Closed-Loop Object Recognition Using Reinforcement Learning

Jing Peng*and Bir Bhanu
College of Engineering
University of California
Riverside, CA 92521
Email: {jp,bhanu}@vislab.ucr.edu

Abstract

Current computer vision systems whose basic methodology is open-loop or filter *type typically use image segmentation followed by object recognition algorithms. These systems are not robust for most real-world applications. In contrast, the system presented here achieves robust performance by using reinforcement learning to induce a mapping from input images to corresponding segmentation parameters. This is accomplished by using the confidence level of model matching as a reinforcement signal for a* team of learning automata *to search for segmentation parameters during training. The use of the recognition algorithm as part of the evaluation function for image segmentation gives rise to significant improvement of the system performance by automatic generation of recognition strategies. The system is verified through experiments on sequences of color images with varying external conditions.*

1 Introduction

Image segmentation, feature extraction and model matching are the key building blocks of a computer vision system for model-based object recognition [5, 10]. The tasks performed by these building blocks are characterized as the low (segmentation), intermediate (feature extraction) and high (model matching) levels of computer vision. While the goal of image segmentation is to extract meaningful objects from an image, model matching uses a representation such as shape features obtained at the intermediate level for recognition. It requires explicit shape models of the object to be recognized. There is an abstraction of image information as we move from low to high levels and the processing becomes more knowledge based or goal directed.

Although there is an abundance of proposed computer vision algorithms for object recognition, there have been hardly any systems that achieve good performance for practical applications, since most such systems do not adapt to changing environments [1]. The main difficulties, typically associated with systems that are mostly open-loop or *filter* type, can be characterized as follows. First, the fixed set of parameters used in various vision algorithms often leads to ungraceful degradation in performance. Second, the image segmentation, feature extraction and selection are generally carried out as preprocessing steps to object recognition algorithms for model matching. These steps totally ignore the effects of the earlier results such as image segmentation on the future performance of the recognition algorithm. Third, generally the criteria used for segmentation and feature extraction require elaborate human designs. When the conditions for which they are designed are changed slightly, these algorithms fail. Finally, object recognition is a process of making sequences of decisions, i.e., applying various image analysis algorithms. Often the usefulness of a decision or the results of an individual algorithm can *only* be determined by the final outcome (e.g. matching confidence) of the recognition process. This is also known as a "vision-complete" problem [4], i.e., one cannot really assign labels to the image without the knowledge of which parts of the image correspond to what objects.

This paper presents a learning based vision framework in which these problems can be adequately addressed. The underlying theory is that any recognition system whose decision criteria for image segmentation and feature extraction, etc. are developed autonomously from the outcome of the final recognition might transcend all these problems. Our system accomplishes this by incorporating a reinforcement learning mechanism that takes the output of the recognition algorithm and uses it as a feedback to influence the performance of the segmentation process. As a result, the recognition performance can be significantly improved over time with this method.

This work is most closely related to the work by Bhanu et al. [2], in which they describe a system that uses genetic and hybrid algorithms for learning segmentation parameters. However, the recognition algorithm is not part of the evaluation function for segmentation in their system. The genetic or hybrid algorithms simply search for a set of parameters that optimize a prespecified evaluation function that may not best serve the overall goal of robust object recognition. Furthermore, the papers assume that the location of the object in the image is known for their surveillance application. In our work, we do not make such an assumption. We use an explicit geometric model of an object, represented by its polygonal approximation, to recognize it in the image.

*Current address: Machine Vision Dept., Amherst Systems, Inc., 30 Wilson Road, Buffalo, NY 14221

1063-6919/96 $5.00 © 1996 IEEE

2 Learning System for Segmentation Parameter Estimation

Consider the problem of recognizing an object in an input image, assuming that the model of the object is given and that the precise location of the object in the image is unknown. The conventional method for the recognition problem is to first segment the input image, then extract and select appropriate features from the segmented image, and finally perform model matching using these features. If we assume that the matching algorithm produces a real-valued output indicating the degree of success upon its completion, then it is natural to use this real-valued output as feedback to influence the performance of segmentation and feature extraction to bring about the system's earlier decisions favorable for more accurate model matching. The rest of this paper describes a reinforcement learning-based vision system to achieve just that.

2.1 Learning to Segment images

Our current investigation into reinforcement learning-based vision systems is focused on the problem of learning to segment images. An important characteristic of our approach is that the segmentation process takes into account the biases of the recognition algorithm to develop its own decision strategies. A consequence of this is that the effective search space of segmentation parameters can be dramatically reduced. As a result, more accurate and efficient segmentation, and thus recognition performance, can be expected.

2.2 Image Segmentation

We begin with image segmentation [6] because it is an extremely important and difficult low-level task. All subsequent interpretation tasks including object detection, feature extraction, object recognition and classification rely heavily on the quality of the segmentation process. The difficulty arises for image segmentation when only local image properties are used to define the region-of-interest for each individual object. It is known [1, 3] that correct localization may not always be possible. Thus, a good image segmentation cannot be done by grouping parts with similar image properties in a purely bottom-up fashion. Difficulties also arise when segmentation performance needs to be adapted to the changes in image quality, which is affected by variations in environmental conditions, imaging devices, lighting, etc.

2.3 Our Approach

Each combination of segmentation parameters produces, for a given input, an unique segmentation image from which a confidence level of model matching can be computed. The simplest way to acquire high pay-off parameter combinations is through trial and error. That is, generate a combination of parameters, compute the matching confidence, generate another combination of parameters, and so on, until the confidence level has exceeded a given threshold. Better yet, if a well-defined evaluation function over the segmentation space is available, then local gradient methods, such as hill-climbers, suffice. While the trial-and-error methods suffer from excessive demand for computational resources, such as time and space, the gradient methods suffer from the unrealistic requirement for an evaluation function. In contrast, reinforcement learning performs trials and errors, yet does not demand excessive computational resources; it performs hill-climbing in a statistical sense, yet does not require an evaluation function. In addition, it can generalize over unseen images as we shall see later. It thus fits our goal nicely here.

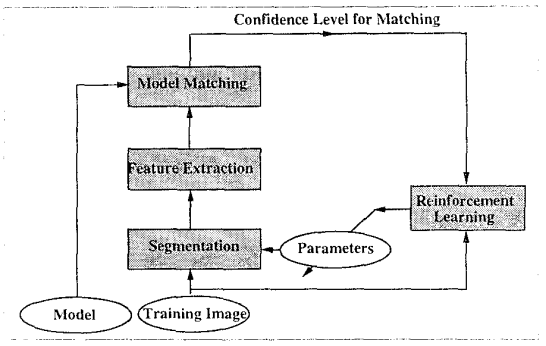

Figure 1: **Reinforcement learning-based multi-level system for object recognition.**

Figure 1 depicts the conceptual diagram of our reinforcement learning-based object recognition system that addresses the parameter selection problem encountered in the image segmentation task by using the recognition algorithm itself as part of the evaluation function for image segmentation.

3 Reinforcement Learning

Reinforcement learning is an important machine learning paradigm. It is a framework for learning to make sequences of decisions in an environment. In this framework, a learning system is given, at each time step, inputs describing its environment. The system then makes a decision based on these inputs, thereby causing the environment to deliver to the system a reinforcement. The value of this reinforcement depends on the environmental state, the system's decision, and possibly random disturbances. In general, reinforcement measuring the consequences of a decision can emerge at a multitude of times after a decision is made. A distinction can be made between associative and non-associative reinforcement learning. In the non-associative paradigm, reinforcement is the only information the system receives from its environment. Whereas, in the associative paradigm, the system receives input information that indicates the state of its environment as well as reinforcement. In such learning systems, a "state" is a unique representation of all previous inputs to a system. In computer vision, this state information corresponds to current input image. Our object recognition applications require us to take into account the changes appearing in the input images. The objective of the system is to

select sequences of decisions to maximize the sum of future reinforcement (possibly discounted) over time. It is interesting to note that for a given state an associative reinforcement learning problem becomes a non-associative learning problem.

3.1 Connectionist Reinforcement Learning

The particular class of reinforcement learning algorithms employed in our object recognition system is the connectionist REINFORCE algorithm [11], where units in such a network (depicted by the picture on the left in Figure 2) are *Bernoulli quasilinear units*, in that the output of such a unit is either 0 or 1, determined stochastically using the Bernoulli distribution with parameter $p = f(s)$, where f is the logistic function, $f(s) = 1/(1+\exp(-s))$ and $s = \sum_i w_i x_i$ is the usual weighted summation of input values to that unit. For such a unit, p represents its probability of choosing 1 as its output value. The picture on the right in Figure 2 depicts the ith unit.

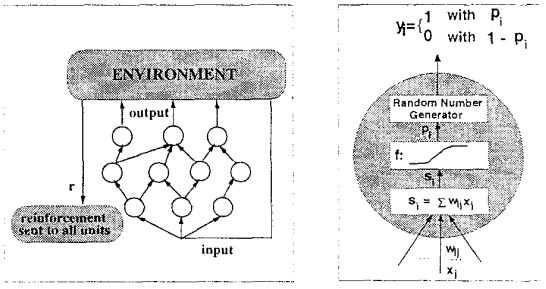

Figure 2: **Left: Connectionist reinforcement learning system. Right: Bernoulli quasilinear unit.**

In the general reinforcement learning paradigm, the network generates an output pattern and the environment responds by providing the reinforcement r as its evaluation of that output pattern, which is then used to drive the weight changes according to the particular reinforcement learning algorithm being used by the network. For the Bernoulli quasilinear units used in this research, the REINFORCE algorithm prescribes weight increments equal to

$$\Delta w_{ij} = \alpha(r - b)(y_i - p_i)x_j \qquad (1)$$

where α is a positive learning rate, b serves as a *reinforcement baseline*, x_j is the input to each Bernoulli unit, y_i is the output of the ith Bernoulli unit, and p_i is an internal parameter to a Bernoulli random number generator. Note that i takes values from 1 to n, and j from 1 to m, where n and m are the number of the units in the network and the number of input features, respectively.

It can be shown [11] that, regardless of how b is computed, whenever it does not depend on the immediately received reinforcement value r, and when r is sent to all the units in the network, such an algorithm satisfies

$$E\{\Delta \mathbf{W}|\mathbf{W}\} = \alpha \nabla_{\mathbf{W}} E\{r|\mathbf{W}\} \qquad (2)$$

where E denotes the expectation operator, $\mathbf{W}$ represents the weight matrix of the network, and $\Delta\mathbf{W}$ is the change of the weight matrix. A reinforcement learning algorithm satisfying (2) has the convergence property that the algorithm statistically climbs the gradient of expected reinforcement in weight space. For adapting parameters of the segmentation algorithm, it means that the segmentation parameters change in the direction along which the expected matching confidence increases. The next two subsections describe the particular network and the algorithm used in this paper.

3.2 The Team Architecture

We use a very simple form of a trial generating network in which all of the units are output units and there are no interconnections between them. This degenerate class of network corresponds to what is called a *team* of automata in the literature on stochastic learning automata. We, therefore, call these networks as *teams of Bernoulli quasilinear units*. Each segmentation parameter is represented by a set of Bernoulli quasilinear units and the output of each unit is binary as we have described earlier.

The weights w_{ij} are adjusted according to the particular learning algorithm used. We note that when $s_i = 0$, and hence $p_i = 0.5$, the unit is equally likely to pick y_i either 0 or 1, while increasing s_i makes a 1 more likely. Adjusting the weights in a team of Bernoulli quasilinear units is thus tantamount to adjusting the probabilities (p_i's) for individual units.

3.3 The Team Algorithm

The specific algorithm we used with the team architecture has the following form: At the tth time step, after generating output $\mathbf{y}(t)$ and receiving reinforcement $r(t)$, i.e., the confidence level indicating the matching result, increment each weight w_{ij} by

$$\Delta w_{ij}(t) = \alpha(r(t) - \overline{r}(t-1))(y_i(t) - \overline{y}_i(t-1))x_j - \delta w_{ij}(t)$$

where α, the learning rate, and δ, the weight decay rate, are parameters of the algorithm. The term $(r(t) - \overline{r}(t-1))$ is called the *reinforcement factor* and $(y_i(t) - \overline{y}_i(t-1))$ the *eligibility* of the weight w_{ij} [11]. Note that this algorithm is a variant of the one described in equation (1), where b is replaced by $\overline{r}$ and p_i by $\overline{y}_i$.

$\overline{r}(t)$ is the exponentially weighted average, or *trace*, of prior reinforcement values $\overline{r}(t) = \gamma\overline{r}(t-1) + (1-\gamma)r(t)$ with $\overline{r}(0) = 0$. The trace parameter γ was set equal to 0.9 for all the experiments reported here. Similarly, $\overline{y}_i(t)$ is an average of past values of y_i computed by the same exponential weighting scheme used for $\overline{r}$. That is, $\overline{y}_i(t) = \gamma\overline{y}_i(t-1) + (1-\gamma)y_i(t)$. The use of weight decay is chosen as a simple heuristic method to force sustained exploration of the weight space since it was found that REINFORCE algorithms without weight decay always seemed to converge prematurely.

- LOOP:
  1. For each image i in the training set do
     (a) Compute matching confidence for image i: $CONFID_i$
     (b) $n_i = MAXCONFID - CONFID_i$
     (c) If $\sum_i n_i$ is 0, then terminate.
     (d) $proportion_i = \frac{n_i}{\sum_i n_i}$
  2. $rr = 0$ (rr: average matching confidence)
  3. For k = 1 to n do
     (a) Sample image i according to $proportion_i$
     (b) Segment image i using current segmentation parameters
     (c) Perform noise clean up
     (d) Get segmented regions (also called blobs or connected components)
     (e) Perform feature extraction for each blob to obtain token sets
     (f) Compute matching of each token set against model, return the highest confidence r
     (g) Obtain new parameters for the segmentation algorithm using r as reinforcement
     (h) $rr = rr + r$
- UNTIL number of iterations is equal to N or $rr/n \geq R_{th}$ (Threshold)

Figure 3: **Main Steps of the Proportional Training Algorithm.**

3.4 Implementation of the Algorithm

We use a proportional training algorithm where, instead of looping through every image in the training set, we samples images proportional to the level of matching confidence the current system achieves. That is, the lower the matching confidence the system gets on an image, the more likely the image will be sampled. In this way training is focused on those images having the lowest matching confidence, and thus faster performance improvement can be achieved. Figure 3 shows the main steps of the proportional training algorithm, where $MAXCONFID$ (=1 in this paper) is the maximum confidence level the system can achieve, i.e., when a perfect matching occurs, n is the number of images in the training set, and N and R_{th} are input parameters to the algorithm.

4 Experimental Demonstration

We have tested our algorithm on a variety of color images.

The *Phoenix* algorithm [8] was chosen as the image segmentation component in our system because it is a well-known method for the segmentation of color images with a number of adjustable parameters. *Phoenix* works by splitting regions using a histogram for color features. Note that any segmentation algorithm with adjustable parameters can be used in our approach.

The *Phoenix* algorithm has a total of fourteen adjustable parameters. The four most critical ones that affect the overall results of the segmentation process are used in learning. These parameters are *Hsmooth*, *Maxmin*, *Splitmin*, and *Height*. The team algorithm searches for a combination of these parameters that will give rise to a segmentation from which the best recognition can be achieved. The ranges for each of these parameters are the same as those used in [2]. The resulting search space is about one million sample points.

Each of the *Phoenix* parameters is represented using 5 bit Gray code that has the advantage over simple binary code in that only one bit changes between representations of two consecutive numbers. Since there are 4 parameters, we have a total of 20 Bernoulli quasi-linear units and each parameter corresponds to the outputs of 5 units.

The feature extraction consists of finding polygon approximation tokens for each of the regions obtained after image segmentation. The polygon approximation is obtained using a split and merge technique [3] that has a fixed set of parameters.

Object recognition employs a cluster-structure matching algorithm [3] that is based on the clustering of translational and rotational transformations between the object and the model for recognizing 2-D and 3-D objects. It outputs a real number indicating the confidence level of the matching process. This confidence level is then used as a reinforcement signal to drive the team algorithm.

The initial parameter values for the *Phoenix* algorithm are chosen at random. We expect, however, that the good starting values of the segmentation parameters affect the convergence rate.

4.1 Results

The segmentation task whose experimental results we report here is a sequence of indoor color images (160 by 120 pixels) having simple geometric objects with varying lighting and motion conditions. These images are shown in Figure 4, where, from left to right, images are moving away from the camera, and within

each column, lighting conditions deteriorate from top to bottom. The training set consists of the images 4(c), (h), (k), and (l) (randomly selected), whereas the testing data come from the rest of the images (8 images). The objective of the task is to find a set of *Phoenix*'s parameters that give rise to a segmentation of the input image that, after appropriate feature extraction, will result in the recognition of the triangular object. The model of the triangular object is represented by a polygonal approximation of its shape. The threshold for matching confidence in this case was set to 0.8. The learning rate parameter α was set to 0.008 in all the experiments. Note that, unlike previous work on image segmentation, the criteria measuring image segmentation quality here are completely determined by the matching algorithm itself.

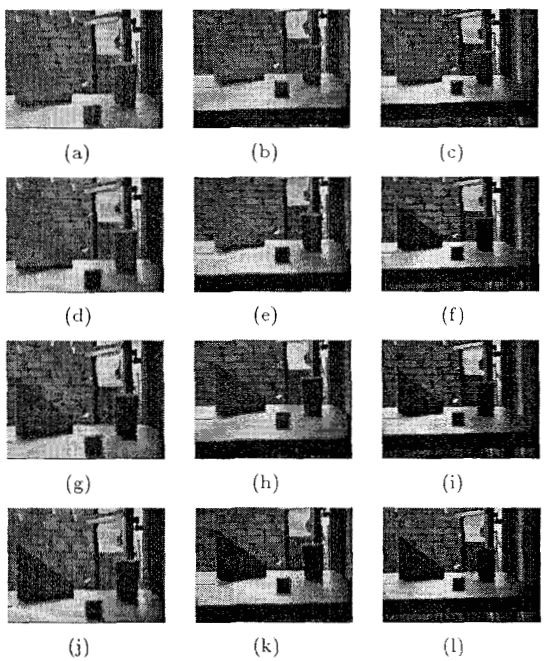

Figure 4: **Twelve color images having simple geometric objects.**

Each unit in the team network has a total of 8 input weights. In the first experiment each of the input weights takes an average grey value of input on a 60 by 40 neighborhood on the input image plane of 120 by 160 pixels. This input image is the luminance image of the corresponding color image. Note that in this experiment the average is normalized to lie between -1 and 1. For weights that are adjacent in a unit, their receptive fields are at least 40 pixels apart in the input image. Thus, the input image is undersampled, which in turn greatly reduces the number of weights in the network.

In the second experiment each input image is projected onto the subspace spanned by the eight eigenvectors corresponding to eight largest eigen values of the original (luminance) image vector space (120 by 160 pixels). These inputs are normalized to lie between -1 and 1. Our goal is to see which method can offer better performance. It turns out that the second method performed slightly better than the first one, as can be seen below (Figures 5 and 6). Note that, unless stated otherwise, all the figures below are obtained under the condition that the system takes input from the subspace spanned by the first 8 major axes corresponding to the eight largest eigenvalues.

Figure 5 shows the segmentation performance (both training and testing) of the *Phoenix* algorithm with learned parameters on the images shown in Figure 4. The training results in Figure 5 are obtained after a mean value (over 5 runs) of 250 passes through the training data. Figure 6 shows the average confidence (over 5 runs) received by the two methods over time during training. Each run consists of a sequence of trials until the average confidence level has exceeded 0.8. The threshold (0.8) serves our purpose well here since it is sufficient to demonstrate the effect of learning for object recognition.

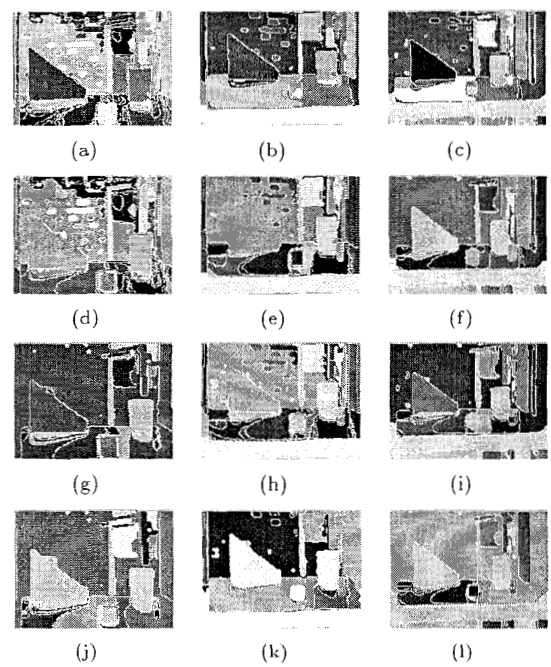

Figure 5: **Segmentation performance of the *Phoenix* algorithm with learned parameters.**

When the segmentation parameters obtained after training were applied to the images in the testing set, recognition results for all the images except 4(f) are acceptable. However, if we include image 4(f) in the

training set and allow learning to continue, experiments have been performed that show that successful recognition can be achieved for all testing images in much less time (less than 50%) compared to the time taken for training on the original training data.

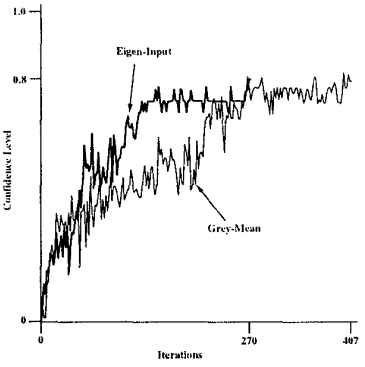

Figure 6: **Average confidence received by the two methods over time during training.**

In comparison, the *Phoenix* algorithm with a default parameter setting was also run on the same images. Figure 7 illustrates the samples of the *Phoenix* performance. This default parameter setting resulted in a total matching failure.

We have also performed a variety of experiments on outdoor images.

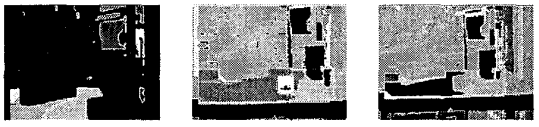

Figure 7: **Samples of segmentation performance of the *Phoenix* algorithm with default parameters on the images (Figures, 4(a), 4(b) and 4(c), respectively).**

5 Conclusions

The key contribution of the paper is the general framework for the usage of reinforcement learning in a model-based object recognition system. Our investigation into reinforcement learning-based object recognition shows conclusively that a robust and adaptive system can be developed that automatically determines the criteria for segmentation of the input images and selects useful features that result in a system with high recognition accuracy when applied to new unseen images. Note that the performance of any learning-based computer vision system depends on the vision algorithms that are used, e.g., the recursive region-splitting *Phoenix* algorithm used in this paper for the segmentation of color images.

We are also exploring other reinforcement learning based techniques for segmentation, feature extraction and model-matching for robust object recognition in real-world environments [9].

Acknowledgements

This work was supported by ARPA/AFOSR grants F49620-93-1-0624 and F49620-95-1-042 at the University of California, Riverside. The contents of the information do not necessarily reflect the position or the policy of the Government.

References

[1] B. Bhanu and T. Jones, "Image understanding research for automatic target recognition," in *Proc. of DARPA Image Understanding Workshop*, pp. 249-259, 1992.

[2] B. Bhanu and S. Lee, *Genetic Learning for Adaptive Image Segmentation.* Boston MA: Kluwer Academic Publishers, 1994.

[3] B. Bhanu and J. Ming, "Recognition of occluded objects: A cluster-structure algorithm," *Pattern Recognition 20(2)*, pp. 199-211, 1987.

[4] D. Chapman, "Intermediate vision: Architecture, implementation, and use," *Cognitive Science 16*, pp. 491-537, 1992.

[5] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Computing Surveys*, pp. 67-108, March 1994.

[6] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer Vision, Graphics and Image Processing 29*, pp. 100-132, 1985.

[7] H. G. John, R. Kohavi and K. Pfleger, "Irrelevant features and the subet selection problem," in *Proc. of the Eleventh International Conf. on Machine Learning*, pp. 121-129, 1994.

[8] K. Laws, "The *Phoenix* image segmentation system: Description and evaluation," SRI International Tech. Rep. TR289, December 1982.

[9] J. Peng and B. Bhanu, "Delayed reinforcement learning for closed-loop object recognition," in *Proc. of ARPA Image Understanding Workshop*, pp 1429-1435, February 1996.

[10] P. Suetens, P. Fua and A. J. Hanson, "Computational strategies for object recognition," *ACM Computing Surveys 24(1)*, pp. 5-59, 1992.

[11] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning 8*, pp. 229-256, 1992.