Explanation-Based Learning for Mobile Robot Perception

Tom M. Mitchell
Joseph O'Sullivan
Sebastian Thrun*
School of Computer Science
Carnegie Mellon University

Abstract

Explanation-based neural network learning (EBNN) has recently been introduced as a method for reducing the amount of training data required for reliable generalization, by relying instead on approximate, previously learned knowledge. We present first experiments applying EBNN to the problem of learning object recognition for a mobile robot. In these experiments, a mobile robot traveling down a hallway corridor learns to recognize distant doors based on color camera images and sonar sensations. The previously learned knowledge corresponds to a neural network that recognizes nearby doors, and a second network that predicts the state of the world after travelling forward in the corridor. Experimental results show that EBNN is able to use this approximate prior knowledge to significantly reduce the number of training examples required to learn to recognize distant doors. We also present results of experiments in which networks learned by EBNN (e.g., "there is a door 2 meters ahead") are then used as background knowledge for learning subsequent functions (e.g., "there is a door 3 meters ahead").

1 Introduction

One crucial problem for robot learning is scaling up. Whereas various research has shown how robots can learn relatively simple strategies from very little initial knowledge (e.g., [5, 7, 6, 10]), theoretical and experimental results indicate that simple learning approaches will require unrealistic amounts of training data to learn significantly more complex functions. One approach to scaling up is to rely on human training (e.g., [3, 4, 5, 12]). We are interested here in methods by which the robot can use its own previously learned knowledge to reduce the need for new data in subsequent learning.

This paper presents early experiments with a robot architecture that utilizes previously learned knowledge about the effects of its actions, to reduce the difficulty of learning to recognize objects that it approaches as it travels through its environment. This architecture is based on explanation-based neural network learning (EBNN) [9, 16], an explanation-based learning algorithm that represents both prior knowledge and the current target function using neural networks. Initial experiments with EBNN have demonstrated its ability to learn robot control knowledge in simulated [9] and real-world robot domains [15]. Here we examine its ability to reduce training data required for learning robot perception skills. The results reported here show that EBNN is able to significantly reduce the amount of training data required for learning, when compared to purely inductive methods such as Backpropagation, when an appropriate domain theory is available.

2 Explanation-Based Neural Network Learning

Explanation-based neural network learning (EBNN) is a method for improving the learner's ability to generalize correctly from limited training data, by using approximate knowledge that has been previously learned by the system. Prior knowledge consists of a collection of previously learned neural networks (the *domain theory networks*), and the function to be learned is represented by an additional neural network (the *target network*). The EBNN algorithm uses the domain theory networks to analyze each training example, in order to extract information about the relevance of the different features of the example. This relevance information is then used, together with standard neural network induction, to constrain the weights of the target network. In this way, the target network is constrained both *inductively* by the training data, and *analytically* by the knowledge implicit in the domain theory.

*Sebastian Thrun is currently at Bonn University.

How can the domain theory networks be used to analyze training examples so that information is extracted in a form useful for refining the target network? The key lies in the TangentProp algorithm [14], an extension to Backpropagation [13] that is able to adjust network weights to minimize the error in both the *values* and the *derivatives* of the function represented by the target network. Consider some target function, F, and a training example, X, consisting is a vector of components x_i. The TangentProp algorithm iteratively adjusts the weights of the target network, NET, to minimize an error measure containing two terms. The first error term is the difference between the target function value, F(X), and the value predicted by the network, NET(X). The second error term is the difference between the partial derivatives of the target function, $\mathrm{dF(X)/d}x_i$, and the partial derivatives of the function represented by the target network, $\mathrm{dNET(X)/d}x_i$. EBNN obtains estimates of target values for $\mathrm{dF(X)/d}x_i$ by using its domain theory to analyze training examples.

Given some training example, X, with target function value F(X), EBNN processes the training example as summarized in Figure 1. First, EBNN *explains* the example; that is, it uses its domain theory to post-facto predict the value of F(X). This requires that it be possible to estimate F(X) by chaining together the domain theory networks (as illustrated in Figure 4). Second, EBNN *analyzes* this explanation to determine the relevance of each example feature: The weights and activations of the domain theory networks in the explanation are examined, in order to analytically extract the partial derivative of F(X) with respect to each component feature x_i of X. Notice this derivative describes the relevance of each feature, x_i: if the value of x_i has no influence on the value of F it will have a derivative of 0, whereas if the feature has a large derivative, it is highly relevant. In this way, information about the relevance of each example feature is extracted from the explanation. Finally, EBNN *refines* the target function network based on both the derivatives extracted from the explanation (analytical component of learning) and the training value F(X) (inductive component of learning). The TangentProp algorithm updates the weights of the target network to best fit both the target derivatives and target value. In order to minimize the impact of incorrect domain theories and explanations, the degree to which TangentProp seeks to fit the target derivatives is reduced by the degree to which the domain theory networks err in explaining the observed training value, F(X).

The analytical component of learning in EBNN is similar to earlier explanation-based learning methods based on symbolic representations [1, 8]: given a target function, F, the domain theory is used as an alternative method for computing F, and the dependencies are then extracted from the trace of this computation. The EBNN algorithm is described in greater detail in [9, 16], and summarized here in Figure 1.

For each training example:

- *Explain* how the training example satisfied the target function: Chain together the domain theory networks to predict the value of the target function for the training example.
- *Analyze* this explanation to determine feature relevance: Examine the weights and activations of the domain theory networks in the above explanation, to extract the partial derivatives of the explained target function value, with respect to each training example feature.
- *Refine* the target function network: Update the target network weights to fit both the observed target value (inductive component), and the target derivatives extracted from the explanation (analytical component).

Figure 1: **Summary of the EBNN Algorithm.**

3 The Problem

Here we consider the problem of learning to recognize objects that a robot is approaching as it travels through a hallway corridor. In this task, it is easy to recognize the object from specific predetermined nearby locations and orientations. However, learning to recognize the object from varying distances and orientations is much more difficult, and is our goal here. The experiments reported here involve learning functions of the form:

$$\text{Door-ahead}_d\text{: S} \rightarrow \{\text{True, False}\}$$

where $s \in S$ is the current robot sensor input (color vision and sonar readings), and Door-ahead$_d$ is True if and only if, after traveling forward d meters in the corridor, the robot will be adjacent to a door. Notice this learning task is a special case of a more general problem: learning to recognize states in which a particular action will reliably lead to future states with a arbitrary desired property.

Figure 2 shows the mobile robot, Xavier [11], used in these experiments. The sensors include the color camera visible on top of the robot, and a ring of sonar sensors visible approximately one third of the way down the robot torso. Although the camera is mounted on a pan/tilt unit controllable by the robot, in these experiments the camera was maintained in a single position pointed 15 degrees downward and 30 degrees to the right, as the robot traveled straight down the center of the corridor.

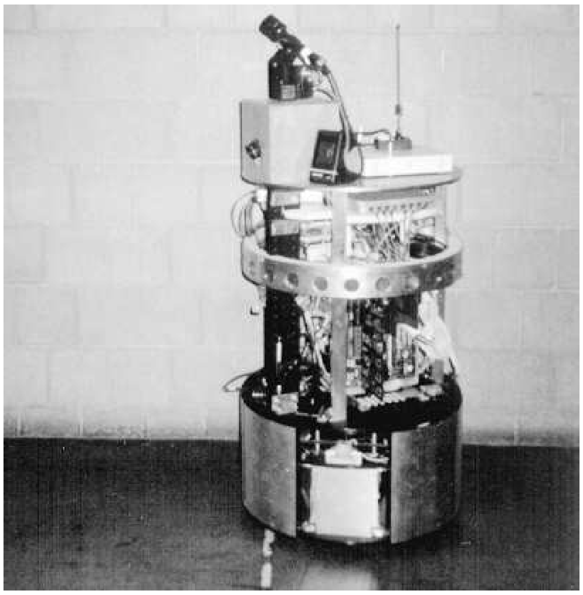

Figure 2: **The Learning Lab Robot, Xavier.**

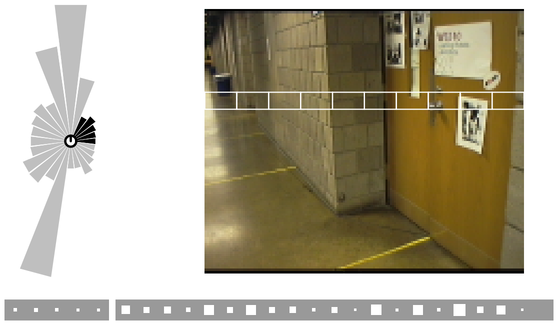

Figure 3: **Typical sensor input and its representation.** *This image was collected by Xavier while it proceeded forward down the center of a corridor. The circle of gray bars on the left indicate the 24 sonar distance echoes detected by Xavier's sonars at the same time the camera image was collected. The long sonar reading extending toward the top of the figure corresponds to the sonar facing forward down the corridor, and the shorter readings to the side reflect the position of the wall. This sonar and image data is subsampled and summarized in the vector of 25 values displayed at the bottom of the figure, as explained in the text.*

In order to limit the complexity of the sensor data for this learning task, Xavier summarizes its vision and sonar sensations in a vector of just 25 sensor readings, as shown in Figure 3. A row of 10 rectangular regions is extracted from the image, as highlighted in the figure. For each such rectangular region, two values are computed: the average red intensity and the average blue intensity. Thus, the image is reduced to a vector of just 20 intensity values, as displayed at the bottom right of the figure. Here, the two leftmost vector values correspond to the red and blue intensities of the leftmost rectangle, and so on. The area of the white box in the vector diagram indicates the magnitude of the corresponding vector value. The sonar data is also subsampled, to include only the 5 sonar readings shaded in dark gray in the sonar display. These are the sonars that face somewhat forward and toward the right, and therefore can be expected to contain relevant information for recognizing upcoming doorways on the right. The length of these sonar values (clipped to a maximum of 300 cm) is displayed in the row of 5 white boxes directly under the sonar display in the figure. Together, the 5 sonar readings, and 20 “”fat pixels” from the color image, form the summary of robot sensations used in these experiments.

4 Approach

Consider the task of learning the target function Door-ahead$_1$, whose value is True when there is a door 1 meter ahead. One way to learn this function is to inductively learn a neural network whose inputs are the vector of 25 sensor readings, and whose single output indicates the boolean value of the function. Given a significant number of training examples, such a network can be learned to a reasonable accuracy using the Backpropagation algorithm [13]. We are interested here in comparing such purely inductive methods with EBNN.

To apply EBNN to the task of learning Door-ahead$_1$, one must have prior knowledge capable of explaining observed training examples of this target function. We therefore allow Xavier to first learn two neural networks. The first represents the function

$$\text{Forward-1M: S} \rightarrow \text{S}$$

which, given the current state (represented by the vector of 25 sensor readings) predicts the sensor readings (next state) that can be expected if the robot drives 1 meter forward in the corridor. Notice that this function is independent of the particular perception learning task, and therefore the cost of learning it can be amortized over a variety of target functions.

The second domain theory network represents the function

$$\text{Door-Here?: S} \rightarrow \{\text{True, False}\}$$

which is True only for states in which the robot is directly adjacent to a door. Although the function Door-Here? is specific to recognizing doors, it is significantly easier to learn than the target function which requires recognizing doors at a distance.

Both of these functions are learned (approximately) from previous experience, in this case using the purely inductive Backpropagation algorithm. These two learned networks are then used by EBNN to explain individual training examples of the target function Door-ahead$_1$, as illustrated in Figure 4.

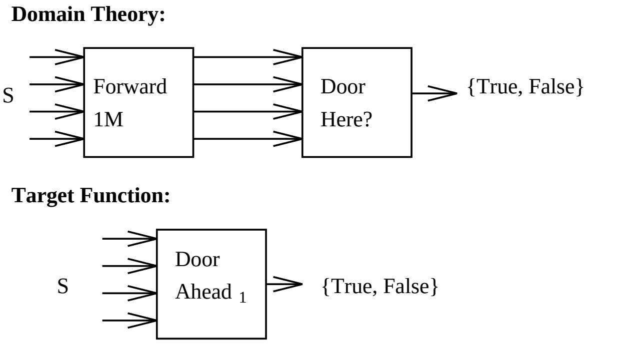

Figure 4: **A Typical EBNN Explanation.** *Each training example of the target function, Door-ahead$_1$, is explained in terms of the previously learned functions Forward-1M and Door-Here?.*

To illustrate the application of EBNN, consider analyzing a single training example. Figure 5 shows the explanation for a single training example of the target function – a particular state, S_{902}, for which there is no door one meter ahead. The training example state is represented by the vector of 25 sensor readings on the left of the diagram. The first domain theory network, applied to this state, predicts the state one meter forward. The second domain theory network is then applied to this predicted state, to estimate whether a door is present in this predicted next state. Taken together, these predictions form EBNN's explanation of the training example.

EBNN uses this explanation to determine the relevance of each feature of the training example, in order to guide its learning of the target function. As explained earlier, it uses the explanation to compute the partial derivative of the target function with respect to each training example feature. Irrelevant inputs (such as camera pixels that provide no information about doors on the right) will have partial derivatives of zero (provided the domain theory is correct!). In contrast, inputs with large derivatives are known to be highly relevant. The partial derivatives of the explanation output with respect to each input (i.e., each training example feature) are computed by applying the chain rule to the derivatives of each individual network in the explanation. The lines in Figure 5 indicate the sign and magnitude of the most significant derivatives for each individual network in the domain theory. Consider, for example, the rightmost network in the explanation: Door-Here?. The thickest solid line indicates that the output of this network increases rapidly with

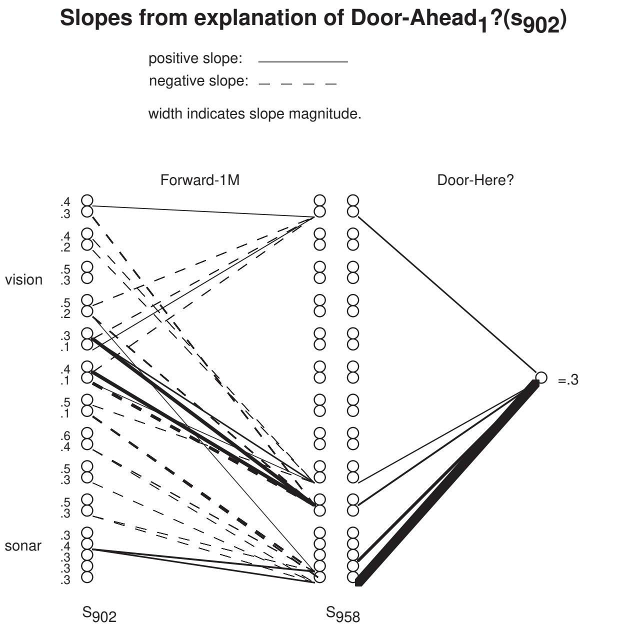

Figure 5: **Derivatives Extracted from an Explanation.** *Sensed state S_{902} is a training example for which target function Door-Ahead$_1$ is False. EBNN explains this value for the target function by using its previously learned neural networks Forward-1M, and Door-Here?. The derivatives of these individual networks, indicated by solid and dashed lines in the diagram, represent information about the presumed relevance of network input features to the network output. Derivatives of individual networks are combined to compute the derivative of the entire explanation. For example, the derivative of the output of Door-Here?, with respect to the second sonar value of S_{902} is .48.*

the value of its bottommost input (the fifth sonar). Because doors in this corridor are typically recessed, a long sonar reading in this direction is a strong indicator that the robot is directly by a door. Similarly, the knowledge captured in the first network indicates that this fifth sonar reading depends positively on the second sonar value of the initial state. This is reasonable, because as the robot drives forward, the object that provided the second sonar echo moves into the field of view of the fifth sonar. Coupled together, these two dependencies indicate that the larger the second sonar value in the current state, the more likely that there is a door one meter ahead.

Using the chain rule, EBNN combines the matrix of partial derivatives from each step in the explanation to compute the derivative of the target function with respect to each training example feature. It then updates the weights of the target network, to fit both these target slopes, and the observed training value.

The former provides an analytical component of learning, transferring knowledge from the domain theory networks into the target network. The later provides an inductive component, tuning the weights of the target network to fit the observed value independent of prior knowledge.

5 Experimental Results

This section considers two questions. First, can EBNN generalize more correctly than purely inductive methods, by relying on its previously learned knowledge? Second, can knowledge acquired by EBNN be successfully built upon, by using it as part of the domain theory for learning subsequent target functions?

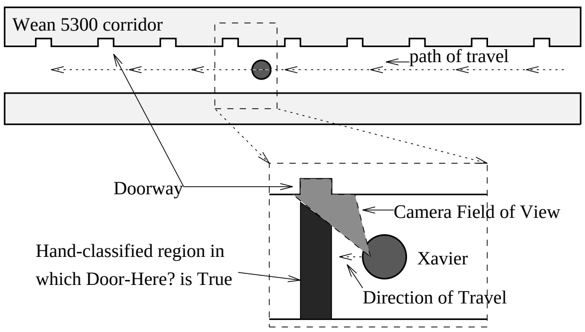

Figure 6: **Corridor Environment** *For these experiments, Xavier traversed an entire corridor, gathering sonar and vision snapshots approximately every 12cm. Snapshots were then hand-classified to indicate when Xavier was immediately next to a door.*

The experimental setup was as follows: Xavier was first allowed to travel an entire corridor (see Figure 6). From this experience, it acquired 403 sensor "snapshots", one approximately every 12 cm. The correct value of Door-Here? was manually provided for each snapshot. The Forward-1M and Door-Here? domain theory networks were then trained by applying the inductive Backpropagation algorithm to this data.

5.1 Generalization Accuracy

To explore the generalization accuracy of EBNN and Backpropagation, we trained the robot to learn the target function Door-ahead$_1$, while varying training set size. Training sets were composed of N contiguous examples, the final examples being directly adjacent to a given door. Values of N were 5, 10, 15, 20, 25, 30, 35, etc., up to the point where a new door was encountered (typically around N=50). Positive examples of Door-ahead$_1$ were typically around examples 10 through 21, and in most training sets approximately 80the examples were negative examples.

Learning experiments were repeated using each of the eight doors in the corridor as training data, and testing on the others. In each of these repeated experiments, the 403 corridor snapshots were divided into two sets – the *training set* of snapshots approaching the selected doorway, and the rest. A *holdout set* of 100 random samples was extracted from the rest, with the remainder defined to be the *evaluation set*. Backpropagation and EBNN were applied to (subsets of) the training set using a learning rate of 0.1, with cross validation performed using the holdout set to determine when to halt training. Generalization accuracy was then measured by applying the learned networks to the evaluation set. This process was repeated for all eight doorways in the corridor, and the results were averaged.

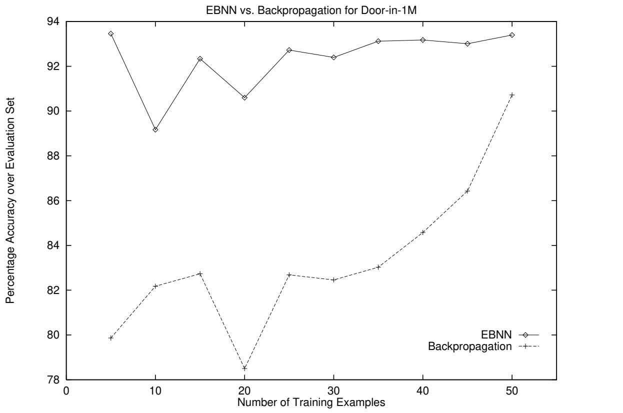

Figure 7: **Generalization accuracy for Door-Ahead$_1$.** *Generalization accuracy averaged over eight learning experiments, using each of the eight available doors as a training door. The dotted line indicates the performance of purely inductive Backpropagation. The solid line indicates the performance of EBNN. Note that an classification accuracy of 79 percent is achievable by classifying every example as negative.*

Figure 7 displays the average results from all eight experiments, plotting generalization accuracy of both learning methods against the number of training examples available. As can be seen, EBNN generalizes more accurately than the inductive Backpropagation algorithm over the entire range of training set sizes. It is interesting to compare the performance of the two approaches on very small sets of training data. The first five training examples were all negative examples of the target concept. As a result, Backpropagation learned a network that always predicted there is no door ahead, leading to an accuracy of 79% on the evaluation data set. Surprisingly, EBNN was able to learn a more accurate network from this same data, despite the fact that it contained no positive examples. This is because EBNN extracted dependencies from its explanations that captured the information that changes to certain feature values would make it more likely that a door would be present. Notice the dip in performance for both methods between 10 and 20 examples. At this point, the data sets were particularly misleading because the heavy number of positive examples was misrepresentative of their distribution in the evaluation sets.

One unrealistic assumption in the above experiment is that the learner has available a set of 100 examples to use for cross validation, even though it uses only 5 to 50 examples for training. Since we are interested in learning when available data is sparse, a more realistic assumption would be that *no* extra data is available for cross validation, and that any attempt to avoid overfitting must be based on the limited available training data. In such cases, a more realistic approach is to use jackknifing [2], or a leave-one-out strategy, in which the number of weight tuning epochs is determined as follows: Divide the N available examples into sets of N-1 train and 1 holdout example, in each of the N possible ways. For each of these training sets, train by cross validating on the remaining example, halting training when performance is maximized on this single holdout example. Compute the desired number of training epochs as the *mean* number of training epochs for these N experiments, then obtain final learning results by training on all N examples for that desired number of training epochs.

We used standard cross validation in our experiments in order to avoid the high computational cost of such jacknifing techniques. In order to test that this was not unfairly biasing the results in favor of EBNN, we compared the effect of jacknifing versus cross validation over one of the training doors. The results, shown in Figure 8, indicate that cross validation in fact biased our results in favor of Backpropagation. This is consistent with our general experience that EBNN is less susceptible than Backpropagation to overfitting (because EBNN considers more training constraints, allowing it fewer degrees of freedom to overfit the data). Because Jacknifing in this case utilized fewer total examples than cross validation, its choice of training epochs was typically less likely to be correct, leading to significant overfitting problems for Backpropagation.

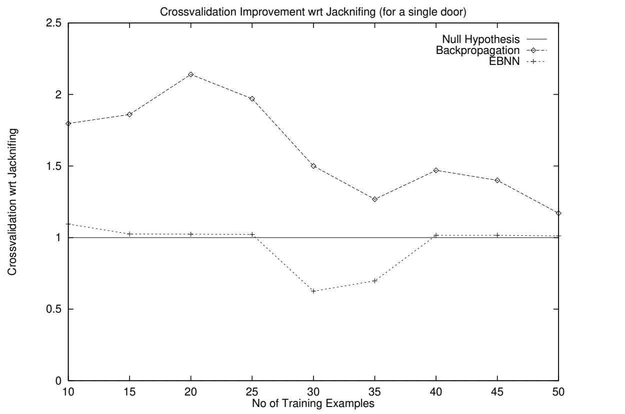

Figure 8: **Comparison of Cross-validation wrt Jacknifing** *Jacknifing techniques would have allowed us to avoid the requirement of gathering extra samples for cross-validation but are too computational intensive. The vertical axis displays the generalization accuracy using cross validation, divided by the accuracy using Jacknifing. The null hypothesis assumes that these two techniques produce identical results. The dotted line indicates that EBNN does not gain addition performance from cross-validation. The dashed line indicates that Backpropagation relies heavily on a validation signal, and improves significantly under cross-validation.*

5.2 Building on Acquired Knowledge

To test the ability of EBNN to learn progressively more difficult functions, and to bootstrap itself by using its earlier learned networks as domain theory for subsequent learning, we extended the previous experiment by learning the additional target functions Door-ahead$_2$, Door-ahead$_3$, and Door-ahead$_4$.

The experimental setup was as above, using the same sets of training examples, holdout examples for cross-validation and evaluation examples. The explanation used by EBNN to learn Door-ahead$_d$ was formed by chaining together the networks for Forward-1M, and Door-ahead$_{d-1}$. For example, EBNN explained examples of Door-ahead$_2$ by first predicting the state one meter ahead (using the Forward-1M network), then estimating whether a door was present one meter ahead of that state (using the Door-ahead$_1$ network learned in the first experiment). Again the training process was repeated for all doorways, and averaged results are reported.

As in the first experiment, results are shown for both EBNN and Backpropagation with Figure 9 indicating the generalization accuracy after training on various numbers of examples, for Door-ahead$_d$, where d=1,2,3 and 4 meters. The generalization accuracy, plotted on the vertical axis, degrades for both methods as d increases (i.e., as the task gets more difficult), with EBNN consistently generalizing more accurately than Backpropagation. This data is shown in different form in Figures 10 and 11.

Figure 12 indicates the relative performance of EBNN versus Backpropagation for N=40 training examples. Figure 13 and Figure 14 represent contour plots resummarizing the data from Figure 9. Notice the monotonic decrease in EBNN performance as problem complexity increases, whereas Backpropagation exhibits a less smooth dropoff.

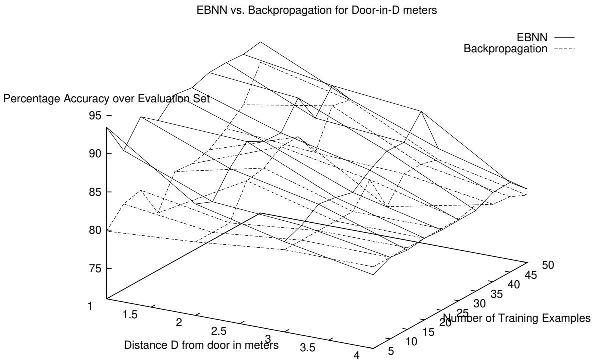

Figure 9: **Generalization accuracy Door-Ahead$_d$.** *Each point corresponds to a different learning task averaged over all doorways: learning Door-Ahead$_d$ for some value of d in meters.*

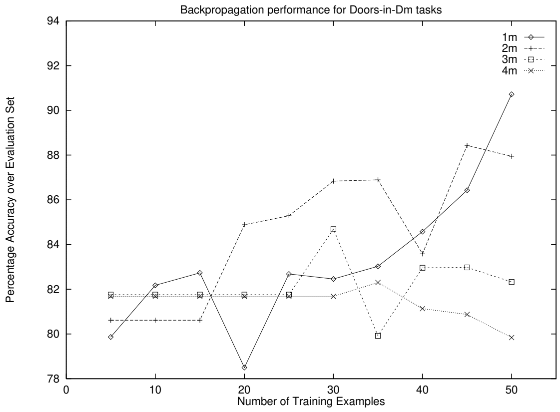

Figure 11: **Generalization accuracy for Door-Ahead$_d$ using Backpropagation** *None of the learning curves compares favorably with the corresponding curve for EBNN.*

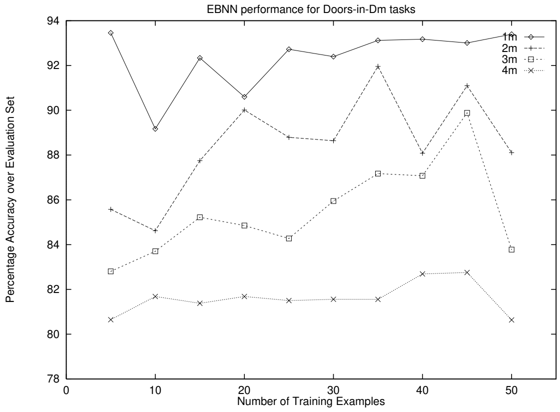

Figure 10: **Generalization accuracy for Door-Ahead$_d$ using EBNN**

6 Conclusions

This paper presents first results applying explanation-based neural network learning (EBNN) to the problem of mobile robot object recognition. These initial experiments demonstrate the ability of EBNN to use approximate, previously learned knowledge about the effects of its actions (traveling forward down a corridor) and about the appearance of a very close object (being directly in front of a door), to guide the learning of neural networks to recognize the object from further distances. The ability of EBNN to utilize information from its approximate domain theory is demonstrated by the fact that it outperforms the standard inductive neural network learning method, Backpropagation, across a variety of data set sizes, and a family of related target functions. While further experimentation is warranted, these results suggest the potential role of EBNN in helping to scale up robot learning by moving away from purely inductive, tabula rasa, learning techniques, toward techniques that bootstrap themselves by leveraging previously learned knowledge to simplify subsequent learning.

7 Acknowledgments

We thank Ryusuke Masuoka for his contributions in helping to develop the EBNN code. This research is sponsored in part by the National Science Foundation under award IRI-9313367, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330.

References

[1] G. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[2] B. Efron. *The Jacknife, the Bootstrap, and Other Resampling Plans.* SIAM, Philadelphia, 1982.

[3] Katsushi Ikeuchi and Takashi Suehiro. Towards an assembly plan from observation. In *Proceedings of IEEE Int. Conf. on Robotics and Automation*, May 1992.

[4] J. Laird, E. Yager, C. Tuck, and M. Hucka. Learning in tele-autonomous systems using soar. In *Proceedings of the 1989 NASA Conference of Space Telerobotics*, 1989.

[5] Long-Ji Lin. *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie Mellon University, School of Computer Science,

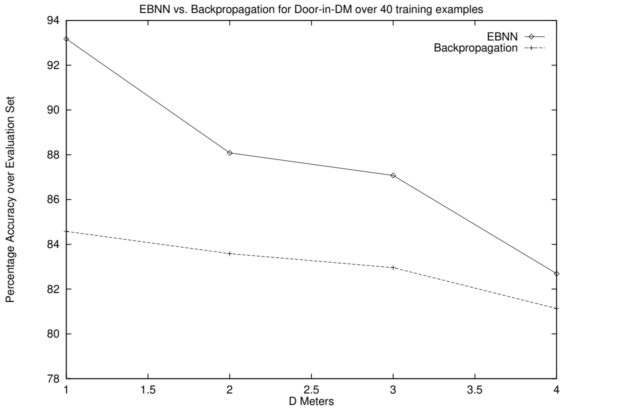

Figure 12: **Generalization Accuracy Door-Ahead$_d$** *We consider the performance of EBNN (solid line) vs Backpropagation (dotted line) over all tasks fixing the number of training examples.*

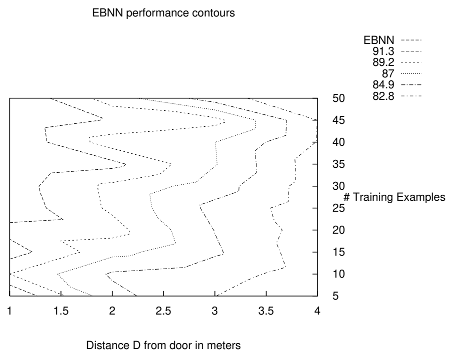

Figure 13: **EBNN performance changes for Door-Ahead$_d$.** *EBNN degrades slowly and smoothly due to building on previously acquired knowledge.*

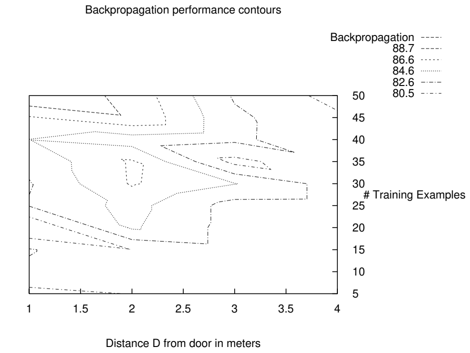

Figure 14: **Backpropagation performance changes for Door-Ahead$_d$.** *Backpropagation does not exhibit a stable generalization*

Pittsburgh, PA 15232, 1993. Technical Report CMU-CS-93-103.

[6] Pattie Maes and Rodney Brooks. Learning to coordinate behaviors. In *Proceedings of AAAI-90*, Boston, August 1990.

[7] Sridhar Mahadevan and Jonathan Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of Machine Learning Workshop '91*, July 1991.

[8] T.M. Mitchell, R.K. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1), 1986.

[9] Tom M. Mitchell and Sebastian B. Thrun. Explanation-based neural network learning for robot control. In J. E. Moody, S. J. Hanson, and R. P. Lipmann, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, December 1993.

[10] Andrew W. Moore. *Efficent Memory-based Learning for Robot Control.* PhD thesis, Trinity Hall, University of Cambridge, England, 1990, 1990.

[11] Joseph O'Sullivan. Xavier manual. Carnegie Mellon University, Learning Robot Lab Internal Document - contact josullvn@@cs.cmu.edu, March 1994.

[12] Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15232, 1992. Technical Report CMU-CS-92-115.

[13] David E. Rumelhart, Geoffery E.Hinton, and Ronald J. Williams. *Learning internal representations by error propagation.* MIT Press, 1986.

[14] Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lipmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903. Morgan Kaufmann, December 1992.

[15] Sebastian B. Thrun. An approach to learning robot navigation. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, September 1994. (to appear).

[16] Sebastian B. Thrun and Tom M. Mitchell. Integrating inductive neural network learning and explanation-based learning. In *Proceedings of IJCAI-93*, Chamberry, France, July 1993. IJCAI.