Computer Vision

Classification with Neural Networks

27 August 2008

j.van.de.loosdrecht@tech.nhl.nl, jaap@vdlmv.nl

Classification with Neural Networks

Overview:

- **Introduction classification**
- **Introduction neural networks**
- **Image classification**
- **Exercise**
- **Feature vector classification (*)**
- **Exercise (*)**

Related presentations:

- **Classification with neural networks part II**
  with examples how to use neural networks in scripts
- **Genetic Algorithms**
  how to tune the parameters of a neural networks

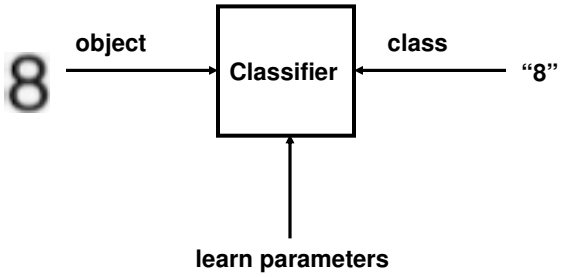
Introduction classification

Classification:
the assignment of an object characterized by a set of features to one of a number of predefined classes.

Example: Optical Character Recognition

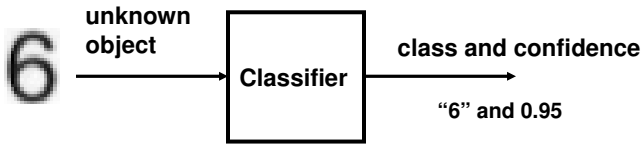
Introduction classification

- **Learning mode**
  - **Learn set: set of objects with known classes**
  - **Each time the total learn set is trained is called an epoch, for successful learn normally many epochs are necessary**



Introduction classification

- **Production mode**



28-Aug-08 Classification with Neural Networks 5

Introduction classification

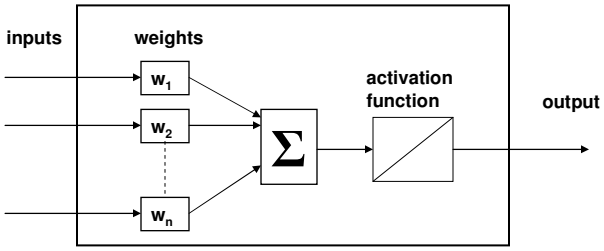
- **Image classification**
  - **'raw' image data**
- **Feature vector classification**
  - **features are extracted from image and used for the classification**
  - **data reduction**

28-Aug-08 Classification with Neural Networks 6

Introduction neural networks

- **Based on a model of our brain**
- **Neuron**
- **Neural networks**
- **Back Propagation Network**
- **Example: learning sinus function**

Neuron

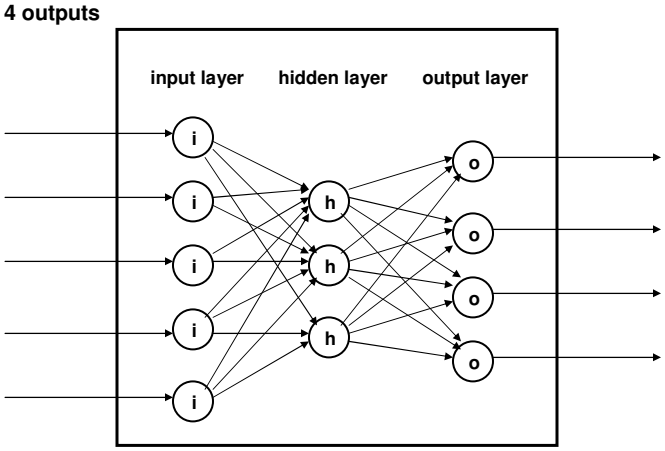


Neural networks

- **Neural network consists of multiple neurons organised in different layers**
- **There are several different architectures for the organization and for the learning algorithms**
  - **Supervised learning**
  - **Unsupervised learning**
- **Most common used architecture is Back Propagation Network (BPN)**

Back Propagation Network

Example with:
- **5 inputs**
- **one hidden layer with 3 neurons**
- **4 outputs**



Example: learning sinus function

- Create BPN with
  - Input layer of 1 neuron
  - One hidden layer of 100 neurons
  - Output layer of 1 neuron
- Train the BPN with 100 random examples: x, sin(x)
  - $in = random –1 1
  - $out = sin $in
  - Note the behaviour of the training error ($t_err)
- Test the BPN with 5 random values
  - $in = random –1 1
  - $res = sinus of $in learned by BPN
  - $exp = sin $in
  - $error = error in result

- use script bpn_sinus.jls

Example: learning sinus function

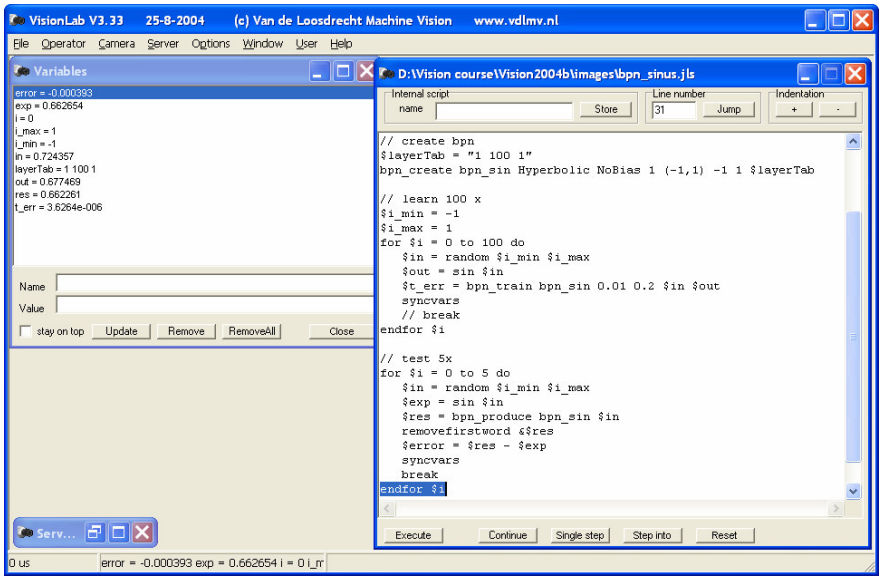


Image classification

- Introduction
- Class Image Sets (CIS)
  - Train set
  - Evaluation sets
- Training the BPN
- Using the BPN

Image classification

- Pixels of image are inputs,
  nr of pixels = size input layer
- Nr of classes = size output layer
- Scale, rotation and lighting variations
  - Train BPN with all combinations
  - Use computer vision operations to make images invariant by normalizing:
    - Scale, use Zoom
    - Rotation, use Rotate(Full)
    - Lighting, use Contrast Stretch

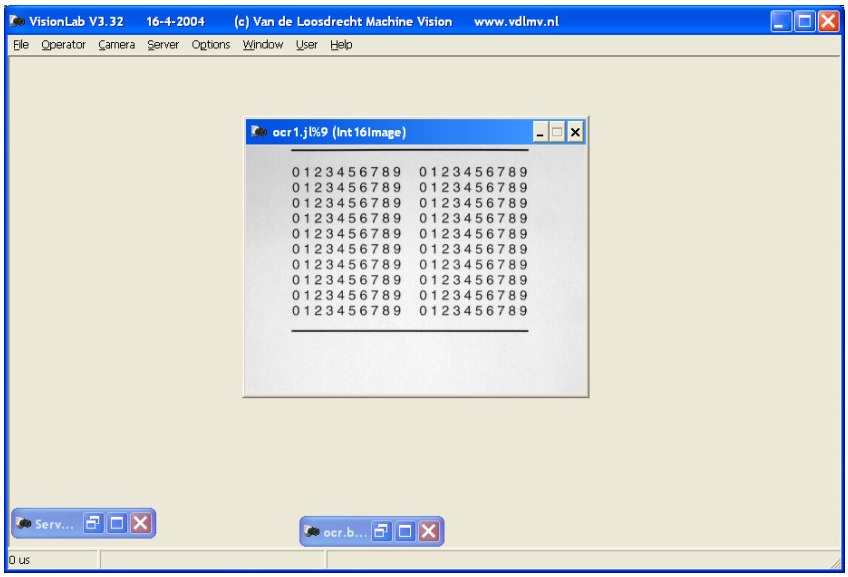
Class Image Set (CIS)

- **A CIS is a collection of images with their associated classes. All images in a CIS must have the same image type and size**
- **CIS:**
  - **Class table with the name and id of each class**
  - **For each class an image table, each image in an image table has its unique image index number**

Demonstration Class Image Set (CIS) (*)

- **Create new CIS and add some classes and images (no slides) use correct image type and image size**

Image ocr1.jl used to generate CIS



28-Aug-08 Classification with Neural Networks 17

Script gen_ocr1_cis.jls

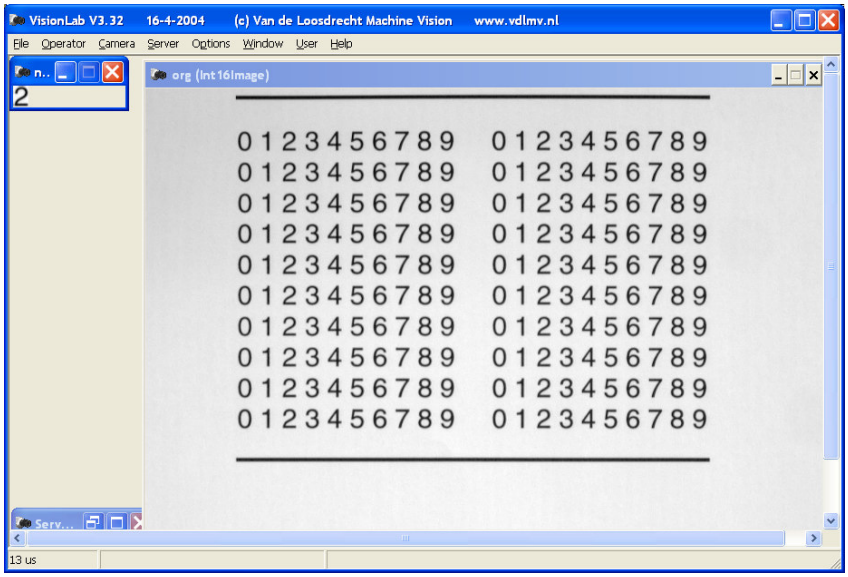
- Create cis
- Threshold image
- Remove big blobs
- Label image
- BlobAnalyse SortDown TopLeft UseX Height TopLeft Width
- for $group = 0 to 1 do
  - for $num = 0 to 9 do
    - for $i = 0 to 9 do
      - Roi $t $l $h $w
      - Contraststretch
      - Zoom NearestPixelInterpolation
      - Add normalized image to cis

28-Aug-08 Classification with Neural Networks 18

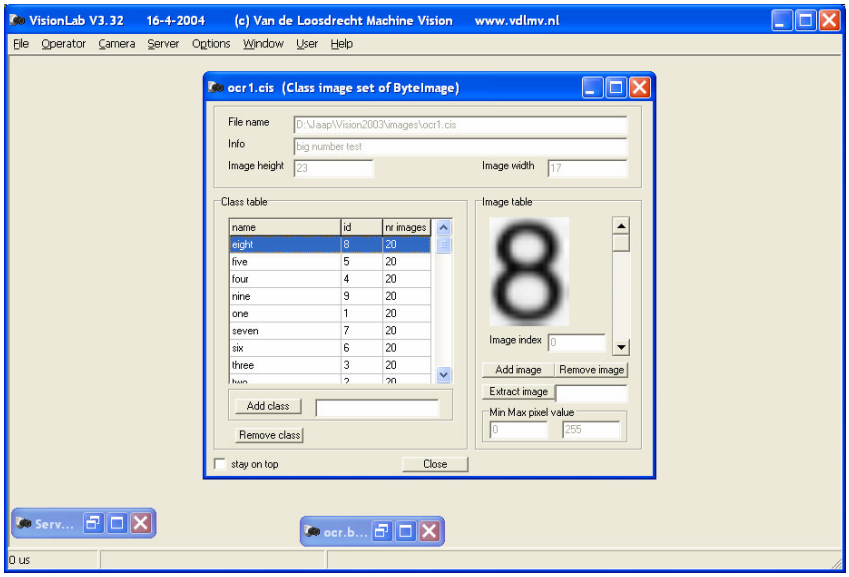
Demonstration Class Image Set (CIS)

- Use script gen_ocr1_cis.jls to generate ocr1.cis,
- Open and examine ocr1.cis

Generation of ocr.cis

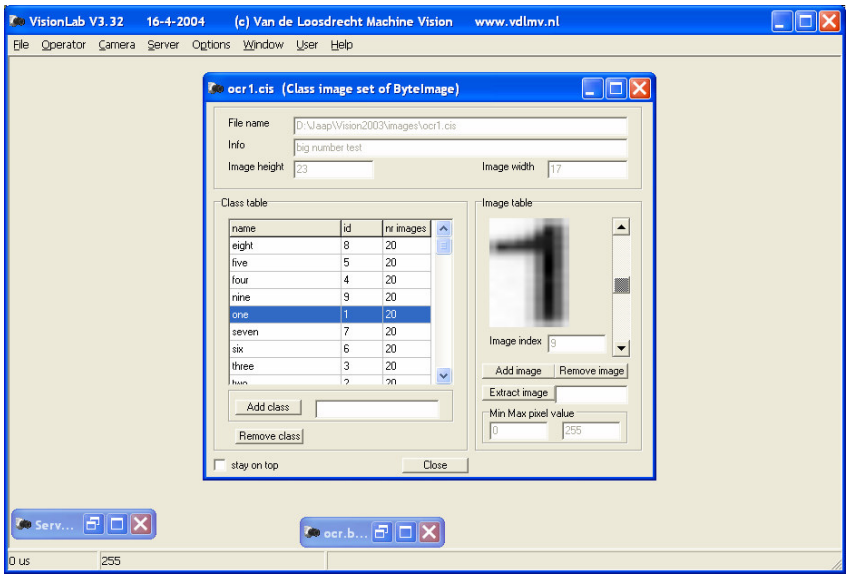


ocr1.cis



28-Aug-08 Classification with Neural Networks 21

ocr1.cis



28-Aug-08 Classification with Neural Networks 22

Different sizes of numbers

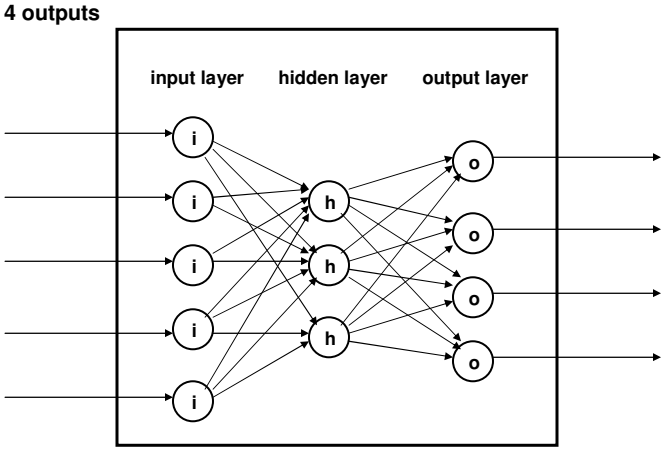
- **Note that number 1 is much smaller in width as number 8**
- **A BPN always works with a fixed number of inputs.**
- **So all numbers have to be rescaled in size to the same height and width of the learn image**

Training the BPN

- **Learning parameters**
- **Train set**
  balancing the train set
- **Evaluation set**
- **Testing the result**

Back Propagation Network

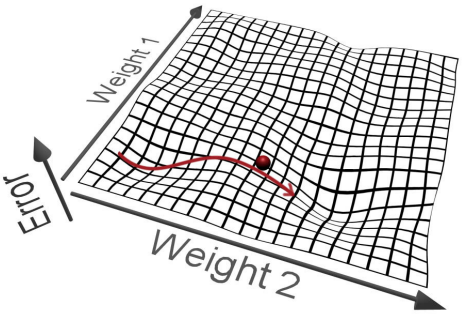
Example with:
- **5 inputs**
- **one hidden layer with 3 neurons**
- **4 outputs**



Learning parameters

- **Learn rate:**
  The learn rate is a small positive value and determines how fast the BPN will learn. If the learn rate is too low learning will be very slow. If the learn rate is too high the training error will oscillate and stay at an high value.
- **Momentum:**
  The momentum is zero or a small positive value. This parameter influences the speed of convergence of the learing process and specifies which fraction of the "learn effect" of the previous training sample should be added to the "learn effect" of the next training sample.

Learn rate and momentum



the direction of the change for the weights is calculated using the derivative of the error function

the 'step size' for the adaptation of the weights is proportional to the the learning rate

the momentum is the fraction of the previous weight change which is added to the new calculated weight change

28-Aug-08 Classification with Neural Networks 27

Learning parameters

- **Size first hidden layer:**
  If too small the training error will remain high, if too high training will be very slow
- **Size second hidden layer:**
  In theory one hidden layer can learn "all problems", but the use of a second hidden layer may speed up the learning process. Advice: start first with an empty 2nd hidden layer.

28-Aug-08 Classification with Neural Networks 28

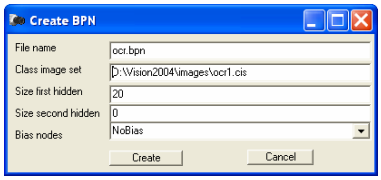
Training the BPN

- **Stopping the training process:**
  - **Max nr of epochs**
  - **Stop error**
- **Confidence:**
  indication how confident the classifier is about the result,
  [1 = confident .. 0 = unconfident].
  In normal conditions a confidence > 0.7 is fine.
- **Reset network:**
  all weights are random initialised with value [min ..max]

Demonstration Training BPN

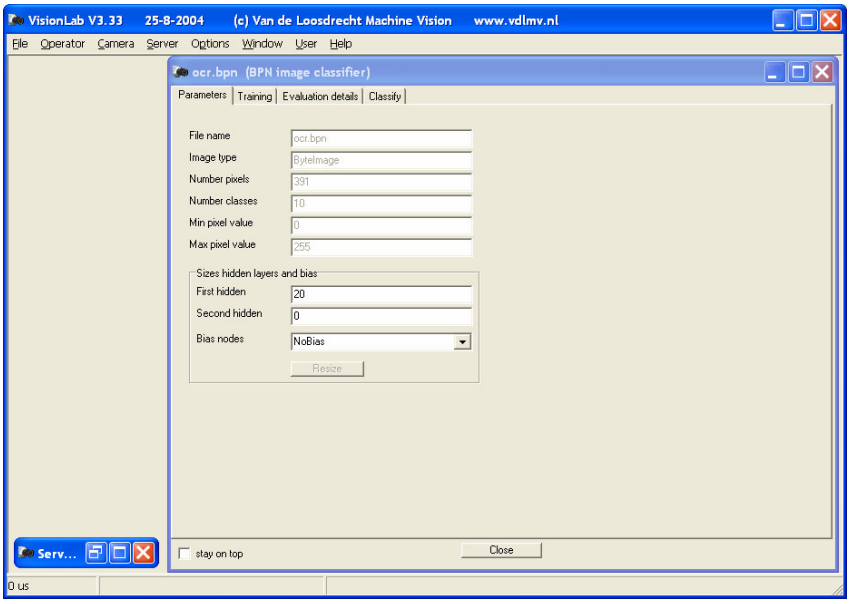
- **Create new BPN ocr.bpn (File menu),**
  use ocr1.cis and size first hidden layer = 20, second = 0, NoBias
- **Go to Training form**
- **Start training, note evaluation set and training set are the same**
  reset: -0.1 .. 0.1
- **Open ocr1.cis and select from a random class a random image and extract that image with name test**
- **Select image with name test as "second selected"**
- **Go to BPN and select Classify form and classify "second selected" image**

Demonstration Training BPN



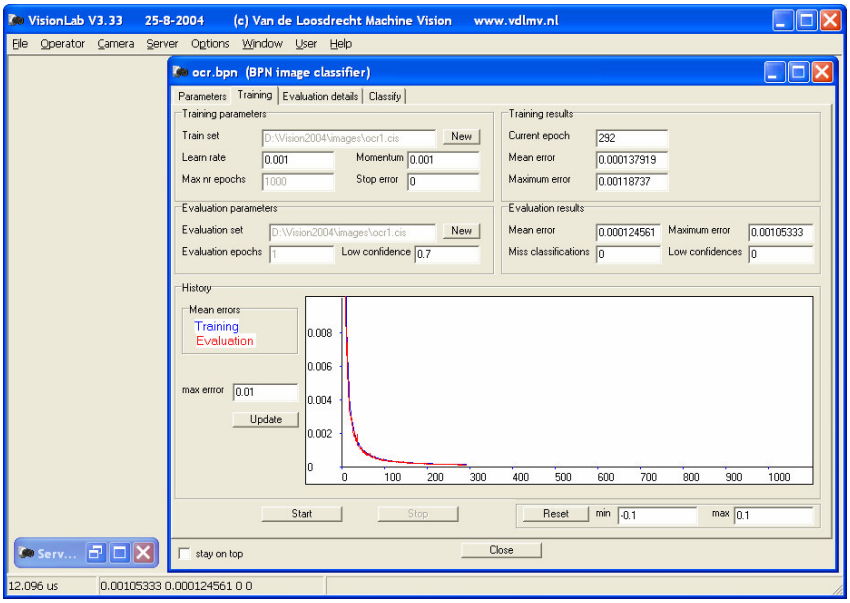
28-Aug-08 Classification with Neural Networks 31

Demonstration Training BPN

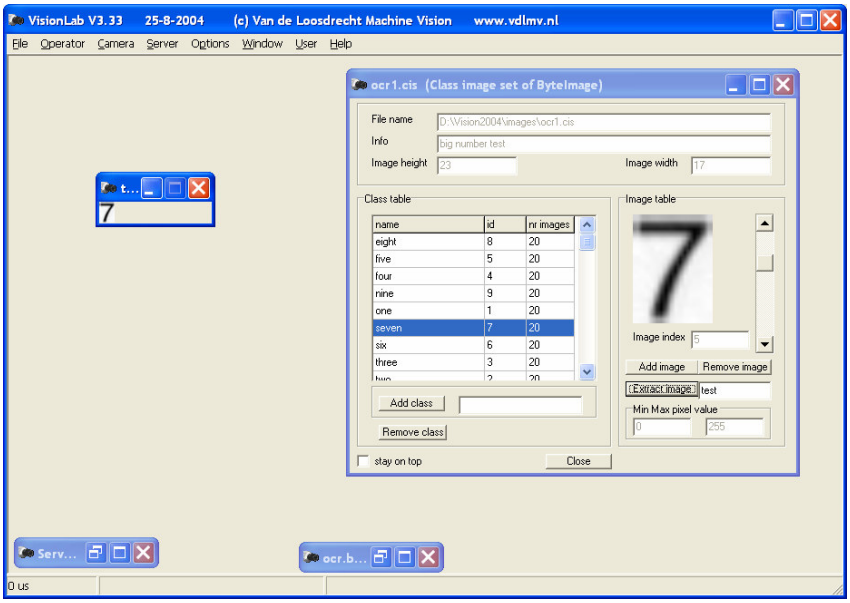


28-Aug-08 Classification with Neural Networks 32

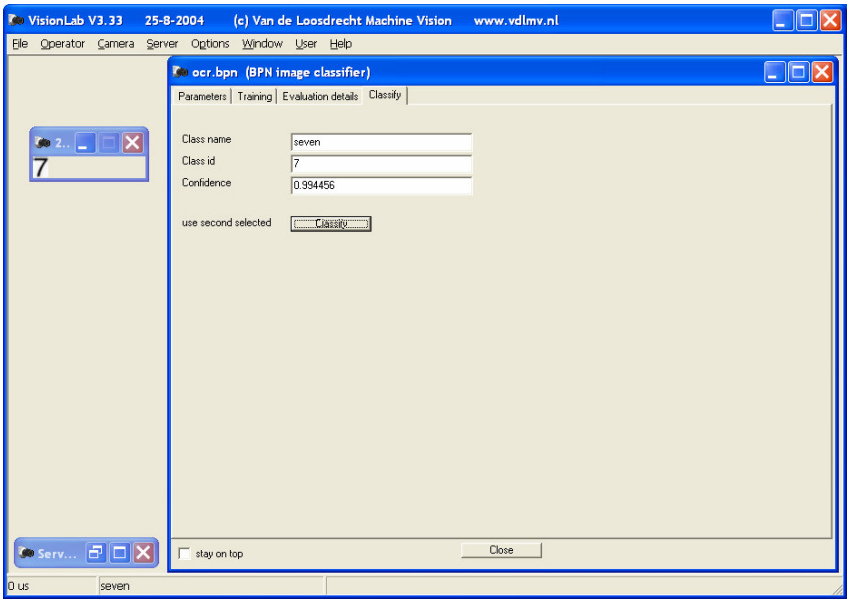
Demonstration Training BPN



Select random image from random class



Classify image



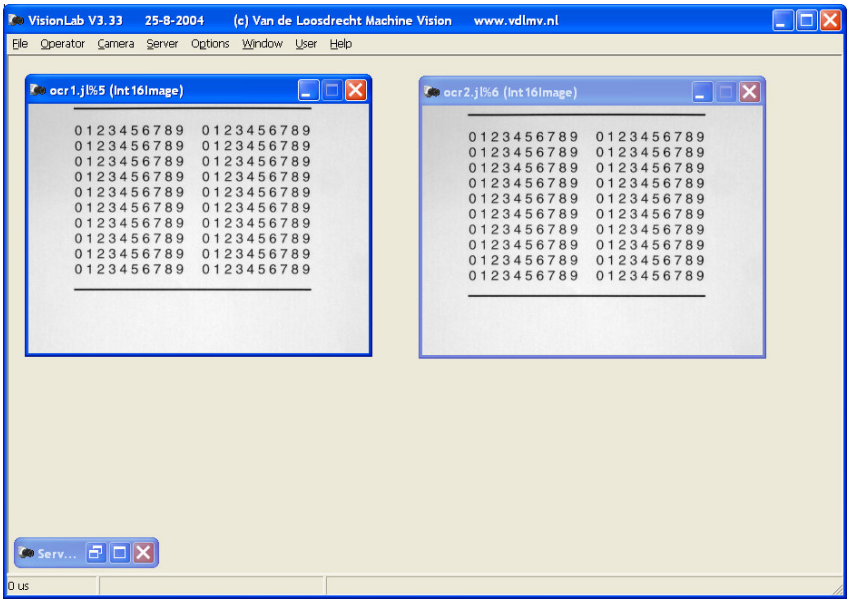
Training with evaluation set

In order to avoid inbreed, it's better to use a separate training and evaluation set

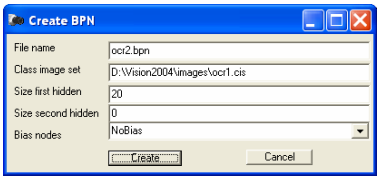
Demonstration Training BPN with evaluation set

- Create new CIS and some classes (no slides)
- Open image ocr1.jl and ocr2.jl, same object slightly different position and lighting condition
- Use script gen_ocr2_cis.jls to generate ocr2.cis
- Create new BPN ocr2.bpn (File menu),
  use ocr1.cis and size first hidden layer = 20, second = 0 , NoBias
- Go to Training form
- Start training, and select as evaluation set ocr2.cis
- Open ocr2.cis and select from a random class a random image and extract that image with name test
- Select image with name test as "second selected"
- Go to BPN and select Classify form and classify "second selected" image

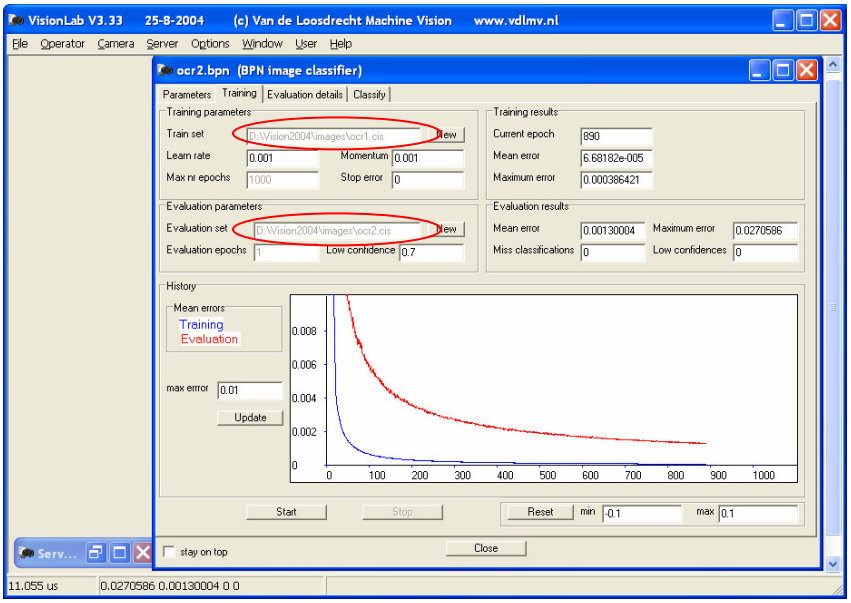
Compare image ocr1.jl and ocr2.jl



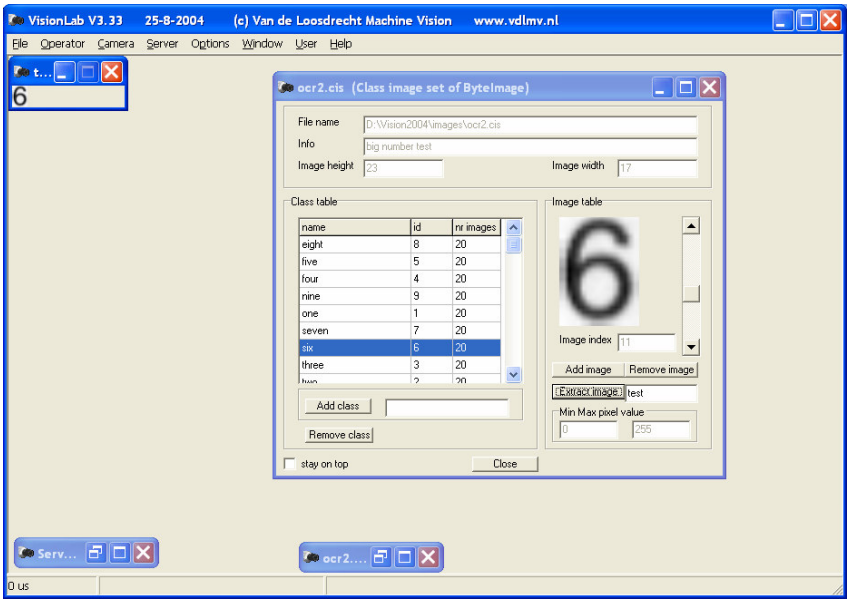
Demonstration Training BPN with evaluation set



Demonstration Training BPN with evaluation set

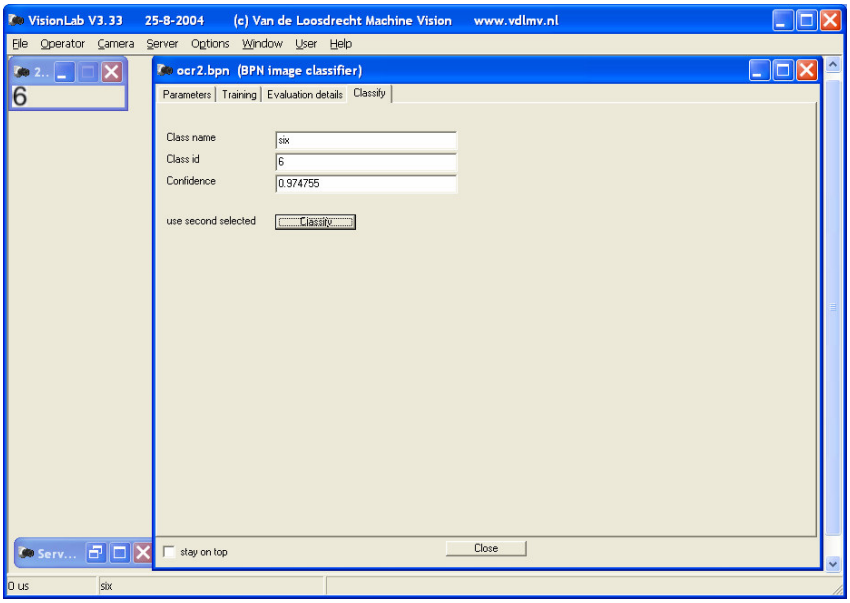


Select random image from random class from evaluation CIS



28-Aug-08 Classification with Neural Networks 41

Classify image

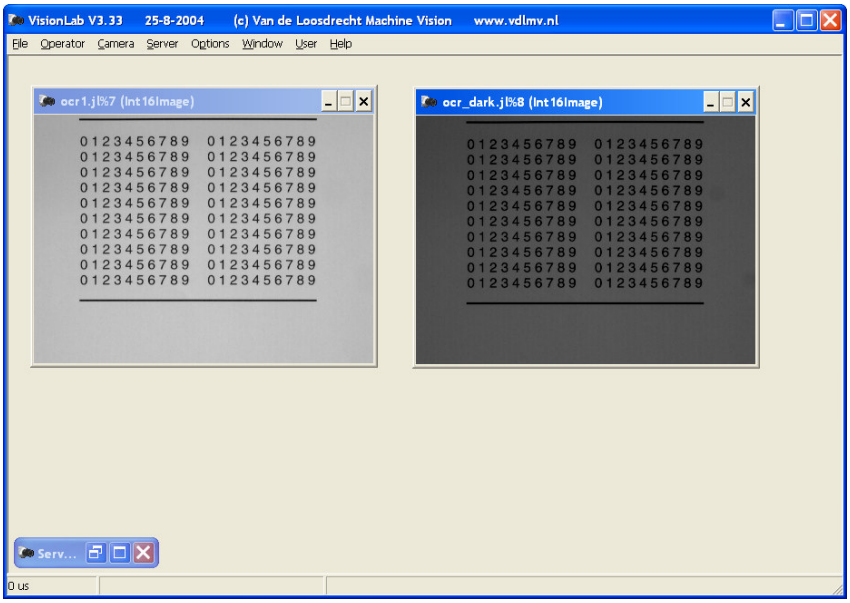


28-Aug-08 Classification with Neural Networks 42

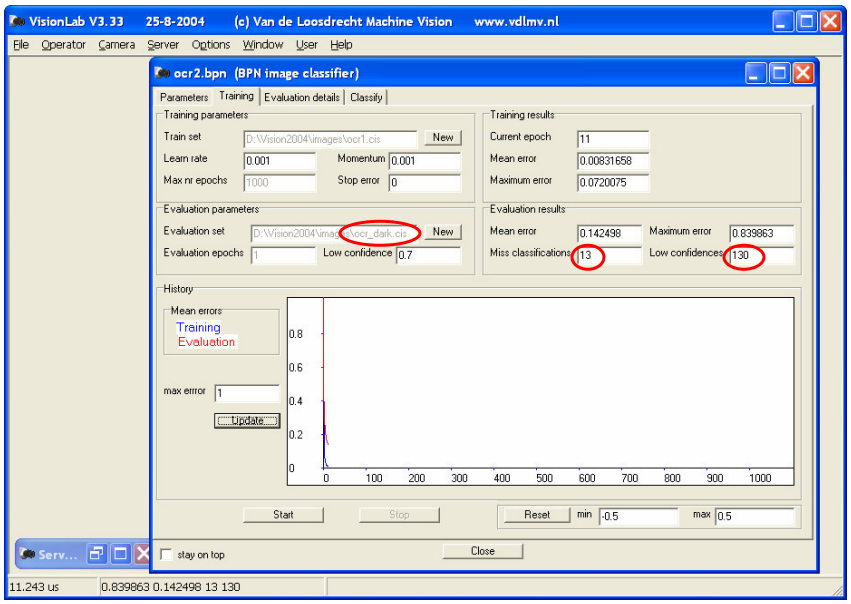
Demo analyzing learning problems (*)

- Compare images ocr1.jl and ocr_dark.jl with LUT clip
- Change evaluation set to ocr_dark.cis,
  this is generated from image ocr_dark.jl under very low lighting conditions
- Reset network with –0.5 and 0.5, ([-0.1 .. 0.1] will give better results)
- Start training and stop after 12 epochs,
  due to the low number of epochs there are now 11 miss classifications and 99 low confidences
- This can be analyzed on the Evaluation details form,
  use low details
- By selecting a row and specifying a name at the Extract Image button the offending image can be extracted for examination
- By changing to high details the individual outputs in the output layer can be examined.
  Output values are in the range [-1..1]. A class is 'strong' if the output value is close to 1 and weak if close to –1.
  On the first row an 'Eight' is classified as a 'Three'

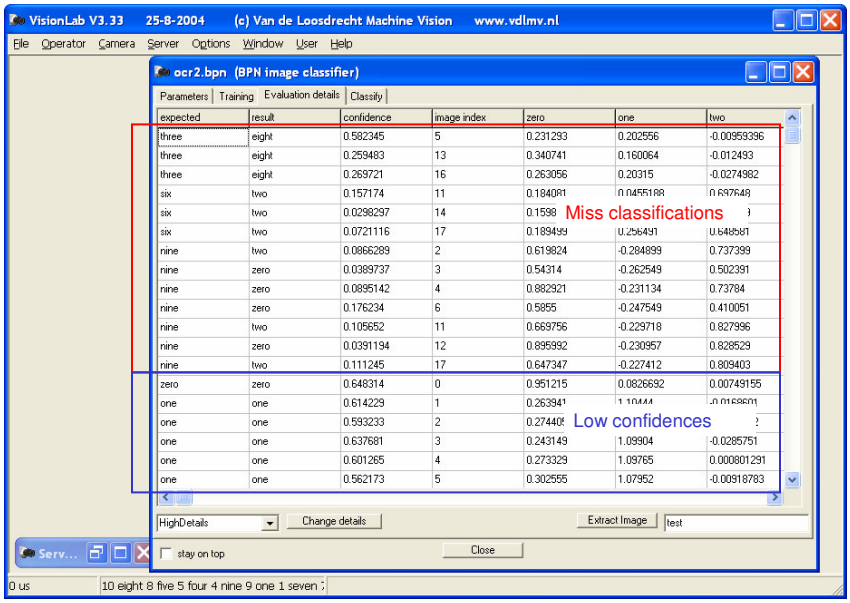
Comparing ocr1.jl and ocr_dark.jl (*)



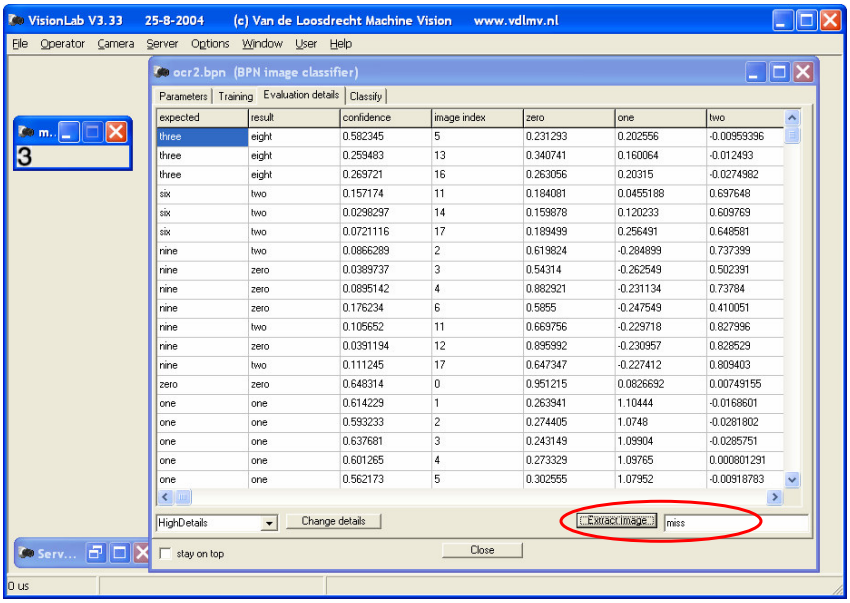
Evaluate using ocr_dark.cis (*)



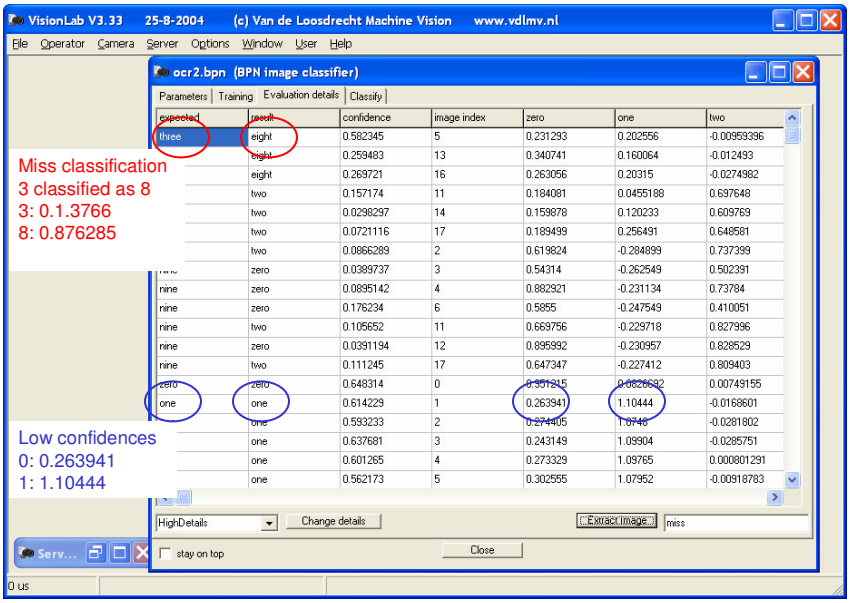
Evaluation details form (*)



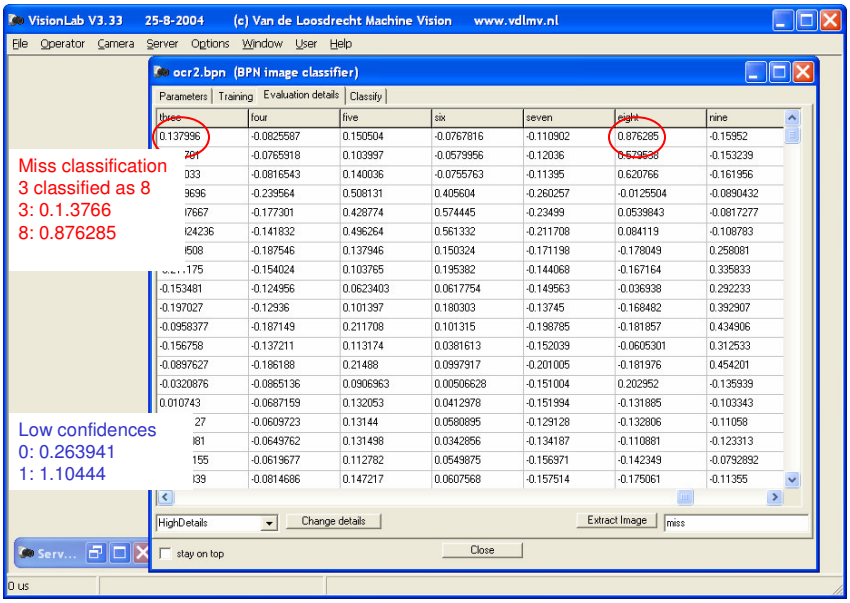
Extract offending image (*)



Examine high details (*)



Examine high details, scrolled to the right (*)



Exercise OCR dark (*)

- **Experiment with learning parameters to get better results**

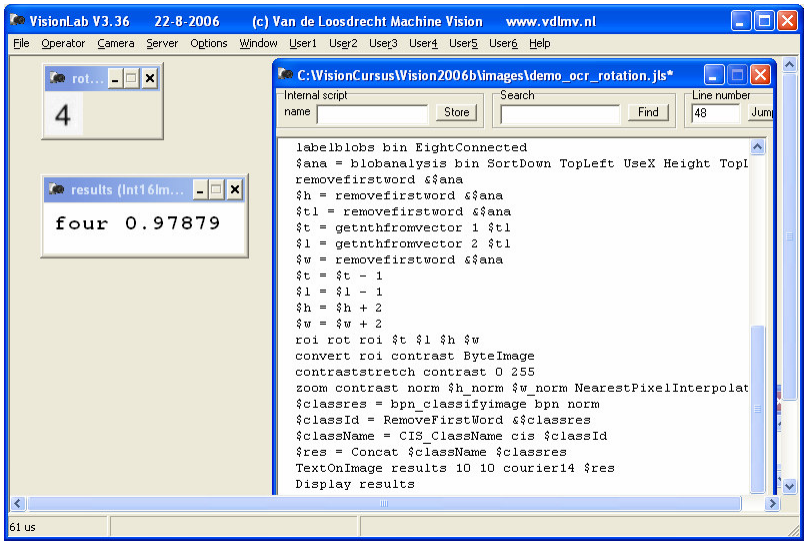
Notes for applications

- **It's better to control the lighting conditions so this kind of dark image can not occur**
- **It's also possible to use more sophisticated contrast normalization algorithms to deal with changing light conditions**
- **The training set can be extended with images under different lighting conditions to make the application more robust**

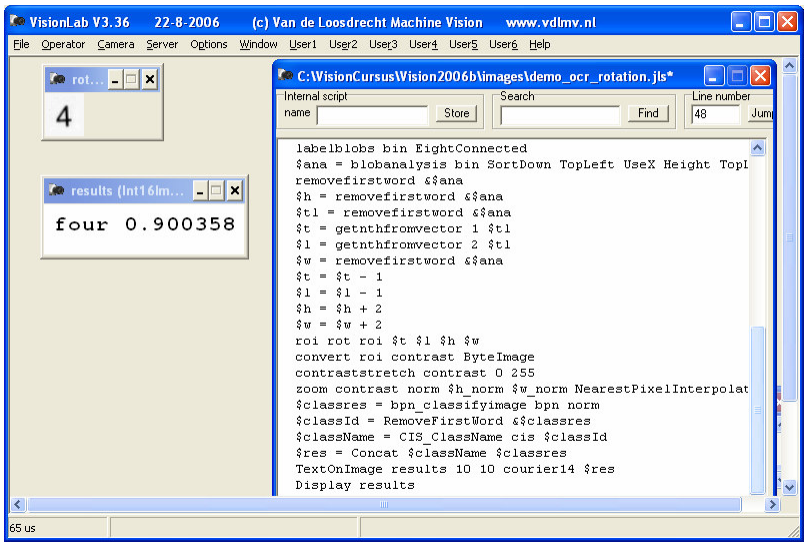
Demo robustness of BPN

- **Execute script demo_ocr_rotation.jls**
- **Execute script demo_ocr_size.jls**
- **Execute script demo_ocr_contrast.jls**

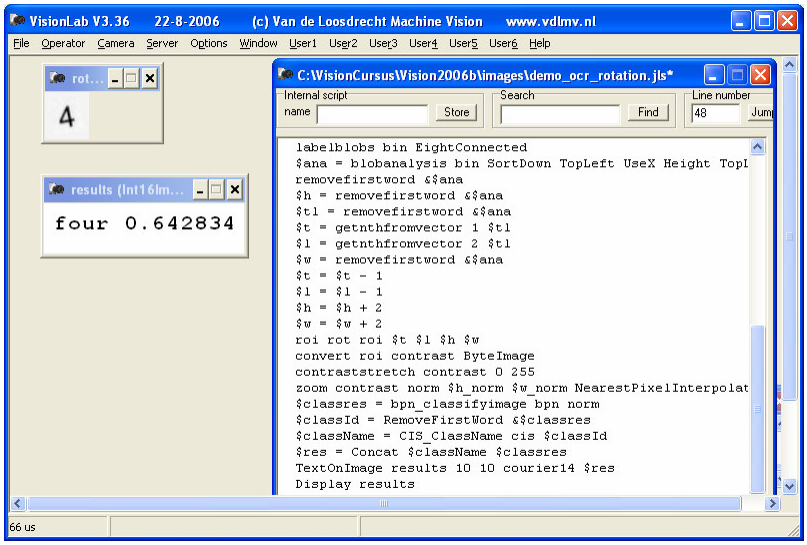
Demo with rotated input images



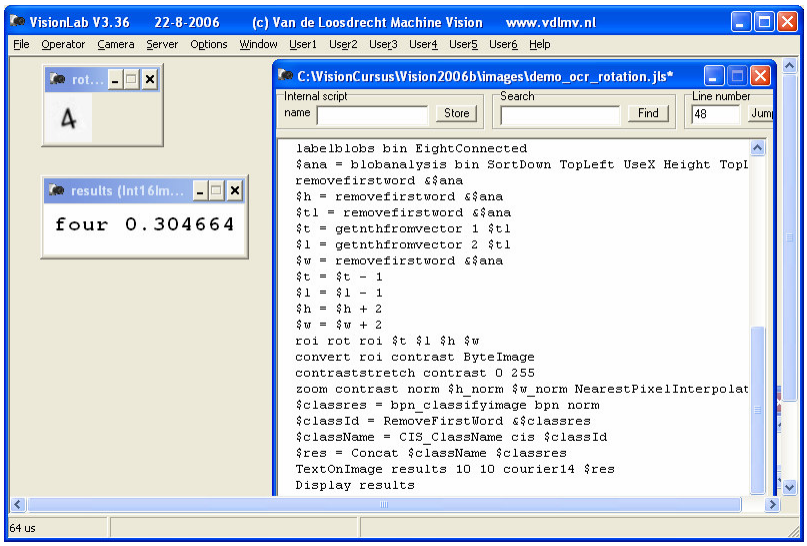
Demo with rotated input images



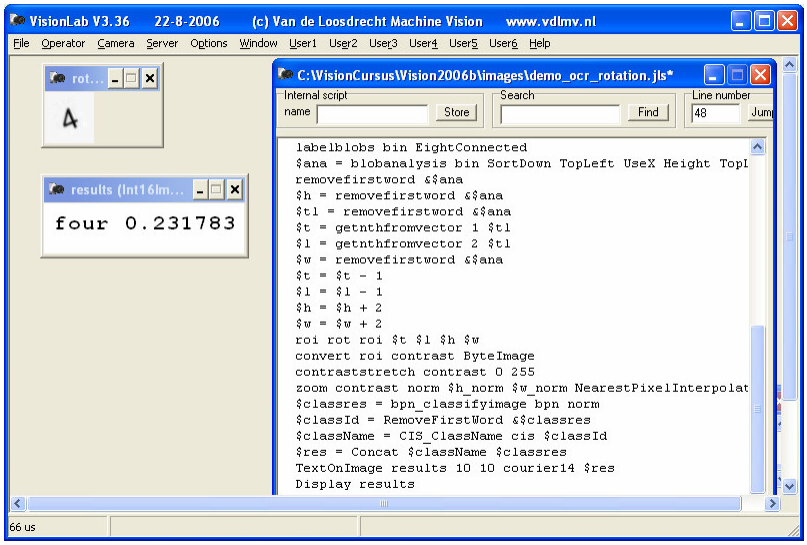
Demo with rotated input images



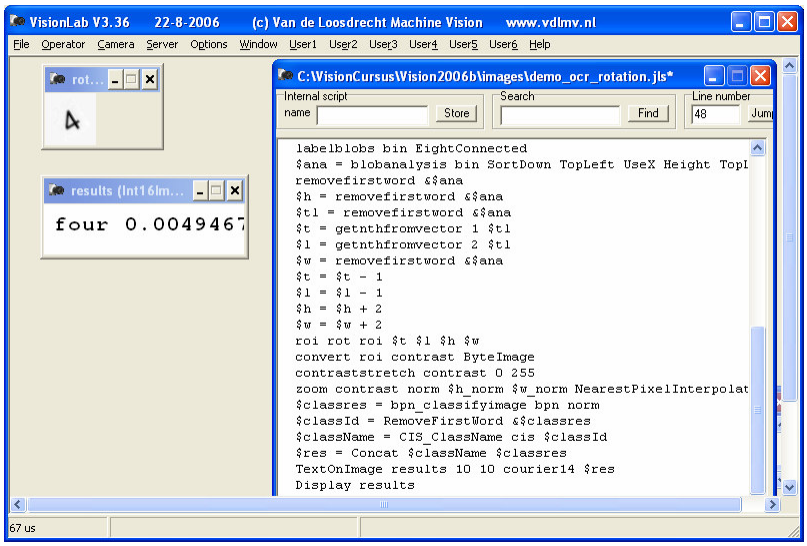
Demo with rotated input images



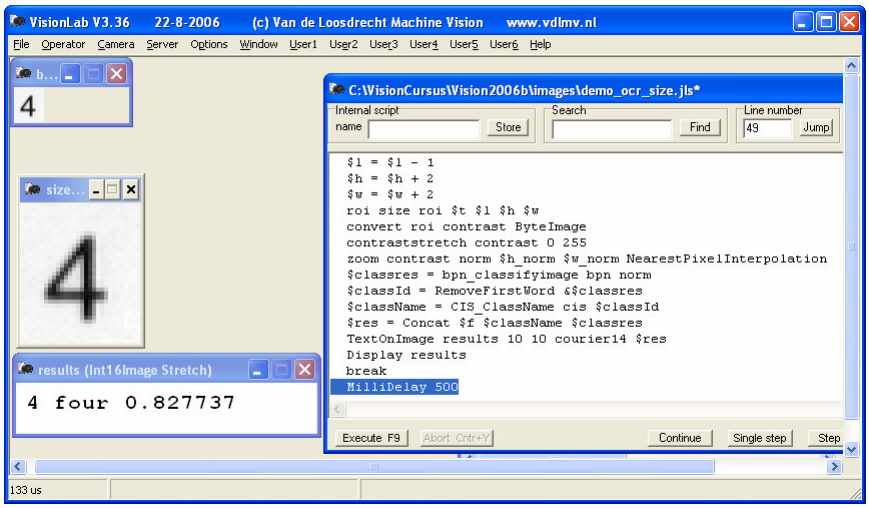
Demo with rotated input images



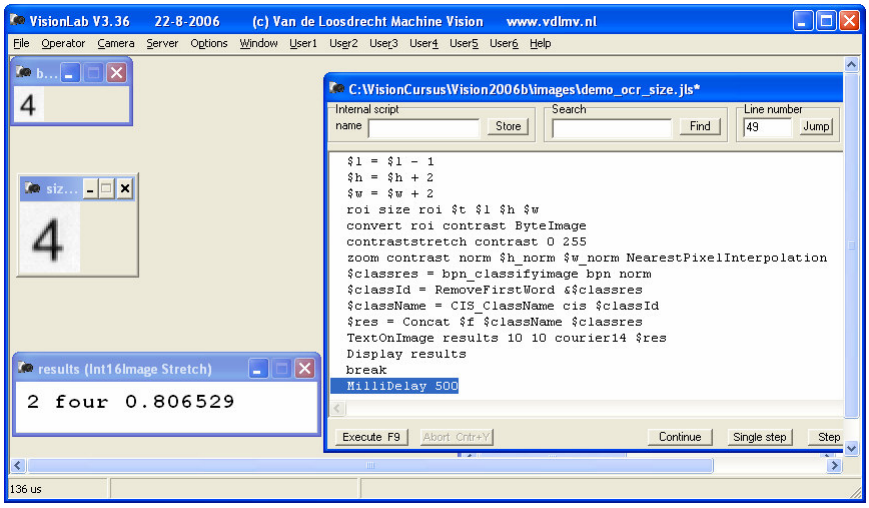
Demo with rotated input images



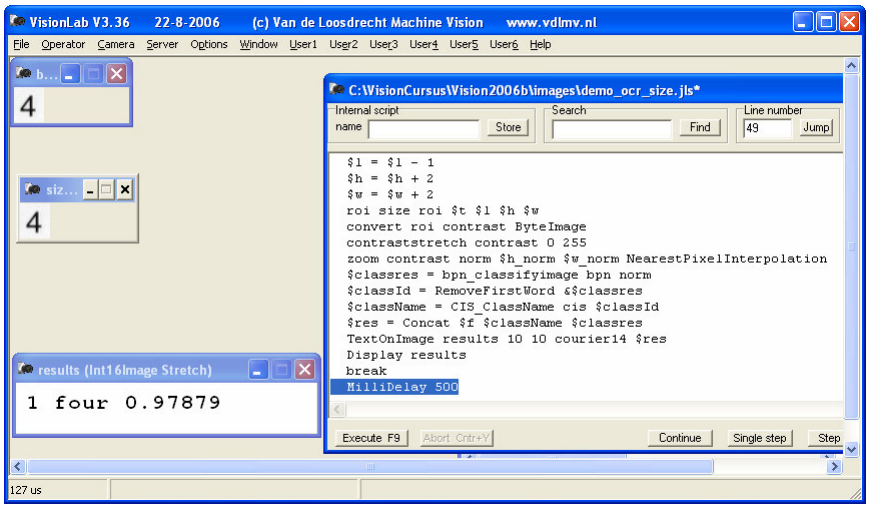
Demo with zoomed input images



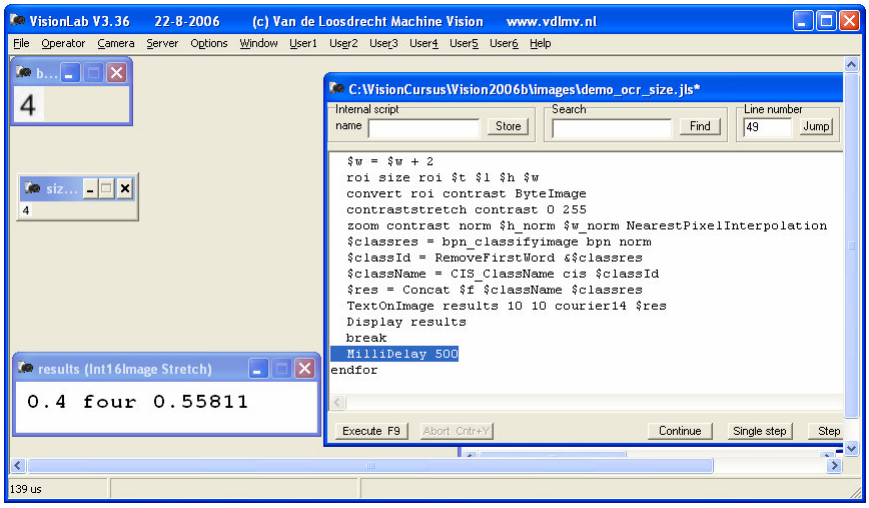
Demo with zoomed input images



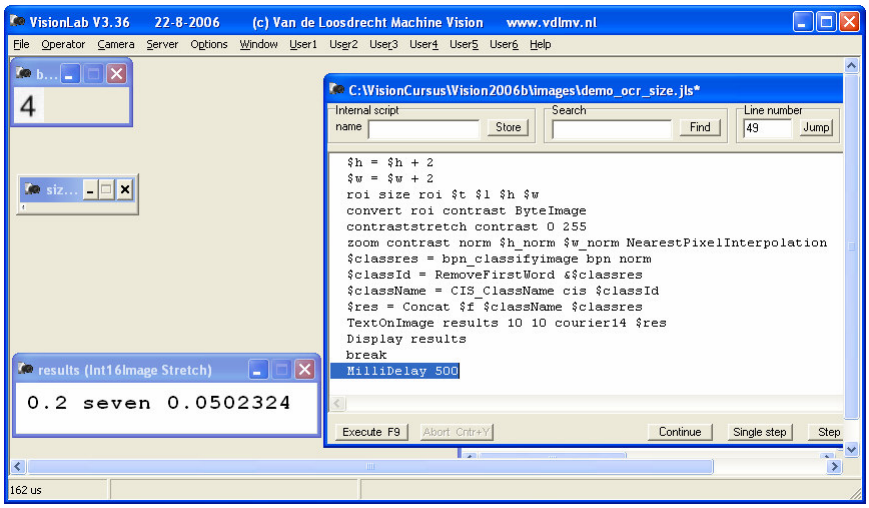
Demo with zoomed input images



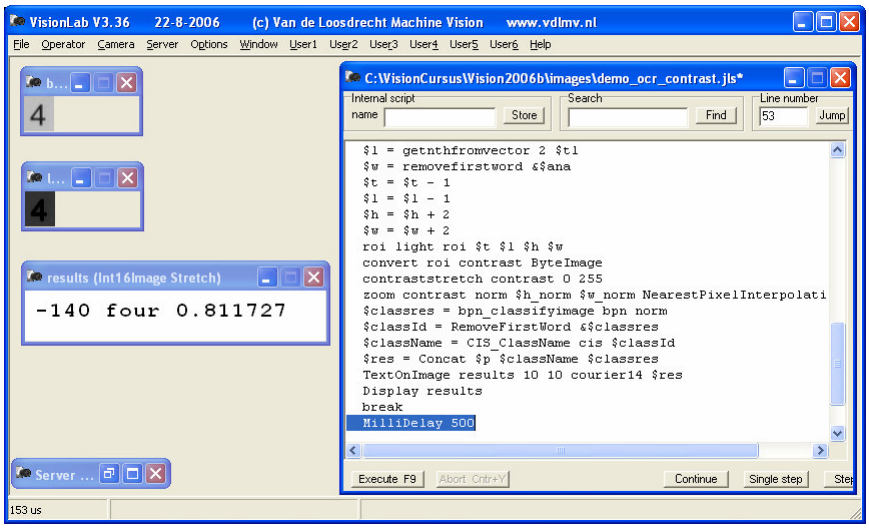
Demo with zoomed input images



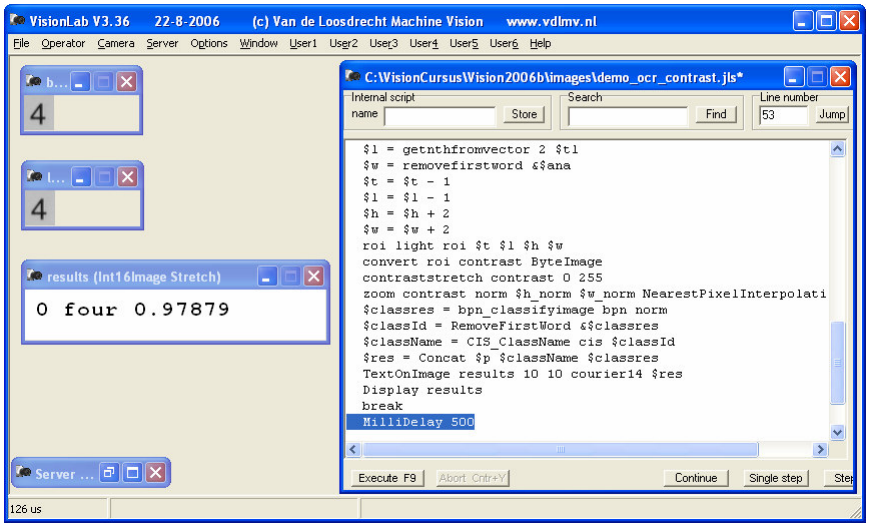
Demo with zoomed input images



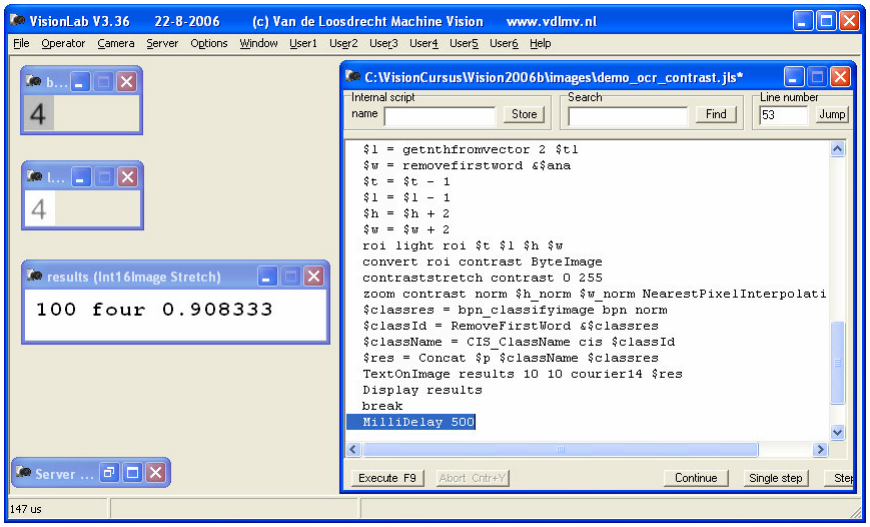
Demo with changed light conditions



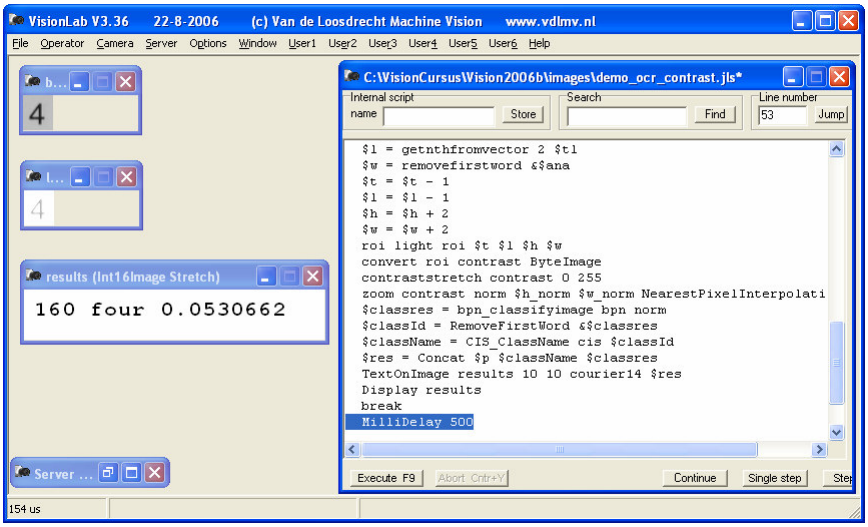
Demo with changed light conditions



Demo with changed light conditions



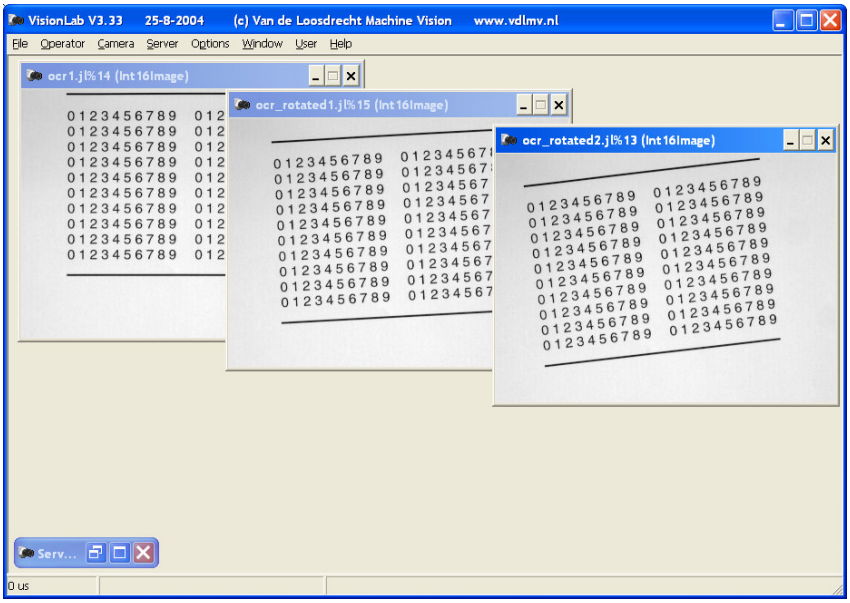
Demo with changed light conditions



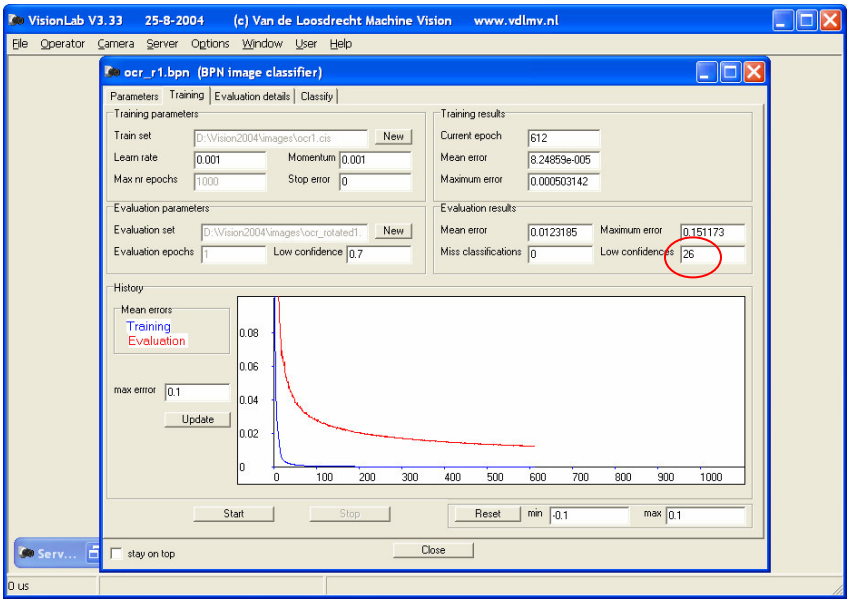
Demo with wrong evaluation set (*)

- **The BPN was trained with numbers without rotation and is expected to perform worse if the digits are rotated**
- **Compare test images ocr1.jl, ocr_rotated1.jl and ocr_rotated2.jl**
- **For the rotated image ocr1 a CIS is generated with gen_ocr_rotated1_cis.jls**
- **For the rotated image ocr2 a CIS is generated with gen_ocr_rotated2_cis.jls**
- **Demonstrate training with ocr1.cis and evaluation sets the rotated CIS's, compare the results.**
  - **Analyse bad results for rotated2, have a look at the generated CIS !**

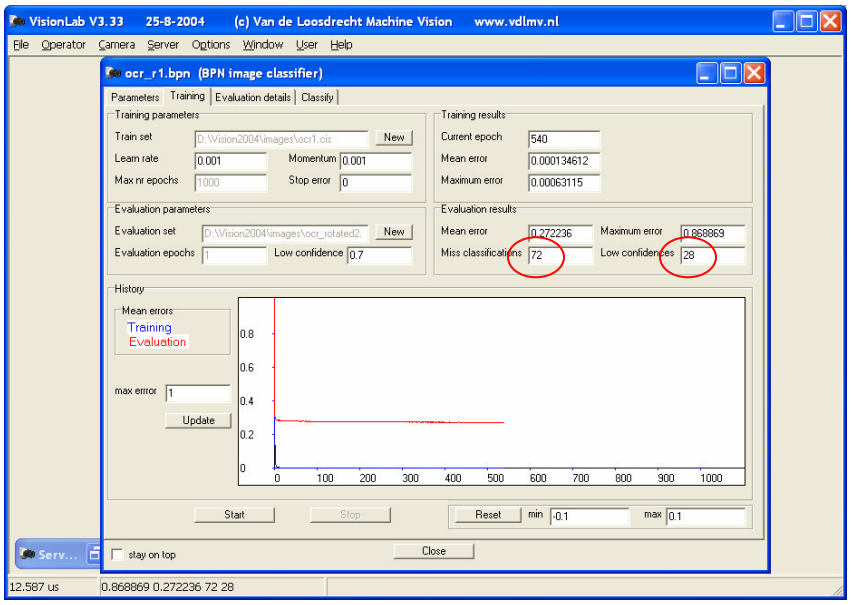
Comparing the rotated images (*)



Evaluation set ocr_rotated1.cis (*)

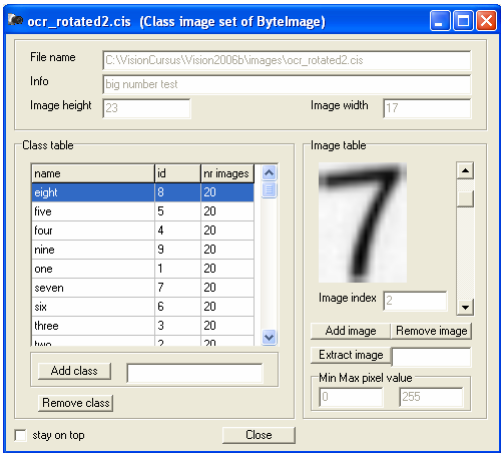


Evaluation set ocr_rotated2.cis (*)



28-Aug-08 Classification with Neural Networks 71

Wrong test data, seven in class “eight” ???? (*)



28-Aug-08 Classification with Neural Networks 72

Testing with rotated input images (*)

- **Conclusion:**
  - **The BPN can cope with small rotations, but the confidence level will drop for classifying**
- **For applications:**
  - **The training set can be extended with rotated images, but training will then be more difficult**
  - **All image are normalized for rotation before training or classification**

Exercises rotation robustness OCR (*)

- **Normalize the training and evaluation sets for rotation and evaluate the BPN.**
  Hints:
  - **adapt script gen_ocr_rotated2_cis.jls**
  - **use the horizontal bars**
- **Make new versions of image ocr2.jl and generate new CIS's for evaluation:**
  - **Scale, use zoom**
  - **Lighting, add pixel value**
- **Use new CIS's to evaluate the BPN**

- **Answer first exercise: gen_ocr_rotated_norm_cis.jls,**

Robustness OCR (*)

- **Image classification with BPN can be reasonable robust for**
  - **Scale**
  - **Rotation**
  - **Lighting**

  This is because the images are normalized for these aspects

Feature vector classification (*)

- **Introduction**
- **Class Feature Sets (CFS)**
- **Training the BPN**
- **Using the BPN**
- **Exercise**

OCR feature vector classification (*)

- Features are extracted from image and used for the classification
- Learning from features instead of pixels
- Data reduction

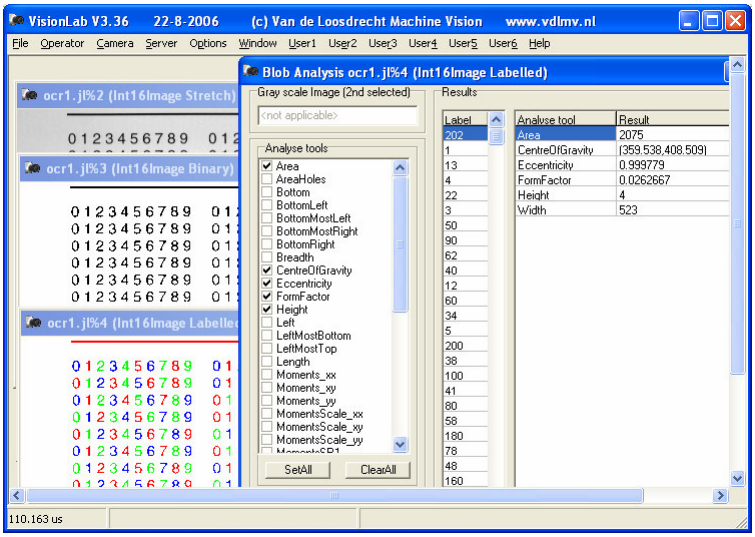
- OCR example:
  - Image: 23 x 17 = 391 pixels
  - Use features like:
    Area, AreaHoles , Breadth, CentreOfGravity,
    MomentsScale_xy, NrOfHoles, SumRows, SumCols, etc

Demo feature classifying (*)

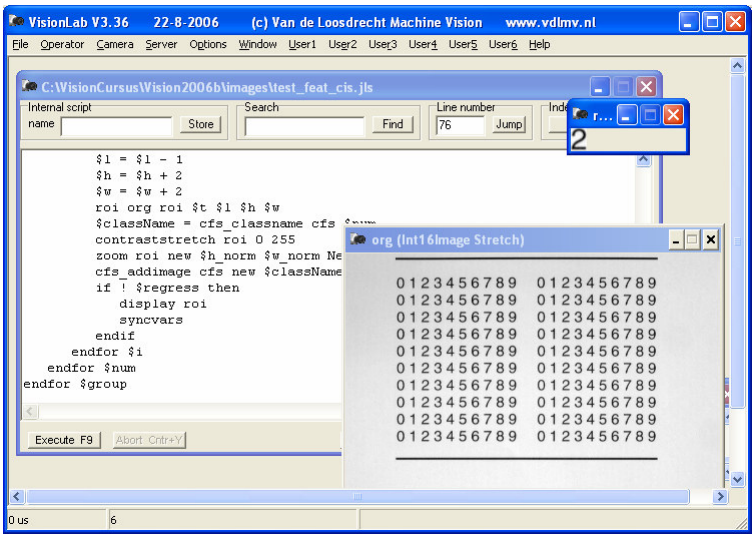
- { Demonstrate how to perform measurements
  - Open ocr1.jl (grayscale image)
  - ThresholdIsodata (binair image)
  - LabelBlobs (labelled image)
  - BlobAnalyse (measurement)
- use test_feat_cfs.jls to generate test.cfs }

- Open test.cfs, show measurements
- CreateFeatureBPN with test.cfs, first hidden = 40 and bias
- Demo learning, no feature: cg.x, cg.y and SumXXX
  (use keyboard space to de-select and and activate selection with select button)
  - Learnrate = 0.002, momentum = 0.01
  - Problem with learning, about 20 miss classifications
  - Look at evaluation details:
    - Problems in classifying 6
    - Neuron output 6 and 9 = ~0.5, others ~0
  - Use cg.x to distinguish between 6 and 9
    - 6: 7.3 < cg.x < 8.1, 10.7 < cg.y < 11.8
    - 9: 8.6 < cg.x < 9.9, 10 < cg.y < 11.2

Perform measurements (*)



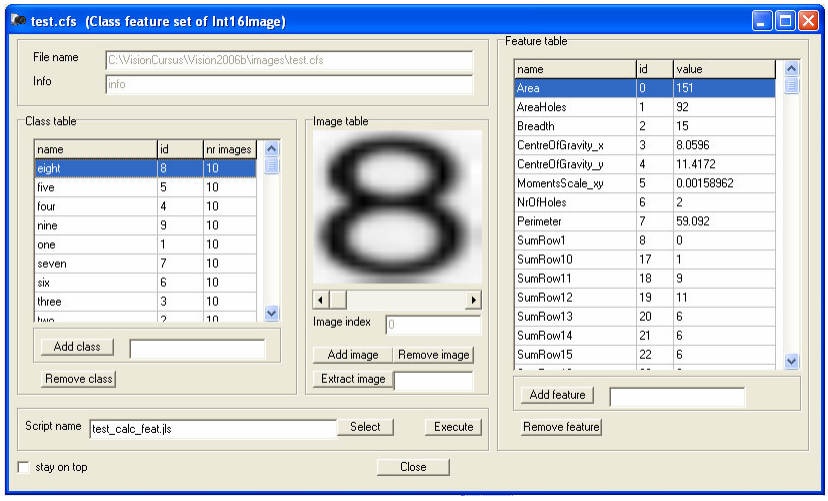
Generate Class Feature Set (*)



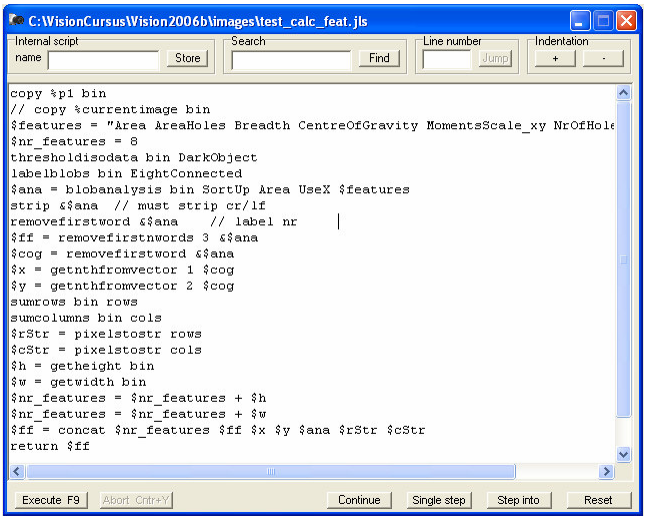
Class Feature Set (CFS) (*)

- **A CFS is a collection of images with their associated classes. All images in a CFS must have the same image type**
- **CFS:**
  - **Class table with the name and id of each class**
  - **For each class an image table, each image in an image table has its unique image index number**
  - **For each image a feature table, the features are the result of executing the script on the image**

Class Feature Set (*)



Script for calculating features (*)



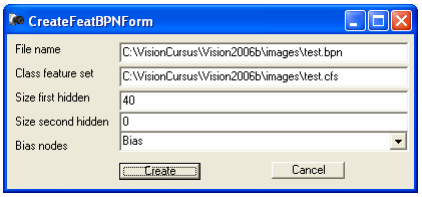
28-Aug-08 Classification with Neural Networks 83

Conventions for script used for CFS (*)

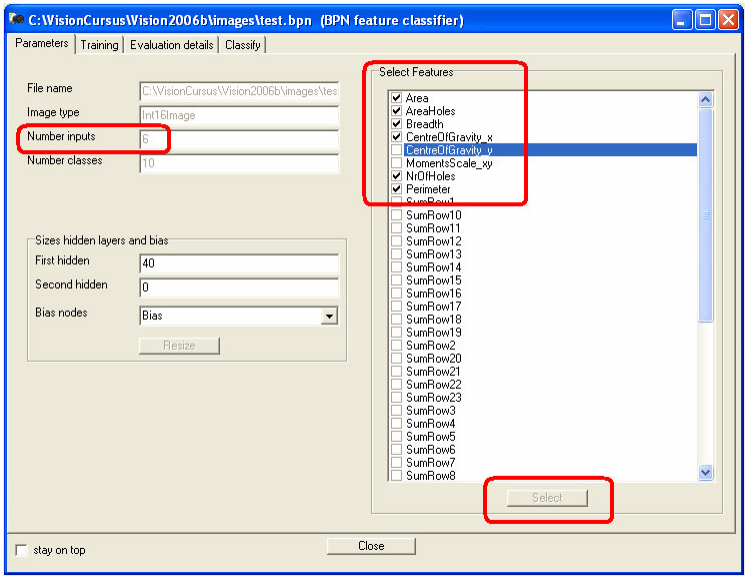
- **Script is used for calculating the features for each image in the CFS**
- **First parameter (%p1) is name of image**
- **Function result is string with features separated by a space and in order of feature id**

28-Aug-08 Classification with Neural Networks 84

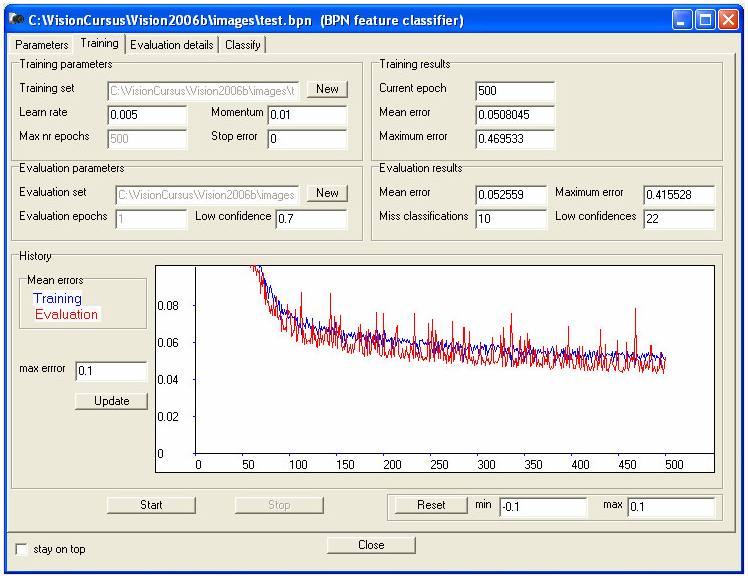
Create BPN Feature Classifier (*)



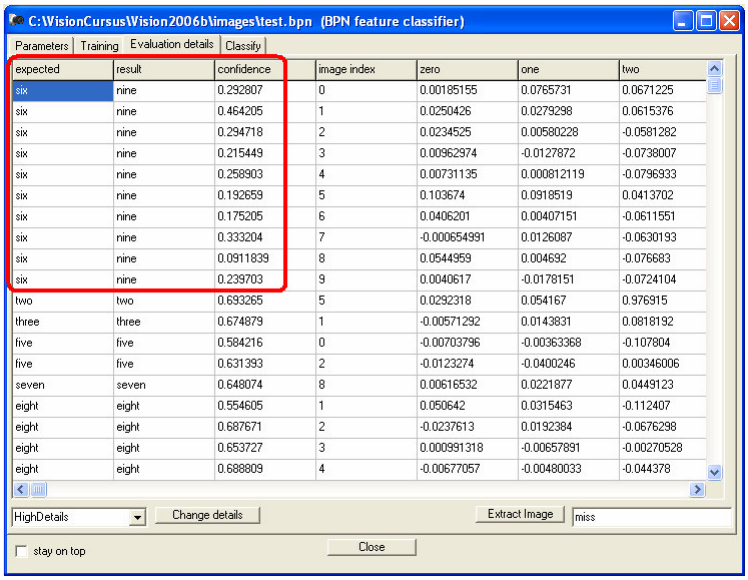
Select Features (*)



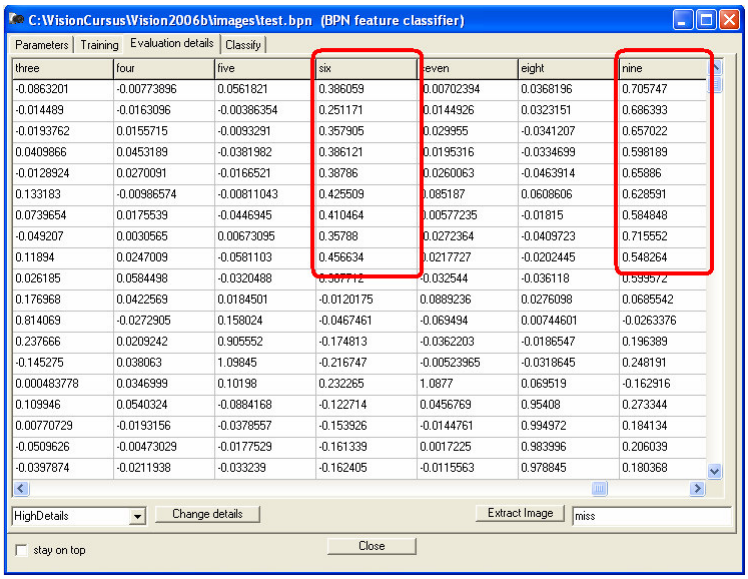
Train (*)



Problems in classifying ‘six’ and ‘nine’ (*)



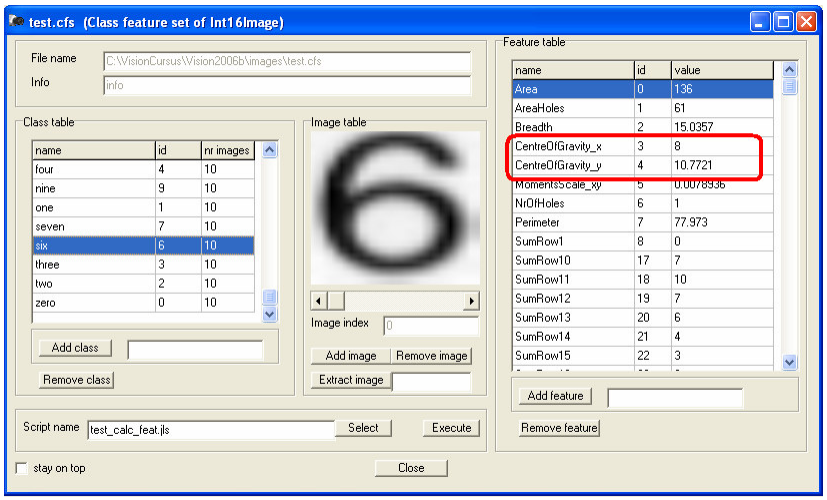
Classifier hesitates between 'six' and 'nine' (*)



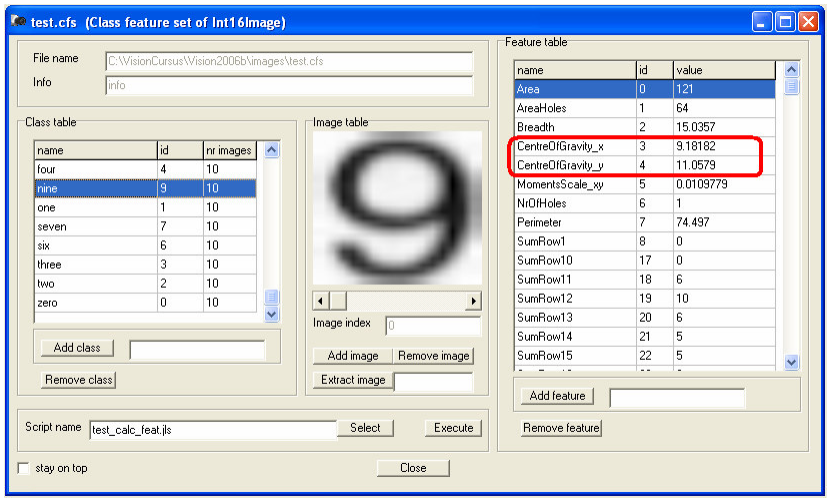
Add extra features (*)

- **Analyse evaluation details:**
  - **Problems in classifying 'six'**
  - **Neuron output 'six' and 'nine' ~ 0.5, others ~0**
- **Use centre of gravity to distinguish between 6 and 9 see feature table in class feature set**
  - **'six' : 7.3 < cg.x < 8.1, 10.7 < cg.y < 11.8**
  - **'nine' : 8.6 < cg.x < 9.9, 10 < cg.y < 11.2**
- **cg.x discriminates between 6 and 9**

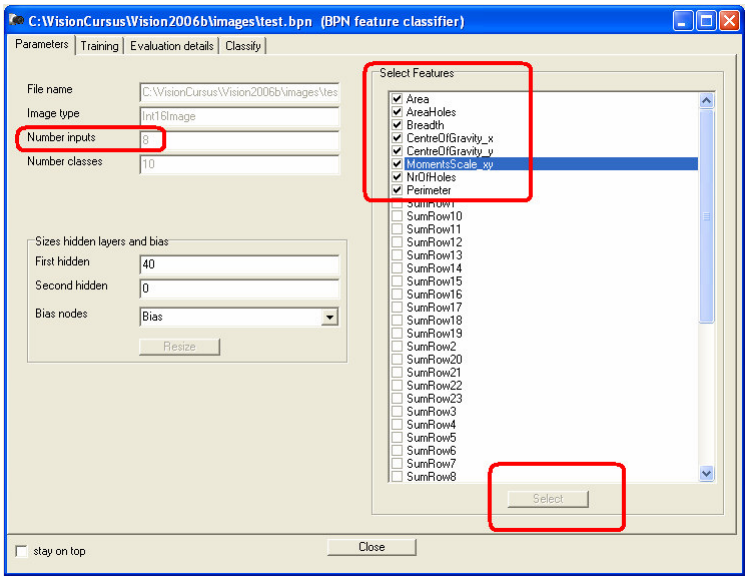
Features for ‘six’ (*)



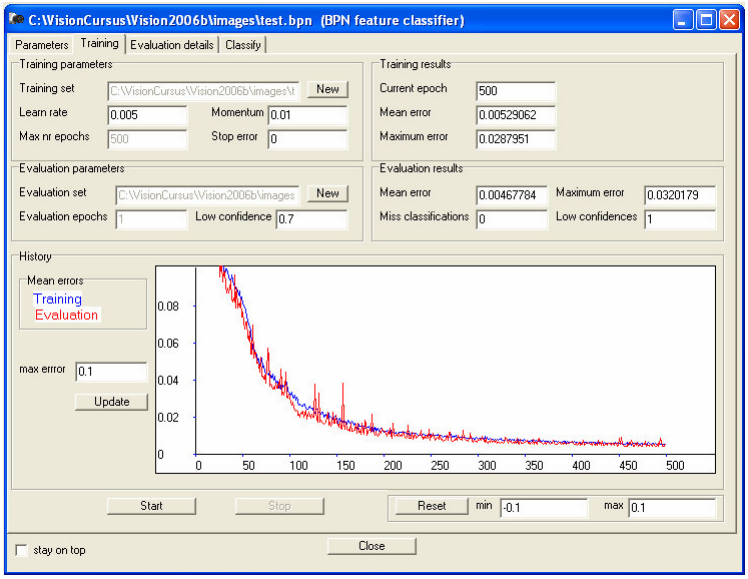
Features for ‘nine’ (*)



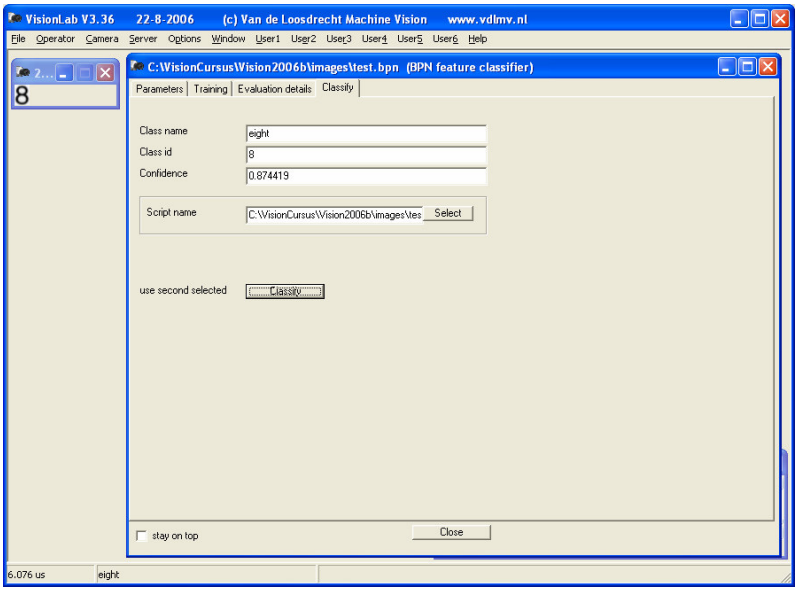
Select extra features cg.x and cg.y (*)



Train with centry of gravity (*)



Classify image (*)



OCR feature vector classification (*)

Conclusions:

- **Complexity has been reduced from 391 dimensions to 8**
- **Selection of which features to use is can be quite difficult**

Note:
Invariant to scale, rotation and lighting can be implemented:

- **In feature extraction**
- **By normalizing the images**

Exercise feature vector classification (*)

Experiment with adding and removing features

Tip: experiment with SumRows and SumCols only