Learning One More Thing

Sebastian Thrun Tom M. Mitchell

September 1994
CMU-CS-94-184

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Most research on machine learning has focused on scenarios in which a learner faces a single, isolated learning task. The lifelong learning framework assumes instead that the learner encounters a multitude of related learning tasks over its lifetime, providing the opportunity for the transfer of knowledge.

This paper studies lifelong learning in the context of binary classification. It presents the *invariance approach*, in which knowledge is transferred via a learned model of the invariances of the domain. Results on learning to recognize objects from color images demonstrate superior generalization capabilities if invariances are learned and used to bias subsequent learning.

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. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of NSF, Wright Laboratory or the United States Government.

Keywords: explanation-based neural network learning, learning invariances, lifelong learning, machine learning, neural networks, object recognition

1 Introduction

Standard inductive supervised learning is concerned with learning an unknown target function from a finite collection of training data. The framework of supervised learning can be characterized as follows. Let F denote the set of all potential target functions. For example, in a robot arm domain F might be the set of all kinematic functions for robots with three joints. It is commonly assumed that all functions in F are defined over a single input space, denoted by I, and a single output space, denoted by O. The learner has a set of hypotheses that it can consider, denoted by H, which might or might not be different from F. For example, the set H could be the set of all artificial neural networks with 20 hidden units [Rumelhart *et al.*, 1986], or, alternatively, the set of all decision trees with depth less than 10 [Quinlan, 1986]. Throughout this paper, let us make the simplifying assumption that all functions in F are binary classifiers, by restricting the output to $O = \{0, 1\}$. We will refer to instances that fall into class 1 as positive instances, and to those that fall into class 0 as negative instances.

To learn an unknown target function $f^* \in F$, the learner is given a finite collection of input-output examples (training examples)

$$X = \{\langle i, f^*(i)\rangle\},$$

which are possibly distorted by noise. The goal of the learner is to generate a hypothesis $h \in H$, such that the deviation

$$E \;=\; \sum_{i \in I} Prob(i)\; ||f^*(i) - h(i)||$$

between the target function f^* and h on future examples will be as small as possible. Here $Prob(i)$ is the probability distribution according to which the examples are generated. $Prob$ is generally unknown to the learner, as is f^*.

Standard supervised learning focusses on learning a single target function f^*, and training data is assumed to be available only for this one function. However, if functions in F are sufficiently related, it can be helpful to have access to training examples of other functions f in F as well. For example, consider a robot whose task is to find and fetch various objects, using its camera for object recognition. Here let F be the set of recognition functions for all objects, one for each potential target object, and the target function $f^* \in F$ corresponds to some object the robot must currently learn to recognize. X, the training set, will consist of positive and negative examples of this object. The task of the learner is to find an h which minimizes E. In order to do so, the robot must learn to recognize the target object invariant of rotation, translation, scaling in size, change of lighting and so on. Clearly, the more profound the learner's initial understanding of these invariances, the fewer training examples in X it will require for reliable learning. Because these invariances are common to all functions in F, images showing other objects can provide additional information for learning about these invariances, and hence augment the training set X.

This example illustrates the lifelong learning framework [Thrun and Mitchell, 1993]. In lifelong learning, a collection of related learning problems is encountered over the lifetime of the system. When learning the n-th task, the learner may therefore employ knowledge gathered in the

Given:

- a space of possible hypotheses H
- a set of training examples X of some unknown target function $f^* \in F$, drawn with probability distribution $Prob$.
- in lifelong learning: a collection of support sets $Y = \{X_k\}$, which characterize other functions $f_k \in F$

Determine:

a hypothesis $h \in H$ that minimizes $\sum_{i \in I} Prob(i) \, ||f^*(i) - h(i)||$

Table 1: Standard supervised learning and lifelong supervised learning.

---

previous $n-1$ tasks to improve its performance. This paper considers a particular form of lifelong learning in which the learning tasks correspond to learning boolean classifications (concepts), and in which the previous experience consists of training examples of other classification functions from the same family F. More formally, in addition to the set of training examples X for the target function f^*, the learner is also provided sets of examples

$$X_k \quad = \quad \{\langle i, g_k(i) \rangle\} \quad (k = 1, \ldots, |G|)$$

of other functions $G = \{g_1, g_2, \ldots\} \subset F$ taken from the same function family F. Since this additional data supports learning f^*, we shall call each X_k a *support set* for X. The set of all available support sets, $\{X_k : k = 1, \ldots, |G|\}$, is denoted by Y. Notice that the support sets X_k are not necessarily generated according to the same probability distribution.

Table 1 summarizes the problem definitions of standard supervised learning and the lifelong supervised learning problem considered here. In lifelong learning, the learner is given a collection of support sets Y in addition to the training set X and the hypothesis space Y.

Support sets can be defined for a variety of real-world learning tasks. For example, consider a personal computer assistant whose task it is to recognize its user's handwriting. F is the set of all mappings from hand-drawn curves to letter labels, one for each potential user. For each user, i.e., for each $f \in F$, the assistant could start learning to recognize handwriting from scratch. However, many characteristics will co-occur for multiple users, and one would expect a learner which incorporates examples from other users to generalize better. A second example, to which inductive machine learning techniques have frequently been applied, is stock market prediction. While many of the successful techniques in this domain remain unpublished, it seems to be well-accepted that stocks can be predicted better by learning from data regarding entire families of stocks, rather than learning about a single stock in isolation. Clearly, certain patterns can be found that apply to various stocks, and knowing about these regularities when training data is limited will most likely improve the prediction accuracy of a learning system.

The lifelong learning framework studied in this paper differs from standard supervised learning in that in addition to the training data X, support sets Y are also provided. This raises two fundamental questions:

1. How can a learner use support sets to generalize more accurately?
2. Under what conditions will a learner benefit from support sets? Obviously, the more closely related the functions in F, the better. But exactly what relation among these functions is required in order for support sets to be useful? What happens if the functions in F are not related at all? Can support sets mislead generalization and, if so, under what conditions?

This paper does not provide general answers to these questions. Instead, it proposes one particular approach, namely learning invariance functions, which relies on certain assumptions on the function set F. It also presents empirical evidence that this approach to using support sets can significantly improve generalization accuracy when learning to recognize objects based on visual data.

2 The Invariance Approach

The invariance approach first learns an invariance function σ from the support sets in Y. This function is then used to bias the learner as it selects a hypothesis to fit the training examples X of the target function.

2.1 Invariance Functions

Let $Y = \{X_k\}$ be a collection of support sets for learning f^* from X. Recall our assumption that all functions in F have binary output values. Hence, each example in a support set is either positive (i.e., output 1) or negative (i.e., output 0). Consider a target function, $f_k \in F$ with $k \in \{1, \ldots, |F|\}$, and a pair of examples, say $i \in I$ and $j \in I$. A local *invariance operator* $\tau_k : I \times I \longrightarrow \{0, 1\}$ is a mapping from a pair of input vectors defined as follows:

$$\tau_k(i,j) \quad = \quad \begin{cases} 1 & \text{if } f_k(i) = f_k(j) = 1 \\ 0 & \text{if } f_k(i) \neq f_k(j) \\ \text{not defined} & \text{if } f_k(i) = f_k(j) = 0 \end{cases}$$

Basically, the local invariance operator indicates whether both instances are members of class 1 (positive examples) relative to f_k. If $\tau_k(i,j) = 1$, then f_k is invariant with respect to i and j. Notice that positive and negative instances are not treated symmetrically in τ.

The local invariance operators τ_k ($k = 1, \ldots, |F|$) define a (global) *invariance function* for F, denoted by $\sigma : I \times I \longrightarrow \{0, 1\}$. For two examples, i and j, $\sigma(i,j)$ is 1 if there exists a k for which $\tau_k(i,j) = 1$. Likewise, $\sigma(i,j)$ is 0 if there exists a k for which $\tau_k(i,j) = 0$:

$$\sigma(i,j) \quad = \quad \begin{cases} 1 & \text{if } \exists k \in \{1, \ldots, |F|\} \text{ with } \tau_k(i,j) = 1 \\ 0 & \text{if } \exists k \in \{1, \ldots, |F|\} \text{ with } \tau_k(i,j) = 0 \\ \text{not defined} & \text{otherwise} \end{cases}$$

The invariance function behaves like an invariance operator, but it does not depend on k. It is important to notice that the invariance function can be ill-defined. This is the case if there exist two examples which in one target function both belong to class 1, whereas in another they fall into different classes:

$$\exists i, j \in I, k, k' \in \{1, \ldots, |F|\} : \quad \tau_k(i,j) = 1 \ \wedge \tau_{k'}(i,j) = 0$$

In such cases the invariance mapping is ambiguous and is not even a mathematical function. A class of functions F is said to obey the *invariance property* if its invariance function is non-ambiguous[1]. The invariance property is a central assumption for the invariance approach to lifelong classification learning.

The concept of invariance functions is quite powerful. Assume F holds the invariance property. If σ is known, every training instance i for an arbitrary function $f_k \in F$ can be correctly classified, given there is at least one positive instance of f_k available. To see, assume $i_{\text{pos}} \in I$ is known to be a positive instance for f_k. Then for any instance $i \in I$, $\sigma(i, i_{\text{pos}})$ will be 1 if and only if $f_k(i) = 1$. Although the invariance property imposes a restriction on the function family F, it holds true for quite a few real-world problems. For example, a function family obeys the invariance property if all positive classes (of all functions f_k) are disjoint. One such function family is the family of object recognition functions defined over distinct objects.

2.2 Learning with Invariants

In the lifelong learning regime studied in this paper, σ is not given. However, an approximation to σ can be learned. Since σ does not depend upon the specific target function f^*, every support set $X_k \in Y$ can be used to train σ, as long as there is at least one positive instance available in X_k. For all $k \in \{1, \ldots, |G|\}$, training examples for σ are constructed from examples $i, j \in X_k$:

$$\langle (i,j), \tau_k(i,j) \rangle$$

Here τ_K must be defined, i.e., at least one of the examples i and j must be positive under f_k. In the experiments described below, σ is approximated by training an artificial neural network using the Backpropagation algorithm [Rumelhart *et al.*, 1986].

Once σ has been learned, one way to mimic f^* is to pick an arbitrary positive training instance in X, and to use σ for classification, as described above. However, σ might not be accurate enough to classify correctly, usually because of modeling limitations, noise, or lack of training data. In fact, the experimental results described in the next section indicate that there are better ways to employ the invariance network.

2.3 Extracting Slopes to Guide Generalization

The remainder of this section describes an alternative approach which employs a hybrid neural network learning algorithm for learning f^*. This algorithm is a special case of both the Tangent-

[1]It is generally acceptable for the invariance function to be ambiguous, as long as the likelihood for generating ambiguously classified pairs of examples is zero.

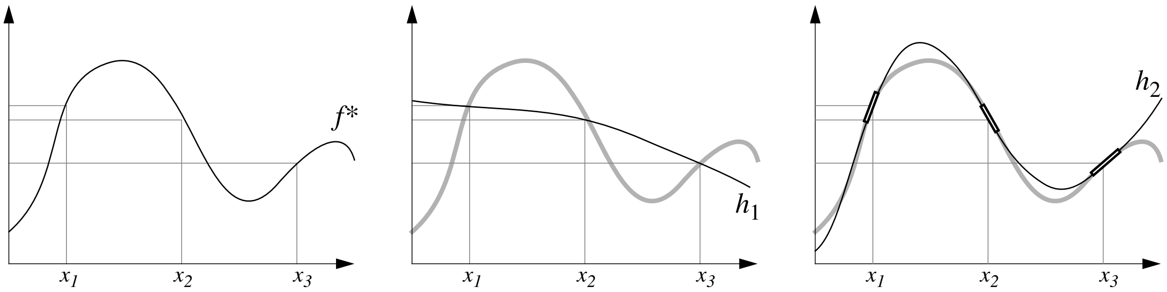

Figure 1: **Fitting values and slopes:** Let f^* be the target function for which three examples $\langle x_1, f^*(x_1)\rangle$, $\langle x_2, f^*(x_2)\rangle$, and $\langle x_3, f^*(x_3)\rangle$ are known. Based on these points the learner might generate the hypothesis h_1. If the slopes are also known, the learner can do much better: h_2.

Prop algorithm [Simard *et al.*, 1992] and the explanation-based neural network learning (EBNN) algorithm [Mitchell and Thrun, 1993]. Here we will refer to it as EBNN.

Suppose we are given a training set X, and an invariance network σ which has been trained using a collection of support sets Y. We are now interested in learning f^*. One could, of course, ignore the invariance network and the support sets altogether and train a neural network purely based on the training data X. The training set X imposes a collection of constraints on the output values for the hypothesis h. If h is represented by an artificial neural network, as is the case in the experiments reported below, the Backpropagation (BP) algorithm can be used to fit X.

EBNN does this, but it also derives additional constraints using the invariance network. More precisely, in addition to the value constraints in X, EBNN derives constraints on the *slopes* (tangents) for the hypothesis h. To see how this is done, consider a training example i, taken from the training set X. Let i_{pos} be an arbitrary *positive* example in X. Then, $\sigma(i, i_{\text{pos}})$ determines whether i and i_{pos} belong to the same class— information that is readily available, since we are given the classes of i and i_{pos}. However, predicting the class using the invariance network also allows us to determine the output-input slopes of the invariance network. These slopes measure the sensitivity of class membership with respect to the input features in i. This is done by computing the partial derivative of σ with respect to i at (i, i_{pos}):

$$\nabla_i \sigma(i) \quad := \quad \frac{\partial \sigma(i, i_{\text{pos}})}{\partial i}$$

$\nabla_i \sigma(i)$ measures how infinitesimal changes in i will affect the classification of i. Since $\sigma(\cdot, i_{\text{pos}})$ is an approximation to f^*, $\nabla_i \sigma(i)$ approximates the slope $\nabla_i f^*(i)$. Consequently, instead of fitting training examples of the type $\langle i, f^*(i)\rangle$, EBNN fits training examples of the type

$$\langle i, f^*(i), \nabla_i f^*(i)\rangle.$$

Gradient descent can be used to fit training examples of this type, as explained in [Simard *et al.*, 1992]. Fig. 1 illustrates the utility of this additional slope information in function fitting.

Notice if multiple positive instances are available in X, slopes can be derived for each one of

1. Let $X_{\text{pos}} \subset X$ be the set of positive training examples in X.

2. Let $X' = \emptyset$

3. For each training example $\langle i, f^*(i) \rangle \in X_{\text{pos}}$ do:

(a) Compute $\nabla_i \sigma(i) \;=\; \frac{1}{|X_{\text{pos}}|} \sum_{i_{\text{pos}} \in X_{\text{pos}}} \frac{\partial \sigma(i)(i_{\text{pos}})}{\partial i}$ using the invariance network σ.

(b) Let $X' \;=\; X' \;+\; \langle i, f^*(i), \nabla_i \sigma(i) \rangle$

4. Fit X'.

Table 2: Application of EBNN to learning with invariance networks.

---

them. In this case, averaged slopes are used to constrain the target function:

$$\nabla_i \sigma(i) \;:=\; \frac{1}{|X_{\text{pos}}|} \sum_{i_{\text{pos}} \in X_{\text{pos}}} \frac{\partial \sigma(i, i_{\text{pos}})}{\partial i} \tag{1}$$

Here $X_{\text{pos}} \subset X$ denotes the set of positive examples in X. The application of the EBNN algorithm to learning with invariance networks is summarized in Table 2.

Generally speaking, slope information extracted from the invariance network is a linear approximation to the variances and invariances of F at a specific point in I. Along the invariant directions slopes will be approximately zero, while along others they will be large. For example, in the object recognition domain described above, it might happen that color is an important feature for classification while brightness is not. This is typically the case in situations with changing illumination. In this case, the invariance network could learn to ignore brightness, and hence the slopes of its classification with respect to brightness would be approximately zero. The slopes for color, however, would be large, given that slight color changes imply that the object would belong to a different class.

When training the classification network, slopes provide additional information about the sensitivity of the target function with respect to its input features. Hence, the invariance network can be said to bias the learning of the classification network. However, since EBNN trains on both slopes and values simultaneously, errors in this bias (incorrect slopes due to approximations in the learned invariance network) can be overturned by the observed training example values in X.

3 Example

3.1 Object Recognition

To illustrate the invariance network in a real-world domain, we collected a database of 700 color camera images of seven different objects, as depicted in Fig. 2 (left columns).

Object	color	size
bottle	green	medium
hat	blue and white	large
hammer	brown and black	medium
can	red	medium
book	yellow	depending on perspective
shoe	brown	medium
glasses	black	small

The objects were chosen so as to provide color and size cues helpful to their discrimination. The background of all images consisted of plain, white cardboard. Different images of the same object varied by the relative location and orientation of the object within the image. In 50% of all recordings, the location of the light source was also changed, producing bright reflections at random locations in various cases. In some of the images the objects were back-lit, in which case they appeared to be black. Fig. 3 shows examples of two of the objects, the shoe and the glasses. 100 images of each object were recorded.

In all our experiments images were encoded by a 300-dimensional vector, providing color, brightness and saturation information for a down-scaled image of size 10 by 10. Examples for the down-scaled images are shown in Figures 2 (right columns) and 3. Although each object appears to be easy to recognize from the original image, in many cases we found it difficult to visually classify objects from the subsampled images. However, subsampling was necessary to keep the networks at a reasonable size.

The set of target functions, F, was the set of functions that recognize objects, one for each object. For example, the indicator function for the bottle, f_{bottle}, was 1, if the image showed a bottle, and 0 otherwise. Since we only presented distinct objects, all sets of positive instances were disjoint. Consequently, F obeyed the invariance property. The set of hypotheses H was the set of all artificial neural networks with 300 input units, 6 hidden units, and 1 output unit, as such a network was employed to represent the target function.

The objective was to learn to recognize shoes, i.e., $f^* = f_{\text{shoe}}$. Five other objects, namely the bottle, the hat, the hammer, the can and the book, were used to construct the support sets Y. In order to avoid any overlap in the training set X and the support sets in Y, we exclusively used pictures of a seventh object, glasses, as counterexamples for f_{shoe}.

3.2 Training the Invariance Network

Each of the five support sets in Y, X_{bottle}, X_{hat}, X_{hammer}, X_{can} and X_{book}, contained 100 images of the corresponding object (positive examples) and 100 randomly selected images of other objects (negative examples). When constructing training examples for the invariance network, we randomly selected a subset of 1,000 pairs of images, 800 of which were taking for training and 200 for cross-validation. 50% of the final training and cross-validation examples were positive examples for the invariance network (i.e., both images showed the same object), and the other 50% were negative examples.

Figure 2: Objects (left) and corresponding network inputs (right). A hundred images of a bottle, a hat, a hammer, a coke can, and a book were used to train and test the invariance network. Afterwards, the classification network was trained to distinguish the shoe from the glasses.

In several attempts to construct an invariance network, using a variety of network topologies with up to two hidden layers, we achieved a maximum generalization accuracy of 62.0%. This result was somewhat unsatisfactory, since random guessing, by comparison, results in 50% accuracy. When applied to the two remaining unseen objects, the shoe and the glasses, the best invariance network classified only 53.2% of all image pairs correctly.

In order to improve these results, we applied a learning technique that focusses learning by incorporating additional training information, adopted from [Suddarth and Kergosien, 1990], [Caruana, 1993]. Their technique rests on the assumption that in addition to the learning task of interest, some related learning tasks, using the same input representation and the same training data (with different target values), are available. Instead of training on a single task, a network is

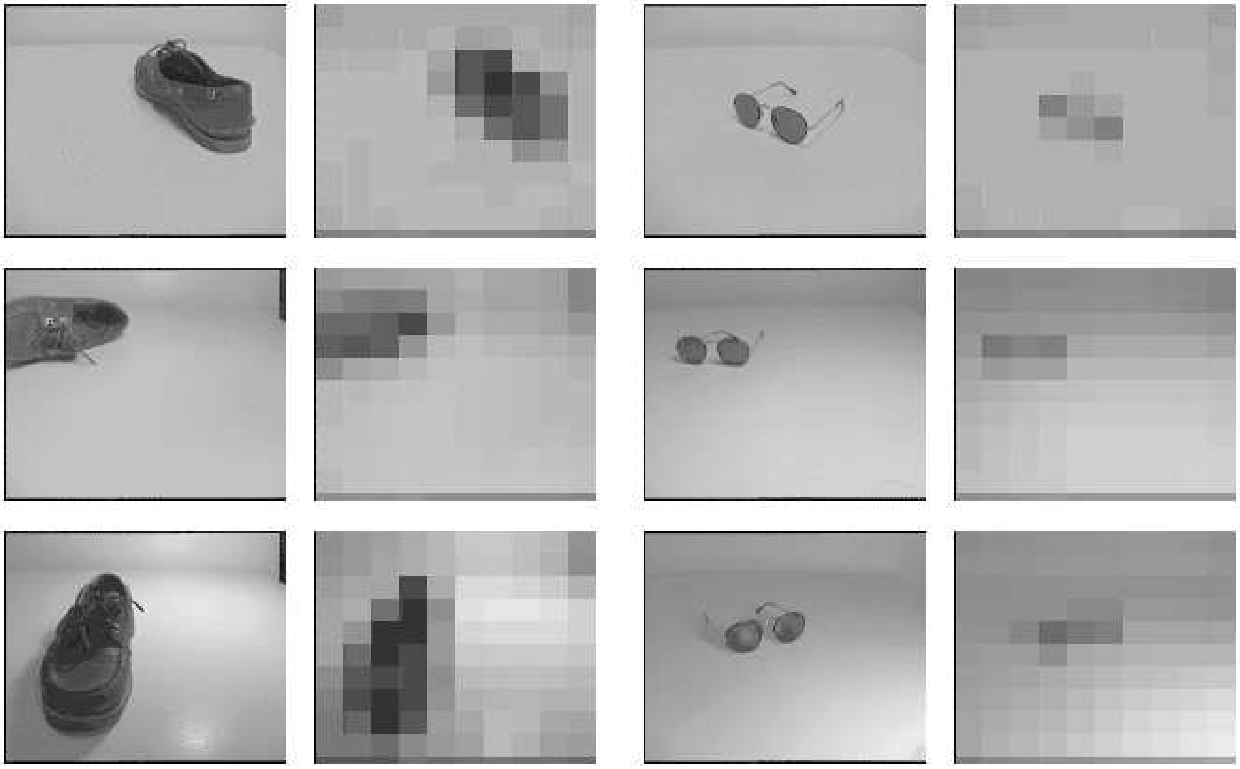

Figure 3: Images, along with the corresponding network inputs, of the objects shoe and glasses. These examples illustrate some of the invariances in the object recognition domain.

trained on all tasks simultaneously, using an augmented output layer that provides additional output units for the additional tasks. This technique, which is called "learning by hints" or "multi-task learning," has been found to yield better generalization accuracies, which can be attributed to the fact that all of these tasks share the same hidden units. If tasks are sufficiently related, it allows better hidden representations to be developed, resulting in improved generalization. In fact, this approach establishes an alternative method for the lifelong learning problem, as discussed in Sect. 4.

In the object recognition domain, a task that is obviously related to determining whether or not two images belong to the same class is the task of actually classifying the two images. Hence, we added two sets of 5 output units to the invariance network, which were trained to determine the classification of the object shown in either image. A local $1\!-\!\text{of}\!-\!n$ encoding was used to encode the 5 different object classes. Hence, the augmented invariance network had 11 output units, one for determining if the two images are the same or not and 10 for classifying images. The latter 10 units, however, were used exclusively during training the invariance network, and did not play any part in subsequently applying the invariance network. The classification accuracy of this network was significantly better than the accuracy of the single output network reported above. After training, the augmented network managed to determine whether or not two objects belong to the same class with 79.5% generalization accuracy. It also exhibited 67.0% classification accuracy in the new task, the recognition of the shoe. Obviously, both accuracy rates are significantly better than those achieved using the single output network.

3.3 Learning to Recognize Shoes

Having trained the invariance network, we were now interested in training the classification network. The network used throughout the experiments reported here consisted of 300 input units, 6 hidden units, and 1 output unit—no effort was made to optimize the network topology. A total of 200 examples of images showing the shoe and the glasses were available for training and testing the shoe classification network.

The central question regarding the invariance approach is to what extent the invariance network, when used to bias the target function, improves the generalization accuracy of the classification network.

In order to elucidate the role of the invariance network, we trained the classification network using only two training examples: a randomly selected image of the shoe (positive example), and a randomly selected image of the glasses (negative example). Slopes were computed using the previously learned invariance network. Since the counterexamples to the target concept, the glasses, form a unique class of images that do not overlap with any other positive class from the support sets, slopes could also be derived using negative examples. Instead of using Eq. (1), slopes were extracted from the invariance net using the extended mixture:

$$\nabla_i \sigma(i) \quad := \quad \frac{1}{|X|} \left(\sum_{i_{\text{pos}} \in X_{\text{pos}}} \frac{\partial \sigma(i, i_{\text{pos}})}{\partial i} - \sum_{i_{\text{neg}} \in X_{\text{neg}}} \frac{\partial \sigma(i, i_{\text{neg}})}{\partial i} \right) \tag{2}$$

Here $X_{\text{pos}} \subset X$ is the set of positive examples in X, and $X_{\text{neg}} = X - X_{\text{pos}}$ is the set of negative examples. Eq. (2) differs from Eq. (1) by taking also negative examples i_{neg} into account, which is justified by the fact that images of glasses form a class disjoint from all other objects.

Fig. 4 shows the average generalization curve as a function of training epochs with and without the invariance network. The generalization accuracy here was measured over all 200 available images. The curve shows the generalization accuracy averaged over 100 experiments, each trained using one randomly selected positive and one randomly selected negative example.[2] Without using the invariance approach, the average generalization accuracy after 10,000 training epochs is 59.7%. However, using EBNN with the invariance network increases accuracy to 74.8% due to the information conveyed by the invariance slopes. This difference can be assessed in multiple ways. In terms of residual error, Backpropagation exhibits a misclassification rate that is 60.1% larger than that of EBNN. A second interpretation is to look at the *performance increase*, which is defined as the difference in classification accuracy after learning and before learning, assuming that the accuracy before learning is 50%. EBNN's performance increase is 24.8%, which is 2.6 times better than Backpropagation's 9.7%.

For example, if a neural network is trained using the two images of a shoe and the glasses depicted in Fig. 2, plain Backpropagation classifies only 52.5% of the testing images correctly.

[2]Note we used a fast learning method that adapted the amount of momentum on-line during learning.

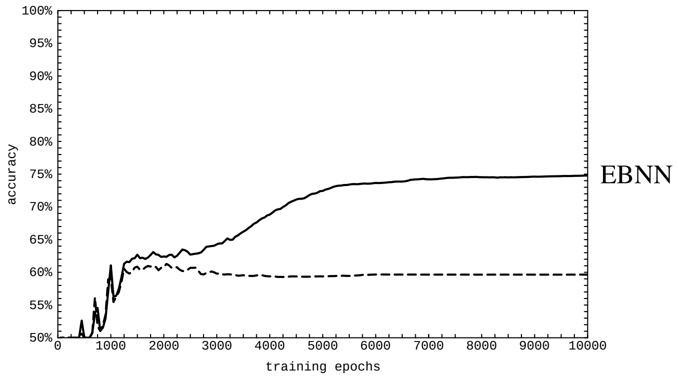

Figure 4: Training curves, with (solid line) and without (dashed line) the invariance network and EBNN, measured on an independent test set and averaged over 100 runs, after providing one positive and one negative training example.

Here the generalization rate is particularly poor, since the location of the objects differ, and Backpropagation mistakenly considers location the crucial feature for object recognition. EBNN using the invariance network produces a network that is much less sensitive to object location, resulting in a 85.5% generalization accuracy in this particular example.

Note that the network learned by EBNN performs significantly better than the invariance network itself used as a classifier (67.0%). This is because EBNN is trained on images of the shoe and the glasses, while the invariance network is not.

The importance of mixing slopes becomes clear by looking at the accuracy that is achieved when slopes are computed differently. If slopes are only extracted by comparing an image with itself (i.e., both images of the invariance network are equivalent), the average final accuracy is 66.4%. When only pairwise different images are used as input to the invariance network, the resulting accuracy is 71.8%. Both accuracies are significantly smaller than the 74.8% accuracy achieved by mixing the two. We also tried experiments weighting the mixtures of slopes. In one case, we used the prediction accuracy of the domain theory (LOB*) as the weighting factor: The more accurate the invariance network for a particular training example, the stronger the weight of the corresponding slope in the mixture. This strategy, which has been found to be useful across a variety of domains [Mitchell *et al.*, 1994], [Thrun, 1994] resulted about equivalent performance (74.1%) in the object recognition domain.

The reader should notice that all these results refer to the classification accuracy after 10,000 training epochs, using just one positive and one negative training example. As can be seen in Fig. 4, Backpropagation suffers from some over-fitting, as the accuracy drops after a peak at about 2,050 training epochs. The average classification accuracy at this point in time is 61.3%. However, due to lack of data, it is impossible in this domain to use early stopping methods that rely on cross validation, and it is unclear whether such methods would have improved the results for Backpropagation significantly.

Fig. 5a shows analogous results for training with two examples of both shoes and glasses. Here, the difference between EBNN and Backpropagation is even wider. EBNN achieves 82.9%

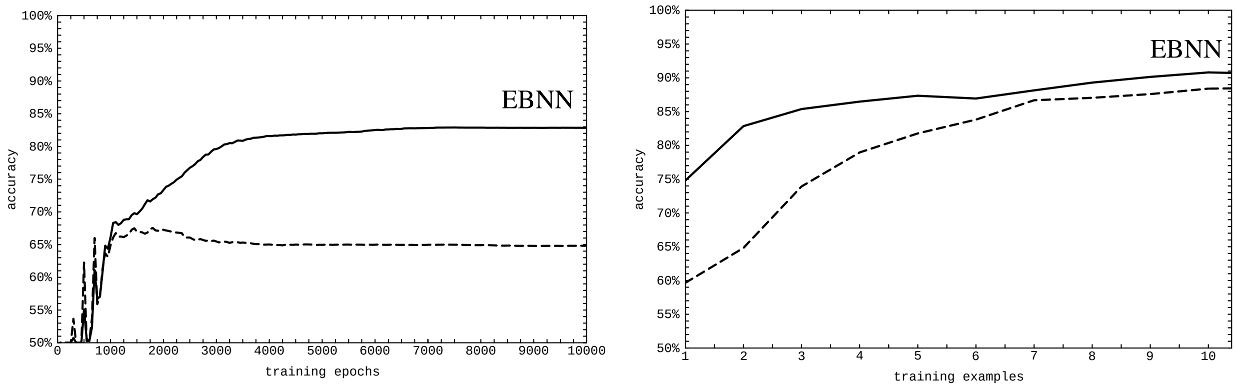

Figure 5: (a) Averaged generalization accuracy when training on two positive and two negative training examples. (b) Generalization accuracy for different number of training examples after 10,000 training epochs.

---

final accuracy, as opposed to 64.8% by plain Backpropagation. Consequently, Backpropagation's misclassification rate exceeds that of EBNN by 105.8%. EBNN's performance increase is 32.9%, which is 2.21 times better than Backpropagation's 14.8%.

Results for experiments with larger training set sizes are depicted in Fig. 5b. As can be seen from this figure, the difference between the methods decreases as the number of training instances increases. EBNN, however, continues to perform slightly better than plain Backpropagation. This matches our expectations, as the need for background knowledge decreases as the number of training examples increases. However, the primary focus of this paper is learning when training data in X is scarce.

3.4 The Role of the Invariance Network

The improved classification rates, which illustrate the successful transfer of knowledge from the support sets via the invariance network, raise the question what exactly are the invariances represented in this network. What type information do the slopes convey?

A plausible (but only approximate) measure of the importance of a feature is the magnitude of its slopes. The larger the slopes, the larger the effect of small changes in the feature on the classification, hence the more relevant the feature. In order to empirically assess the importance of features, average slope magnitudes were computed for all input pixels, averaged over all 100 pairs of training instances. The largest average slope magnitude was found for color information: 0.11. In comparison, saturation slopes were, on average, only 0.063 (this is 57% of the average for color slopes), and brightness slopes only 0.056 (51%).

These numbers indicate that, according to the invariance network, color information was most important for classification. To verify this hypothesis, we repeated our experiments omitting some of the image information. More specifically, in one experiment color information was omitted, in a second saturation, and brightness in a third. The results

	standard supervised	with support sets
invariance network classification, no EBNN		67.0%
1 training example per class	59.7%	74.8%
same, after 2,050 training epochs	61.3%	
image shown in Fig. 3	52.5%	85.5%
2 training examples per class	64.8%	82.9%
only objects of same class used for deriving slopes		66.4 %
only objects of different class used for deriving slopes		71.8%
weighting slopes by classification accuracy		74.1%
no color	52.4%	57.9%
no saturation	59.0%	72.9%
no brightness	58.7%	76.3%

Table 3: Summary of the classification results listed in the paper. All numbers are average classification rates on unseen testing data.

---

	without invariance network	with invariance network
no color	52.4%	57.9%
no saturation	59.0%	72.9%
no brightness	58.7%	76.3%
full information	59.7%	74.8%

confirmed our belief that color information indeed dominates classification. It is clear that without color the generalization rates in the testing set are poor, although EBNN still generalizes better. If saturation or brightness is omitted, however, the generalization rate is approximately equivalent to the results obtained for the full images reported above. However, learning required significantly more training epochs in the absence of brightness information (not shown here). These and other results reported here are summarized in Table 3.

Fig. 6 shows average slope matrices for the target category (shoes) with respect to the three input features, color, brightness and saturation. Grey colors indicate that the average slope for an input pixel is zero. Bright and dark colors indicate strongly positive and strongly negative slopes, respectively. Notice that these slopes are averaged over all 100 explanations used for training.

As is easily seen, average color slopes vary over the image, showing a slight positive tendency on average. Average saturation slopes are approximately zero. Brightness slopes, however, exhibit a strong negative tendency which is strongest in the center of the image. One possible explanation for the latter observation is the following: Both the shoe and the glasses are dark compared to the background. Shoes are, on average, larger than glasses, and hence fill more pixels. In addition, in the majority of images the object was somewhere near the center of the image, whereas the border pixels showed significantly more noise. Lack of brightness in the image center is therefore a good indicator for the presence of the shoe, as is clearly reflected in the brightness slopes derived

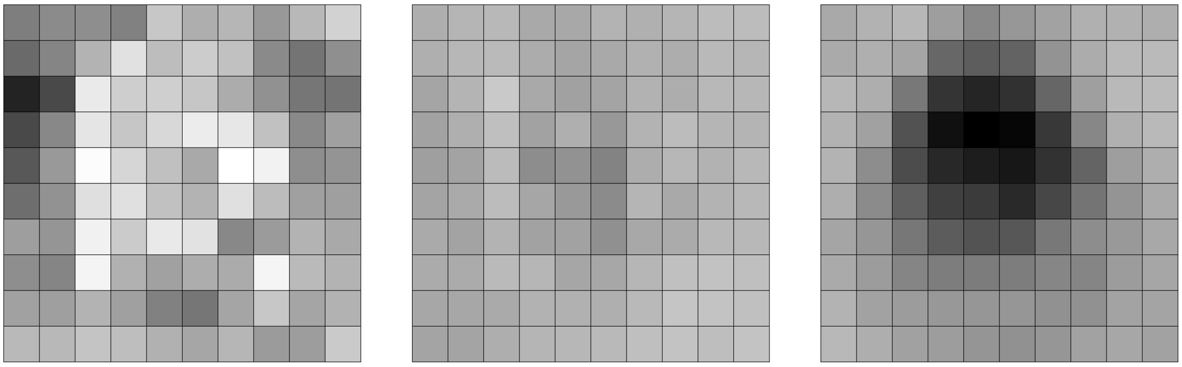

Figure 6: Slopes of the target concept (glasses) with respect to (a) color, (b) saturation, and (c) brightness. Every slope is averaged over 400 explanations. White (black) color represents positive (negative) values.

from the invariance network. The less obvious results for color and saturation can be attributed to the effect that optimal classifiers are non-linear in color and saturation. In order to discriminate objects by color, for example, the network has to spot a specific interval in color space. Hence, the correct slopes can be either positive or negative depending in the particular color of a pixel, cancelling each other out in this plot.

As pointed out earlier, slopes provide first-order information, and invariances may well be hidden in higher-order derivatives. However, both the superior performance of EBNN as well as the clear correlation of slope magnitudes and generalization accuracy show that EBNN manages to extract useful invariance information in this domain, even if these invariances defy simple interpretation.

4 Alternative Approaches

While in this paper we have presented one particular approach to lifelong learning in the context of classification, many others are possible and several can be found in recent literature.

- **Learning internal representations.** Other researchers report techniques to develop more appropriate hidden layer representations from multiple tasks. For example, Pratt proposed a method which transfered information by using an internal representation that was developed in earlier learning tasks [Pratt, 1993]. A similar technique has been proposed in [Sharkey and Sharkey, 1992]. A second example of learning internal representations using multiple target functions is Caruana's multi-task learning algorithm. In his approach, multiple, related tasks are trained simultaneously in a single neural network, forcing the networks to share hidden units. He reports that hidden internal representations are developed which lead to improved generalization [Caruana, 1993]. Notice that these results match our findings when training the invariance network. All these approaches develop better internal representations of the data by considering multiple functions in F with the goal of improving generalization.

- **Spotting relevant features.** Another approach, which bears close resemblance to learning invariances and learning representations, is to spot irrelevant features [Littlestone, 1987], [Caruana and Freitag, 1994]. If the set of target functions F is such that—across the board—only a subset the features is relevant (e.g., the time of day may not matter for object recognition), a learning system can employ support sets to find the most relevant features. Once they are discovered, the remaining hypothesis space is smaller, which reduces the sample complexity in learning. Notice that EBNN weakens the influence of irrelevant features through zero-valued slopes EBNN.

- **Adapting the data.** A different approach to lifelong learning is to modify the data, either in the training set X or in the support sets Y. For example, imagine there is a general-purpose module that can be applied to all functions in F, but it requires that data is pre-filtered, and each $f \in F$ requires an individual filter. This is the case, for example, in approaches to speaker adaptation. Speaker adaptation comprises a family of techniques studied in speech recognition, in which a computer quickly adapts to the accent, voice, pitch, or speed of an individual speaker (see [Hild and Waibel, 1993] for an example). Typically, speech is translated to a more machine-understandable speech by a user-specific module that allows quick adaptation. Speaker adaptation is an example of an approach in which training data X is adapted to fit previously learned modules.

In essence, all these approaches change the bias of the function approximator. They differ in the way bias is represented, and in the assumption they make on the underlying function class F.

A variety of approaches aim to change the bias of an inductive function approximator in a more direct way. For example, Sutton [Sutton, 1992] describes an approach that employs Kalman filters to determine optimal learning rates. Atkeson [Atkeson, 1991] proposes techniques for optimizing the distance metric in a nearest neighbor generalizer. In [Maron and Moore, 1994], an incremental method for the selection of nearest neighbor models is described. Starting with a hypothesis set H, this technique uses training data and cross-validation to gradually reduce H. However, many of the approaches listed here have not been proposed in the context of learning more than one task.

It should be noted that the invariance approach bears some resemblance to training schemes found in the context of autonomous driving and letter recognition. Simard and colleagues [Simard *et al.*, 1992] employed the Tangent-Prop algorithm for recognizing hand-printed letters. Since letter recognition should be invariant to translation and rotation, they manually provided zero-valued target slopes in the directions of these invariances, very much like those automatically generated by EBNN. Pomerleau [Pomerleau, 1989] reports a similar technique that was used for training an autonomous vehicle. Based on knowledge about the relation of camera images and steering direction, he constructed additional training data that were used when training the network. His technique can be viewed as an approximative version of Simard's approach. In both cases, domain knowledge was employed to incorporate invariances into neural network learning. The invariance approach presented here differs in that a model of the invariances is constructed from training data, making it applicable in situations where the appropriate expert knowledge is not available from a human designer.

5 Discussion

In the lifelong learning framework, the learner faces a collection of related learning tasks. The challenge of this framework is to transfer knowledge across tasks, in order to generalize better from fewer training examples of the target function itself.

The experimental results presented in this paper provide clear evidence of superior generalization in the object recognition domain, when invariances learned from related tasks are used to augment the training data for a new object recognition task. However, the success of this invariance approach relies on several critical assumptions:

- Well defined invariance functions rest on the assumption that F obeys the invariance property. However, even if the invariance property is only approximately satisfied by F, the support sets can be used to train an invariance network. This network will, in the ideal case, approach the *expected* σ-value. The object recognition domain presented above provides an example in which the invariance property may hold only approximately. This is because different objects may look alike in the coarse-grained, noisy images, in which case they violate the invariance property.
- It is also assumed that functions in F possess certain invariances which can actually be learned by the invariance network. This fact does not follow from the invariance property. The exact invariances that will be learned depend crucially on the input representation and function approximator used for σ.
- We also assumed that the output space O of functions in f is binary. However, this assumption is not essential for the invariance approach. In principle, invariance functions may be defined for arbitrary, high-dimensional output spaces, given that a notion of difference between output vectors is available. For example, if the function space F contains multi-dimensional real-valued functions of the type $f : \Re^m \longrightarrow \Re^n$, the canonical vector difference $\tau_k : \Re^{2m} \longrightarrow \Re^n$ with $\tau_k(i,j) = |f_k(i) - f_k(j)|$ $(i, j \in \Re^m)$ establishes a local invariance operator and hence an invariance function. This function, however, differs from σ used in the binary case in that no clear class boundaries are defined, and no distinction is made between positive and negative examples.

In the experiments reported above, all three assumptions were at least approximately fulfilled. We conjecture that the real world offers a variety of tasks where learned invariances can boost generalization. For example, problems such as face recognition, cursive handwriting recognition, stock market prediction and speech recognition, which possess non-trivial but important invariances. For example, consider the problem of learning to recognize faces of various individuals. Here certain aspects are important for the successful recognition (e.g., the distance between the eyes), whereas others are less important (e.g., the direction in which the person is looking). After training on a number of individuals, we conjecture that the invariance network might grasp some of these invariances, reducing the difficulty of learning faces of new individuals.

It should be noted that in this paper lifelong learning is applied to one particular class of learning problems, namely binary classification tasks. In this context, several restrictive assumptions have been made, most of which are adopted from standard supervised learning. It is assumed that all functions (and hence all instances in X and Y) are drawn from a monolithic function class F, in which all functions share the same input and output space. Moreover, O is restricted to be $\{0, 1\}$. In applying the invariance network, further assumptions were made on the relation of the target functions in F.

The lifelong learning framework, however, is more general. It only specifies that a learner encounters a multitude of related learning tasks over its entire lifetime. The task of the learner need not necessarily be classification. For example, lifelong learning can be studied in function approximation, unsupervised learning, control learning or other learning paradigms. It is also not required that the support sets, which contain related training data, and the training set stem from a class of functions sharing a common input and output space. For example, control learning may benefit from training data collected in a classification domain, although the control function operates over different input and output spaces.

For example, in [Mitchell and Thrun, 1993] and [Thrun, 1994] control learning is studied in a lifelong learning framework. In control learning, the target function f^* is a control function which maps percepts, denoted by $s \in S$, to actions, denoted by $a \in A$:

$$f^* : S \longrightarrow A$$

Actions, when executed by the agent, result in some scalar penalty/reward, and the goal of learning is to maximize reward. A popular method for learning control is reinforcement learning [Barto *et al.*, to appear], [Sutton, 1990], [Watkins and Dayan, 1992]. Reinforcement learning constructs value function that can be used to select optimal actions. Embedded in a lifelong learning framework, a control learning agent may face a variety of control learning tasks over its lifetime. As shown in [Mitchell and Thrun, 1993], learning action models, which are functions of the type

$$g : S \longrightarrow S,$$

can significantly reduce the amount of training required for subsequent control learning tasks. EBNN is used to transfer knowledge across tasks, as in the invariance approach presented in this paper.

The lifelong learning problem can also be understood as a meta-level learning problem. Consider, for example, its application to classification, as presented in this paper. Each training example at the meta-level corresponds to a whole support set X_k at the base-level. The set of support sets $Y = \{X_k\}$ thus forms the set of training examples at the meta-level. The testing set is X, which is the set of training examples for the target function f^* at the base-level. A convenient assumption to be made is that the base-level hypothesis space H is a superset of F. Then, the hypothesis space at the meta-level is a set of restrictions on H, or, in other words, a set of subsets of H. The goal of learning on the meta-level is to reconstruct F (or find a minimal superset of F) as a hypothesis space for the base-level. Each training example X_k reduces the meta-level hypothesis space. If F is a candidate hypothesis in the meta-hypothesis space, one expects that in the limit $H = F$ is the only hypothesis left at the meta-level. Since each support set characterizes a function in F, every training example at the meta-level will be a positive example of the target concept F. Clearly, there can be no useful bias-free learning at the meta-level any more than there can be at the base-level.

However, despite the striking similarities between standard supervised learning and meta-learning, there are significant differences. Given a particular target function, $f^* \in F$, the ultimate goal of learning is to minimize the prediction error for f^*. Recognizing F is a secondary goal, since it is useful only in support of learning f^*. X, which establishes a single testing pattern in the meta-level, does not specify f^* uniquely. Instead, it provides a potentially small and noisy set of input-output examples of f^*. In addition, examples on the meta-level may vary in length, since the number of training examples in a support set may vary. In order to learn at the meta-level, more flexible encodings are needed than those that are typically studied in supervised learning. The invariance network establishes one particular such encoding, which works only if the original function space F holds the invariance property. The invariance network does not directly describe a hypothesis class—rather, it imposes shape constraints that, when incorporated into the training of the base-level recognizer, constrain its space of hypotheses.

The central question of this paper is whether learning can be made easier when the learner has already learned other related tasks. Will a system that is "trained" to learn generalize better than a novice learner? This paper provides encouraging results in an object recognition domain. However, most questions that arise in the context of lifelong learning still lack satisfactory, more general answers. We expect that future research in this direction will be important to going beyond the intrinsic bounds associated with learning single isolated functions.

Acknowledgment

We thank Astro Teller for thoughtful comments on an earlier draft of this paper.

References

[Atkeson, 1991] Christopher A. Atkeson. Using locally weighted regression for robot learning. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 958–962, Sacramento, CA, April 1991.

[Barto *et al.*, to appear] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, to appear.

[Caruana and Freitag, 1994] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, San Mateo, CA, 1994. Morgan Kaufmann.

[Caruana, 1993] Richard Caruana. Multitask learning: A knowledge-based of source of inductive bias. In Paul E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48, San Mateo, CA, 1993. Morgan Kaufmann.

[Hild and Waibel, 1993] Hermann Hild and Alex Waibel. Multi-speaker/speaker-independent architectures for the multi-state time delay neural network. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages II 255–258. IEEE, April 1993.

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

[Maron and Moore, 1994] Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems 6*, San Mateo, CA, 1994. Morgan Kaufmann.

[Mitchell and Thrun, 1993] Tom M. Mitchell and Sebastian B. Thrun. Explanation-based neural network learning for robot control. In S. J. Hanson, J. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 287–294, San Mateo, CA, 1993. Morgan Kaufmann.

[Mitchell *et al.*, 1994] Tom M. Mitchell, Joseph O'Sullivan, and Sebastian B. Thrun. Explanation-based learning for mobile robot perception. In *Workshop on Robot Learning, Eleventh Conference on Machine Learning*, 1994.

[Pomerleau, 1989] D. A. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. Technical Report CMU-CS-89-107, Computer Science Dept. Carnegie Mellon University, Pittsburgh PA, 1989.

[Pratt, 1993] Lori Y. Pratt. Discriminability-based transfer between neural networks. 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.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

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

[Sharkey and Sharkey, 1992] Noel E. Sharkey and Amanda J.C. Sharkey. Adaptive generalization and the transfer of knowledge. In *Proceedings of the Second Irish Neural Networks Conference*, Belfast, 1992.

[Simard *et al.*, 1992] 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. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903, San Mateo, CA, 1992. Morgan Kaufmann.

[Suddarth and Kergosien, 1990] Steven C. Suddarth and Y. L. Kergosien. Rule-injection hints as a means of improving network performance and learning time. In *Proceedings of the EURASIP Workshop on Neural Networks*, Sesimbra, Portugal, Feb 1990. EURASIP.

[Sutton, 1990] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning, June 1990*, pages 216–224, 1990.

[Sutton, 1992] Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceeding of Tenth National Conference on Artificial Intelligence AAAI-92*, pages 171–176, Menlo Park, CA, July 1992. AAAI, AAAI Press/The MIT Press.

[Thrun and Mitchell, 1993] Sebastian B. Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 1993. (to appear). Also appeared as Technical Report IAI-TR-93-7, University of Bonn, Dept. of Computer Science III.

[Thrun, 1994] Sebastian B. Thrun. A lifelong learning perspective for mobile robot control. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, September 1994. (to appear).

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