Learning Connectedness in Binary Images

Erik van der Werf   Jos Uiterwijk   Jaap van den Herik

Department of Computer Science, Institute for Knowledge and
Agent Technology, Universiteit Maastricht, P.O. Box 616,
6200 MD Maastricht, The Netherlands
{e.vanderwerf,uiterwijk,herik}@cs.unimaas.nl

Abstract

This paper proposes a new Eye-based Recurrent Network Architecture (ERNA) for image classification. The new architecture is trained by a combination of Q-learning and RPROP. The classification performance is compared with other network architectures on the task of determining connectedness between pixels in small binary images. The experiments show that ERNA outperforms both the standard multi-layer perceptron network and the fully-connected recurrent network on the task mentioned above. This performance leads us to the conclusion that the eye facilitates learning in the topologically-structured domain of image classification.

1 Introduction

Determining connectedness is a fundamental task in the field of image processing and pattern recognition. In 1969 Minsky and Papert [2] showed that perceptrons cannot learn the attribute of connectedness. In the 1988 epilogue of the expanded edition they argue that the same holds for multi-layer perceptron (MLP) networks. Comparing this statement with the well-known fact that MLPs can approximate any function arbitrarily close when given sufficient hidden units, we may raise the question whether it is possible to develop a network architecture that learns to determine connectedness.

This paper addresses the issue for the case of connectedness between pixels, i.e., we examine if it is possible to determine that two remote pixels are connected by a path of neighboring pixels. This type of connectedness is underlying global connectedness, where the task is to detect unconnected objects. Recently Wang [7] proposed a special kind of recurrent neural-network architecture based on coupled oscillators, able to determine the number of unconnected objects in an image. Although this result seems promising, his network is prewired rather than trained by examples.

As an alternative we introduce the Eye-based Recurrent Network Architecture (ERNA) that learns to classify images. The new architecture is inspired by the human eye function and is applied to the task of learning connectedness; its performance is compared with three other network architectures.

The remainder of this paper is organized as follows. In section 2 the new network architecture is introduced. Section 3 explains the training procedure. In section 4 we discuss the generation of the data set. Section 5 contains experimental results. Finally, section 6 provides conclusions and future research.

2 ERNA

The standard feed-forward multi-layer perceptron architecture (MLP) for pattern classification usually has one hidden layer with non-linear transfer functions, is fully connected to all inputs, and has an output layer with one neuron assigned to each class. Several training algorithms exist for this network architecture, that can find reasonably good solutions for a great variety of supervised-learning tasks.

The disadvantage of using the MLP for image classification is that the architecture does not exploit any knowledge about the topological ordering of pixels. Although pixels in an image are topologically fixed on a structured grid, the conventional network architectures treat every pixel just as an (arbitrary) element of the input vector, thus ignoring the spatial order of the original representation. For humans this disadvantage becomes evident in the task of recognizing natural images in which the spatial order of pixels is removed either by random permutation or by concatenation into a linear array. Clearly, for methods dealing with low-level image properties, the topological ordering is useful. This observation inspired us to employ a special input for our new network architecture.

Guided by the unrivaled performance of human vision and the fact that humans (and many other animals) have eyes we designed ERNA, an Eye-based Recurrent Network Architecture. Figure 1 shows the main components of ERNA. In our architecture, the eye is an input structure covering a local subset of pixels surrounding a movable point of fixation (see upper left corner). The focusing and scanning operations of the eye impose spatial order onto the input, thus automatically providing information about the topological ordering of pixels.

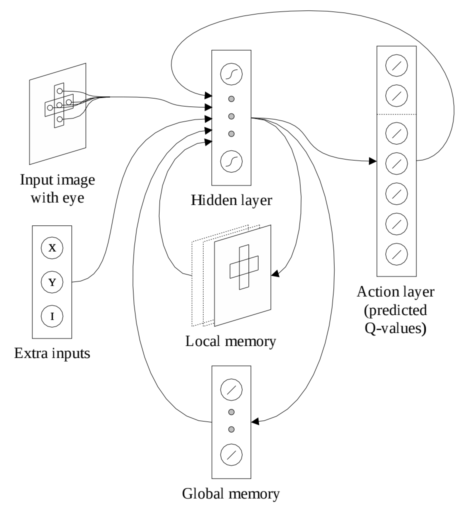

Figure 1: ERNA

The movement of the eye is controlled by five action neurons (left, right, up, down, stay). Together with two action neurons for classification (connected, unconnected) they form the action layer (see upper right corner).

Focusing the eye on relevant pixels usually requires multiple actions. Since knowledge about previously observed pixels may be needed a memory seems necessary. A memory is implemented by adding recurrent connections to the network architecture. The simplest way to do this is linking the output of the hidden layer directly to the input. However, since information is partially redundant, an additional linear layer, called global memory, is applied to compress information between the output of the hidden layer and the input for the next iteration.

Since the global memory has no topological ordering (with respect to the image structure) and is overwritten at every iteration, it is not well suited for long-term storage of information related to specific locations in the image. Therefore, a local memory formed by linear neurons coupled to the positions of the input pixels is devised. At each iteration, the hidden layer is connected to the neurons of the local memory associated with the area visible by the eye. In ERNA the number of local memory neurons for a pixel as well as the readable and writable window size are defined beforehand. The operation of the network is further facilitated by three extra input neurons representing the co-ordinates of the eye's point of fixation (X,Y) and the maximum number of iterations left (I).

Below we briefly discuss the operation of ERNA. At each iteration step the hidden layer performs a non-linear mapping of input signals from the eye, the local memory, the global memory, the action layer and the three extra inputs to the local memory, the global memory and the action layer. The network then executes the action associated with the action neuron with the largest output value. The network iterates until the selected action performs the classification, or a maximum number of iterations is reached.

We note that, next to the normal recurrent connections of the memory, in ERNA the action layer is also recurrently connected to the hidden layer, thus allowing (back)propagation of information through all the action neurons.

Since the eye automatically incorporates knowledge about the topological ordering of pixels into the network architecture, we expect it to facilitate learning in topologically-oriented image-classification tasks, i.e., with the same number of training examples a better classification performance should be obtained. To evaluate the added value of the eye and that of the recurrent connections, ERNA is compared with three other network architectures.

The first network is the MLP, which has a feed-forward architecture with one non-linear hidden layer. The second network is a feed-forward network with an eye. This network is a stripped-down version of ERNA. All recurrent connections are removed by setting the number of neurons for local and global memory to zero. Previous action values are also not included in the input. The third network is a recurrent network with a fully-connected input, a fully-connected recurrent hidden layer with non-linear transfer functions, and a linear output layer with three action neurons (connected, unconnected, thinking). The difference with the MLP is that the hidden layer has recurrent connections and the output layer has a third action-neuron for performing more iterations. This network architecture is very similar to the well-known Elman network [1] except that signals also propagate recurrently between the action layer and the hidden layer (as happens in ERNA).

3 Training procedure

In our experiments, ERNA and the other three networks were trained with the resilient propagation algorithm (RPROP) developed by Riedmiller and Braun [5]. RPROP is a gradient-based training procedure that overcomes the disadvantages of gradient-descent techniques (slowness, blurred adaptivity, choice of learning parameters, etc.).

The gradient used by RPROP consists of partial derivatives of each network weight with respect to the (mean square) error between the actual output values and the target output values of the network. For the MLP the target values are directly derived from class information (connected / unconnected). For ERNA the calculation of targets is less trivial and will be discussed in subsection 3.1.

When the target values are known, the gradient for feed-forward networks can be calculated by repeated application of the chain rule, using standard backpropagation. For the recurrent networks several techniques can be applied for calculating the gradient [3]. In our experiments the gradient is calculated with backpropagation through time [9], which corresponds to performing standard backpropagation on the network unfolded in time.

The quality of the weight updates strongly depends on the generalization of the calculated gradient. Therefore, all training was done in batch. This means that the gradient was averaged over all training examples before performing the RPROP weight update.

3.1 Action values

The calculation of the gradient requires target values. Since ERNA has to control actions that do not directly perform classification, reinforcement learning is used. However, in reinforcement learning there are no pre-defined targets. Instead, incidental positive (reward) or negative (punishment) reinforcement signals constitute the teaching signal. The network must be trained to maximize the sum of the reinforcements for a complete sequence of actions. An appropriate reinforcement-learning method is Q-learning [8].

Q-learning is a method for learning state-action values. A state-action value, or Q-value, is the maximum expected sum of reinforcements that can be obtained from a given state when performing the associated action. For neural networks this means that for each possible action, the network has an associated action neuron that is trained to predict the Q-value. By definition the optimal Q-values must satisfy

$$Q(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{1}$$

where r_t is the immediate reward after executing action a_t in state s_t at time t, and γ is a discount factor for long-term consequences of actions. Although for finite sequences γ can be set exactly to 1, it is customary to use somewhat smaller values to favor quick results. However, if γ is chosen too low, the network tends to behave probabilistically.

The target function for Q-learning is directly derived from (1) in the form of

$$T(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) \tag{2}$$

for which Q' means that the estimation of the network is used instead of the optimal Q-value (which is unknown).

Although the targets calculated by formula (2) give reasonable results, convergence is usually quite slow. The reason is that long chains of actions delay learning from distant reinforcement signals. To overcome this problem Q(λ)-learning [4, 6, 8] can be used. The target function for Q(λ)-learning can be defined recursively as

$$T(s_t, a_t) = r_t + \gamma((1-\lambda) \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) + \lambda \max_{a_{t+1}} T(s_{t+1}, a_{t+1})) \tag{3}$$

in which λ is a weighting factor between 0 and 1 that determines the relative contribution of future reinforcements to the estimated target value. Since (3) uses the target value for the optimal action at the next iteration, it can only be applied in a chain of optimal actions. The execution of non-optimal actions during training, known as exploration, is necessary to ensure that optimal Q-values can be learned. Therefore, when at time $t+1$ a non-optimal action is executed (2) is used instead of (3). When a_t is a final action (usually classification) only the direct reinforcement signal r_t is used as target value.

4 The data set

Two binary-valued pixels are connected if both their values equal 1 and their distance is 1 (in our square images diagonal connections are not used). Furthermore, if pixel A is connected to B and B is connected to C, A is also connected to C.

For the experiments, square 4×4, 5×5, and 6×6 images were created. Images with the upper left pixel connected to the lower right pixel were labelled connected, all others were labelled unconnected.

The binary images were not generated completely at random, because on such data all networks perform almost optimally. The reason is that in 75% of the cases the class unconnected can be determined from the two crucial corner pixels alone (both must be 1 for being connected), and in addition, most bright images (more ones) are connected, and most dark images (more zeros) are unconnected.

We define a *minimal connected path* as a path of ones in which each pixel is crucial for connectedness (if any pixel is flipped to 0 the two corner pixels are no longer connected). To build a reasonably difficult data set, we started to generate the set of all minimal connected paths between the two corners. From this set a new set was generated by making copies and randomly flipping 15% of the pixels. For all images the two crucial corner pixels were set at 1. Duplicate images and images with less ones than the minimal path length (for connecting the two corners) were removed from the data set.

After applying this process for creating the 4×4, 5×5 and 6×6 images, the three data sets were split into independent training and test sets, all containing an equal number of unique positive and negative examples. The three sets contained 300, 1326, and 1826 training examples and 100, 440, and 608 test examples, respectively.

5 Experimental results

The experiments presented here compare the generalizing ability of ERNA with those of three other network architectures. The learning task is to determine binary connectedness between pixels. It is done by focusing on the relation between the number of training examples and the classification performance on an independent test set.

To prevent over-training, in each run a validation set was selected from the training examples and was used to find the optimal point for stopping the training. For the experiments with the 4×4 images 100 validation samples were used. For both the 5×5 and 6×6 images 200 validation samples were used.

Because of limited computational resources and the fact that reinforcement learning is much slower than supervised learning, the size of the hidden layer was tested exhaustively only for the MLP. For ERNA we established reasonable settings, for the architecture and training parameters, based on some initial tests on 4×4 images. Although these settings were kept the same for all our experiments, other settings might give better results especially for the larger images. The architecture so obtained was as follows. For the hidden layer 25 neurons, with tangent sigmoid transfer functions, were used. The area observed by the eye contained the pixel on the fixation point and the four direct neighbors, i.e., the observed area was within a Manhattan-distance of one pixel from the center point of focus. The output to the local memory was connected only to the center point. For each pixel three linear neurons were assigned to the local memory. The global memory contained 15 linear neurons. All memory and action neurons were initialized at 0. During training, actions were selected randomly 5% of the time. In the rest of the cases, the best action was selected directly 75% of the time, and 25% of the time actions were selected with a probability proportional to their estimated Q-value. During validation and testing of course no exploration was used. The maximum number of iterations per example was set at the number of pixels. Negative reinforcements of -1 were returned for moving the eye out of range, exceeding the maximum number of iterations or performing the wrong classification. A positive reinforcement of $+1$ was returned for the correct classification. The Q-learning parameters λ and γ were set at 0.3 and 0.97. All network weights were initialized with small random values. Training was performed in batch for a maximum of 5000 epochs.

The MLP was tested with hidden layers of 3, 6, 12, 25, 50 and 100 neurons. In each run, the optimal layer size was selected based on the performance on the validation set. Supervised training with RPROP was performed in batch for a maximum of 2000 epochs.

The stripped-down version of ERNA (the feed-forward network with eye) was kept similar to ERNA as much as possible. The sizes of the hidden layer and the eye were kept the same and training was done with exactly the same learning parameters.

The fully-connected recurrent network (without eye) also used a hidden layer of 25 neurons, and training was done with exactly the same learning parameters except that this network was allowed to train for a maximum of 10,000 epochs.

It should be noted that the eye was always initialized in the upper left corner. This is a reasonably good initialization point that significantly improves performance for small training sets. We did not consider this to be unfair for comparing the networks for two reasons: first, even when in our initial experiments the eye was initialized in one of the other two corners, ERNA always outperformed the other networks for larger training sets. And second, at least for the task discussed in this paper, finding a good starting position can be done automatically by trial and error based on the performance on the validation set (which is strongly correlated with the performance on the test set).

In figures 2, 3 and 4 the average performance is plotted for the four network architectures tested on the 4×4, 5×5 and 6×6 images, respectively. The horizontal axis shows the number of training examples, with logarithmic scaling. The vertical axis shows the fraction of correctly-classified test samples (1.0 for perfect classification, 0.5 for pure guessing).

The plots show that for all image sizes both ERNA and the stripped-down version of ERNA outperform the two networks without eye. Moreover, we can see that the recurrent connections are only useful for ERNA,

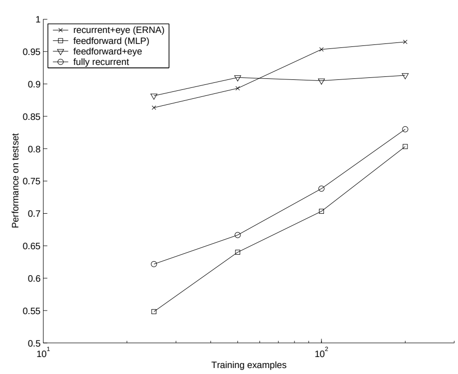

Figure 2: Classification of 4×4 images

and then only when sufficient training examples are available.

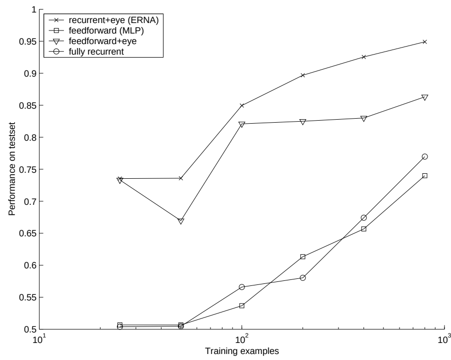

Figure 3: Classification of 5×5 images

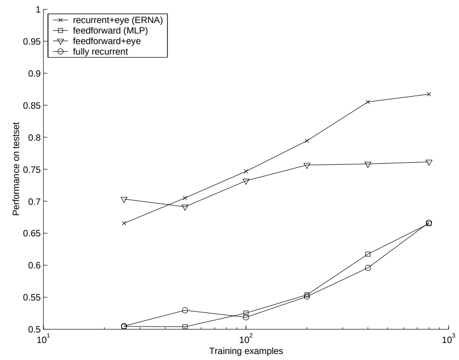

Figure 4: Classification of 6×6 images

6 Conclusions and future research

Recognizing connectedness with neural networks is a problem with a history dating back to Minsky and Papert [2]. Their results led several people to believe that MLPs would not be able to learn connectedness. In this paper it is shown that the problem of binary connectedness between two pixels can be learned from examples. Our experiments show that ERNA greatly improves generalization compared to both the MLP and the fully-connected recurrent network. However, since our experiments also showed that some generalization can even be expected from the MLP, this network still might be a good choice in the case of huge training sets.

The experiments presented in this paper seem to confirm our intuitive idea that an eye-like input structure can facilitate learning by automatically incorporating knowledge about the topological ordering of pixels into the network architecture. For eye-based network architectures, which control more actions than just the classification, the use of recurrent connections is important. Although our experiments show that the fully-connected recurrent network (without eye) does not benefit from its recurrent connections, this should be verified for other sizes of the hidden layer.

Training recurrent neural networks with simple gradient-descent usually requires a long time to converge to reasonable solutions. In some initial experiments we found RPROP to behave 2 to 20 times faster than standard gradient-descent. It would therefore be interesting to see how RPROP performs on other tasks.

While some initial experiments have been done, ERNA still has many parameters to be tuned. For instance, the number of neurons assigned to the eye, their distribution over the field of view, the sizes of the local memory, global memory and the hidden layer, possible trade-offs between these numbers, and the effect of recurrently linking the action neurons should be studied more thoroughly.

Although in this paper ERNA was only tested on one specific task, the architecture in principle supports any kind of image classification. Our future research will especially focus on determining features that characterize local properties and neighborhood relations in the game of Go. However, ERNA should also be tested on tasks like image segmentation (which is essentially the classification of single pixels) and large-scale image classification.

An interesting point about the new architecture is the fact that ERNA operates independently of the image size. Future research should give insight into the question how well this feature can be exploited when scaling up to larger images.

Another line of future research we consider is to see whether it is possible to extent Wang's [7] or some closely related type of cellular network so that it can be trained from examples, which is essential for comparison to both the MLP and ERNA.

Acknowledgments

We are grateful to Eric Postma and Levente Kocsis for useful discussions and comments on earlier drafts of this paper.

References

[1] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[2] M.L. Minsky and S.A. Papert. *Perceptrons: An introduction to computational geometry*. MIT Press, Cambridge, MA, 1969. Expanded Edition, 1988.

[3] B.A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995.

[4] J. Peng and R. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22:283–290, 1996.

[5] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation: the rprop algorithm. In *Proceedings of the IEEE Int. Conf. on Neural Networks (ICNN)*, pages 586–591, 1993.

[6] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[7] D.L. Wang. On connectedness: a solution based on oscillatory correlation. *Neural Computation*, 12:131–139, 2000.

[8] C.J.C.H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

[9] P.J. Werbos. Backpropagation through time; what it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560, 1990.