1 Explanation Based Learning for Mobile Robot Perception

Joseph O'Sullivan and *Tom M. Mitchell*
Carnegie Mellon University

Sebastian Thrun
University of Bonn

Abstract

Although machine learning techniques have been applied with remarkable success to several problems of computer perception and vision, most of these problems have been fairly simple in nature. The difficulty with scaling up to more complex tasks is that inductive learning methods require a very large number of training examples in order to generalize correctly from complex sensor data.

This chapter proposes an approach to overcoming this difficulty, by relying on previously learned information to augment the available training data. In particular, we consider the task faced by a mobile robot learning to recognize new objects within an already-familiar environment. Because the robot has previously operated within this environment (here the corridors of a particular building), it has already had the opportunity to learn certain regularities that can be useful in subsequent learning tasks. Given a new task, such as learning to recognize the distance to the next door in the corridor, knowledge of these regularities enables the system to learn the new task more accurately from a limited quantity of new training data. We describe the Explanation-Based Neural Network (EBNN) algorithm for utilizing previously learned knowledge, and examine its performance for the mobile robot perception task of door recognition in a familiar corridor based on color vision and sonar sensor data. Experimental results indicate that EBNN is able to generalize more accurately than purely inductive methods such as Backpropagation, even when its prior knowledge is only approximately correct.

1. Introduction

Recent results in robot learning have demonstrated that robots can learn simple strategies from very little initial knowledge in restricted environments [9, 2, 17].

While these results indicate the potential role of machine learning for robot perception and control, new approaches are needed to scale up to more complex problems and many realistic environments.

The fundamental roadblock to scaling up learning algorithms is that as the complexity of the learning task (e.g., the number of distinct sensors, complexity of individual sensor inputs, amount of sensor noise, complexity of the function to be learned) increases, it becomes increasingly difficult to correctly generalize from the limited quantity of training data available. Both experimental and theoretical research indicates that purely inductive learning methods, which basically rely on detecting statistical regularities among the training data, scale poorly with such increases in complexity [7, 30, 10].

A number of approaches have been proposed for scaling up learning, by introducing constraints in addition to observed training data. These include, for example, engineering human knowledge into the system [29, 20], and using constraints from related learning tasks [21, 1]. This chapter considers methods by which a robot can use previously learned knowledge about its environment to augment the available training data when confronting new perceptual learning tasks. Figure 1 represents this framework.

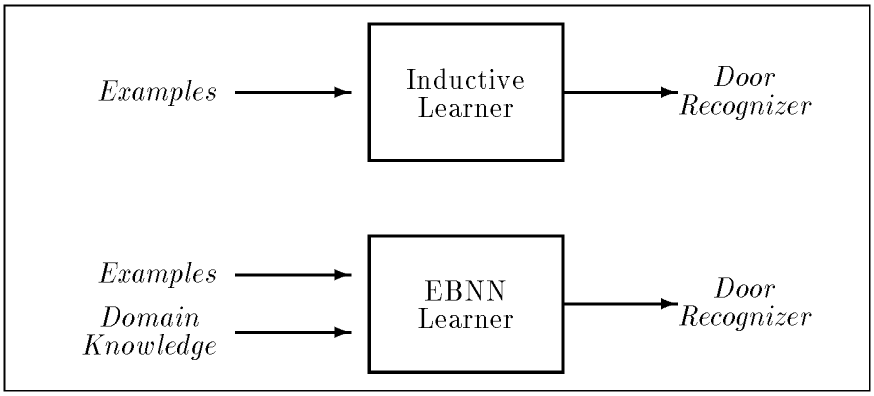


Figure 1: *Using prior knowledge to scale up learning. Inductive learning systems rely upon detecting statistical regularities in data. The EBNN algorithm also uses previously learned knowledge to further constrain learning. This chapter discusses the form of this prior knowledge, and how it is created and used within the EBNN framework. In particular, we shall see how to learn a doorway recognizer given labeled snapshots of the world, and how EBNN improves upon this by using knowledge about the motion of the robot.*

To illustrate the potential importance of previously learned knowledge in generalization, consider a mobile robot that operates over a long period of time within a particular building. It is occasionally presented with new perceptual learning tasks such as "learn to recognize doors", "learn to recognize trash bins," or "learn to recognize trash." There are typically many regularities that occur in the world of any given robot and environment, arising from factors such as the type of lighting, the location of the sensors on the robot, the typical positions and motions of objects within the environment, etc. Such regularities, if known to the robot, can provide useful knowledge to constrain learning tasks undertaken in that setting, and would enable the robot to generalize better from less training data.

This chapter presents the Explanation-Based Neural Network (EBNN) learning algorithm. EBNN learns such regularities, then uses this knowledge to guide generalization for new learning tasks within this familiar environment. As a result, it is able to generalize more accurately from less training data when presented with new learning tasks.

The primary type of learned regularity considered in this chapter is knowledge of how mobile robot actions affect robot sensory inputs. More specifically, consider a robot roaming the corridors of a particular office building. A sequence of camera images perceived by the robot as it travels down the corridor is shown at the top of Figure 2.

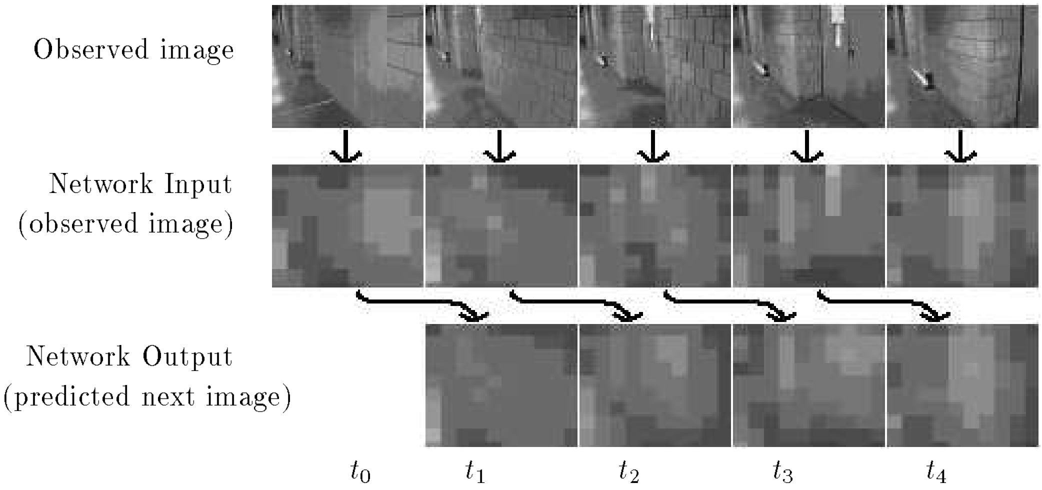


Figure 2: *Results of a neural network trained to predict future images. The images in the top sequence are observed by the robot moving forward in the corridor. Images in the middle sequence are down-sampled (coarse grain) descriptions of the top images. Images in the bottom sequence are predictions by the learned network of the next image, based on the current image. These particular images were not seen by the network during training.*

Here, each image is separated by a "go forward 1 meter" robot action. The robot uses this self-collected data to train a neural network that takes the current image as input, and produces a prediction of the next image as output. The second row of images in Figure 2 shows the actual network inputs, which are a coarse-grain down-sampled version of the original image. The third row of images shows the trained neural network's prediction of the next image. Notice that the predicted images, while imperfect, do capture certain features of the actual next images (shown directly above them).

Such knowledge as learned by this network captures, imperfectly, regularities of this robot in this environment. We will refer to this as *domain knowledge*, or a *domain theory* that characterizes useful information about the robot and its environment, independent of any particular object recognition task. EBNN is a learning method that can use this kind of approximate domain knowledge to improve performance at new learning tasks such as "learn to estimate the distance to the next door." We argue that often, domain knowledge is easy to obtain, as it does not depend on the particular learning task at hand. For example, domain knowledge learned in previous object recognition tasks can be re-used if the robot faces a new recognition task.

The reminder of this chapter is organized as follows. In section 2, we describe the robot, Xavier, used in the experiments reported here. We also describe the learning tasks, all related to detecting the location of doorways in hallways, as well as the types of domain knowledge provided to the learner.

Section 3 describes the EBNN algorithm in detail and presents results applying it to learning to recognize doors. The main result is that EBNN learns more accurately than the standard inductive Backpropagation algorithm, by relying on prior knowledge as well as the training data. Notice EBNN differs from Backpropagation solely in the fact that EBNN exploits further domain knowledge for generalization. The study also illustrates that as the domain knowledge becomes less accurate, EBNN's performance degrades gracefully, and that EBNN can itself be used as a source of domain knowledge. These experiments are carried out using a hand-selected subset of the larger set of available sensor features.

Section 4 explores the effect of weakening the accuracy of the domain theory and of increasing the number of input features. It also considers a more complex task: learning the exact distance to the next doorway along the robot's path.

Section 5 discusses the main lessons of the experiments reported here. In section 6, we review other research on using prior knowledge in learning. Section 7 describes future research and Section 8 our conclusions.

This chapter expands upon, and includes portions of, work previously presented in [18].

2. Learning for Mobile Robot Perception

Figure 3 shows the mobile robot, Xavier, used in the experiments reported here. Xavier was developed at CMU, and is built on a 4 wheeled omni-directional RWI base, 24 inches in diameter. The sensors we shall use are 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, it was fixed in a single position pointed 15 degrees downward and 30 degrees to the right.

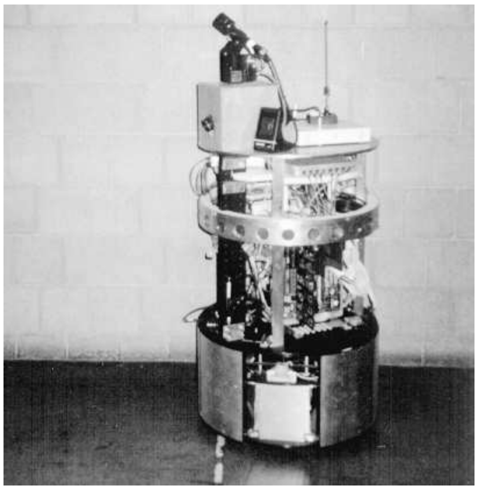


Figure 3: *The robot used in the experiments in this chapter, Xavier of Carnegie Mellon University's Learning Robot Laboratory.*

2.1. The Setting

In the experiments considered here, Xavier operates within the 40 meter long building corridor shown in Figure 4. Within this environment, there are various landmarks and movable objects which are useful for Xavier to recognize.

(a) (b)

Figure 4: *The corridors at CMU as seen day to day with bins, open doors and people (a), and as cleaned up for the reported experiments (b).*

In general, each learning task can be viewed as the problem of estimating some target function, given examples of the input-output relationship for that function. As demonstration, this chapter studies the recognition of doors as a learning task. Two classes of target functions, which are the basis for two separate series of experiments respectively, are:

- **Object Recognition:**

  The task here is to learn a boolean-valued function classifying the current state of the world according to whether there is a door some fixed distance d ahead. We consider learning target functions of the form

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

where $d = 1, 2, 3, 4$ is some fixed distance in meters, $s \in S$ 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. For example, the target function Door-Ahead$_2$?(s) is true if and only if there is a door 2 meters ahead of state s.

- **Object Localization:**

  Here the task is to learn a real-valued function describing the distance to the next door. The target function in this case is

$$\text{Door-Distance: } S \rightarrow d$$

  where $s \in S$ is again the current robot sensor input, and d is a real number indicating distance in meters. The value of Door-Distance(s) is the smallest distance d such that after traveling d meters forward in the corridor, the robot will be adjacent to a door.

Both tasks are cast as supervised learning tasks, in which during training a person notifies the robot when it is adjacent to a door. This information is easily converted into accurate estimates of distances to doors from any given state, using Xavier's dead-reckoning of distance traveled, obtained from the sensed number of wheel turns.

It is important to notice that learning to recognize nearby doors is significantly easier than learning to recognize distant doors, because the door becomes a much smaller feature within the image as distance increases, and sonar information becomes much less useful. Thus, this family of target functions provides a range of difficulty for testing our learning methods.

2.2. The Learning Approach

The learning methods considered here use artificial neural networks to represent the learned target function. Neural networks provide a general mechanism for estimating, or learning, a large class of real-valued or discrete-valued functions. Given a set of training examples of the target function, the Backpropagation algorithm [22] can be used to tune the weights of the network to fit these training examples. Whether this learned network will generalize to perform accurately on examples outside the training set depends on many factors, notably the number of training examples provided. We will refer to the accuracy of the learned network on future examples taken from an independent testing set as the *generalization accuracy* of the learner.

As noted earlier, the key question considered in this paper is how to improve the generalization accuracy of learning, by using previously learned knowledge. Plain Backpropagation is unable to exploit such knowledge. Therefore, we use an alternative method to train artificial neural networks, EBNN, which will be described in the next section. In the experiments reported here, different types of prior knowledge are considered, including knowledge of the form "how does traveling forward alter sensory input?" (Forward-1M), and "am I currently adjacent to a door?" (Door-Here?). This knowledge, like the target functions to be learned, is represented by neural networks that have been previously learned by the robot operating within the same corridor. Because this knowledge is itself learned from limited data, it provides only an approximate characterization of the true environment (recall Figure 2). For each learning experiment, a collection of networks summarizing prior knowledge is used as a domain theory to guide learning the new target function. These experiments are summarized in Table 1, and described in more detail in the following sections.

Data	Domain Theory	Target Function
$\{< S, \{T,F\} >\}$	Forward-1M: S $\rightarrow$ S Door-Ahead$_{d-1}$?: S $\rightarrow$ {T, F}	Door-Ahead$_d$?: S $\rightarrow \{T, F\}$
$\{< S, d >\}$	Door-Distance: S $\rightarrow d$	Door-Distance: S $\rightarrow d$

Table 1: *The learning tasks addressed in this chapter. Inductive techniques use just raw data in learning. EBNN improves learning performance by using both raw data and domain theories acquired from previous experience. Section 3 investigates the learning of Door-Ahead$_d$, section 4 uses a trivial domain theory to explore the robustness and scalability of EBNN when learning Door-Distance.*

3. Explanation-Based Neural Network Learning

This section explores the use of domain knowledge to improve learning for a mobile robot in the corridor environment. It also presents the EBNN learning algorithm in detail. The experiments reported here demonstrate that domain knowledge can be used by EBNN to improve the accuracy of learning, and that EBNN's performance is robust to errors in this domain knowledge.

3.1. The Task

We will introduce the main concepts of the EBNN learning algorithm using an example. Throughout this section, the specific task for the robot is to learn the target function

$$\text{Door-Ahead}_1\text{?: } S \rightarrow \{\text{True, False}\}$$

whose value is True when there is a door 1 meter ahead. In the initial experiments the sensed world state, S, is represented by a simple vector of 25 sensor readings as shown in Figure 5. Each state is collected by Xavier as it proceeds forward down the center of a corridor. A row of 10 rectangular regions is extracted from the image and for each region, two values are computed: the average red intensity and the average blue intensity. The circle of gray bars on the left in the figure indicate the 24 sonar distance echoes detected by Xavier's sonars at the same time the camera image was collected. The sonar data is sub-sampled 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. This sonar and image data is summarized in the vector of 25 values displayed at the bottom. The area of the white box in the vector diagram indicates the magnitude of the corresponding vector value.

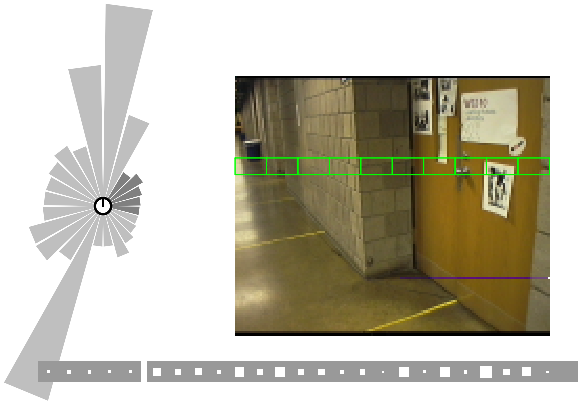


Figure 5: *A typical reduced sensor input and its representation, as described in the text.*

3.2. The Domain Theory

To learn the target function, Door-Ahead$_1$?, EBNN uses previously learned knowledge called the domain theory. The domain theory used in this experiment consists of two neural networks that have been previously learned by the robot. The first is similar to that shown in Figure 2, and represents the function

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

Given the current state represented by the vector of 25 sensor readings, the Forward-1M network predicts the 25 sensor readings (next state) that it expects to perceive if the robot drives 1 meter forward in the corridor.

The second domain theory network represents the function

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

which is True only when the robot is in a state where it is directly adjacent to a door.

Notice that taken together, these two domain theory networks can be used to infer values of the target function Door-Ahead$_1$?, as illustrated in Figure 6. These networks are called domain theory, because, if correct, they allow to logically infer the target concept. In EBNN, however, both of the networks of the domain theory are learned from previous experience using the Backpropagation algorithm, and hence might be incorrect.

This ability to use the domain theory to infer, or *explain*, values of the target function is the key to the EBNN algorithm. To see why, consider the generalization task faced by the learner: Given only a sample of the possible input-output pairs of the target function Door-Ahead$_1$?, it must hypothesize the value of the function for all possible inputs. It must therefore determine which subset of the input sensor readings imply which output value under which conditions. Inductive learning methods rely on the statistical properties of the training examples to determine which input sensor readings are relevant under which conditions. In EBNN, the domain theory provides an alternative means for determining which features are relevant. This is accomplished by using the domain theory to infer, or explain, the output value for each observed training example, then extracting this relevance information from the constructed explanation.

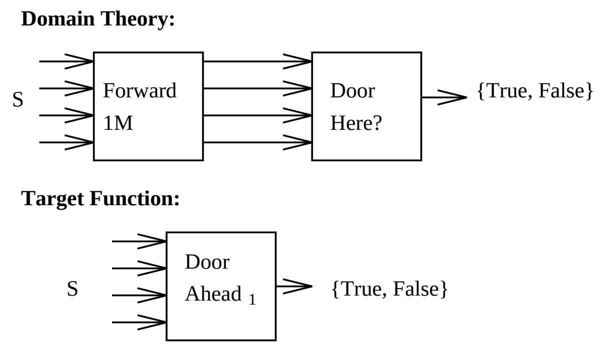


Figure 6: *A Typical EBNN Domain Theory. Each training example of the target function, Door-Ahead$_1$?, is explained in terms of the previously learned functions Forward-1M and Door-Here?.*

3.3. Extracting Derivatives

To see how the domain theory is used by the EBNN algorithm, consider an example: analyzing a single negative training example: state S_{902}, for which there is no door one meter ahead. EBNN first constructs an explanation. An explanation is a prediction of the target concept based on the domain theory, which is obtained by chaining together the domain theory networks as shown at the top of Figure 6.

The 25 sensor readings describing state S_{902} are input to the first domain theory network, which outputs the predicted state one meter forward. The second network is then applied to this predicted state, to estimate whether a door is present in this predicted next state. We will refer to this prediction of the (already known) training example value as the *explanation* of the training example.

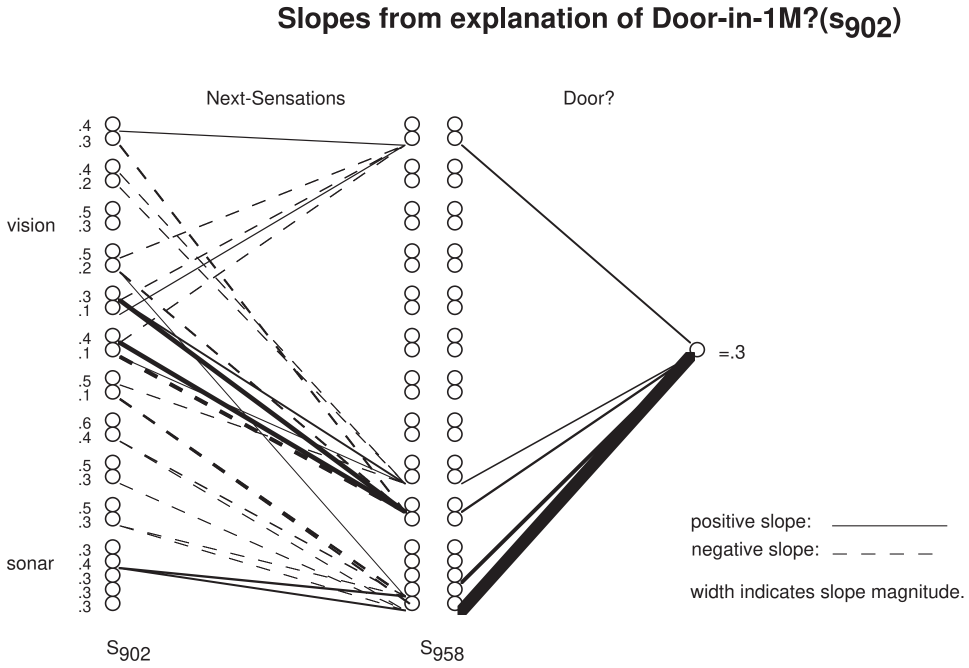


Figure 7: *Sensed state S_{902} is a training example for which target function Door-Ahead$_1$? is False. EBNN explains this value for the target function by using its previously learned neural networks Forward-1M, and Door-Here?. The derivatives of these individual networks, indicated by solid and dashed 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.*

The EBNN algorithm estimates the relevance of each feature of the training example from such explanations. In particular, it uses the explanation to compute the partial derivative of the target function value 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.

Because neural networks represent continuous real-valued functions, the derivative of their output with respect to each input is always well defined. Furthermore, this derivative can be calculated from the network weights and activations. 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 7 indicate the sign and magnitude of the most significant derivatives for each individual network in the domain theory for the explanation of the negative example S_{902}. 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 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. The kind of dependency captured by this and other partial derivatives is useful for determining how to generalize from the specific feature values of the training example.

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 network that represents target function, to fit both these target derivatives, 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.

3.4. The General EBNN Learning Algorithm

The general EBNN algorithm is summarized in Figure 8. For each training example, EBNN constructs an explanation, analyzes it to calculate the partial derivatives of the target function with respect to each example feature, then refines the target function to fit these derivatives as well as the training example value. The third step, refining the target network weights, utilizes a gradient descent algorithm to minimize the combined error function

$$E = E_{values} + \alpha E_{derivatives}$$

where E_{values} is the difference between the target network output values and the known values of the training examples, and where $E_{derivatives}$ is the difference between the target network derivatives and those inferred from the explanation. Gradient descent to minimize this error function is accomplished by the TangentProp algorithm [24], an extension to Backpropagation that iteratively adjusts

For each training example:

Explain how the training example satisfies the target function: Chain together the networks of the domain theory 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 networks of the domain theory 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 8: *Summary of the EBNN Algorithm.*

network weights to minimize this combined error function. EBNN uses a modified version [13] of the TangentProp algorithm, which takes advantage of the exact form of the EBNN derivative error (This error term adds up derivative errors with respect to all axial direction vectors, instead of just picking one direction in the input space). Figure 9 illustrates the advantage of fitting derivatives in addition to values.

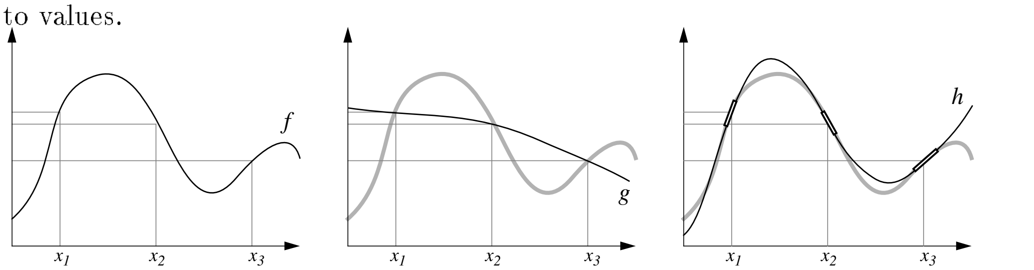


Figure 9: *Fitting values and derivatives in EBNN. Let f be the target function for which three input-output examples $\langle x_1, f(x_1)\rangle$, $\langle x_2, f(x_2)\rangle$, and $\langle x_3, f(x_3)\rangle$ are known. Based on these points the learner might generate the hypothesis g. If the derivatives are also known, the learner can do much better: h.*

One important issue in using the above error function is choosing an appropriate value for α, the coefficient that determines the relative importance of fitting derivatives versus values. Because target derivatives are calculated from the explanation, their accuracy is strongly determined by the accuracy of the underlying domain theory. Clearly, a very poor domain theory will give rise to inaccurate explanations that provide misleading estimates of the true derivatives of the target function. Given that explanations are simply predictions of known training values, a simple way to evaluate the accuracy of an explanation is to compare the known training value to the value predicted by the explanation. For this reason, α is calculated separately for each distinct training example. In the case that the example is accurately explained by the domain theory, it is given a high value placing strong emphasis on fitting the derivatives for this example. In the case that the example is poorly explained, it is given a low value, de-emphasizing this analytical component of learning.

Thus, a reasonable heuristic for choosing α_i for the ith training example is

$$\alpha_i = 1 - \frac{\delta_i}{\delta_{threshold}}.$$

Here δ_i is the root-mean square difference between the true target value and the prediction by the domain theory for training example i. $\delta_{threshold}$ denotes the prediction error threshold for all the examples, and is used for normalization. Negative values of α_i, that is, values for which $\delta_i > \delta_{threshold}$, are set to 0. This heuristic, LOB* [15], for setting α_i is based on the assumption that the accuracy of the explanation's derivatives are correlated to the accuracy of the explanation's predictions.

3.5. Experimental Results

The experiments here address the following two questions, which are central to EBNN and the role of prior knowledge in learning.

1. For a given set of training examples, can EBNN generalize more correctly than purely inductive methods by relying on its previously learned knowledge?
2. Can knowledge acquired by EBNN be successfully built upon, by using it as part of the domain theory for learning subsequent target functions?

The experimental setup was as follows: Xavier was first allowed to travel an entire corridor cleared of clutter (see Figure 10). From this experience, it acquired 403 sensor "snapshots," one approximately every 12 cm, as it followed a path approximately down the center of the corridor, with its camera pointed downward toward the right. The correct value of Door-Here? was manually provided for each snapshot.

First, we trained the domain theory networks using standard Backpropagation. The Forward-1M domain knowledge network was learned by generating, from the data, input-output pairs of states approximately 1 meter (in fact 96cm) apart. 25% of these pairs were then used as a hold-out set for cross-validation, and a neural network was trained to approximate Forward-1M. Cross-validation uses the prediction error over the hold-out set to determine the number of training iterations (epochs) which produces the best performance over this hold-out set. Training is terminated at this point.

Similarly, input-output pairs were generated from the data by hand-classifying each state with the correct Door-Here? label. Backpropagation was used to learn

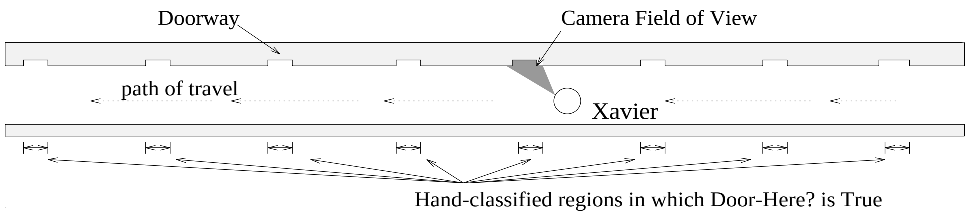


Figure 10: *For these experiments, Xavier traversed an entire corridor, gathering sonar and vision snapshots approximately every 12cm. Snapshots were then hand-classified to indicate when Xavier was immediately next to a door.*

the Door-Here? domain theory network using this data, using 25% of the pairs as a hold-out set for cross-validation.

The Door-Ahead$_d$ function was then learned using training data approaching each of the eight doors in the corridor. In each of these repeated experiments, the 403 corridor snapshots were divided into three sets – the sequence of snapshots approaching the selected doorway (the master *training* set), 100 random samples for cross-validation during training (the *hold-out* set), with the remainder defined to be the *evaluation* set.

Therefore each master training set contained the snapshots between two adjacent doors. Training sets of N examples were generated from each master set, where values of N were 5, 10, 15, etc. Examples in the training sets started at the final doorway snapshot, and progressed backwards towards the previous doorway (typically at N=50). Positive examples of Door-Ahead$_1$? were typically around examples 10 through 21 within the sequence, and in most evaluation sets approximately 80% of the examples were negative examples.

3.5.1. Generalization Accuracy

To explore the generalization accuracy of EBNN and Backpropagation, we trained the robot to learn the target function Door-Ahead$_1$?, while the training set size varied from 5 to 50.

Backpropagation and EBNN were applied to the training sets, subsets of the master training set, using a learning rate of 0.1. The explanation used by EBNN to learn Door-Ahead$_1$? was formed by chaining together the previously trained networks for Forward-1M, and Door-Here?. $\delta_{threshold}$ in LOB* was set to δ_{max}, the maximum prediction error, which means that the derivative error term was used to some degree in refining all of the training examples. Generalization accuracy was then measured by applying the trained networks to the evaluation set. This process was repeated for all eight doorways in the corridor, and the results were averaged.

Figure 11 displays the average results from all eight repetitions, plotting generalization accuracy of both learning methods against the number of training examples available. As can be seen, EBNN generalizes more accurately than the

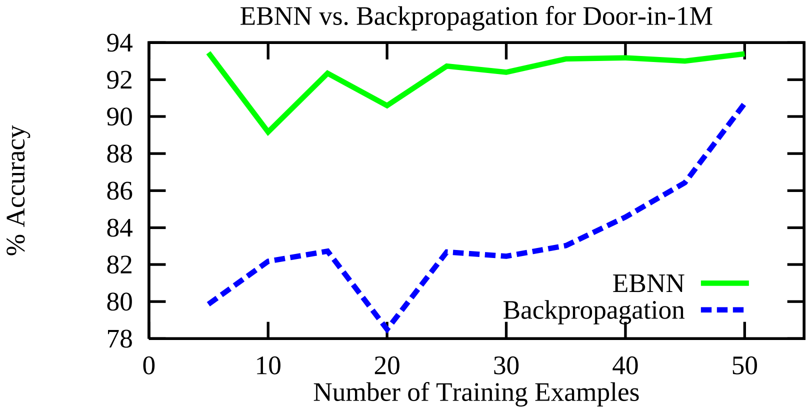


Figure 11: *Generalization accuracy for Door-Ahead$_1$? averaged over eight learning experiments, using each of the eight available doors as a training door. The dotted line indicates the performance of Backpropagation, the solid line that of EBNN. Note that a classification accuracy of 79 percent is achieved by classifying every example as negative.*

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. However, 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 large number of positive examples misrepresented 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 for 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 over-fitting must be based on the limited available training data. In such cases, a more realistic approach is to use jacknifing [4], 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 hold-out 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 hold-out 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 for one of the eight training doors. The results, shown in Figure 12, indicate that cross-validation in fact biased the experimental results in favor of Backpropagation. This is consistent with our general experience that EBNN is less susceptible than Backpropagation to over-fitting (because EBNN considers more training constraints, allowing it fewer degrees of freedom to over-fit 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 over-fitting problems for Backpropagation.

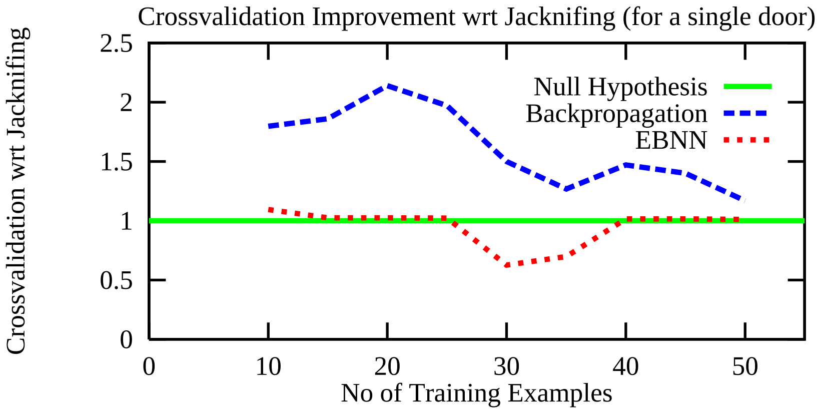


Figure 12: *Jacknifing techniques would have allowed us to avoid gathering extra samples for cross-validation but is more computationally intensive. The vertical axis displays the generalization accuracy using cross-validation, divided by the accuracy using Jacknifing. The null hypothesis assumes that these two techniques produce identical results. The dotted line indicates that EBNN performs nearly as accurately with jacknifing as with cross-validation. The dashed line indicates that Backpropagation performs worse using jacknifing.*

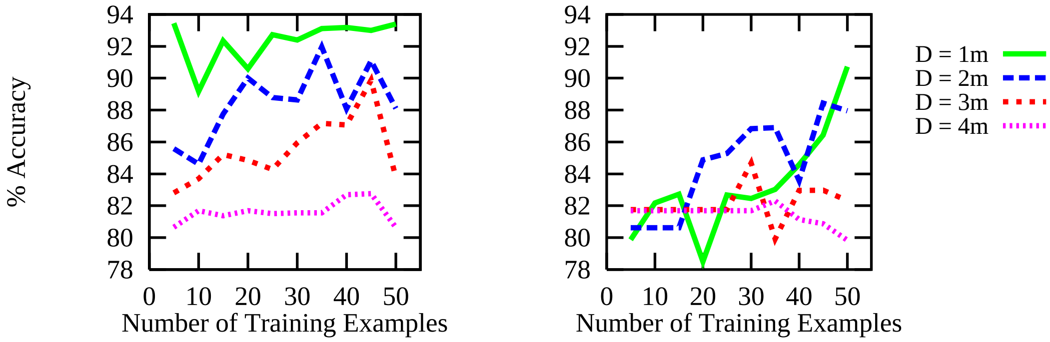
3.5.2. Iterating EBNN

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$_4$?, which must recognize doors further away.

The experimental setup was as above, using the same sets of training, hold-out and evaluation examples. In general, 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 as in the previous experiments), then estimating whether a door was present one meter ahead of that state (using the trained Door-Ahead$_1$? network that had achieved the best generalization performance). Again the training process was repeated for all doorways, and averaged results are reported.

Generalization Performance for Door-Ahead$_d$? tasks



(a) EBNN (b) Backpropagation

Figure 13: *The generalization accuracy for Door-Ahead$_d$? comparing training by EBNN (a) versus training by Backpropagation (b).*

As in the first experiment, results are shown for both EBNN and Backpropagation with Figure 13 indicating, after training on the various numbers of examples, the generalization for Door-Ahead$_d$?, where d=1,2,3 and 4 meters. This 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. Starting with d = 4 meters, both learning methods basically perform as well as simple majority voting, i.e., always producing "no." Notice the monotonic decrease in EBNN performance as problem complexity increases, whereas Backpropagation exhibits a more erratic behavior. To summarize, these results suggest that EBNN out-generalizes Backpropagation consistently, if it is given the right domain knowledge. This is particularly the case if training data is scarce. When sufficient training data is available, both methods exhibit approximately equivalent generalization rates.

4. Attributes of Prior Knowledge

This section explores the effect on generalization accuracy of inaccurate domain knowledge and of increasing the number of inputs. It also demonstrates the application of EBNN to learning the distance to the next door – a real valued function.

The specific task for the robot is now to learn the target function

$$\text{Door-Distance: } S \rightarrow d$$

whose value d is the distance the robot should travel to arrive at the next door.

The dimension of the training examples is increased from the 25 inputs of the previous section to 152, as shown in Figure 14. A grid of 8×8 rectangular regions is extracted from the original image of the world. Each rectangular region is encoded in YUV space. Y gives information about the intensity at a point, while U and V jointly encode color information and are shared between adjacent pixels. (The YUV values available on our robot had been converted to RGB color-space in the previous series of experiments to allow us to explicitly remove irrelevant color information). The sonar values are also included as input features, providing 24 range readings equally spaced around the circumference of the robot. A single output corresponds to the real value of the function Door-Distance. Alternative techniques for modeling real numbers in neural networks (such as activating one of many outputs or Gaussian weighting of outputs [20]) were not considered since using a single output turns out to be sufficient for solving the task.

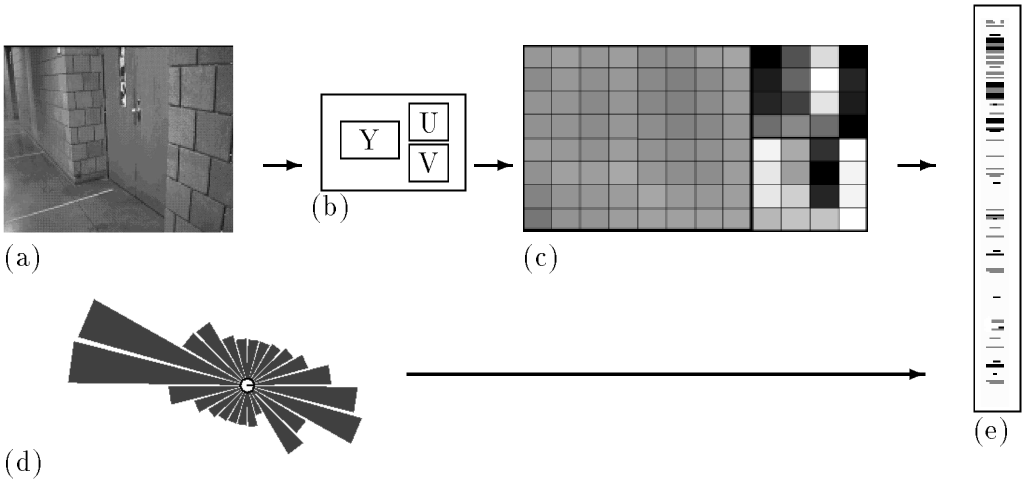


Figure 14: *The original view of the world (a) from the camera is down-sampled to a reduced YUV image (b) in order to make experimentation practical. (The Y components roughly approximate intensity, while the U and V components approximate color). The sensor data shown in (c) and the 24 sonar readings in (d) yield the final input representation (e). The darkness of the individual lines in the vector diagram (e) indicates the magnitude of the corresponding vector value.*

4.1. The Domain Theory

EBNN again requires a domain theory capable of explaining observed training examples of the target function

$$\text{Door-Distance: } S \rightarrow d.$$

As the domain theory, we use a neural network which represents the same function

$$\text{Door-Distance: } S \rightarrow d.$$

Because this domain theory network has the same type and structure as the target network, it can be analyzed via the same means. This simplifies our goal of acquiring and using domain theories of varying accuracy, based on input encodings of varying complexity.

In the first of the following experiments, a high quality domain theory is used to learn Door-Distance. The quality of the domain theory is degraded in the second experiment by training it on progressively smaller data sets, to study how this will affect the generalization accuracy of the final network learned by EBNN from this domain theory. The final experiment varies the size of input representation to examine how this will affect the improvement of EBNN over Backpropagation.

4.2. Experimental Results

Xavier traveled the entire corridor (cf. Figure 10) several times over a period of several days to gather data, encountering different lighting positions and different positions of the robot relative to the center of the corridor. During these runs, Xavier was informed of all states for which Door-Here? was true. This information was easily converted into accurate estimates of Door-Distance for all snapshots in the data, by using Xavier's dead-reckoning of distance traveled. Different subsets of this data were then used to train the domain theory and the target function networks, as described in the following subsections.

For each of the experiments, four different starting points in the corridor, each more than 1 meter from the next doorway, were used. Training sets were generated by Xavier starting at these points and moving forward to collect up to N contiguous examples, where N ranged in size from as little as 4 to nearly 170 examples. For each starting point, a separate hold-out set leading up to a doorway was used for cross-validation and a separate evaluation set of 70 examples was used for calculating the generalization performance. As detailed in section 3, cross-validation uses the prediction error over the hold-out set to determine the number of training iterations which produces the best performance over this hold-out set. Results were reported by averaging the prediction performance over the examples in the unseen evaluation sets, the accuracy being measured by the averaged error in distance.

$\delta_{threshold}$ in LOB* was set at 10% of δ_{max}, the maximum prediction error over the training examples. This reduced the risk associated with degrading the performance of EBNN by using incorrect derivatives, but also reduced the number of examples used by EBNN in refining of the target function (Figure 15).

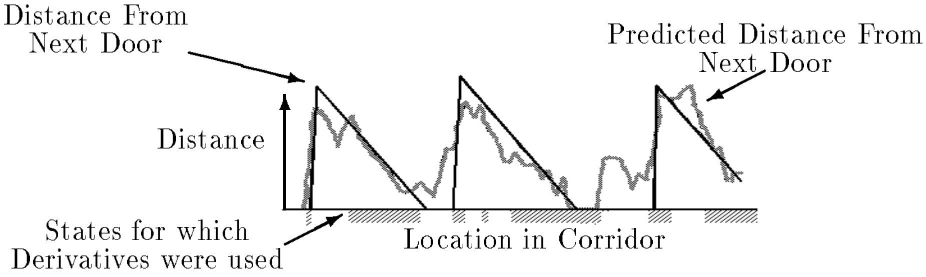


Figure 15: *The LOB* heuristic selects which explanations of the domain theory to use in learning. The grey line shows the results of the domain theory explanation of Door-Distance as the robot approaches and passes 2 doors in the corridor. The black line indicates the actual distance. The shaded states indicate when the two are sufficiently close that derivatives from the explanations are used for training (i.e., when $\alpha > 0$).*

4.2.1. "Accurate" Domain Knowledge

A baseline case was required, to show the performance of high quality domain knowledge in this setting. To achieve this, the Door-Distance domain theory network was trained over approximately 200 examples using Backpropagation. 10% of these examples were used for cross-validation during training.

A different set of examples was used in the training of the Door-Distance target network. A maximum of 170 examples was used during training, with Backpropagation and EBNN performance nearly converging for this amount of data.

Figure 16 compares the performance of EBNN using this Door-Distance domain theory, against the performance of Backpropagation. The error in predicting the door distance over the unseen evaluation sets is reported for both. As in the experiments of Section 3, EBNN generalizes more accurately than Backpropagation. Notice the domain theory had an average error of 109 cm over the same evaluation set. The inductive learning component of EBNN allows it to learn a target function more accurate than this domain theory, once it considers sufficient training data. Figure 16 also shows the result of a naive learner which predicts a constant Door-Distance: the average of the distances in the observed training data (i.e., the *expected* value of the distance). This provides a baseline for comparing performance of more intricate learning methods.

4.2.2. Degrading Domain Knowledge

In this experiment, the accuracy of the domain theory was degraded by reducing the number of examples from which it was learned. Four distinct domain theories were learned, whose average error over the evaluation set ranged from 138 cm to 106 cm. The previous experiment was then repeated for EBNN using each of these four domain theories, with results shown in Figure 17.

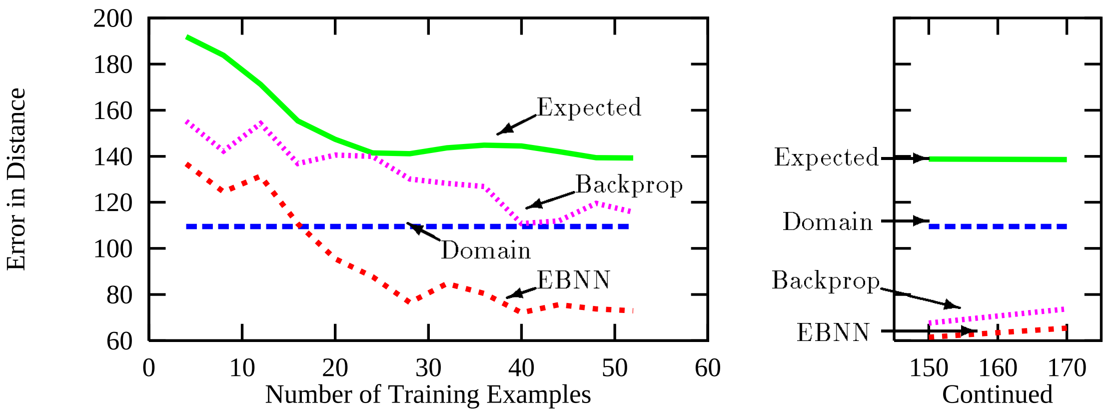


Figure 16: *Generalization accuracy for Door-Distance averaged over the evaluation sets. The graph is continued to show when the performance of Backpropagation approaches that of EBNN. We also present the performance when the domain theory is applied to the evaluation set. For comparison, when the average of the training examples is used as the predicted distance, the expected case is plotted.*

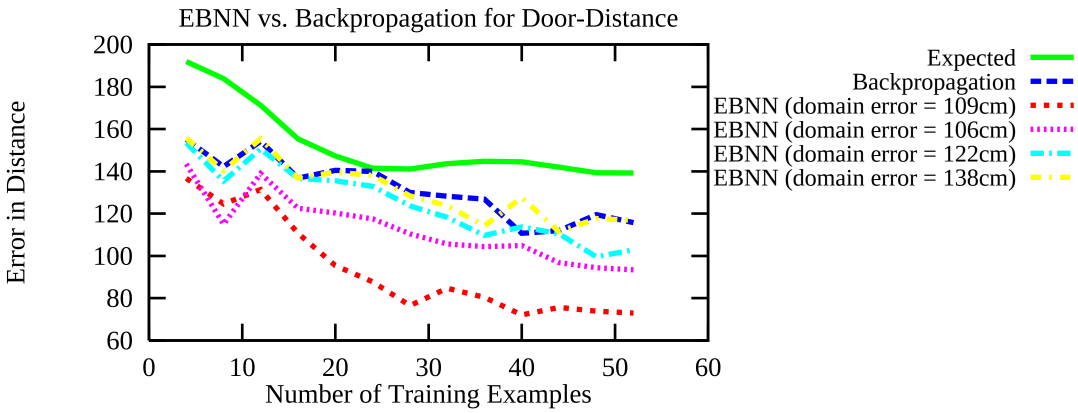


Figure 17: *Generalization accuracy for Door-Distance as the quality of the domain theory used by EBNN degrades.*

Note the correlation between the degradation of the domain theories, and that of the generalization performance of Door-Distance trained under EBNN. In addition, the effect of the LOB* heuristic is noted by the difference in performance between the domain theories achieving accuracies of 106cm and 109cm. Despite the 106cm accurate domain theory generating better performance over the evaluation sets, it did not explain as many training examples as did the 109cm accurate domain theory, and as such did not have as strong an effect upon the generalization performance.

These findings suggest that LOB* successfully prevents EBNN from suffering from misleading domain knowledge. Similar results have been reported for EBNN applied in the context of reinforcement learning [25].

4.2.3. Scaling to High Dimensional Input Spaces

Here, we were interested in characterizing the role of prior knowledge with varying dimension of the input space. We varied the dimension of the input by using just sonar data, just vision data, or both as input when learning Door-Distance using the high quality domain theory. We calculated the relative improvement in performance over the expected accuracy for EBNN compared to Backpropagation. The ratio of these two,

$$improvement\ ratio = \frac{EBNN\ accuracy - expected\ accuracy}{Backpropagation\ accuracy - expected\ accuracy}$$

is shown in Table 2. This ratio, averaged over the various training set sizes, improves as the number of inputs increases. This also reflects our experience from prior research on synthetic learning tasks [13].

Representation	Sonar	Vision	Sonar + Vision
Input Size	24	128	152
Improvement Ratio	2.6	5.0	5.2

Table 2: *We repeat the experiments of Section 4.2.1. using three different input representations. The improvement ratio, defined in the text, indicates the relative improvement of using EBNN versus using Backpropagation.*

Notice that as the input dimension increases, so does the benefit of using EBNN over Backpropagation. We conjecture the reason for this difference lies in the fact that the number of derivative constraints extracted from explanations grows as the dimension of the input space. Hence, as the input dimension increases, EBNN produces more constraints for the target function, whereas Backpropagation does not.

5. Discussion

We have now presented the results of two series of experiments involving learning to recognize doors in a corridor, having varied the complexity of the input representation, the quality of the domain knowledge, the type of the domain knowledge, and the type of the target function to be learned. This section reconsiders our initial hypothesis that EBNN improves generalization and reduces the required training examples, in the light of these results.

5.1. Can Prior Knowledge Improve Generalization?

As demonstrated by a variety of the above experiments, the EBNN algorithm provides a method for using prior knowledge to improve generalization accuracy when learning a perceptual task for a mobile robot. For small training sets, EBNN was shown to generalize more accurately than Backpropagation when learning the target function "Is the next door d meters ahead?" In this case, EBNN relied on prior knowledge in the form of previously learned neural networks representing the functions "What will the sensor readings be after driving forward 1 meter?" and "Is there a door here?" As shown in Figure 11, whereas Backpropagation required 50 examples to reach an accuracy of 90% on this task, EBNN was able to reach the same accuracy in less than 20 examples.

In Section 4, EBNN also exhibits improved generalization accuracy when learning the real-valued function "How far away is the next door?", as long as it is provided sufficiently strong prior knowledge. As shown in Figure 16, after training on the approach to a single door (ranging from 20 to 30 training examples) EBNN predicts the door distance to within 80 cm on future doors, compared to Backpropagation's prediction error of 130 cm. It required over 150 training examples for Backpropagation to approach EBNN's prediction accuracy, which at that point was 65 cm.

Given this evidence, the following discussion examines conditions which affect the degree to which prior knowledge improves generalization.

5.2. EBNN Robustness to Errors in Prior Knowledge

In general, when building on previously learned knowledge, one would like a learning algorithm that benefits from this knowledge when it is sufficiently correct, but is not harmed when it is incorrect.

The effect of increasing domain theory error on generalization accuracy was shown in Figure 17, for the task of learning "How far away is the next door?". In these experiments, the domain theory accuracy was varied from an average error of 106 cm through 138 cm (compared with an error of 140 cm achievable by simply predicting the mean distance over the training set). These results show that the generalization accuracy of EBNN degrades gracefully with increasing domain theory error, to the point where performance is no better than Backpropagation. However, even under the least accurate of these domain theories, EBNN at worst equals the generalization accuracy of Backpropagation learning. Hence, in all our experiments EBNN degrades gracefully as the quality of the domain theory degrades.

The LOB* heuristic is largely responsible for the ability of EBNN to survive incorrect domain knowledge, since it suppresses the use of this domain knowledge for training examples that are incorrectly explained.

This same trend is apparent in the experiments from section 3, learning "Is the next door d meters ahead?", for d equal to 1, 2, 3, and 4. In this case, the domain theory error increases with d, due to the greater difficulty of recognizing more distant objects. Figure 13 shows that the generalization accuracy of EBNN degrades as error increases, but remains superior to that of Backpropagation.

5.3. EBNN and Scaling Up to Higher Dimensional Inputs

Problem complexity is a possible influence upon the improvement in generalization accuracy of EBNN over that of Backpropagation. One measure of problem complexity is the number of inputs to the network. The experiment reported in section 4.2.3. was created to specifically address this issue. The results reported in Table 2 demonstrated that as the number of inputs increased so did the improvement of EBNN over Backpropagation. From this, we suggest that for increasingly complex inputs there is increased benefit in using prior knowledge.

We note that this experiment did not control for the information content of the inputs. In setting up the experiments of section 3, we noticed that a bank of 6 sonars contained all the sonar information useful for detecting presence of the next doorway. In section 4, the extra 18 sonars acted as irrelevant input features. That is, these 18 extra inputs have low or zero information content. In section 4.2.3., it may be seen that the visual features have some information content, as a strong ratio of improvement is obtained when using visual features as the sole input. Thus, when increasing the input size by combining sonar and vision features, we do not simply introduce irrelevant features, but in fact also introduce a different type of information content.

5.4. The Role of LOB*

The LOB* heuristic often improves the performance of EBNN by filtering out the analytical learning component for training examples that are poorly explained. Previous work has demonstrated that EBNN performance degrades substantially if the LOB* heuristic is removed, and derivatives derived from explanations are simply used in all cases [16]. This heuristic is based on the assumption that domain theory accuracy is a good predictor of the quality of the derivatives used for generalization. Consider creating an antagonistic domain theory, which has perfect predictions over the training examples, but random derivatives at these points. Such a domain theory would mislead the LOB* heuristic, resulting in very poor generalization to novel points. While such an antagonistic domain theory would not necessarily occur in the real world, it is the case that LOB* is an additional sensitive parameter of EBNN. For instance, in the series of experiments of Section 4, LOB* was found insufficient to obtain good generalization when $\delta_{threshold}$ was greater than 10% of δ_{max}.

5.5. Computational Costs

EBNN has a greater computational cost than Backpropagation, due to the additional constraints of training on derivative information. Theoretical research [13] has indicated that training with EBNN is about $2I + \frac{7}{2}$ times that of Backpropagation, where I is the number of network inputs. This assumes that all explanations have yielded correct predictions, and thus all derivative information is being used for training. In practice, we found that on the same machine, a single EBNN training epoch takes on the order of 2 to 5 times that of Backpropagation. Given that EBNN was found in some experiments to converge using less than half the examples required by Backpropagation, the increased computational cost per example may be more than offset by the reduced number of examples.

In the mobile robot perception tasks considered here, the cost of acquiring training examples overrides computational costs. This data acquisition cost includes both the cost of the robot capturing training examples, and the cost of human labeling of examples.

5.6. Cost of Learning the Domain Theory

In all our experiments described above, we ignored the costs of learning an appropriate domain theory.

Consider, for example, the cost of learning the domain theory in the first set of experiments from Section 3. Here the domain theory consists of two networks that represent the two functions "What will the sensor readings be after driving forward 1 meter?" and "Is there a door here?". The cost of learning the first is much less significant than that of learning the second. In learning the first, the robot is self-supervising. In addition, this *task-independent* network can be reused for explaining data for other tasks (such as "distance to end of corridor"), so its cost can be amortized over several learning problems. In contrast, data for the second network is human-labeled, which typically is expensive. Furthermore, the cost of learning this *task-specific* network cannot be amortized over different object recognition tasks.

Given that the cost of learning the domain theory is dominated by the task-specific component, we would like methods that minimize this portion of the cost.

6. Related Work

The EBNN method provides one means of utilizing prior domain knowledge. The benefit of this method is that it combines analytical and inductive processes to learn from a combination of previously learned knowledge and new training data. By being able to reuse learned knowledge, this framework is suitable for a robot agent which has to face multiple tasks over a period of time. This section considers related research on utilizing prior knowledge to guide learning.

Three of the best known methods for inserting prior knowledge into neural network learning are Fu's method [6], KBANN [23] and RAPTURE [12]. These methods use prior knowledge in the form of hand-coded propositional rules to guide learning of neural networks. The approach is to first construct an initial network that produces exactly the same predictions as the domain theory, then to refine this network using the available training examples and the Backpropagation algorithm. The key difference between these methods and EBNN is the difference in representation of prior knowledge (symbolic rules versus neural networks) and the way in which this knowledge is used to constrain learning (initializing the target function, versus using a modified criterion for tuning weights).

Earlier explanation-based learning methods that used symbolic representations [3, 14] and the analytical learning component of EBNN are based on the same principle: given a target function, F, the domain theory is used as an alternative method for computing F, and the relevance of the various training example features is then extracted from the trace of this computation. However, EBNN differs in that it represents information about these dependencies in terms of partial derivatives, whereas purely symbolic methods represent them in terms of logical preconditions to the explanation. The original work on symbolic explanation-based learning algorithms also made the restrictive assumption that the domain theory was perfectly correct. This restriction is overcome by more recent work such as FOCL [19].

Simard and colleagues [24] TangentProp algorithm, which is used in EBNN to fit derivatives extracted from the domain theory networks, was originally applied to the task of learning hand-written digits. In his case, the human designer provided certain directions along which the derivatives of the target network were trained to be zero. EBNN differs from this approach in that this derivative knowledge is learned itself, represented by the domain theory networks.

One technique for learning of domain knowledge to aid robot navigation has used a sonar-specific means of storing prior knowledge, rather than connectionist networks [11]. In this instance, occupancy grids were used to predict the effects of the actions for a sonar based robot. The robot was able to learn domain knowledge using these domain specific structures while executing one task, and then, by applying this knowledge, outperformed attempts which did not use domain knowledge at a second task. In presenting a system which learns domain knowledge using connectionist networks, we present a system which can be applied to a greater variety of domains.

7. Future Work

While the results reported here illustrate the potential of EBNN and of using prior knowledge to guide learning, a number of open research issues remain.

Perhaps the most important open issue arising from the experiments shown here is reducing the cost of acquiring the domain theory used by EBNN. As discussed in section 5.6., certain types of knowledge are independent of the particular object recognition task (e.g., the Forward-1M network), and learning these can be amortized over the entire lifetime of the robot and over all object recognition tasks. However, other knowledge (e.g., Door-Here?) is object-specific, and therefore its cost cannot be amortized in this way. It is possible to construct an alternative domain theory in which the target function is itself used as a component of the domain theory, thereby eliminating the cost of gathering data to learn an additional object-specific network. In other work [15] this type of structure was found to be successful for learning to control a simulated robot. Research is needed to understand the conditions under which using the target function as part of the domain theory can be successful, and to explore alternative approaches to reducing the per-task costs of EBNN.

A related topic for future research is developing stronger methods for determining the relative weights for the analytical (derivative) and inductive (value) components of EBNN. The current LOB* heuristic weights the analytical learning component on an example-by-example basis, by considering the accuracy of the domain theory prediction to be a good indicator of the accuracy of the extracted derivatives. Although this heuristic is helpful in most cases, in some it may provide poor estimates of the quality of derivatives. One alternative to explore is learning additional networks trained to predict the quality of the domain theory for individual examples, similar to [28].

Whereas this paper considered only single explanations as a source of derivative information, it is possible in many cases to have multiple explanations. For instance, if we included a “Turn right 30 degrees” action, we would have the additional means of explaining each example by “Turn right 30 degrees and perform recognition.” Combining results from multiple explanations has been found to produce increased generalization accuracy in an computer vision domain [27]. An alternative use for multiple explanations is in situations when two partially correct explanations are available. In the above situation of turning right, we would also expect the explanations produced to be accurate close to the door. The actual domain theory we used in section 4 was weak in these states. By combining these two explanations, the domain theory would now be valid over greater portions of the task space.

Whereas EBNN rests on the life-long learning assumption, that a robot must confront a variety of learning tasks over its lifetime, our research to date has not addressed the issue of what software architecture would support such long-term operation. This raises issues such as how will the robot decide what to learn, when to learn it, or which domain theory to apply? How will it organize its long-term memory in order to make relevant prior knowledge accessible and obvious when and where appropriate? Under what conditions should the agent rely on its learned target function to be more reliable than the domain theory? Some of these architectural issues have been explored in systems such as Soar [8] and

FORR [5], but many open issues remain.

8. Conclusion

The lesson of this chapter for computer vision learning is that when data is scarce, more accurate learning is possible by relying on available prior knowledge. This chapter has introduced one particular means of using prior knowledge to improve the generalization accuracy when learning object recognition procedures for a mobile robot. The EBNN algorithm accomplishes this by explaining training examples, extracting information about feature relevance in terms of derivatives, and using this as an additional constraint for generalization. The experimental studies have demonstrated that EBNN learns more accurate object recognition procedures (represented by neural networks) than the standard neural network learning algorithm, Backpropagation. Furthermore, the EBNN algorithm is robust to significant errors in the prior knowledge it employs. As this prior knowledge becomes less accurate, the improvement of EBNN over Backpropagation decreases, to the point where EBNN reaches approximately the same accuracy as Backpropagation. We examined a variety of learning tasks, to observe the effect of increasing the dimensionality of sensor inputs, weakening the accuracy of the domain theory, and varying the logical structure of the domain theory. EBNN is found to outperform or match the performance of Backpropagation over the entire range of situations considered.

This model fits in with a lifelong learning approach to robotics [26], in which we assume the robot must confront a variety of learning tasks over a long period of time. Because the environment and robot sensors will be quite similar for this entire family of tasks, knowledge that captures the regularities inherent in this task family offers a potentially important method for scaling up learning for robot perception.

References

[1] Rich Caruana. Learning Many Related Tasks At the Same Time With Backpropagation. In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, December 1994.

[2] Jonathan Connell and Sridhar Mahadevan. Rapid Task Learning for Real Robots. In *Robot Learning*, chapter 4, pages 105–150. Kluwer Academic Publishers, 1993.

[3] G. DeJong and R. Mooney. Explanation-Based Learning: An Alternative View. *Machine Learning*, 1(2):145–176, 1986.

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

[5] Susan L. Epstein. For the Right Reasons: The FORR Architecture for Learning in a Skill Domain. *Cognitive Science*, 18:479–511, 1994.

[6] Li-Min Fu. Integration of neural heuristics into knowledge-based inference. *Connection Science*, 1(3):325–339, 1989.

[7] D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and other Learning Applications. *Inform. Comput.*, 100:78–150, 1992.

[8] Laird, J.E. and Rosenbloom, P.S. and Newell, A. Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning*, 1(1):11–46, 1986.

[9] Long-Ji Lin. *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15232, 1993. also available as Technical Report CMU-CS-93-103.

[10] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1987.

[11] Sridhar Mahadevan. Enhancing Transfer in Reinforcement Learning by Building Stochastic Models of Robot Actions. In Sleeman D. and Edwards P., editors, *Proceedings of Ninth International Workshop on Machine Learning*, pages 290–299. Morgan Kaufmann, July 1992.

[12] J. Jeffrey Mahoney and Raymond J. Mooney. Combining neural and symbolic learning to revise probabilistic rule bases. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann.

[13] Ryusuke Masuoka, Sebastian Thrun, and Tom M. Mitchell. Constraining Neural Networks to Fit Target Slopes. Internal Memo, Learning Robot Laboratory, School of Computer Science, Carnegie Mellon University, 1993.

[14] T.M. Mitchell, R.K. Keller, and S. Kedar-Cabelli. Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1(1), 1986.

[15] 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.

[16] Tom M. Mitchell and Sebastian B. Thrun. Learning Analytically and Inductively. In Steier and Mitchell, editors, *Mind Matters: A Tribute to Allen Newell.* Erlbaum, 1995.

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

[18] Joseph O'Sullivan, Tom M. Mitchell, and Sebastian B. Thrun. Explanation Based Learning for Mobile Robot Perception. In *Proceedings of MLC-Colt Workshop on Robot Learning*, Rutgers University, New Brunswick, N.J, July 1994.

[19] Micheal J. Pazzani and Dennis Kibler. The role of prior knowledge in inductive learning. *Machine Learning*, 9:54–97, 1992.

[20] Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance.* Kluwer Academic Publishers, 1993.

[21] Lorien Y. Pratt. Experiments on the Transfer of Knowledge between Neural Networks. In *Computational Learning Theory and Natural Learning Systems, Constraints and Prospects*, chapter 14, pages 523–560. MIT Press, 1994.

[22] David E. Rumelhart, Geoffery E.Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rymelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol I+II.* MIT Press, 1986.

[23] Jude W. Shavlik. An approach for Combining Explanation-Based and Neural Learning algorithms. *Connection Science*, 1:231–253, 1989.

[24] 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.

[25] 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.

[26] Sebastian B. Thrun and Tom M. Mitchell. Lifelong Robot Learning. In *Robotics and Autonomous Systems*, 1993.

[27] Sebastian B. Thrun and Tom M. Mitchell. Learning One More Thing. Technical Report CMU-CS-94-184, School of Computer Science, Carnegie Mellon University, Sept 1994.

[28] Sebastian B. Thrun and Knut Möller. Active Exploration in Dynamic Environments. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 531–538, San Mateo, CA, 1992. Morgan Kaufmann.

[29] Geoffry G. Towell, Jude W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of AAAI-90*, pages 861–866. Morgan Kaufmann, July 1990.

[30] L. G. Valiant. A Theory of the Learnable. *Comm. ACM*, 27:1134–1142, 1984.

ACKNOWLEDGEMENTS

We thank Ryusuke Masuoka for his contributions in helping to develop the EBNN code, and the Xavier team at CMU for developing an effective platform for research into learning for mobile robot perception. Shumeet Baluja, Rich Caruana, Sven Koenig, Conrad Poelman and Astro Teller kindly provided comments on earlier drafts of this chapter. 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.