Learning Defect Classifiers for Visual Inspection Images by Neuro-Evolution using Weakly Labelled Training Data

Nils T Siebel, Gerald Sommer

Abstract— **This article presents results from experiments where a detector for defects in visual inspection images was learned from scratch by EANT2, a method for evolutionary reinforcement learning. The detector is constructed as a neural network that takes as input statistical data on filter responses from a bank of image filters applied to an image region. Training is done on example images with weakly labelled defects. Experiments show good results of EANT2 in an application area where evolutionary methods are rare.**

I. Introduction

AUTOMATED industrial inspection is an important area for quality control. It is naturally dominated by engineering approaches to the computer vision task. It would therefore be unthinkable that an evolutionary method could ever be used in such a setting—or would it? In the machine vision industry there is a trend away from engineered solutions with tuned parameters towards learning, self-optimising methods. If a method existed that could learn to detect defects in inspection images from simple training data this would save development cost and setup time.

With this long-term goal in mind the application of evolutionary and neural network-based methods to industrial inspection tasks does not seem so unnatural anymore. Neural networks are universal approximators, so they are able to map image features to classification results. Evolutionary methods are good at global optimisation which is useful in the automatic tuning of parameters like those of the networks.

We aim to learn neural networks that act as classifiers for defect detection. The network is learnt from scratch by EANT2, an evolutionary reinforcement learning method. As network inputs we calculate statistics on filter responses from a bank of image filters applied to an image region.

The remainder of the article is as follows. Section II describes related work. Details on our neuro-evolutionary method can be found in Section III. The test setup and results are located in Section IV, followed by conclusions in Section V.

II. Related Work

In this section we will discuss related work concerning methods for learning neural networks by evolutionary algorithms. To our knowledge, no related work exists for the application of these neuro-evolutionary methods to visual inspection tasks.

The authors are with the Cognitive Systems Group, Institute of Computer Science, Christian-Albrechts-University of Kiel, Germany (e-mail: {nts,gs}@ks.informatik.uni-kiel.de).

Until recently, only small neural networks have been evolved by evolutionary algorithms [1]. According to Yao, a main reason is the difficulty of evaluating the exact fitness of a newly found structure: In order to fully evaluate a *structure* one needs to find the optimal (or, some near-optimal) *parameters* for it. However, the search for good parameters for a given structure has a high computational complexity unless the problem is very simple *(ibid.)*.

Most recent approaches evolve the structure and parameters of the neural networks simultaneously. Examples include EPNet [2], GNARL [3] and NEAT [4]. EPNet uses a modified backpropagation algorithm for parameter optimisation (i.e. a local search method). The mutation operators for searching the space of neural structures are addition and deletion of neurons and connections (no crossover is used). A tendency to remove connections/nodes rather than to add new ones is realised in the algorithm. This is done to counteract "bloat"—i.e. ever growing networks with only little fitness improvement, also called "survival of the fattest" [5]. GNARL is similar in that is also uses no crossover during structural mutation. However, it uses an evolutionary algorithm for parameter adjustments. Both parametrical and structural mutation use a "temperature" measure to determine whether large or small random modifications should be applied—a concept known from simulated annealing [6]. In order to calculate the current temperature, the algorithm needs some knowledge about the "ideal solution" to the problem, e.g. the maximum fitness expected to be reached.

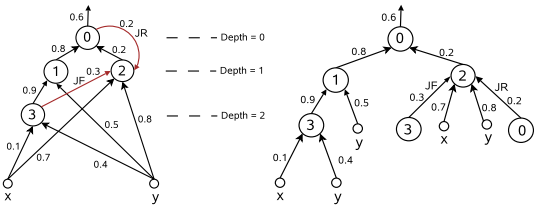
Unlike EPNet and GNARL, NEAT uses a crossover operator that allows to produce valid offspring from two given neural networks. It works by first aligning similar or equal subnetworks and then exchanging differing parts. Like GNARL, NEAT uses evolutionary algorithms for both parametrical and structural mutation. However, the probabilities and standard deviations used for random mutation are constant over time. NEAT also incorporates the concept of speciation, i.e. separated sub-populations that aim at cultivating and preserving diversity in the population [5].

III. Learning Neural Networks with EANT2

A. The Algorithm

EANT2, "Evolutionary Acquisition of Neural Topologies Version 2", is an evolutionary reinforcement learning system that realises neural network learning with evolutionary algorithms both for the structural and the parametrical part. It is based on the previous method EANT [7] but uses different algorithms for structural mutation and parameter optimisation [8]. EANT2 represents neural networks and

978-1-4244-1823-7/08/$25.00



(a) Original neural network (b) Network in tree format

N 0 W=0.6	N 1 W=0.8	N 3 W=0.9	I x W=0.1	I y W=0.4	I y W=0.5	N 2 W=0.2	JF 3 W=0.3	I x W=0.7	I y W=0.8	JR 0 W=0.2

(c) Corresponding Linear Genome

Fig. 1. An example of encoding a neural network using a linear genome

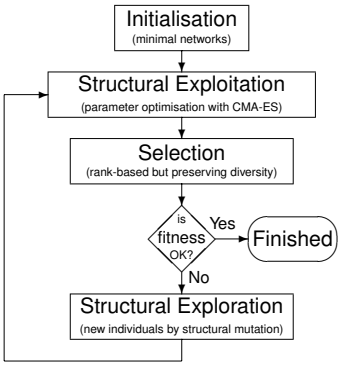


Fig. 2. The EANT2 algorithm. Please note that CMA-ES has its own optimisation loop which creates a nested loop in EANT2.

their parameters in a compact genetic encoding, the "linear genome". It encodes the topology of the network implicitly by the order of its elements (genes). The following basic gene types exist: neurons, network inputs, biases and forward connections. There are also "irregular" connections between neural genes which we call "jumper connections". Jumper genes can encode either forward or recurrent connections. Figure 1 shows an example encoding of a neural network using a linear genome. The figures show (a) the neural network to be encoded. It has one forward and one recurrent jumper connection; (b) the neural network interpreted as a tree structure; and (c) the linear genome encoding the neural network. In the linear genome, `N` stands for a neuron, `I` for an input to the neural network, `JF` for a forward jumper connection, and `JR` for a recurrent jumper connection. The numbers beside `N` represent the global identification numbers of the neurons, `x` and `y` are the inputs coded by input genes. As can be seen in the figure, a linear genome can be interpreted as a tree based program if one considers all the inputs to the network and all jumper connections as terminals.

Linear genomes can be evaluated, without decoding, similar to the way mathematical expressions in postfix notation are evaluated. For example, a neuron gene is followed by its input genes. In order to evaluate it, one can traverse the linear genome from back to front, pushing inputs onto a stack. When encountering a neuron gene one pops as many genes from the stack as there are inputs to the neuron, using their values as input values. The resulting evaluated neuron is again pushed onto the stack, enabling this subnetwork to be used as an input to another neuron. Connection ("jumper") genes make it possible for neuron outputs to be used as input to more than one neuron, see `JF3` in the example above.

The steps of our algorithm, shown in Figure 2, are explained in detail below.

Initialisation: EANT2 usually starts with minimal initial structures. An minimal network has no hidden layers or recurrent connections, only 1 neuron per output, connected to some or all inputs. EANT2 gradually develops these simple initial structures further using the structural and parametrical evolutionary algorithms discussed below. On a larger scale new neural structures are added to a current generation of networks. We call this "structural exploration". On a smaller scale the current structures are optimised by changing their parameters: "structural exploitation".

Structural Exploitation: At this stage the structures in the current EANT2 population are exploited by optimising their parameters. Parametrical mutation is realised using CMA-ES ("Covariance Matrix Adaptation Evolution Strategy") [9]. CMA-ES is a variant of Evolution Strategies that avoids random adaptation of the strategy parameters. Instead, the search area that is spanned by the mutation strategy parameters, expressed here by a covariance matrix, is adapted at each step depending on the parameter and fitness values of current population members. CMA-ES uses sophisticated methods to avoid problems like premature convergence and is known for fast convergence to good solutions even with multi-modal and non-separable functions in high-dimensional spaces *(ibid.)*.

Selection: The selection operator determines which population members are carried on from one generation to the next. Our selection in the outer, structural exploration loop is rank-based and "greedy", preferring individuals that have a larger fitness. In order to maintain diversity in the population, it also compares individuals by structure, ignoring their parameters. The operator makes sure that not more than 1 copy of an individual and not more than 2 similar individuals are kept in the population. "Similar" in this case means that a structure was derived from an another one by only changing connections, not adding neurons. Again, no network parameters are considered here.

Structural Exploration: In this step new structures are generated and added to the population. This is achieved by applying the following structural mutation operators to the existing structures: Adding a random subnetwork, adding or removing a random connection and adding a random bias. Removal of subnetworks (i.e. neurons together with all their connections) is not done as we found out that this almost

never helps in the evolutionary process. The same is valid for a crossover operator, modelled after the one used in NEAT, which is currently not used. New hidden neurons are connected to approx. 50 % of inputs; the exact percentage and selection of inputs are random to enable stochastic search for new structures.

B. Comparison with Other Methods

EANT2 is closely related to the methods described in Section II. One main difference is the clear separation of structural exploration and structural exploitation. By this we try to make sure a new structural element is tested ("exploited") as much as possible before a decision is made to discard it or keep it, or before other structural modifications are applied. Another main difference is the use of CMA-ES in the parameter optimisation. Further differences of EANT2 to other recent methods are a small number of user-defined algorithm parameters (the method should be as general as possible) and the explicit way of preserving diversity in the population (unlike NEAT's speciation).

In the past we have compared EANT2 with NEAT by applying both algorithms to the same problem [10]. The test environment was a visual servoing task run in a simulation. Both methods were to develop neural networks to control a robot in 3 degrees of freedom in order to align its gripper to an object. The robot movement was determined based on 10 image measurements, so the networks had 10 inputs and 3 outputs. The results showed that NEAT had more problems than EANT2 finding good parameters for given networks. This, together with other features of NEAT, inhibited the development of networks such that EANT2 was at a clear advantage in this comparison. More details on the results of these experiments can be found in [10].

IV. Experiments and Results

A. Motivation and Context

On the DAGM Symposium 2007, the yearly conference of the DAGM (German Association for Pattern Recognition) Bosch organised a competition on a visual inspection task. From the description given on the competition website[1]:

"The particular challenge of this contest is that the algorithm must learn, without human intervention, to discern defects automatically from a weakly labeled (i.e., labels are not exact to the pixel level) training set, the exact characteristics of which are unknown at development time. During the competition, the programs have to be trained on new data without any human guidance.

"The provided data is artificially generated, but similar to real world problems. It consists of multiple data sets, each consisting of 1000 images showing the background texture without defects, and of 150 images with one labeled defect each on the background texture. The images in a single data set are very similar, but each data set is generated by a different texture model and defect model.

[1]http://klimt.iwr.uni-heidelberg.de/dagm2007/prizes.php3, visited on August 11 2007

"Not all deviations from the texture are necessarily defects. The algorithm will need to use the weak labels provided during the training phase to learn the properties that characterize a defect. (...) For the development of the algorithm, the participants will be provided with 6 different data sets, each simulated using a different texture and defect model. During the competition, the performance of the participants' algorithms will be tested on 4 different data sets, previously unknown to the users. Note that these data sets are generated by texture models and defect models different from the models of the first 6 data sets."

The preparation of the image data was organised by the Bosch R&D department for automated optical inspection systems which suggests that the data is reasonably realistic.

Figure 3 shows the images available on the website for the development of algorithms. Shown are for each of the 6 classes of defects 2 images with defects. It can be seen that the data is very difficult since

- a large variation exists in the background (non-defective image regions),
- some defects are small and/or very similar to the background and
- the weak label, given as an elliptical region, often contains many more background pixels than defect pixels.

B. Approach and Features Used

Our approach to a solution was to set up a range of feature detectors that describe a given image region. Statistical data on these feature responses is then fed as an input to a neural network which outputs a scalar value for the classification: positive (there is a defect) or negative (there is no defect).

Two methods for image feature extraction are used: the Structure Multivector by Felsberg and Sommer [11] and the feature point detector by Harris and Stephenson [12].

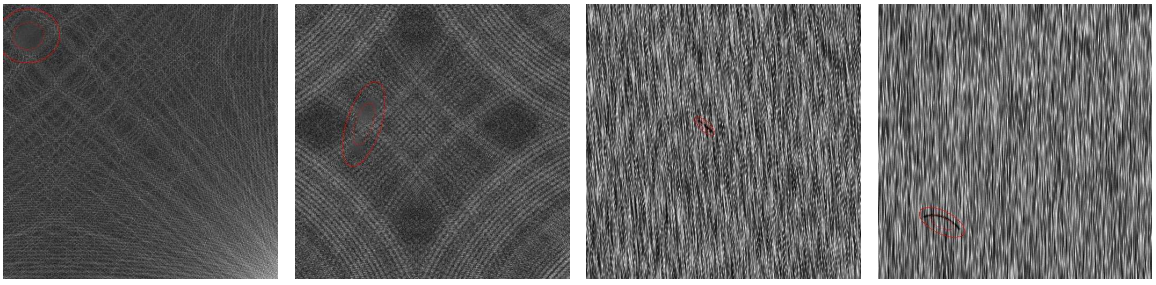
The **Structure Multivector** [11] is an embedding of local image structure in a Clifford algebra [13]. It allows to extract structural information about the image, like local amplitude, phase and intrinsic dimensionality. Derived within the framework of geometric algebra, it is implemented by simple operations in the Fourier domain. Like most methods that work in the Fourier space it allows the restriction of analysis to certain frequencies (i.e. image scales). Since we do not know in advance which scale is the correct one we apply the filter at a range of scales. In figure 4 are shown the major and minor components of the local amplitude from the response of the structure multivector applied on the image shown in Figure 3(e) with a bandpass filter range of 1 to 2 pixels. It can be seen that at the position of the defect these filter responses are relatively strong.

The **Harris feature detector** is based on the operator

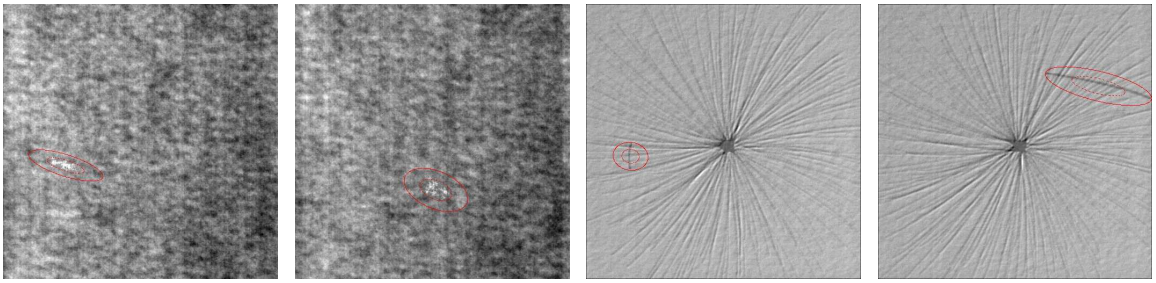
$$R(x,y) = \det \hat{C} - \kappa \cdot \mathrm{trace}^2(\hat{C})$$

where

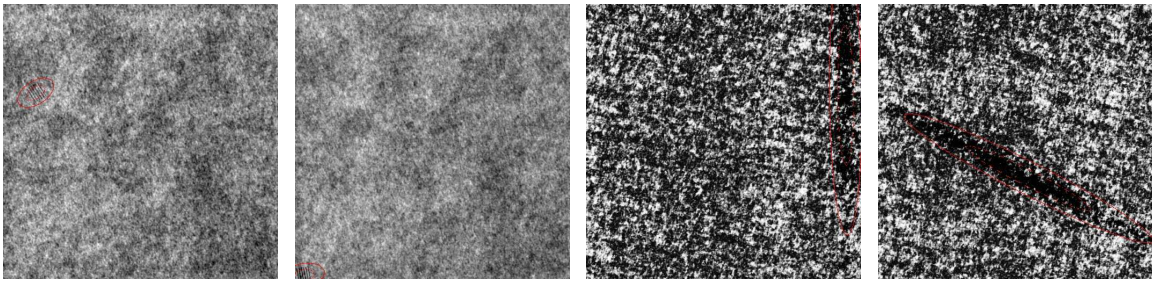
$$\hat{C} = \begin{bmatrix} \hat{I}_x^2 & \widehat{I_x I_y} \\ \widehat{I_x I_y} & \hat{I}_y^2 \end{bmatrix}$$



(a) Class 1, defective, im. 20 (b) Class 1, defective, im. 76 (c) Class 2, defective, im. 30 (d) Class 2, defective, im. 40



(e) Class 3, defective, im. 10 (f) Class 3, defective, im. 30 (g) Class 4, defective, im. 20 (h) Class 4, defective, im. 40



(i) Class 5, defective, im. 40 (j) Class 5, defective, im. 150 (k) Class 6, defective, im. 120 (l) Class 6, defective, im. 140

Fig. 3. Development data, Classes 1 through 6. Shown for every class are 2 out of 150 images with defects. Defects are marked with the weak label provided with the data: the bigger of the 2 red ellipses drawn here. The smaller ellipses (half the diameter of the larger ones) was not given but is drawn here to show that in many cases the defect only takes $\frac{1}{4}$ or less of the area of the ellipse given. In some cases, however, (e.g. in (b) above) the inner ellipse only contains a small or no part of the defective area. Difficulties also include defects that extend outside the image; in (j) only a small part of it is in inside the image. Ellipses in class 6 are much larger than in other sets.

and $\hat{I}_{\cdot}$ denotes the smoothed image derivatives in the given spatial direction. κ can be used to steer the detector between a corner (large κ) and an edge detector (small κ).

In our system we use a fixed value of $\kappa = 0.01$ and applied this detector at 3 scales, using mask widths of 2, 4 and 8 pixels. Figure 5 shows the responses for the image in Figure 3(j): strong responses are marked with a pink dot, weaker responses in green. It can be seen that at the density of Harris responses, especially strong ones, is higher in the region of the defect: by a factor of 4.9 (coarse scale) and 25.5 (fine scale).

C. Feature Extraction and Training

In order to use a natural data description we worked with elliptical image regions. Given 150 training examples of defect ellipses a simple algorithm estimates the distribution of their size and eccentricity. It then generates a set of 50–200 similar ellipses that cover the whole image, as in Figure 6.

On each image the structure multivector and the Harris detector are applied at 3 different scales, which yields feature

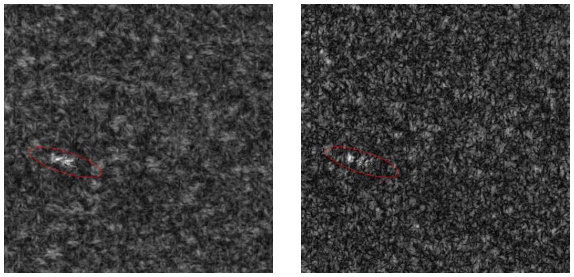


Fig. 4. Normalised major and minor amplitude response of the Structure Multivector applied to im. 10 of class 3, development data (see Figure 3(e))

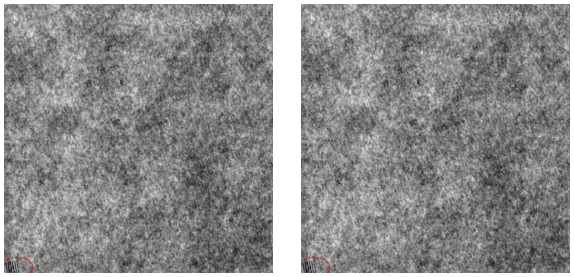


Fig. 5. Harris feature point responses at coarse (left) and fine (right) scales. Strong responses are marked in pink, weaker ones in green. Class 5, im. 150.

images and lists, respectively. On the 150 positive training images the weak label (ellipse) was used to extract feature data, on the 1000 negative images one of the generalised ellipses was used per image. Given an elliptical region the mean value/density, variance and entropy of feature responses within the region are determined using standard methods from statistics. These values are set in relation to the values obtained with the same method outside an ellipse of twice the size, giving a relative measure.

This gives altogether 81 scalar values describing the structure in that region. The result is an 81-dimensional input vector to the neural networks, the output being a scalar ranging from -1 (no defect) to 1 (defect found), using a tanh activation function. The fitness function f used for evaluating a given network N on a set of positive training data D_P and negative training data D_N during EANT2 development is the negative mean difference of the currently calculated responses to ground truth from the training data, i.e.

$$f(N) = -\sum_{d \in D_P} \mathrm{m}(|N(d) - 1|) - \sum_{d \in D_N} \mathrm{m}(|N(d) + 1|),$$

where the function

$$\mathrm{m}(x) = \begin{cases} x, & \text{if } x > 0.01 \\ 0, & \text{else} \end{cases} \tag{1}$$

is used to avoid optimisation problems (very large weights)

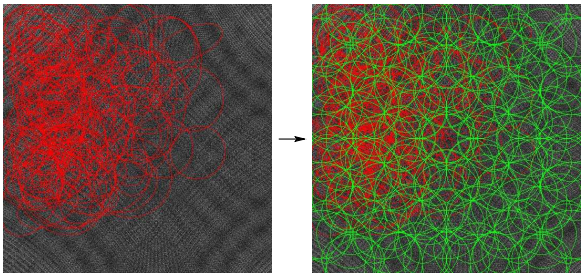


Fig. 6. Generalisation of ellipses given training examples

due to the nature of the tanh activation function. No information about the task or desired properties of the solution are explicitly given to the EANT2 algorithm.

D. Network Development on Development Data

Figure 7 shows the development of the fitness value plotted against the EANT2 generation, i.e. the number of structural mutations. In each EANT2 generation (outer loop in Figure 2) CMA-ES optimises parameters for a maximum of 7,500 generations. For 4 of the 6 defect classes a training error of near 0 is achieved with the initial networks only by optimising their parameters. For classes 1 and 2 the error is reduced to 0.0794 and 0.1125 at generation 15. At this stage the process was stopped since in the original setting from the competition a time limit of 24 hours for the development of detectors was given. Network sizes range from 75 to 203 at this stage.

These results look very good, however, in this training data the "exact" (weak) labels of the defects were given, and only one ellipse region per image is used to extract data. In the testing/application stage the generalised ellipses are used instead to sample the whole image, and the maximum response of the neural detector used as a classification result for a given image. It is very unlikely that a defect will be examined by a perfectly matching ellipse (size, orientation). Also, with up to 200 samples of the image the likelihood of false positives increases accordingly.

The fast reduction of the error for classes 3 through 6 came as a surprise because no structural development except for the random initialisation of networks (EANT2 population size: 30) was done. We believe that in these cases simple networks are already sufficient to model the training data due to the rich information provided by the feature detectors. (This did not occur with the Competition data which suggest that it is more difficult.)

E. Results on Competition Data

The data of the actual competition is shown in Figure 9. 4 new defect classes with 4 new classes of background pattern were again provided with 150 positive (defective) and 1000 negative (no defect) images. Weak labels as seen in Figure 9 were given for positive images.

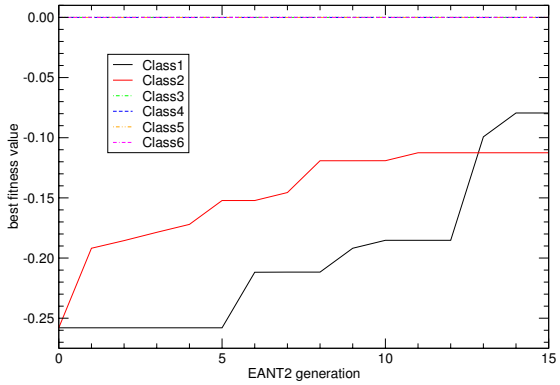


Fig. 7. Development of training fitness plotted against EANT2 generation, for all 6 defect classes of the development data.

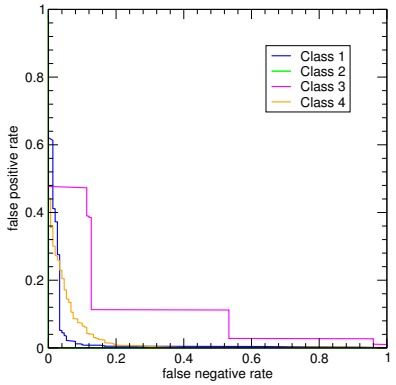


Fig. 8. ROC curves for all 4 classes of the Competition data, EANT2 classifier.

Our algorithm was trained, as before, on the feature responses of the structure multivector and Harris detector. Features were extracted using the given defect ellipse (positive examples) or one of the "generalised ellipses" per image (negative examples). The training data obtained in this way was used to develop neural networks with EANT2.

When performing classification on a given image it is scanned using the generalised ellipses, e.g. 151 for Defect Class 3. The maximum of the 151 responses of the neural network is used as a (real-valued) classification value for the whole image. A threshold τ can then be used as follows to determine the result $R(I)$ of the classifier given the neural network responses $N(e_i)$ for i ellipses e_i covering an image I:

$$R(I) = \begin{cases} \text{defect}, & \text{if } \max\limits_{e_i \subset I} N(e_i) \geq \tau \\ \text{no defect}, & \text{if } \max\limits_{e_i \subset I} N(e_i) < \tau \end{cases}$$

Figure 8 shows a ROC-type curve[2] of the classifier applied to all 4 defect classes of the training data (where ground truth is available). It shows how a trade-off between false positives (defect detected when there is none) and false negatives (no detection when there is a defect) can be made by adjusting the threshold τ.

For Defect Class 2 a perfect separation between positive and negative responses is achieved such that no wrong classification is calculated; the smallest response for an image with defect is -0.7551, the largest response of an image without defects is -0.8586. Therefore any $\tau : -0.7551 < \tau \leq -0.8586$ will yield 100 % correct classification results. The result for Classes 1 and 4 is also very good. For example, within a threshold range $-0.6915 < \tau < -0.7039$ the correct classification rate for Class 1 is 97.9 %. Class 4 reaches 96.7 % correct classification.

For Class 3 the detection results are not as good, with a maximum classification rate of 90.6 %. We believe this is due to the small size of the defect in relation to the image and the similar appearance of the defects to parts of the non-defective image area. (To our consolation even a human test subject identified only 148 of 150 defect images hidden among 1000 non-defective images although he examined all 1150 images 4 times.)

[2]Receiver Operating Characteristic, see e.g. [14]. In this case we have plotted the false positives against the false negatives.

V. Summary and Conclusions

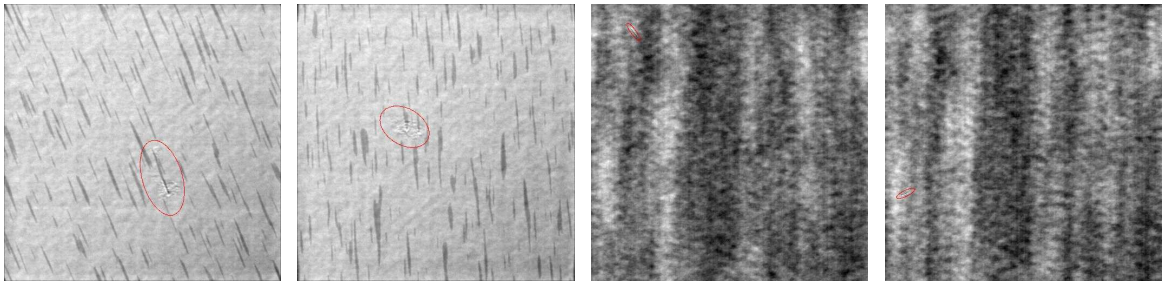
In this article we dealt with the task of creating a system that could automatically train a classifier for defect detection in visual inspection images using a set of weakly labelled training data. Our implementation of such a system uses EANT2, a method for creating neural networks by evolutionary reinforcement learning, as its learning component. Feature extraction from a given image uses the Structure Multivector and the Harris detector, yielding an 81-dimensional feature vector to describe the appearance of an image region. This is used as an input to the neural networks, the output being the scalar classification value.

Our experiments showed that the data used during training, using 1 example region per image, could be relatively easily classified by networks evolved by EANT2. When the classifier is used to examine a whole image, however, the detector is applied to 50–200 regions covering the image area. In this case the correct classification rate of our method ranges from 90 % to 100 %, depending on the type of defect and background used in the image.

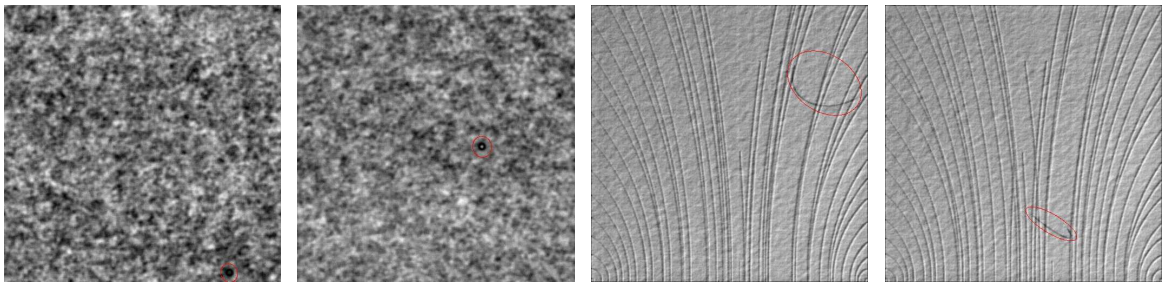
In conclusion, our experiments have shown that EANT2 can generate neural networks that correctly classify almost all defects in visual inspection images. Improvements on the data processing and/or the learning method are necessary to improve the classification rate for some difficult types of images.

Acknowledgements

The author wishes to thank the Robert Bosch GmbH Stuttgart, especially Christian Perwass, for organising the DAGM 2007 competition and providing the image data.



(a) Class 1, defective, im. 3 (b) Class 1, defective, im. 129 (c) Class 2, defective, im. 120 (d) Class 2, defective, im. 121



(e) Class 3, defective, im. 116 (f) Class 3, defective, im. 121 (g) Class 4, defective, im. 109 (h) Class 4, defective, im. 132

Fig. 9. Competition training data, all 4 Classes. Shown are 2 out of 150 images of every class with defects. Red: weak label provided with the data.

REFERENCES

[1] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, September 1999.

[2] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, May 1997.

[3] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[4] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[5] Á. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Germany: Springer Verlag, 2003.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[7] Y. Kassahun and G. Sommer, "Efficient reinforcement learning through evolutionary acquisition of neural topologies," in *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN 2005)*, Bruges, Belgium, April 2005, pp. 259–266.

[8] N. T. Siebel and Y. Kassahun, "Learning neural networks for visual servoing using evolutionary methods," in *Proceedings of the 6th International Conference on Hybrid Intelligent Systems (HIS'06), Auckland, New Zealand*, December 2006, p. 6 (4 pages).

[9] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[10] N. T. Siebel and G. Sommer, "Evolutionary reinforcement learning of artificial neural networks," *International Journal of Hybrid Intelligent Systems*, vol. 4, no. 3, pp. 171–183, October 2007.

[11] M. Felsberg and G. Sommer, "Structure multivector for local analysis of images," in *Proceedings of the 10th International Workshop on Theoretical Foundations of Computer Vision, Dagstuhl, Germany*, 2001, pp. 93–104.

[12] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference, Manchester, UK*, 1988, pp. 147–151.

[13] C. B. U. Perwass, "Applications of geometric algebra in computer vision," PhD thesis, Geometric Algebra Research Group, University of Cambridge, Cambridge, UK, 2000.

[14] T. Fawcett, "ROC graphs: Notes and practical considerations for data mining researchers," HP Labs, Technical Report HPL-2003-4, 2003.