Pomerleau, D.A. (1991) Efficient **Training of Artificial Neural Networks for Autonomous Navigation. In *Neural Computation3:1* pp. *88-97.***

Efficient Training of Artificial Neural Networks for Autonomous Navigation

Dean A. Pomerleau
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The ALVINN (Autonomous Land Vehicle In a Neural Network) project addresses the problem of training artificial neural networks in real time to perform difficult perception tasks. ALVINN is a back-propagation network designed to drive the CMU Navlab, a modified Chevy van. This paper describes the training techniques which allow ALVINN to learn in under 5 minutes to autonomously control the Navlab by watching a human driver's reactions. Using these techniques ALVINN has been trained to drive in a variety of circumstances including single-lane paved and unpaved roads, and multi-lane lined and unlined roads, at speeds of up to 20 miles per hour

1 Introduction

Artificial neural networks *sometimes* require prohibitively long training times *and* large training data sets to learn interesting task. As a result, few attempts have been made to apply artificial neural networks to complex real-world perception problems. In addition, domains *such* as phoneme recognition [Waibel et al., 1988] and character recognition [LeCun et al., 1989] [Pawlicki et al., 1988] where connectionist techniques have been applied successfully, results have come only after careful preprocessing of the input to segment and label the training exemplars. In short, artificial neural networks have never before been successfully trained using sensor data in real time to perform a real-world perception task.

The ALVINN (Autonomous Land Vehicle In a Naval Network) system remedies this shortcoming. ALVINN is a back-propagation network designed to drive the CMU Navlab, a modified Chevy van (See Figure 1). Using real time training techniques, the *systan* quickly learns to autonomously control the Navlab by watching a human driver's reactions. ALVINN has *been* trained *to* drive in a variety *of* circumstances including single-lane paved and unpaved roads, and multi-lane lined and unlined roads, *at* speeds of *up* to 20 miles per hour.

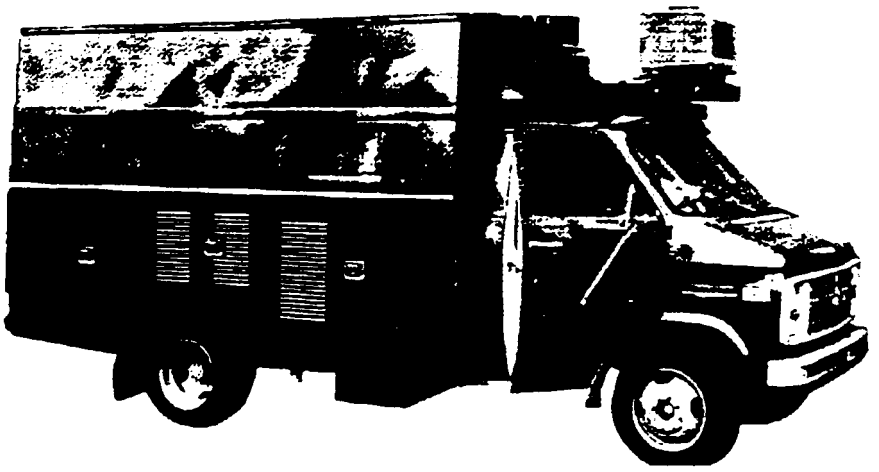

Figure 1: The CMU Navlab Autonomous Navigation Testbed

2 Network Architecture

ALVINN's current architecture consists of a single hidden layer back-propagation network (See **Figure** 2). The input layer of the network consists of a 30x32 unit "retina" onto which a video camera image is projected. Each of the 960 units in the input retina is fully connected to the hidden layer of 5 units. which is in turn fully connected to the output layer. The output layer consists of 30 units and is a Linear representation of the direction the vehicle should travel in order to keep the vehicle on the road. The centermost output *unit* represented the "travel straight ahead" condition, while units to the left and right of center represented successively sharper left and right turns.

To drive the Navlab, a video image from the onboard camera is reduced to a low-resolution 30x32 pixel image and injected into the input layer. After completing a forward pass through the network a steering command is read off the output layer. The steering direction dictated by the network is taken to be the center of mass of the "hill" of activation surrounding the output *unit* with the highest activation level. Using the center of mass of activation instead of the most active output unit when determining the direction to *steer* permits finer steering corrections, and hence improves ALVINN's driving accuracy.

3 Training

To train ALVINN, the network is presented with road images as input and the corresponding correct steering direction as the desired output. The weights in the network are altered using the back-propagation algorithm so the network's output more closely

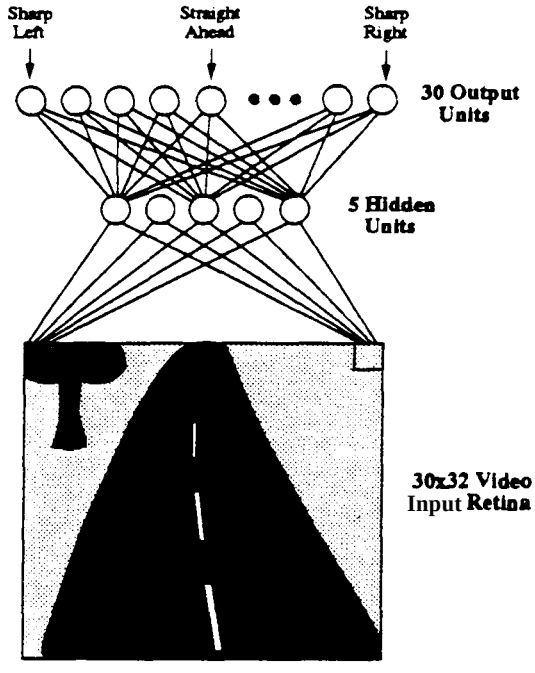

Figure 2: ALVINN Architecture

corresponds to the correct steering direction. The only modifications to the standard back-propagation algorithm itself used in this work are a weight change momentum factor which is steadily increased during training, and a learning rate constant for each weight which is scaled by the fan-in of the unit to which the weight projects. ALVINN's ability to learn quickly results from the output representation and the exemplar presentation scheme to be discussed next.

Instead of training the network to activate only a single output unit. ALVINN is trained to produce a gaussian distribution of activation centered around the steering direction which will keep the vehicle centered on the road. As in the decode stage, this steering direction may fall between the directions represented by two output units. The following approximation to a gaussian equation is used to precisely interpolate the correct output activation levels:

$$x_i = e^{-\frac{d_i^2}{10}}$$

where x_i represents the desired activation level for unit i and d_i is the ith unit's distance from the correct steering direction point along the output vector, which may fall between two output units. The constant 10 in the above equation is an empirically-determined scale factor that determines over how many output units the gaussian should spread

As an example, consider the situation in which the correct steering direction falls halfway between the steering directions represented by output units j and $j + 1$. Using the above equation, the desired output activation levels for the units successively farther to the left and right of the correct steering direction point halfway between units j and $j + 1$ will fall off rapidly with the values 0.98, 0.80, 0.54, 0.29, 0.13, 0.04, 0.01, etc.

This gaussian desired output vector can be thought of as representing the probability density function for the correct steering direction, in which a unit's probability of being correct decreases with distance from the gaussian's center. By requiring the network to produce a probability distribution as output, instead of a "one of N" classification, the learning task is made easier since slightly different road images require the network to respond with only slightly different output vectors. This is in contrast to the highly non-linear output requirement of the "one of N" representation in which the network must completely change its output vector (from having one unit on and the rest off to having a different unit on and the rest off) on the basis of fine distinctions between slightly shifted road scenes.

3.1 Original Training Scheme

The source of training data has evolved substantially over the course of the project. Training was originally performed using simulated road images designed to portray roads under a wide variety of weather and lighting conditions. The network was repeatly presented with 1200 synthetic road scenes along with the corresponding correct output vector, while the weights between units in the network were adjusted with the back-propagation algorithm, using the Warp systolic array supercomputer [Pomerleau et. al., 19881. The network required between 30 and 40 presentations of these 1200 synthetic road images in order to develop a representation capable of accurately driving over the single-lane Navlab test road. Once trained, the network was able to drive the Navlab at up to 1.8 meters per second (3.5 mph) along a 400 meter path through a wooded area of the CMU campus under a variety of weather conditions including snowy, rainy, sunny and cloudy situations.

Despite its apparent success, this training paradigm had serious drawbacks. From a purely logistical standpoint, generating the synthetic road scenes was quite time consuming, requiring approximately 6 hours of Sun-4 CPU time. Once the road scenes were generated, training the network required an additional 45 minutes of Warp computation time. In addition, differences between the synthetic road images on which the network was trained and the real images on which the network was tested often resulted in poor performance in real driving situations. For example, when the network was trained on synthetic road images which were less curved than the test road, the network would become confused when presented with a sharp curve during testing. Finally, while relatively effective at training the network to drive under the limited conditions of a single-lane road it quickly became apparent that extending the synthetic training paradigm to deal with more complex driving situations like multi-lane and off-road driving would require prohibitively complex synthetic training data generators.

3.2 Training "On-the-fly"

To deal with these problems, I have developed a scheme, called training "on-the-fly", which involves teaching the network to imitate a human driver under actual driving conditions. As a person drives the Navlab, back-propagation is performed to train the network using the current video camera image as input and the direction in which the person is currently steering as the desired output

There are two problems associated with this scheme. First, since the human driver normally steers the vehicle down the road center during training, the network will never be presented with situations where it must recover from errors, such as being slightly off the road center. When driving for itself, the network is bound to eventually make a mistake, so it must be prepared to recover by steering the vehicle back to the road center. The second problem is that naively training the network with only the current video image and steering direction runs the risk of overlearning from repetitive inputs. If the human driver takes the Navlab down a straight stretch of road during part of a training run, the network will be presented with a long sequence of similar images. This sustained lack of diversity in the training set will cause the network to "forget" what it had learned about driving on curved roads and instead learn to always steer straight ahead

Both problems associated with training on-the-fly stem from the fact that back-propagation requires training data which is representative of task to be learned. To provide the necessary variety of exemplars while still training on real data, the simple training on-the-fly scheme described above must be modified. Instead of presenting the network with only the current video image and steering direction, each original image is laterally shifted in software to create 14 additional images in which the vehicle appears to be shifted by various amounts relative to the road center (Set Figure 3). The shifting scheme maintains the correct perspective by shifting nearby pixels at the bottom of the image more than far away pixels at the top of the image as illustrated in Figure 3. The correct steering direction as dictated by the driver for the original image is altered for each of the shifted images in order to account for the extra lateral vehicle displacement in each. The use of shifted training exemplars eliminates the problem of the network never learning about situations from which recovery is required Also, overtraining on repetitive images is less of a problem, since the shifted training exemplars add variety to the training set. However as additional insurance against the effects of repetitive exemplars, the training set diversity is further increased by maintaining a buffer of recently encountered road scenes.

So in practice, training on-the-fly works as follows. A video image is digitized and reduced to the low resolution image required by the network. This single original image is shifted 7 times to the left and 7 times to the right in 025 meter increments to create 15 new training exemplars. Fifteen old exemplars from the current training set of 200 road scenes are chosen and replaced by the 15 new exemplars. The 15 exemplars to be replace in the training set are chosen in the following manner. The 10 tokens in the training set with the lowest error are replaced in order to prevent the network from overlearning frequently encountered situations such as straight stretches

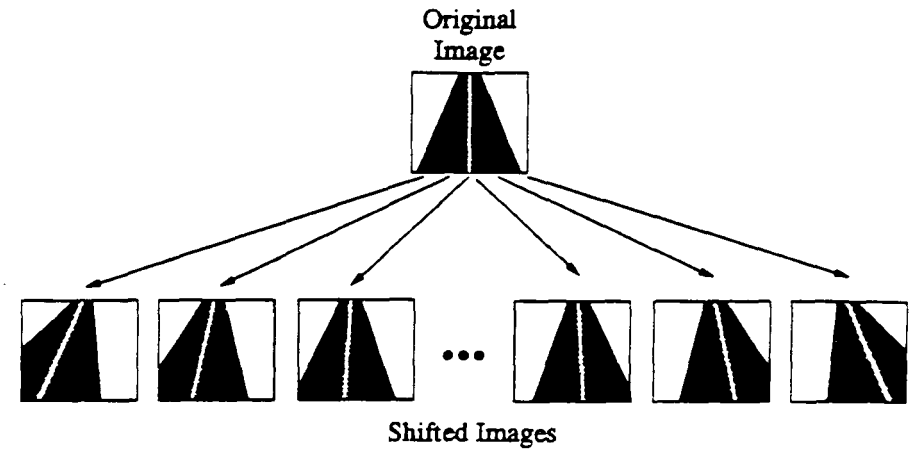

Figure 3: The single original video image is laterally *shifted* to create multiple training exemplars in which the vehicle appears to be at different locations relative to the road.

of road. The other 5 exemplars to be replaced are *chosen* randomly from the training set. This random replacement is done to prevent the training *set* from becoming filled with erroneous road images which the network is unable to correctly learn. These erroneous exemplars result from occasional momentary incorrect steering directions by the human driver.

After this replacement process, one forward and one backward sweep of the back-propagation algorithm is performed on these 200 exemplars to incrementally update the network's weights, and then the process is repeated. The network requires approximately 50 iterations through this *dig* —replace train *cycle* to learn to drive on the roads *that* have been tested. Running on a Sun-4, this takes approximately five minutes during which a person drives at about 4 miles per hour over the test road. *After* this training phase, not only *can* the network imitate the person's driving along the *same* stretch of road it can also generalize to drive along parts of the road it has never encountered, under a wide variety of weather conditions. In addition, since determining the steering direction from the input image merely involves a forward *sweep* through the network, the *system* is able to process 25 images per second, allowing it to drive at up to the Navlab's maximum *speed* of 20 miles per hour[1]. This is over twice as fast as any other sensor-based autonomous system has driven the Navlab [Crisman and Thorpe, 1990] [Kluge and Thorpe, 1990].

4 Discussion

The training on-the-fly scheme gives ALVINN a flexibility which is novel among autonomous navigation systems. It has allowed me to successfully train individual

[1] The Navlab has a hydraulic drive system which allows for very precise speed control, but which prevents the vehicle from driving over 20 miles per hour.

Figure 4: Video images taken on three of the test roads ALVINN has been trained to drive on. They are, from left to right, a single lane dirt access road, a single lane paved bicycle path, and a lined two-lane highway.

networks to drive in a variety of situations, including a single lane dirt access road a single-lane paved bicycle path, a two-lane suburban neighborhood street, and a lined two-lane highway (See Figure 4). ALVINN networks have driven in *each* of these situations for up to 1/2 mile, until reaching the end of the road or a difficult intersection. The development of a system for *each* of these domains using the "traditional approach" to autonomous navigation would require the programmer to 1) determine what features are important for the particular task, 2) program detectors (using statistical or symbolic techniques) for finding these important features and 3) develop an algorithm for determining which direction to *steer* from the location of the detected features.

An illustrative example of the traditional approach to autonomous navigation is the work of Dickmanns [Dickmann and Zapp, 1987] on high *speed* highway driving. Using specially designed hardware and software to track programmer chosen features such as the lines painted on the road and to use their position to determine how to steer, Dickmanns' system has achieved impressive results, driving at up to 60 miles per hour on the German autobahn. However to achieve these results in a hand-coded system. Dickmanns has had to sacrificed much in the way of generality. Dickmanns emphasizes accurate vehicle control in the limited domain of highway driving which in his words, "put relatively low requirements on image processing."

In contrast, ALVINN is able to *learn* for *each* new domain what image features are important, how to detect them and how to use their position to steer the vehicle. Analysis of the hidden unit representations developed in different driving situations suggest that the network develops detectors for the image features which correlate with the correct steering direction. When trained on multi-lane roads, the network develops hidden unit feature detectors for the lines painted on the road, while in single-lane driving situations, the detectors developed are sensitive to road edges and road shaped regions of similar intensity in the image. Figure 5 illustrates the evolution of the weights

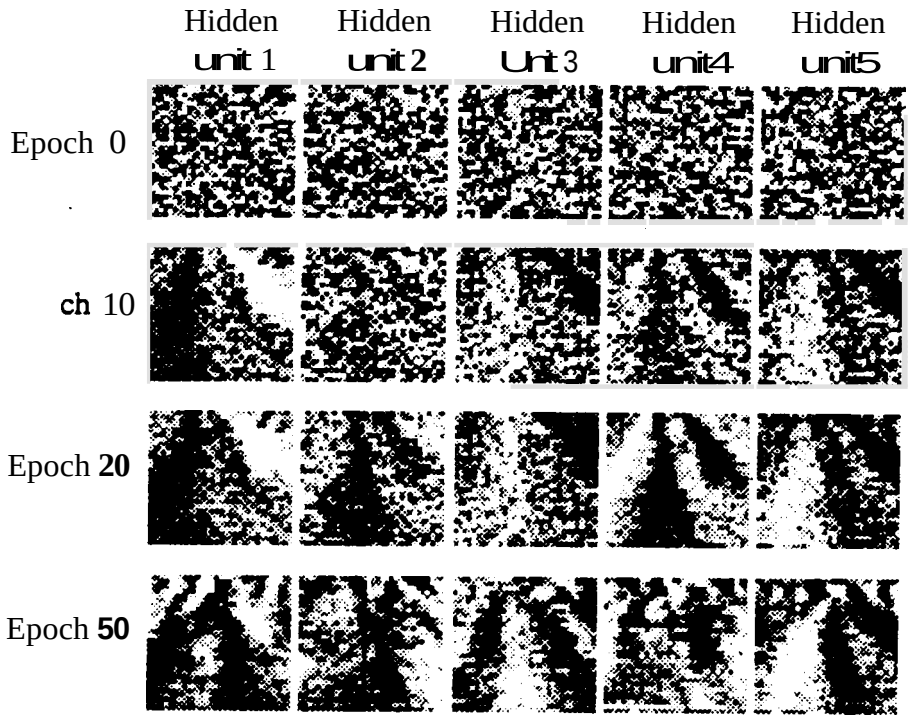

Figure 5: The weights projecting from the input retina to the 5 hidden units in an ALVINN network at four points during training on a lined two-lane highway. Black squares represent inhibitory weights and white square represent excitatory weights. The diagonal black and white bands on weights represent detectors for the yellow line down the center and the white line down the right edge of the road

projecting to the 5 hidden units in the network from the input retina during training on a lined two-lane highway. For a more detailed analysis of ALVINN's internal representations see [Pomerleau, 1990] [Pomerleau, 19891.

As a result of this flexibility, ALVINN has been able to drive in a wider variety of situations than any other autonomous navigation system. ALVINN has not yet achieved the impressive speed of Dickmanns' system on highway driving, but the primary barrier preventing faster driving is the Navlab's physical speed limitation. In fan. at 25 frames per second, ALVINN cycles twice as fast as Dickmanns' system. A new vehicle capable of traveling at normal highway speeds is currently being built at CMU which should allow ALVINN to drive significantly faster.

Other improvements I am developing include connectionist and non-connectionist techniques for combining networks trained for individual driving situations into a single system capable of handling them all. In addition, I am integrating symbolic knowledge sources capable of planning a route and maintaining the vehicle's position on a map into the system. These modules will allow ALVINN to make high level, goal oriented decisions such as which way to turn at intersections and when to stop at a predefined destination.

Acknowledgements

This work would not have been possible without the input and support provided by Dave Touretzky, John Hampshire, and especially Charles Thorpe, Omead Amidi, Jay Gowdy. Jill Crisman, James Frazier and rest of the CMU ALV group.

This research was supported by the Office of Naval Research under Contracts N00014-87-K-0385, N00014-87-K-0533 and N00014-86-K-0678, by National Science Foundation Grant EET-8716324, by the Defense Advanced Research Projects Agency (DOD) monitored by the Space and Naval Warfare Systems Command under Contract N00039-87-C-0251, and by the Strategic Computing Initiative of DARPA, through contracts DACA76-85-C-0019, DACA76-85-C-0003 and DACA76-85-C-0002, which are monitored by the U.S. Army Engineer Topographic Laboratories.

References

[Crisman and Thorpe, 1990] Crisman, J.D. and Thorpe C.E. (1990) Color vision for road following. In *Vision and Navigation: The CMU Navlab* Charles Thorpe, (Ed) Kluwer Academic Publishers.

[Dickmann and Zapp, 19871 Dickmanns, E.D. and Zapp, A. (1987) Autonomous high speed road vehicle guidance by computer vision. In Proceedings *of the 10th World Congress on Automatic Control, Vol. 4,* Munich, West Germany.

[LeCun et al., 19891 LeCun, *Y.,* Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D. (1989) Backpropagation applied to handwritten zip *code* recognition. Neural *Computation 1(4)* pp. 541-551.

[Kluge and Thorpe, 1990] Kluge, K and Thorpe C.E. (1990) *Explicit models for robot road following.* In *Vision and Navigation: The CMU Navlab* Charles Thorpe, (Ed.) Kluwer Academic Publishes.

[Pawlicki et al, 1988] Pawlicki, T.F., Lee, D.S. .Hull, JJ.. Srihari, S.N. (1988) Neural network models and their application to handwritten digit recognition. *Proceedings of IEEE International Conference on Neural Networks*, San Diego, CA.

[Pomerleau, 1990] Pomerleau, D A (1990) Neural network based autonomous navigation. In *Vision and Navigation: The CMU Navlab* Charles Thorpe, (Ed.) Kluwer Academic Publishers.

[Pomerleau, 1989] Pomerleau, D.A. (1989) ALVINN: An Autonomous Land Vehicle In a Neural Network. *Advances in Neural Information Processing Systems, 1.* D.S. Touretzky (ed.), Morgan Kaufmann.

[Pomerleau et. al., 19881 Pomerleau, D.A., Gusciora, G.L., Touretzky, D.S., and Kung, H.T. (1988) Neural network simulation at *Warp speed:* How We got 17 million connections per second. *Proceedings of IEEE International* Joinz *Conference on Neural Networks,* San Diego, CA.

[Thorpe et al., 19871 Thorpe, C., Herbert, M., Kanade, T., Shafer S. and the members of the Strategic Computing Vision Lab (1987) Vision and navigation for the Carnegie Mellon Navlab. *Annual Review of Computer Science Vol. II,* Ed Joseph Traub, Annual Reviews Inc., Palo Alto, CA.

[Waibel et al., 1988] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. (1988) Phoneme recognition: Neural Networks vs. Hidden Markov Models. *Proceedings from Int. Conf. on Acoustics, Speech and Signal Processing.* New York, New York.