Evolution of hierarchical translation-invariant feature detectors with an application to character recognition

Marc Ebner

Eberhard-Karls-Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Köstlinstraße 6, 72074 Tübingen, Germany
ebner@informatik.uni-tuebingen.de

Abstract. The task of feature extraction is usually performed by chaining a series of well known elementary operators. We are trying to automate the process by using a genetic algorithm to evolve the required feature detectors. In this case one only needs to specify the fitness function for the given problem. In this paper we present our results with the evolution of feature detectors for the task of character recognition.

1 Introduction

1.1 Motivation

In image processing one is often faced with the task of extracting certain features from an image. Usually this problem is approached by selecting certain well known operators such as edge extraction or Gaussian smoothing and applying them to the images to get the desired result for the problem at hand. We are trying to automatically evolve feature detectors for arbitrary tasks using evolutionary algorithms. The process of feature extraction may then be reduced to formulating an appropriate fitness function. In this paper we show that automatically evolved feature detectors can be successfully applied to the task of extracting an example character from a set of characters using the digits zero through nine.

1.2 Background

Roth and Levine [3] have used a genetic algorithm [4] to evolve feature detectors to extract geometric elements. A necessary prerequisite is that the elements can be described by an equation in the form $f(\bar{x}, \bar{a}) = 0$ where $\bar{x}$ is a point on the element and $\bar{a}$ are the parameters of the equation.

Katz and Thrift [1] have generated image filters for target recognition with a genetic algorithm. To extract an object from an image they have used a single linear filter. To classify an extracted object they have used a fixed number of linear filters with a fixed size which resulted in a single feature vector. This feature vector is then used for classification. A special 2D-crossover operation is used to evolve the filters. This 2D-crossover cuts out an area of the filter and swaps the contents with the corresponding area selected from the other parent.

Koza has successfully applied his genetic programming paradigm [5] to the field of letter recognition [6]. He evolved detectors which were able to recognize the letter "I" or "L". The detectors examine the pixel grid by moving over the pixels and looking if the pixel at the current position is set or not. Koza introduced five automatically defined functions [6] which enabled the detectors to examine the pixels in the local area around them at the current position.

Andre [2] has evolved individuals which locate a particular number or letter in an image using genetic programming [5]. The detectors are required to look for the features by moving themselves over the image. They are equipped with five 3×3 hit-miss matrixes which can be used for a matching operation at the current location. Andre also used a 2D-crossover operation to evolve the hit-miss matrixes. Two random areas of equal size are selected from the two parents and exchanged to create two offspring.

Lohmann [7] used structure evolution to evolve an operator which computes the Euler number [11] of an image. Lohmann evolved the parameters and the structure of a local filter. The sum of all local filters then gives the output for the complete image. Each local filter has a 3×3 input field. Elements are combined using binary multiplication. Finally the output of the local detector is the sum over all binary products of the input field.

Johnson et al. [10] used Genetic Programming to evolve visual routines. They used point operators, feature detectors and point list filters as primitive functions for the problem of determining the position of the left or right hand in a silhouette of a person.

Kröner and Schulz-Mirbach [13] focused on the task of adaptive calculation of invariant features. They used a fixed hierarchy of transfer functions which are symmetric in their input arguments. The weights of the transfer functions are trained by minimizing an error measure on the number of incorrectly classified patterns.

2 Evolution of hierarchical translation-invariant detectors

To evolve our feature detectors we have used a variant of an evolutions strategy [12] called structure evolution. Lohmann [8, 9, 7] introduced structure evolution to evolve individuals containing discrete and continuous variables. For our feature detectors, we have used variable sized hierarchical individuals much like the individuals used in the genetic programming paradigm [5].

2.1 Individuals

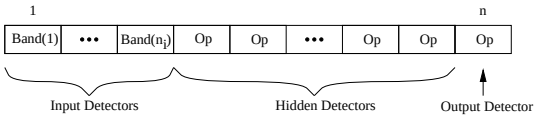

Fig. 1. General structure of one individual.

Each individual (figure 1) is an array of a set of elementary detectors. It consists of input detectors, hidden detectors and output detectors analogous to

the neurons of a neural net [14]. Each of the elementary detectors has a predefined structure. The types used as elementary detectors are described in the following subsection. To impose a hierarchical treelike structure on the resulting detector each elementary detector may receive input from one or two other elementary detectors. This input consists of a single band image. Each detector is only allowed to use the output of the detectors to its left. Thus using this method we automatically generate a tree structure. An individual with its corresponding hierarchical structure can be found in figure 4.

The input that the individual receives is also part of this structure. For every band in the input image we place a detector in the individual which simply outputs the particular band. All input detectors are located at the beginning of the array. If we have a n_i band input image, the lower n_i detectors in the array supply the individual bands to be used for processing by the following detectors.

2.2 Input to the detectors

The input to the detectors consists of several bands calculated from the input image. In case of a single band image the input is exactly this single band input image. The bands calculated from an color input image could be the individual color components red, green and blue or hue, saturation and intensity or both.

2.3 Elementary operators

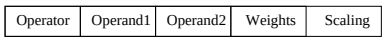

Fig. 2. Structure of one elementary operator.

Our evolving feature detector is composed of several elementary operators (figure 2). These elementary operators constitute the set of primitive functions [5] for our algorithm. We have provided the system with monadic and dyadic operators. Monadic operators operate on a single band image. Dyadic operators operate on two single band images.

In the following text $f_i(x, y)$ denotes the output of the i-th feature detector at position (x, y). We set the resulting output image by looping over all image positions. Every operator only uses local information. Let $N(x, y)$ be the set of points p in the neighbourhood of (x, y) which the operator can use for its calculation. The weight from point p to the operator i is given by w_{pi}. The factor a is used to scale the result.

Monadic operators For a monadic operator the input image is given by $I_{op(i)}$.

Maximum $f_i(x, y) = a \max_{p \in N(x,y)} \{w_{pi} \cdot I_{op(i)}(p)\}$

Minimum $f_i(x, y) = a \min_{p \in N(x,y)} \{w_{pi} \cdot I_{op(i)}(p)\}$

Sum $f_i(x, y) = a \sum_{p \in N(x,y)} w_{pi} \cdot I_{op(i)}(p)$

Product $f_i(x, y) = a \prod_{p \in N(x,y)} I(p)$

Sum of Gaussian $f_i(x, y) = a \sum_{p \in N(x,y)} \delta(w_{pi}, I_{op(i)}(p))$

Product of Gaussian $f_i(x, y) = a \prod_{p \in N(x,y)} \delta(w_{pi}, I_{op(i)}(p))$ where $\delta(x, y)$ is defined as $\delta(x, y) = e^{-(x-y)^2/\epsilon}$ and ϵ is a scaling parameter with a fixed value.

Dyadic operators For a dyadic operator the input images are $I_{op1(i)}$ and $I_{op2(i)}$.

Maximum $f_i(x,y) = a \sum_{p \in N(x,y)} w_{pi} \max\{I_{op1(i)}(p), I_{op2(i)}(p)\}$
Minimum $f_i(x,y) = a \sum_{p \in N(x,y)} w_{pi} \min\{I_{op1(i)}(p), I_{op2(i)}(p)\}$
Sum $f_i(x,y) = a \sum_{p \in N(x,y)} w_{pi}(I_{op1(i)}(p) + I_{op2(i)}(p))$
Difference $f_i(x,y) = a \sum_{p \in N(x,y)} w_{pi}(I_{op1(i)}(p) - I_{op2(i)}(p))$
Product $f_i(x,y) = a \sum_{p \in N(x,y)} w_{pi}(I_{op1(i)}(p) \cdot I_{op2(i)}(p))$

3 Evolution

To evolve the feature detectors described here we have to evolve the structure of the complete detector as well as the weights associated with each elementary detector. Since the structure evolution is associated with discrete steps and the weights are continuous variables, which need to be optimized, we use structure evolution [7] to evolve the detectors.

Structure evolution consists of an encapsulated evolutions strategy [12] where the continuous parameters are evolved in the inner loop and the discrete parameters are evolved in the outer loop.

We use mutation on the scaling factor a and the weights w and single point crossover on the linear array consisting of the scaling factor a and the weights w in the inner loop whereas for the outer loop we only use mutation. Mutation in the outer loop consists of a random elementary detector which is added at the far right of the individual. This random detector receives input from one or two (depending on the elementary function of the detector) other elementary detectors or input detectors. The operands are chosen with uniform probability. However, other probability distributions can also be used (e.g. one which has a higher probability of selecting higher order elementary detectors).

Since the two operators of the newly added detector are chosen from the list of existing detectors there always remains the possibility of starting over by choosing the input detectors as operands. Since the operands are chosen with uniform probability from the list of detectors the probability of starting over decreases as the size of the individual increases. In the beginning, while the individual is still small, the algorithm may start over several times evolving detectors for different features. As the size of the individual increases the probability of choosing one of the hidden detectors as operands increases. This produces a series of more and more specialized detectors.

Fitness is calculated by summing over the squared differences between actual and desired output: fitness $= -\frac{1}{n} \sum_{p \in \text{Image}} (o_d(p) - o(p))^2$ where $o_d(p)$ and $o(p)$ is the desired and actual output at position p and n denotes the number of points used for the comparison. Since we are using operators with a square mask we exclude the boundary pixels of the image in calculating the fitness measure.

4 Experiments

We have performed a number of experiments with the system. Two of them have been selected for presentation here.

4.1 Detector for multiple symbols

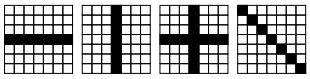

Fig. 3. Symbols used in first experiment.

We have applied our algorithm to the task of finding a translation invariant filter for four different yellow symbols (with RGB color [230,230,30]). The symbols are shown in figure 3. Input to the detector consists of the bands hue, saturation and intensity with the range [0,1]. The symbols are presented to the input detectors in a completely random order. Note that the first two symbols are a subset of the third symbol. Thus is should be relatively easy for the algorithm to develop individuals which are successful in detecting them. The third and fourth symbol are completely independent.

As a neighbourhood $N(x, y)$ for the elementary detectors we have chosen $N(x_0, y_0) = \{(x, y) \quad | \quad |x_0 - x| \leq 1 \quad \text{and} \quad |y_0 - y| \leq 1\}$. This gives every elementary detector a neighbourhood of 3×3 points which can be used as input. The desired output for the individuals is 1.0 at the center of the symbol. No negative examples have been presented in this experiment. Negative examples are only present implicitly since the same detector is applied to all positions of the input image. A perfect individual must have at least depth 2 to determine, if the first, second or fourth individual is shown. The third individual can be detected by an individual of depth zero due to the prominent center of the symbol. A perfect detector could be produced for all four symbols using only the elementary detectors which have been presented above. This detector works by using three elementary detectors. Each detects either a horizontal, vertical or diagonal line of size 7. The fitness for this experiment and the resulting individual

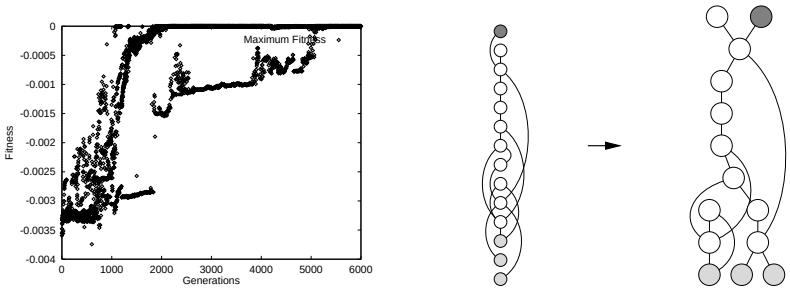

Fig. 4. Improvement of maximum fitness for the first experiment with a $[5, 25(5+25)^{150}]$ evolution strategy ($p_{Mut} = 0.1$, $p_{Cross} = 0.6$). The mutation probability of the outer loop was set to 0.5. The structure shown on the right is the resulting individual. Input bands are colored with a light gray and the output detector is colored with a dark gray.

is shown in figure 4. One can clearly see two different responses of the algorithm depending on the input presented. The algorithm quickly evolved a detector for the first three symbols and finally incorporated the fourth symbol. Since we present the symbols in a completely random order this is analogous to the problem of finding an optimum of a generation dependent fitness function.

4.2 Character recognition

As a second experiment we have applied our detector to the problem of finding one character from a set of characters. The characters we have used for the results reported here have been introduced by Andre [2] for a problem called "the single example digit discrimination task". The problem was to return a "yes" response for the example digit and a "no" for all others. Since our detectors differ from those used by Andre we did not present the digits in isolation and then calculated the fitness. Instead we generated a random 4×4 matrix with numbers from the set $\{0, \ldots, 9\}$. This gives us a 29×29 pixel image with 16 digits (digits are separated by one pixel and the border is 3 pixels wide on all four sides) which is presented as input to the detectors. This single band image has a 1.0 for every set pixel in the digit and a 0 otherwise. The desired output consisted of a 1.0 at the center of all example digits.

Because the matrix of digits is constantly changing the resulting detector has to focus its attention. Only the information in a 5×5 area of the matrix is relevant in determining if an example digit has been found. Pixels outside of the 6×6 area of the digit had to be neglected for the desired response.

As a neighbourhood $N(x, y)$ for the elementary detectors we have again chosen $N(x_0, y_0) = \{(x, y) \quad | \quad |x_0 - x| \leq 1 \quad \text{and} \quad |y_0 - y| \leq 1\}$. We have been able to evolve a correct detector for every digit. In figure 6 the maximum fitness statistics for the problem of locating the corresponding digits are shown.

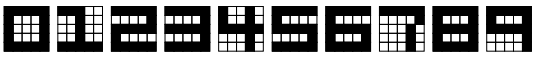

Fig. 5. Numbers used in second experiment (from [2]).

5 Conclusion and ongoing research

The experiments presented here have shown that it is possible to automatically evolve hierarchical translation-invariant feature detectors using structure evolution. Increasingly complex detectors may be evolved using a single mutation operation starting from primitive individuals.

We are currently evolving feature detectors for several different problems including feature extraction from real world images. In using real world images it is interesting to see which input detectors will actually be used in the solution to a particular problem and which will be neglected.

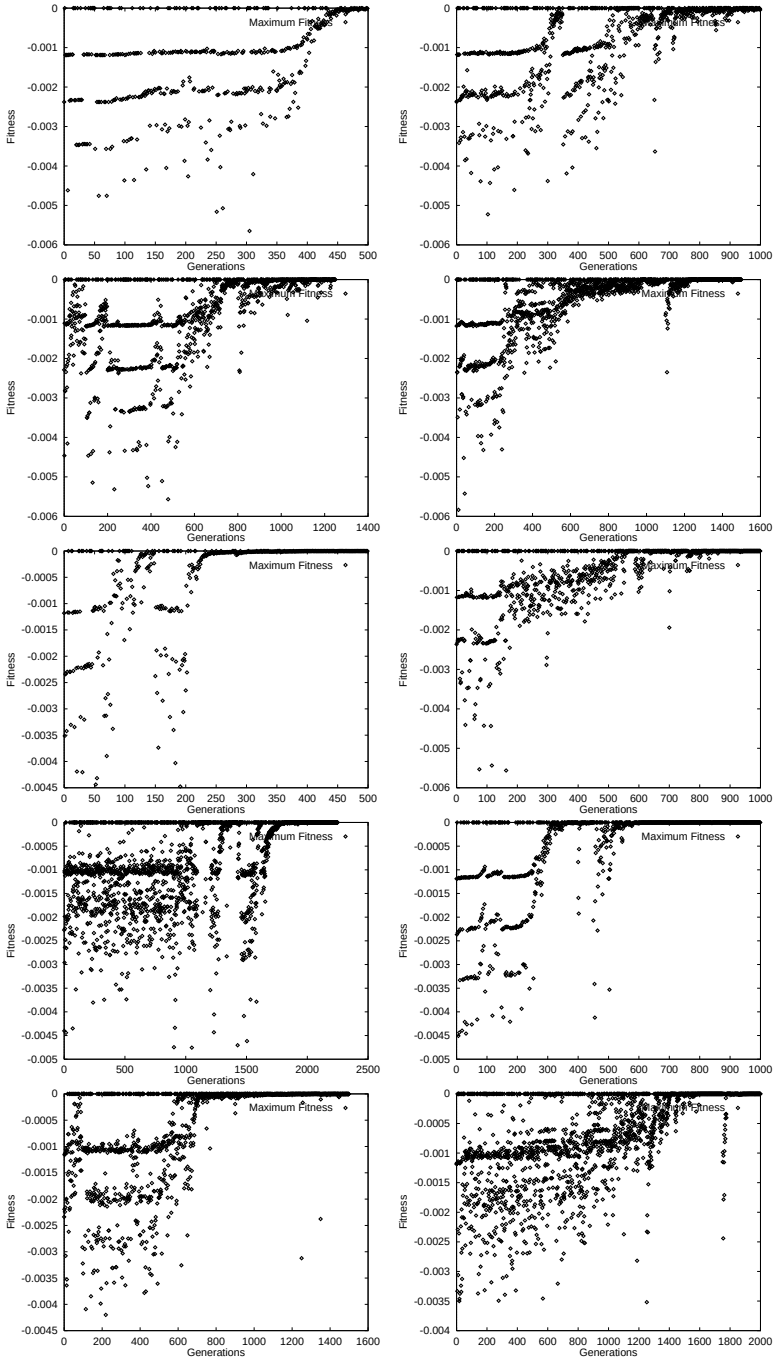

Fig. 6. Improvement of maximum fitness for the digits zero through nine (shown from left to right, top to bottom). Detectors for zero and one were evolved with a $[5, 25(5+25)^{50}]$ evolution strategy. Detectors for two, three, four, seven, eight and nine were evolved with a $[5, 30(5+30)^{50}]$ evolution strategy. The detector for five was evolved with a $[5, 30(5+30)^{100}]$ evolution strategy and finally the detector for six was evolved with a $[5, 30(5+30)^{75}]$ evolution strategy. We have set $p_{Mut} = 0.1$ for all detectors and $p_{Cross} = 0.6$ except for the detectors five, seven and eight where $p_{Cross} = 0.9$. The mutation probability of the outer loop was set to 0.5.

6 Acknowledgements

The author is currently supported by a scholarship according to the Landesgraduiertenförderungsgesetz.

References

1. A. J. Katz and P. R. Thrift. Generating image filters for target recognition by genetic learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):906–910, September 1994.
2. David Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In Jr. Kenneth E. Kinnear, editor, *Advances in Genetic Programming*, pages 477–494, Cambridge, Massachusetts, 1994. The MIT Press.
3. Gerhard Roth and Martin D. Levine. Geometric primitive extraction using a genetic algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):901–905, September 1994.
4. John H. Holland. *Adaptation in natural and artifical systems: an inroductory analysis with applications to biology, control, and artificial intelligence.* The MIT Press, Cambridge, Massachusetts, 1992.
5. John R. Koza. *Genetic Programming, On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.
6. John R. Koza. *Automatic Discovery of Detectors for Letter Recognition*, pages 389–416. The MIT Press, Cambridge, Massachusetts, 1994.
7. R. Lohmann. Structure evolution and incomplete induction. *Biological Cybernetics*, 69:319–326, 1996.
8. Reinhard Lohmann. Selforganization by evolution strategy in visual systems. In Hans-Michael Voigt, Heinz Mühlenbein, and Hans-Paul-Schwefel, editors, *Evolution and Optimization '89*, pages 61–68. Akademie-Verlag, 1990.
9. Reinhard Lohmann. Structure evolution in neural systems. In Branko Soucek and the IRIS Group, editors, *Dynamic, Genetic, and Chaotic Programming*, pages 395–411. John Wiley & Sons, Inc., 1992.
10. Michael Patrick Johnson, Pattie Maes, and Trevor Darrell. Evolving visual routines. In Rodney A. Brooks and Pattie Maes, editors, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 198–209, Cambridge, Massachusetts, 1994. The MIT Press.
11. Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine Vision.* McGraw-Hill, Inc., New York, 1995.
12. Ingo Rechenberg. *Evolutionsstrategie '94.* frommann-holzboog, Stuttgart, 1994.
13. Sabine Kröner and Hanns Schulz-Mirbach. Fast adaptive calculation of invariant features. In G. Sagerer, S. Posch, and F. Kummert, editors, *Tagungsband Mustererkennung 1995, Verstehen akustischer und visueller Informationen, 17. DAGM-Symposium*, pages 23–35, Berlin, 1995. Springer-Verlag.
14. Andreas Zell. *Simulation Neuronaler Netze.* Addison-Wesley (Deutschland) GmbH, Bonn, 1994.

This article was processed using the LaTeX macro package with LLNCS style