200.000 pixels, a size far above what could be processed directly by *Version Vector* with resources available for this project. A reduction of the fragments of the original images to 40*40 pixel coded as 5 separate 8*8 coarse coded images were used for the TC experiments.

The preprocessing setting options chosen for the first experiment is displayed in *Figure 5*.

Batch name:	<input type="text" value="train_tc40-8"/>
Image name(.jpg):	<input type="text" value="experimentTC/T"/>
Image target category:	<input type="text" value="T"/>
Fragmentation threshold:	<input type="text" value="60"/>
Fragmentation pixel cut-off:	<input type="text" value="500"/>
Reduced image size:	<input type="text" value="40"/>
Contour threshold:	<input type="text" value="25"/>
Coarse image size:	<input type="text" value="8"/>
Angle precision:	<input type="text" value="3"/>
Print details (0/1):	<input type="text" value="1"/>
	<input type="button" value="Submit"/>

Figure 5: Parameter setting for train_tc40-8_b

The *batch name* is the identifier used for the results of the preprocessing, and is also used in the names of all results from the experiment. In all experiments, the first part of a result file name, *train* or *test*, refers to whether this is a set preferred for training or testing. The second part, *tc40-8*, reflects the characteristics of the preprocessing, i.e. the shortcut *tc*, the reduction size *40*, and the coarse matrix size, *8* elements per row. The image name refers to the directory of the image collection and the name of the image (*.jpg* format).

Parameters including threshold parameters for finding fragments and for detecting contours were based on tuning experiments. Since the range between black and white in the grey images is 0 to 256, threshold value equal to 60 was a reasonable starting point for the fragmentation threshold used for identification of fragments. For the contour threshold a value of 25 was used.

The preprocessing component in *Vector* requires a list of different triangles. The length of the list depends on the precision used. Before the experiments were started, a selection of alternative triangle lists varying in precision from 1 to 20 degrees was developed and tried. The list corresponding to 3 degrees precision seemed to be the best and was used in the reported experiments. This list contains 5710 different triangles.

Since the preprocessing produces results which should be invariant for scale, rotation and location, only one image per object are required for training. **Figure 6** shows the 2 images used for training the system.



Figure 6: Training set for TC experiments

The output from the preprocessing, the encoded training image vectors, was used as input for the neural net training component with the following settings.

The settings for the training component are shown in **Figure 7**.

Name of training batch:	<input type="text" value="train_tc40-8"/>
Initial weight range:	<input type="text" value="0.1"/>
Activation function (Threshold=b, Exponential=s):	<input type="text" value="b"/>
Training rate:	<input type="text" value="0.1"/>
Training tolerance:	<input type="text" value="0.05"/>
Max no. of iterations:	<input type="text" value="100"/>
	<input type="button" value="Submit"/>

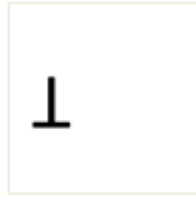
Figure 7 Settings for training using train_40-8

The range for initial random weights was left at default ± 0.1 . In the first experiment, binary threshold functions were used. According to these functions, the output value of a category neuron is 0 if the sum of weighted inputs are less than 0, and 1 if the sum is equal or larger than 0. The larger the value of the training rate the larger are the weight adjustment steps made by the training process and the faster the final weights may be computed. The risk is, however, that the point of optimal weights may be stepped over. The value used, 0.1, is a conservative choice, but the process will usually converge to the optimal set of weights. The training tolerance, 0.05, and the maximum number of iterations, 100, are process stop conditions which by experience seem to work well.

The test set included 18 different images and are shown in **Figure 8**.



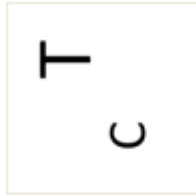
TC1



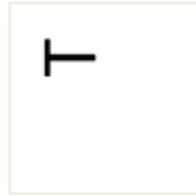
TC2



TC3



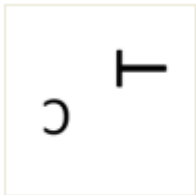
TC4



TC5



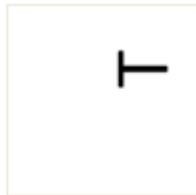
TC6



TC7



TC8



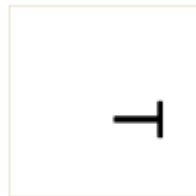
TC9



TC10



TC11



TC12

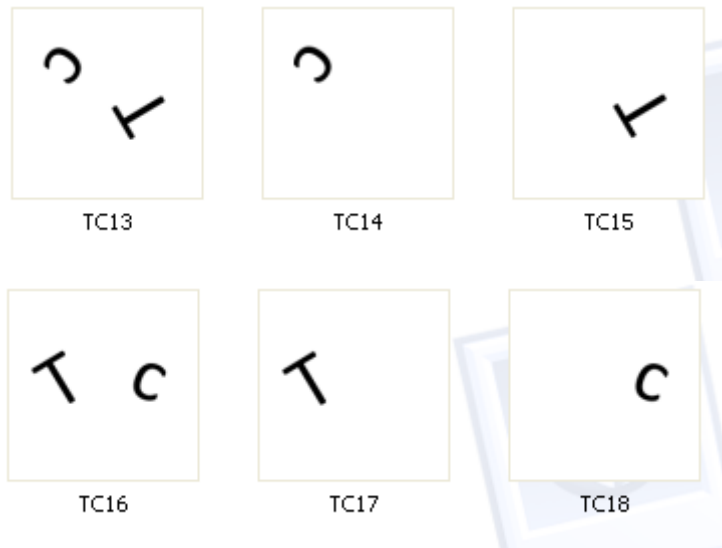


Figure 8: Test images used in the tc experiments

Figure 9 illustrates some of the intermediate results during the preprocessing of image TC13 in the test set of images.

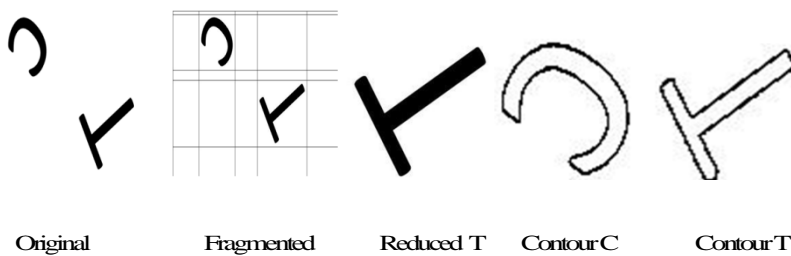


Figure 9: Output of selected preprocessing steps of image TC13

The settings of the Test component are displayed in **Figure 10**. By convention, the weight matrix from the training is named by the batch name with ‘weights’ appended. If a binary activation is applied, a *b* is added, while *s* is used to denote the use of a sigmoid activation function.

Name of testing batch:	<input type="text" value="test_tc40-8"/>
Name of weight matrix:	<input type="text" value="train_tc40-8_weights_b"/>
Activation function(Threshold=b, Exponential=s):	<input type="text" value="b"/>
Testing tolerance:	<input type="text" value="0.1"/>
	<input type="button" value="Submit"/>

Figure 10: Specification for the test_tc40-8

The test results are given in Table 1. In addition to target and predicted category for each fragment, the table also contains 2 columns displaying the sums of weighted inputs to the C and T category neurons. The size of the sums for each fragment can be interpreted as an indicator of the prediction strength.

The table shows that the 18 images were fragmented in 24 fragments corresponding to 12 single letter images and 6 images with 2 letters fragments each.

Inspection of the fragmentation sub-table, the trained network component managed to correctly identify all letters except for the T in fragment 12-1 and in fragment 14-0. (N.B. The image numbers referred to in the table run from 0 to 17, while in **Figure 8** the images are numbered from 1 to 18). The most likely explanation for the incorrect predictions is that the neural network component does not manage the specific rotation of the T in these two fragments.

The small differences between sums in fragments 15-0 and 16-0 indicate that the computed predictions are uncertain.

In the second sub-table of Table 1, the results are consolidated for images. No additional information appears in the sub-table, but it can be more convenient to use when content of images should be referred to.

Experiment 40-8s

The same training and testing image sets used in the previous experiment were used with sigmoid activation functions in training and testing components. Results are given in Table 2.

The predictions are in this experiment presented by probability estimates. It is best illustrated in the sub-table for image consolidated results in which we can see that the probabilities for a T and a C being present in image 15 (picture TC16) are 1.000000 and 0.880797, respectively. In image 16 (picture TC17), the corresponding probabilities are 0.331812 and 0.880797 indicating the uncertainty of the recognition of the letters of the 2 images. Using the highest probabilities, 1.000000 and 0.880797, the correct letters, T and C, are selected.

The conclusion we can draw from the TC experiments are that improved results may be obtained if the rotation recording is performed more precisely. Another alternative is using a higher resolution in the reduced image/fragment representation, for example 80*80 pixels.

5.2 Experiment with Shapes

The objectives of the second set of experiments are to investigate the system's ability to recognize frequently used colored shapes, shapes which cannot be separated by horizontal or vertical lines, and shape representations partly hidden behind another. The four colored shapes shown in *Figure 11* are used as training set. In parenthesis, it should be pointed out that in this application the color would have solved all problems!



Figure 11: The training set of Experiment Shapes

The parameter setting for the pre-processing of shape images is given in *Figure 12*. The setting of this experiment differs from the setting of the previous experiments by image reduction to 80×80 pixels, which preserves more details and seem to work better for this type of images. The change implies that the number of coarse matrices is increased from 5 to 10.

Batch name:	<input type="text" value="train_shapes80-8"/>
Image name(.jpg):	<input type="text" value="xperimentShapes/circle"/>
Image target category:	<input type="text" value="circles"/>
Fragmentation threshold:	<input type="text" value="60"/>
Fragmentation pixel cut-off:	<input type="text" value="500"/>
Reduced image size:	<input type="text" value="80"/>
Contour threshold:	<input type="text" value="30"/>
Coarse image size:	<input type="text" value="8"/>
Angle precision:	<input type="text" value="3"/>
Print details (0/1):	<input type="text" value="1"/>
	<input type="button" value="Submit"/>

Figure 12: Settings for preprocessing of train_shapes80-8

The 4 different shapes were learned by the neural network in 20 iterations as indicated in *Figure 13*. The figure also indicates the convergence of the learning process as reflected by the metric *MSE*.

TRAINING SUMMARY

Batch name: train_shapes80-8
Categories: 4
Activity type: b
Initial weight range: 0.1
Training rate: 0.1
Tolerance: 0.05
Max number of iterations:
Input nodes: 5710
Number of images: 4

Category catalog number with name

0: circles

1: squares

2: stars

3: triangles

Target name of images

Image: 0; Target name: circles

Image: 1; Target name: squares

Image: 2; Target name: stars

Image: 3; Target name: triangles

Target matrix

1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, 0,

0, 0, 0, 1,

MSE: 0.75

MSE: 0.375

MSE: 0.25

MSE: 0.1875

MSE: 0.25

MSE: 0.25

MSE: 0.0625

MSE: 0.125

MSE: 0.0625

MSE: 0.25

MSE: 0.25

MSE: 0.0625

MSE: 0.125

MSE: 0.0625

MSE: 0.25

MSE: 0.25

MSE: 0.0625

MSE: 0.0625

MSE: 0.0625

MSE: 0

Predicted matrix

1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, 0,

0, 0, 0, 1,

Number of iterations: 20

The trained weights are stored in: archive/train_shapes80-8_weights.txt as a matrix corresponding to the set of 4 and 5710 input vector. The target catalog is stored as: archive/train_shapes80-8_category_catalog.

Figure 13: Summary of training report for train_shapes80-8

The series of *MSE* values indicates how the learning is developing in cycles to the ending point $MSE = 0$.

To test the system's performance with respect to the abilities we wanted to investigate, 2 test sets were constructed. The first set contained the 6 shapes shown in ***Figure 14***.

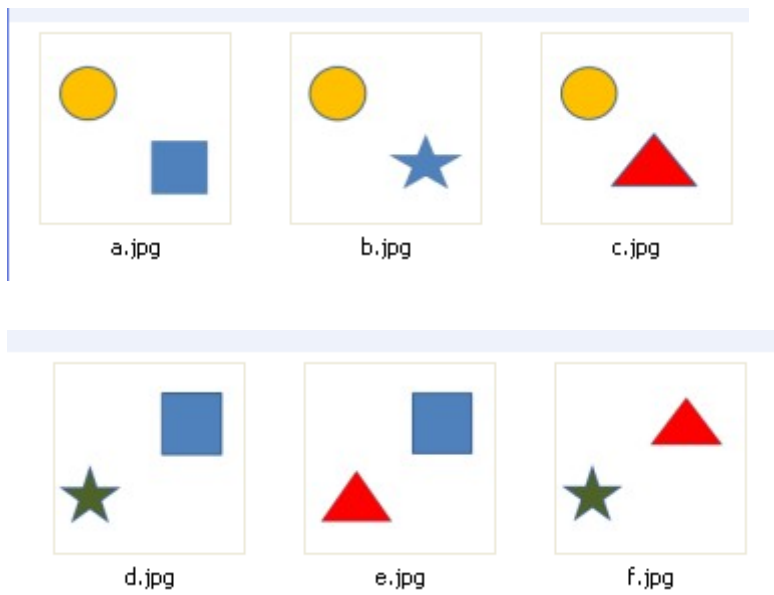


Figure 14: First test set for Experiment Shapes

The 6 images represent different combinations of shape pairs. Common for all these images are that they do not overlap and that the object shapes can all be separated by horizontal/vertical lines.

Table 3 displays the trained system's ability to separate and recognize the different shapes. All images and fragments were correctly recognized.

The second test set for Shapes displayed in **Figure 15** is more challenging. It contains 5 images, each containing 2 object reflections which are inseparable with horizontal/vertical lines, and 3 images with partly overlapping object reflections.

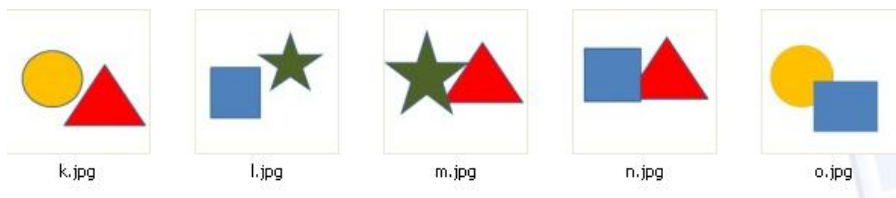


Figure 15: Second test set for Shapes

The test results are reproduced in Table 4. As expected, the system was unable to separate the multiple fragments in these images. For the 3 images k , m and o , a single object was correctly recognized from each even though the circle of image o was vaguely indicated. For the remaining 2 images, images l and n , no correct recognitions were made.

We can conclude that the success of the system for this kind of application depends on separation of the object reflection, and that this part of the Preprocessing component should be improved.

5.3 Experiments with Icons

So far, the images we have been experimenting with had clean shapes with no details or noise disturbing the identification of the object shapes. Also the test images used in the previous experiments were manipulations of the images used in training the system.

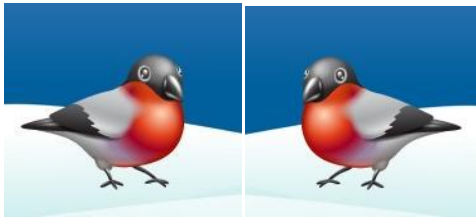
In the Icons set of experiments, we shall use images which do not have clearly defined forms and which also contain varying amount of details as well as independent test images independent of the training images.

These experiments are based on 32 different icons and paired with their mirror images. The icons are downloaded from the net. The different image pairs, original and mirror image, are pre-classified as belonging to one of 4 categories: *animals*, *buildings*, *cars* and *persons* with 8 image pairs in each category randomly ordered and assigned a sequence number from 1 to 8. The original images are identified by the pair number, and the mirror images by the pair number appended with a . Pairs nos. 1, 3, 5 and 7 of each category are used for training the system, while pairs nos. 2, 4, 6, and 8 of each category are used for testing. Note that the test set images are independent, not manipulated copies, of the training image set.

Figure 16 displays the 16 image pairs used for training the system. Note the difference between the rotated images used in previous experiments and the mirror images in this experiment. The mirror images are the result of rotation

around a vertical axis through the image while the previous rotations are around a point in the image.

Figure 17 displays the 16 test image pairs. As pointed out above, they are completely independent of the set used for training the system. The only common features of the 2 sets are those which humans can perceive and use for assigning each reflected object to one of the four categories: *animals, buildings, cars* and *persons*. The crucial question is to which degree the system developed and trained can mimic the humans in extracting and utilizing these features in assigning the icons to the correct categories



animal1



animal1a



animal3



animal3a



animal5



animal5a



animal7



animal7a



building1



building1a



building3



building3a



building5



building5a



building7



building7a



car1



car1a



car3



car3a

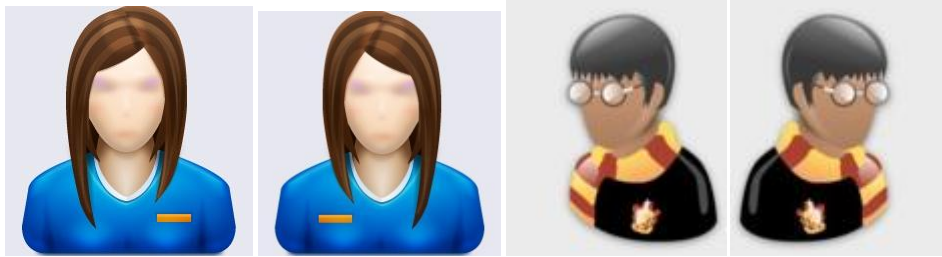


car5

car5a

car7

car7a

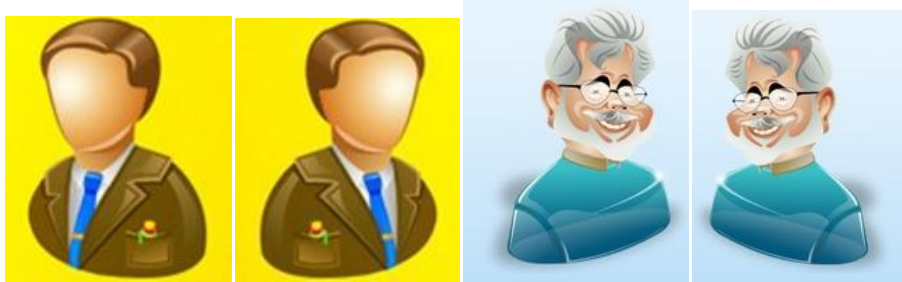


person1

person1a

person3

person3a



person5

person5a

person7

person7a

Figure 16: Icon image set for training



animal2

animal2a

animal4

animal4a



animal6

animal6a

animal8

animal8a



building2

building2a

building4

building4a

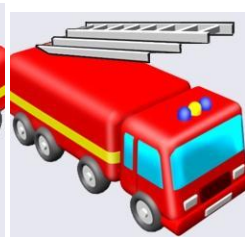
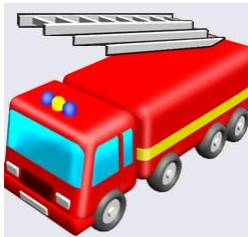


building6

building6a

building8

building8a



car2

car2a

car4

car4a



car6

car6a

car8

car8a



Figure 17: Icon image set for testing

To give an impression of the quality of contour images from train and test sets, the following *Figure 18* to *Figure 21* are a selection.



Figure 18: Contour image of animal8



Figure 19: Contour image of building2



Figure 20: Contour image of car2



Figure 21: Contour image of person8

The first experiment with icons was designed for *Vector* followed by a couple of experiments based on *Matrix*. In all experiments, sigmoid activation functions are used for better results and more convenient interpretation.

Experiment: icons80-8_s

First, an experiment using *Version Vector* with sigmoid activation functions is presented. In all icons experiments, the original images were reduced to 80*80 pixels. Since we did not have a capacity to process more than 8*8 bits matrices with *Vector* encoding, 10 coarse matrices of the reduced matrix for each image were required, and the preprocessing time for each icon image consumed up to 10 minutes in processing time.

Figure 22 displays the setting used for training of the *Vector* representation training set. The preprocessed batch of images was denoted *train_icons80-8*. Training rate 0.05 indicates that a small training rate was used because tuning runs showed that

Name of training batch:	<input type="text" value="train_icons80-8"/>
Initial weight range:	<input type="text" value="0.1"/>
Activation function (Threshold=b, Exponential=s):	<input type="text" value="s"/>
Training rate:	<input type="text" value="0.05"/>
Training tolerance:	<input type="text" value="0.05"/>
Max no. of iterations:	<input type="text" value="100"/>
	<input type="button" value="Submit"/>

Figure 22: Specification for icons_80-8 training

the weight adjustments during training tended to jump past the optimal weight point with larger rates. The required training tolerance was defined as $MSE < 0.05$.

Figure 23 reports the main facts from the training on *train_icons80-8*.

TRAINING SUMMARY

Batch name: train_icons80-8
 Categories: 4
 Activity type: s
 Initial weight range: 0.1

Training rate: 0.05
Tolerance: 0.05
Max number of iterations: 500
Input nodes: 5710
Number of images: 34

Category catalog number with name

0: animals
1: buildings
2: cars
3: persons

Last 10 iteration values of MSE:

MSE: 0.14705882352941
MSE: 0.073529411764706
MSE: 0.088235294117647
MSE: 0.10294117647059
MSE: 0.14705882352941
MSE: 0.051470588235294
MSE: 0.080882352941176
MSE: 0.10294117647059
MSE: 0.17647058823529
MSE: 0.022058823529412

Number of iterations: 467

The trained weights are stored in: archive/train_icons80-8_weights_s.txt as a matrix corresponding to the set of 4 and 5710 input vector. The target catalog is stored as: archive/train_icons80-8_category_catalog.

Figure 23: Training report from Experiment: icons80-8

The report shows that the invariant image encoded input vectors each contained 5710 inputs, i.e. elements in the encoded vector representation. The system required 467 iterations through the set of 32 encoded images to satisfy the condition $MSE < 0.05$. The values for MSE after each of the last 10 iterations, demonstrate how the weight adjustment expressed by MSE varied in cycles, but

converged to 0.02. The final training result demonstrates that the system has learned to correctly recognize and classify the 32 images of the training set.

Before the test results are discussed, the following consideration can be instructive. Assume that no information exists about the distribution of icons by category in the test set. If the 32 images were put into separate envelopes and mixed, the best strategy of predicting the enclosed icon's type category would be to select one of the 4 categories by random and note the result on each envelope.. The expected percentage of correct predictions when opening the envelopes would be 25% within the whole set (8 correctly predicted images), and within each category (2 correctly predicted images). This can serve as a basis for evaluating the following test results.

The results from the test performed on icons *test_icons80-8* are displayed in Table 5. Each row of the table refers to an image and each cell of the row to the conditional probability prediction that an object belonging to the column category is reflected in the row image. Remember that the probability predictions are independent and do not add up to 1 because an image can reflect more objects with different probabilities.

Introducing the convention that the category associated with the highest probability prediction is adopted as the predicted object category, inspection of the table shows that for 8 images were correct predicted. The remaining 24 images were incorrectly and with probabilities which in general were as high as the 8 correct predictions. Random assignments of category to each image would give results which would compete with those from this experiment.

A restriction with the preprocessing used in *Version Vector* is the exponentially growth of triangles with increasing resolution of the image representation. We bypassed this problem in the previous experiments by introducing coarse code representation. It can be intuitively seen, and objectively proved, however, that the more details and noise are present in the original image, the less satisfactory will the coarse representation be. This can serve as a possible explanation why the previous experiments were successful except the last icon experiment.

The triangle encoding was introduced to take care of the rotation problem. In the TC experiments we verified that the encoding was effective as a means to obtain rotational invariance. In applications, when rotation does not an important factor, as in the present icon application, the *Version Matrix* providing binary matrix representations of the reduced images can substitute the triangle coded output vectors.

In the experiment, *icons80-80_s_matrix1* based on *Matrix*, we use a bit matrix corresponding to a reduced $80*80$ pixel image resolution as output for each image. No triangle encoding or coarse representations are required and all details from the $80*80$ matrix can be utilized.

The *Version Matrix* preprocessing generates binary matrices with 6400 binary elements. In the present experiment we use a *SLN* with the same structure as in all the above experiments with about 25000 weight (6400 elements * 4 categories) connections. The program used (BrainMaker Professional) is however more powerful than that used so far. The training rate used was set to 0.1 , and the training tolerance to 0.05 . The weight matrix was adjusted to the training icons in 23 training iterations.

Table 6 displays the main results for this experiment. The number of correct predictions in this experiments was 15 , an improvement compared with the previous *icons80-8_s* experiment, but still not very useful results.

Experiment: icons80-80_s_matrix2

One objection to the simple, SLN used so far is that it has a limited capability to adjust to complicated inputs. In a MLN, the outputs of one layer of neurons are used as inputs to another layer and so on. We use a 2-layer model in which the 6400 input elements from preprocessing are connected to a first layer of 400 hidden neurons. The outputs from the 400 neurons are used as inputs in the final layer of 4 neurons. The 2 weight matrices established count for about 2.5 million connections or weights to be adjusted. This software can also make small random changes in the input elements each time they are read during training. This has a similar effect as having a larger number of training icons and making the weight matrices more reliable. The software used also permits a linearly varying learning rate. In this experiment, it was specified to vary from rate 0.1 when no image was correctly classified, to rate 0 when all were correctly classified. In other words, the closer to perfect predictions, the smaller the adjustment steps to avoid stepping past the optimal goal.

The training progress, measured by the RMS metric, can in BrainMaker be inspected during the training. When to stop training is to decide when the system has learned the general feature of the training set to be able to discover these features in new images. If training continues with more iteration, the specific features of the training set images are also learned, but these make the system unfitted to generalize. In the *icons80-80_s* experiment, trials indicated that training for about 40 iterations were about optimal.

The results are presented in Table 7. It shows that 23 predictions were correct corresponding to an overall precision value of 72% for the complete test set. This result seems interesting, and is followed up further in the next section.

6. Indexing and retrieval

6.1 Indexing

The system trained with *train_icons80-80* and the independent set *test_icons80-80* by means of the *MLN* were also used for experiments with the prototype components *Indexing* and *Retrieval*.

The *Indexing* component read preprocessed images of batch *test_icons80-80* for which the probabilities of the different categories have been predicted using the trained *MLN*. The component generates 2 indices, a *text index* for text queries and a *probability vector index* for image queries. One or more index items are created for each image of the image collection (batch) depending on its fragmentation.

Each item of the text index contains 3 parts, the name of the category, the predicted probability for this category and a link to the image indexed. Applied on the icons of *test_icons80-80*, the index contains the same information as Table 7, but ordered differently. **Figure 24** displays a part of the text index for *test_icons80-80*.

Category	Probability	Link
animals	0.0964	../collection/experimentIcons/animal2.jpg
animals	0.2144	../collection/experimentIcons/animal2a.jpg
animals	0.4064	../collection/experimentIcons/animal4.jpg
animals	0.3364	../collection/experimentIcons/animal4a.jpg
animals	0.3172	../collection/experimentIcons/animal6.jpg
animals	0.3594	../collection/experimentIcons/animal6a.jpg
animals	0.7610	../collection/experimentIcons/animal8.jpg
animals	0.4714	../collection/experimentIcons/animal8a.jpg
animals	0.0656	../collection/experimentIcons/building2.jpg
animals	0.6776	../collection/experimentIcons/building2a.jpg
animals	0.0776	../collection/experimentIcons/building4.jpg
animals	0.0820	../collection/experimentIcons/building4a.jpg
animals	0.2016	../collection/experimentIcons/building6.jpg
animals	0.0520	../collection/experimentIcons/building6a.jpg
animals	0.2994	../collection/experimentIcons/building8.jpg
animals	0.0802	../collection/experimentIcons/building8a.jpg
animals	0.0490	../collection/experimentIcons/car2.jpg
animals	0.4100	../collection/experimentIcons/car2a.jpg

animals	0.0882	../collection/experimentIcons/car4.jpg
animals	0.1754	../collection/experimentIcons/car4a.jpg
animals	0.5842	../collection/experimentIcons/car6.jpg
animals	0.2800	../collection/experimentIcons/car6a.jpg
animals	0.2746	../collection/experimentIcons/car8.jpg
animals	0.7314	../collection/experimentIcons/car8a.jpg
animals	0.1214	../collection/experimentIcons/person2.jpg
animals	0.0200	../collection/experimentIcons/person2a.jpg
animals	0.1664	../collection/experimentIcons/person4.jpg
animals	0.4462	../collection/experimentIcons/person4a.jpg
animals	0.1340	../collection/experimentIcons/person6.jpg
animals	0.0560	../collection/experimentIcons/person6a.jpg
animals	0.2734	../collection/experimentIcons/person8.jpg
animals	0.0484	../collection/experimentIcons/person8a.jpg
buildings	0.2450	../collection/experimentIcons/animal2.jpg
buildings	0.1866	../collection/experimentIcons/animal2a.jpg
buildings	0.1180	../collection/experimentIcons/animal4.jpg
buildings	0.1182	../collection/experimentIcons/animal4a.jpg
buildings	0.0862	../collection/experimentIcons/animal6.jpg

Figure 24: Part of the text index for the image collection test_icons80-8

The index files is used by the retrieval component. A few examples illustrate how the retrieval process works.

6.2 Text query retrieval

A query form requiring 3 parameters is shown in *Figure 25*. The displayed query illustrates search of references to icons representing *animals* with predicted probability > 0.5 from collection *test_icons80-80*.

Text based retrieval

This component is a simple text-based retrieval from a collection of images which has been indexed

Wanted category (mode=1):	<input type="text" value="animals"/>
Name of batch to be retrieved from:	<input type="text" value="test_icons80-80"/>
Lower probability limit:	<input type="text" value="0.5"/>
<input type="button" value="Submit"/>	

Figure 25: The query form

The *Retrieval* component searches the index for possible candidates and returns a table with results ordered by predicted probability. **Figure 26** shows the response to the query in the previous figure.

Retrieval Results

Category:	Probability:	Link:
animals	0.7610	../collection/experimentIcons/animal8.jpg
animals	0.7314	../collection/experimentIcons/car8a.jpg
animals	0.6776	../collection/experimentIcons/building2a.jpg
animals	0.5842	../collection/experimentIcons/car6.jpg

Figure 26: Response to text query for animals.

The precision obtained for this query, 25%, is rather low and the recall is only 12%.

Similar queries for *buildings*, *cars* and *persons* with the same probability limit give the following result:

Retrieval Results

Category:	Probability:	Link:
buildings	0.6040	../collection/experimentIcons/building2a.jpg
buildings	0.5514	../collection/experimentIcons/building6.jpg

Retrieval Results

Category:	Probability:	Link:
cars	0.8424	../collection/experimentIcons/car2a.jpg
cars	0.7826	../collection/experimentIcons/car4.jpg
cars	0.6404	../collection/experimentIcons/person8a.jpg
cars	0.5256	../collection/experimentIcons/building4.jpg
cars	0.5206	../collection/experimentIcons/car6a.jpg

Retrieval Results

Category:	Probability:	Link:
persons	0.8916	./collection/experimentIcons/person6a.jpg
persons	0.6354	./collection/experimentIcons/person2a.jpg
persons	0.5444	./collection/experimentIcons/person4a.jpg
persons	0.5260	./collection/experimentIcons/person2.jpg

Figure 27: Response to text queries for buildings, cars and persons

The responses show *precisions* to be 100 %, 60 % and 100%, and *recalls* 25%, 37% and 50% for the 3 queries. The query results would vary with the probability limit used. In general, a higher lower probability limit will increase precision, but decrease recall values.

By specifying 0.0 as probability limit, the responses give a basis for deriving the average precision-recall curve for the 4 retrieval types as shown in **Figure 28**.

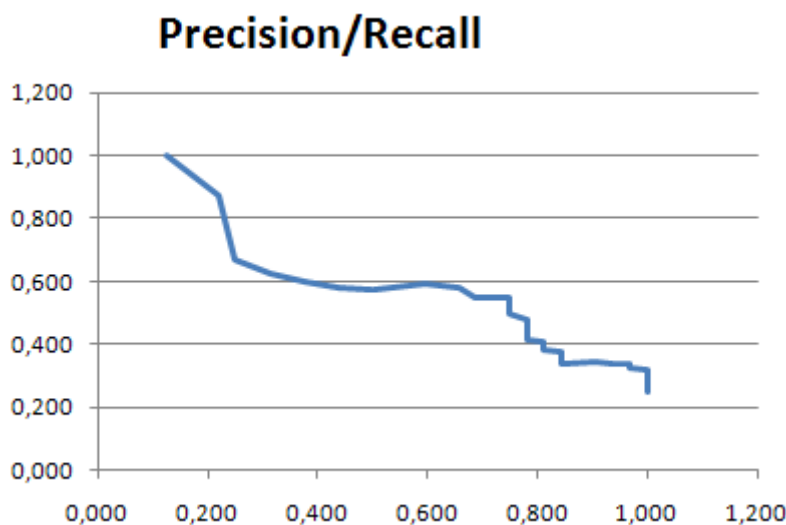


Figure 28: Average precision-recall curve for the icons80-80 application

The curve illustrates the typical form of an average precision-recall curve with decreasing precision for increasing recall.

6.3 Image query retrieval

When loaded into the collection, each image is also indexed by the system with a vector of predicted probabilities. Similarly, a query when presented for the system is evaluated and assigned a probability vector. The search and selection of responses is based on the similarity (the sum of squared differences between the query image vector and each of collection image vectors) and a ranked response is returned.

For illustration an image set of queries were created by drawing an *animal*, a *building*, a *car* and a *person* and preprocessed, indexed and saved as *query_draw80-80*. (This example is using an alternatively trained system). The set is reproduced in **Figure 29**.

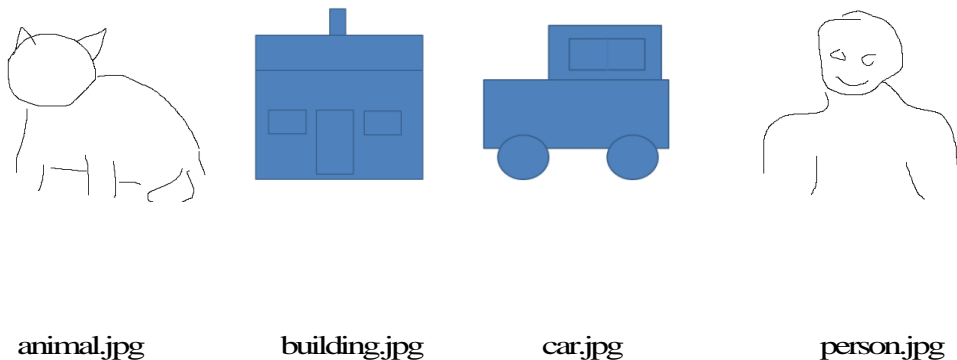


Figure 29: Image query set

The image query search is activated by the interface presented in **Figure 30**

Image based retrieval

This component is a simple retrieval from a collection of images which has been indexed based on an image query.

Query name:	<input type="text" value="query_draw80-80"/>
Name of batch to be retrieved from:	<input type="text" value="test_icons80-80"/>
Max. response limit:	<input type="text" value="3"/>
	<input type="button" value="Submit"/>

Figure 30: Image based retrieval menu

The results from the drawing query set is reproduced in **Figure 31** below

Query 0: [../collection/experimentdraw/animal.jpg](http://collection/experimentdraw/animal.jpg)

Rank: Link:

- 0 [../collection/experimenticons/animal4.jpg](http://collection/experimenticons/animal4.jpg)
- 1 [../collection/experimenticons/car2.jpg](http://collection/experimenticons/car2.jpg)
- 2 [../collection/experimenticons/person4.jpg](http://collection/experimenticons/person4.jpg)

Query 1: [../collection/experimentdraw/building.jpg](http://collection/experimentdraw/building.jpg)

Rank: Link:

- 0 [../collection/experimenticons/animal4a.jpg](http://collection/experimenticons/animal4a.jpg)
- 1 [../collection/experimenticons/animal2.jpg](http://collection/experimenticons/animal2.jpg)
- 2 [../collection/experimenticons/building8.jpg](http://collection/experimenticons/building8.jpg)

Query 2: [../collection/experimentdraw/car.jpg](http://collection/experimentdraw/car.jpg)

Rank: Link:

- 0 [../collection/experimenticons/car4.jpg](http://collection/experimenticons/car4.jpg)
- 1 [../collection/experimenticons/animal2a.jpg](http://collection/experimenticons/animal2a.jpg)
- 2 [../collection/experimenticons/car6.jpg](http://collection/experimenticons/car6.jpg)

Query 3: [../collection/experimentdraw/person.jpg](http://collection/experimentdraw/person.jpg)

Rank: Link:

- 0 [../collection/experimenticons/animal8.jpg](http://collection/experimenticons/animal8.jpg)
- 1 [../collection/experimenticons/person8.jpg](http://collection/experimenticons/person8.jpg)
- 2 [../collection/experimenticons/person4.jpg](http://collection/experimenticons/person4.jpg)

Figure 31: Response to the query set: query_draw80-80

Only the 3 highest ranked responses are shown. The first drawing *animal* in the query set, received the icon *animal4* as the most similar icon. Also the drawing *car* generated icon *car4* as the highest ranked returned response. The drawings

building and person gave also relative useful responses (1 and 2 relevant responses out of 3).

It is possible to compute an average recall-precision curve also for this query for comparison with the one developed for the text-based retrieval. A comparison of the 2 retrieval methods in the present application is shown in *Figure 32*. (NB. note the text based curve which differs from that in Figure 28 because of an alternatively trained system is used.)

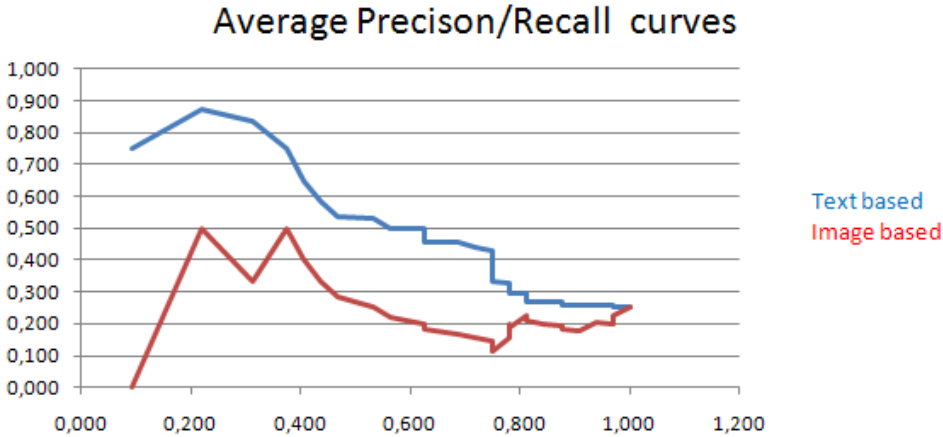


Figure 32: Comparison of precision/recall curves for text and image based retrieval

The curves indicate that the text based retrieval has a higher precision than the image based retrieval. However, it should be strongly emphasized that both curves are based on only 4 queries and the results have little statistical significance.

7. Conclusions and future tasks

The number of experiments performed is not large enough to draw any final conclusions. However, some preliminary conclusions and intentions may be offered:

- Invariant vector representation requires too much resources and *Coarse representation* is ineffective with matrices highly populated. Work on the *Vector* version will not be continued
- *Matrix representation* and *MLN* seem promising
- Preparation of *contour images* needs improvement
- *Fragmentation algorithm* must be refined
- Experiments with *higher resolution* image reduction (100*100 pixels per fragment) will be initiated
- Experiments with *photographic* images will be started
- Redesign of the preprocessing for taking *colors* and their image space distribution to account will be considered

8. Tables

Table 1: Results from Experiment: tc40-8_b

Batch matrix name: test_TC40-8

Weights matrix: train_tc40-8_weights_b

Number of input nodes: 2855

Number of categories: 2

Number of images: 18

Number of fragments: 24

Table of all fragment results

Image no.:	Fragment no.:	Target:	Predicted:	Sum: C:	Sum: T:
0	0	TC	T	-130.400000	65.200000
0	1	TC	C	53.600000	-21.400000
1	0	T	T	-130.400000	65.200000
2	0	C	C	52.200000	-20.400000
3	0	TC	T	-196.300000	104.600000
3	1	TC	C	47.800000	-20.300000
4	0	T	T	-196.300000	104.600000
5	0	C	C	48.800000	-20.800000
6	0	TC	T	-234.000000	124.500000
6	1	TC	C	56.800000	-24.800000
7	0	C	C	55.000000	-23.900000
8	0	T	T	-234.000000	124.500000
9	0	TC	C	50.800000	-21.800000
9	1	TC	T	-125.200000	62.600000
10	0	C	C	50.000000	-21.400000
11	0	T	T	-125.200000	62.600000
12	0	TC	C	84.400000	-40.700000
12	1	TC	C	42.900000	-19.800000
13	0	C	C	90.400000	-44.900000
14	0	T	C	42.900000	-19.800000
15	0	TC	T	-0.700000	2.000000
15	1	TC	C	71.600000	-35.200000
16	0	T	T	-0.700000	2.000000
17	0	C	C	75.600000	-37.200000

Table of consolidated image results

Image no.: Target: Category C: Category T:

0	TC	C	T
1	T		T
2	C	C	
3	TC	C	T
4	T		T
5	C	C	
6	TC	C	T
7	C	C	
8	T		T
9	TC	C	T
10	C	C	
11	T		T
12	TC	C	
13	C	C	
14	T	C	
15	TC	C	T
16	T		T
17	C	C	

Table 2: Results from Experiment: tc40-8_s

Batch matrix name: test_tc40-8

Weights matrix: train_tc40-8_weights_s

Number of input nodes: 2855

Number of categories: 2

Number of images: 18

Number of fragments: 24

Table of all fragment results

Image no.:	Fragment no.:	Target:	Predicted:	Prob.:	C:	Prob.:	T:
0	0	TC	T	0.000000	1.000000		
0	1	TC	C	1.000000	0.000000		
1	0	T	T	0.000000	1.000000		
2	0	C	C	1.000000	0.000000		
3	0	C	T	0.000000	1.000000		
3	1	C	C	1.000000	0.000000		
4	0	T	T	0.000000	1.000000		
5	0	C	C	1.000000	0.000000		
6	0	TC	T	0.000000	1.000000		
6	1	TC	C	1.000000	0.000000		
7	0	C	C	1.000000	0.000000		
8	0	T	T	0.000000	1.000000		
9	0	TC	C	1.000000	0.000000		
9	1	TC	T	0.000000	1.000000		
10	0	C	C	1.000000	0.000000		
11	0	T	T	0.000000	1.000000		
12	0	TC	C	1.000000	0.000000		
12	1	TC	C	1.000000	0.000000		
13	0	C	C	1.000000	0.000000		
14	0	T	C	1.000000	0.000000		
15	0	TC	T	0.331812	0.880797		
15	1	TC	C	1.000000	0.000000		
16	0	T	T	0.331812	0.880797		
17	0	C	C	1.000000	0.000000		

Table of consolidated image results

Image no.:	Target:	Prob.:	C: Prob.:	T:
0	TC	1.000000	1.000000	
1	T	0.000000	1.000000	
2	C	1.000000	0.000000	
3	TC	1.000000	1.000000	
4	T	0.000000	1.000000	
5	C	1.000000	0.000000	
6	TC	1.000000	1.000000	
7	C	1.000000	0.000000	
8	T	0.000000	1.000000	
9	TC	1.000000	1.000000	
10	C	1.000000	0.000000	
11	T	0.000000	1.000000	
12	TC	1.000000	0.000000	
13	C	1.000000	0.000000	
14	T	1.000000	0.000000	
15	TC	1.000000	0.880797	
16	T	0.331812	0.880797	
17	T	1.000000	0.000000	

Table 3: Results 1 from Experiment: shapes80-8_b

Batch matrix name: test_shapes80-8
 Weights matrix: train_shapes80-8_weights_b
 Number of input nodes: 5710
 Number of categories: 4
 Number of images: 6
 Number of fragments: 12

Table of all fragment results

Image no.:	Fragment no.:	Target:	Predicted:	Sum: circles:	Sum: squares:	Sum: stars:	Sum: triangles:
0	0	circle-square	circles	343.900000	-438.600000	-152.600000	-152.500000
0	1	circle-square	squares	-229.700000	146.900000	-238.500000	-237.800000
1	0	circle-star	circles	343.900000	-438.600000	-152.600000	-152.500000
1	1	circle-star	stars	-366.900000	-566.100000	280.500000	-39.500000
2	0	circle-triangle	circles	338.600000	-441.400000	-149.000000	-150.400000
2	1	circle-triangle	triangles	-169.300000	-379.900000	-95.000000	211.700000
3	0	star-square	squares	-264.100000	227.600000	-234.400000	-303.300000
3	1	star-square	stars	-289.100000	-626.500000	332.100000	-143.400000
4	0	triangle-square	squares	-264.100000	227.600000	-234.400000	-303.300000
4	1	triangle-square	triangles	-266.300000	-504.200000	-57.400000	437.100000
5	0	star-triangle	triangles	-222.400000	-480.400000	-47.600000	336.000000
5	1	star-triangle	stars	-273.400000	-629.800000	309.900000	-111.800000

Table of consolidated image results

Image no.:	Target:	Category circles:	Category squares:	Category stars:	Category triangles:
0	circle-square	circles	squares		
1	circle-star	circles		stars	
2	circle-triangle	circles			triangles
3	star-square		squares	stars	
4	triangle-square		squares		triangles
5	star-triangle			stars	triangles

Table 4: Results 2 from Experiment: shapes80-8_b

Batch matrix name: test2_shapes80-8

Weights matrix: train_shapes80-8_weights_b

Number of input nodes: 5710

Number of categories: 4

Number of images: 5

Number of fragments: 5

Table of all fragment results

Image no.:	Fragment no.:	Target:	Predicted:	Sum: circles:	Sum: squares:	Sum: stars:	Sum: triangles:
0	0	circle-triangle	squares	14.600000	282.300000	-564.500000	186.900000
1	0	square-star	circles	256.100000	-98.300000	-413.600000	234.600000
2	0	star-triangle	triangles	-79.700000	46.700000	-402.700000	349.100000
3	0	square-triangle	squares	-180.800000	289.800000	-373.200000	33.400000
4	0	circle-square	squares	172.900000	344.200000	-603.000000	3.600000

Table of consolidated image results

Image no.:	Target:	Category circles:	Category squares:	Category stars:	Category triangles:
0	circle-triangle	circles	squares		triangles
1	square-star	circles			triangles
2	star-triangle		squares		triangles
3	square-triangle		squares		triangles
4	circle-square	circles	squares		triangles

Table 5: Results from Experiment: icons80-8_s_vector

Batch matrix name: test_icons80-8
Weights matrix: train_icons80-8_weights_s
Number of input nodes: 6400
Number of categories: 4
Number of images: 32
Number of fragments: 32

Test results

Table of all fragment results

Image no.:	Fragment no.:	Target:	Computed:	Prob.: animals:	Prob.: buildings:	Prob.: cars:	Prob.: persons:
0	0	animals	buildings	0.000200	0.999400	0.000200	0.001000
1	0	animals	persons	0.000200	0.649400	0.000200	0.754200
2	0	animals	buildings	0.100200	0.742600	0.000200	0.000200
3	0	animals	buildings	0.000200	0.999000	0.000200	0.000200
4	0	animals	buildings	0.063200	0.949600	0.000200	0.020600
5	0	animals	animals	0.516200	0.247600	0.237200	0.058000
6	0	animals	buildings	0.062000	0.950600	0.000200	0.020600
7	0	animals	buildings	0.062000	0.950600	0.000200	0.020600
8	0	buildings	animals	0.346200	0.004400	0.317600	0.151600
9	0	buildings	animals	0.280600	0.021600	0.208400	0.206600
10	0	buildings	persons	0.000200	0.176200	0.000200	0.931200
11	0	buildings	buildings	0.000200	0.996200	0.000200	0.211600
12	0	buildings	persons	0.000200	0.136000	0.000200	0.956000
13	0	buildings	persons	0.000200	0.128200	0.000200	0.961200
14	0	buildings	persons	0.000200	0.102200	0.000200	0.958400
15	0	buildings	persons	0.000200	0.149400	0.000200	0.943200
16	0	cars	buildings	0.062000	0.950600	0.000200	0.020600
17	0	cars	buildings	0.062600	0.950200	0.000200	0.020600
18	0	cars	animals	0.645200	0.000200	0.000200	0.000200
19	0	cars	cars	0.103000	0.000200	0.988600	0.000400
20	0	cars	persons	0.255400	0.012200	0.188000	0.265000
21	0	cars	animals	0.689200	0.063400	0.531600	0.022200
22	0	cars	cars	0.098000	0.000200	0.986000	0.001000
23	0	cars	cars	0.174200	0.001000	0.951200	0.004400

24	0	persons	persons	0.138200	0.001400	0.024000	0.815600
25	0	persons	persons	0.001000	0.000200	0.000200	0.999400
26	0	persons	animals	0.618600	0.000200	0.016200	0.000200
27	0	persons	cars	0.302600	0.000200	0.964000	0.000200
28	0	persons	cars	0.003000	0.000200	0.689000	0.424200
29	0	persons	persons	0.001600	0.000200	0.260400	0.953400
30	0	persons	animals	0.986200	0.002600	0.000200	0.000200
31	0	persons	animals	0.986200	0.002600	0.000200	0.000200

Table 6: Results from Experiment a: icons80-80_s_matrix1

1 layer

6400/4

Training tolerance 0.1

23 iterations training

Test results

Table of all fragment results

Image no.:	Fragment no.:	Target:	Computed:	Prob.: animals:	Prob.: buildings:	Prob.: cars:	Prob.: persons:
0	0	animals	persons	0.036400	0.080400	0.021200	0.971200
1	0	animals	animals	0.500000	0.432600	0.412000	0.259600
2	0	animals	cars	0.220200	0.345400	0.533200	0.306000
3	0	animals	buildings	0.356000	0.369200	0.286200	0.059200
4	0	animals	animals	0.404200	0.144400	0.380000	0.384400
5	0	animals	animals	0.416200	0.227000	0.077000	0.395000
6	0	animals	animals	0.620400	0.080800	0.514200	0.375000
7	0	animals	animals	0.424600	0.042000	0.057600	0.299200
8	0	buildings	animals	0.690600	0.070600	0.405600	0.066600
9	0	buildings	persons	0.163600	0.016200	0.026400	0.365600
10	0	buildings	buildings	0.120200	0.207400	0.200200	0.010400
11	0	buildings	persons	0.022000	0.197600	0.254200	0.707200
12	0	buildings	persons	0.350800	0.177200	0.283400	0.437400
13	0	buildings	animals	0.346600	0.070800	0.195200	0.190000
14	0	buildings	buildings	0.409000	0.423000	0.367000	0.126400
15	0	buildings	animals	0.531400	0.090000	0.359400	0.160200
16	0	cars	persons	0.310000	0.130800	0.122400	0.409200
17	0	cars	persons	0.215400	0.224400	0.146400	0.263200
18	0	cars	cars	0.006200	0.122600	0.135400	0.041000
19	0	cars	persons	0.040800	0.064200	0.182600	0.218600
20	0	cars	cars	0.095200	0.049600	0.235600	0.004200
21	0	cars	animals	0.344400	0.140400	0.163200	0.010000
22	0	cars	cars	0.182400	0.220400	0.236600	0.028200
23	0	cars	persons	0.010000	0.038200	0.206000	0.230200

24	0	persons	animals	0.150200	0.111400	0.145200	0.003600
25	0	persons	persons	0.202400	0.128200	0.312600	0.559600
26	0	persons	persons	0.065400	0.100400	0.270000	0.675400
27	0	persons	buildings	0.219200	0.537200	0.380600	0.336400
28	0	persons	persons	0.043400	0.017400	0.075200	0.657400
29	0	persons	persons	0.075000	0.091400	0.175600	0.283600
30	0	persons	persons	0.015200	0.076400	0.491600	0.700400
31	0	persons	animals	0.232000	0.180000	0.180000	0.192400

Table 7: Results from Experiment b: icons80-80_s_matrix2

2 layer network

6400/400/4

Training tolerance 0.1-0.0

40 iterations training

Table of all fragment results

Image no.:	Fragment no.:	Target:	Computed:	Prob.: animals:	Prob.: buildings:	Prob.: cars:	Prob.: persons:
0	0	animals	buildings	0.096400	0.245000	0.097200	0.111000
1	0	animals	animals	0.214400	0.186600	0.042200	0.180000
2	0	animals	animals	0.406400	0.118000	0.033400	0.052400
3	0	animals	animals	0.336400	0.118200	0.080800	0.071600
4	0	animals	animals	0.317200	0.086200	0.095600	0.118400
5	0	animals	animals	0.359400	0.009400	0.030200	0.116400
6	0	animals	animals	0.761000	0.140800	0.196400	0.080800
7	0	animals	animals	0.471400	0.364000	0.367600	0.009400
8	0	buildings	cars	0.065600	0.082600	0.285200	0.207400
9	0	buildings	animals	0.677600	0.604000	0.285600	0.034600
10	0	buildings	cars	0.077600	0.175400	0.525600	0.115200
11	0	buildings	buildings	0.082000	0.444200	0.306000	0.081200
12	0	buildings	buildings	0.201600	0.551400	0.243400	0.033000

13	0	buildings	buildings	0.052000	0.271200	0.122200	0.047200
14	0	buildings	cars	0.299400	0.099600	0.379200	0.018600
15	0	buildings	buildings	0.080200	0.194600	0.178200	0.076000
16	0	cars	cars	0.049000	0.028000	0.480200	0.398000
17	0	cars	cars	0.410000	0.337600	0.842400	0.028400
18	0	cars	cars	0.088200	0.072200	0.782600	0.272000
19	0	cars	cars	0.175400	0.070000	0.457000	0.044000
20	0	cars	animals	0.584200	0.270400	0.140200	0.089000
21	0	cars	cars	0.280000	0.115600	0.520600	0.051000
22	0	cars	cars	0.274600	0.123000	0.429400	0.136600
23	0	cars	animals	0.731400	0.357600	0.393000	0.009000
24	0	persons	persons	0.121400	0.113000	0.102200	0.526000
25	0	persons	persons	0.020000	0.053200	0.129600	0.635400
26	0	persons	animals	0.166400	0.131400	0.055200	0.067000
27	0	persons	persons	0.446200	0.004400	0.059400	0.544400
28	0	persons	persons	0.134000	0.139200	0.010000	0.284000
29	0	persons	persons	0.056000	0.071600	0.076400	0.891600
30	0	persons	buildings	0.273400	0.409600	0.114000	0.056000
31	0	persons	cars	0.048400	0.156200	0.640400	0.350200

8. References

Bishop, C.M. (1995): *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

California Scientific Software (1998): *BrainMaker Professional User's Guide and Reference Manual*. California Scientific Software. Nevada City, CA.

Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, Eds. (1988): *Parallel Distributed Processing*. Volume 1 and 2. MIT Press. Cambridge, MA.

Spirkovska, L. and Reid, M. B. (1992): *Higher Order Neural Networks in Position, Scale, and Rotation Invariant Object Recognition*. In Soucek, B. and The Iris Group: *Fast Learning and Invariant Object Recognition*. John Wiley & Sons. NY 1992.