Multitask Connectionist Learning

Richard A. Caruana
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
caruana@cs.cmu.edu

The standard methodology in connectionism is to break hard problems into simpler, reasonably independent subproblems, learn the subproblems separately, and then recombine the learned pieces [15]. This modularization can be counterproductive because it eliminates a potentially critical source of inductive bias: the inductive bias inherent in the similarity between different tasks drawn from the same domain. Hinton [5] proposed that generalization in artificial neural networks improves if networks learn to represent underlying regularities of the domain. A learner that learns related tasks *at the same time* can use these tasks as inductive bias for each other and thus better learn the the domain's regularities. This can make learning faster and more accurate and may allow some hard tasks to be learned that could not be learned in isolation. This paper explores the issue of enabling connectionist networks to learn domain regularities by training networks on multiple tasks drawn from the same domain.

MULTITASK LEARNING AND INDUCTIVE BIAS

We learn embedded in a world that simultaneously requires us to learn many related things. These things obey the same physical laws, derive from the same human culture, are preprocessed by the same sensory hardware.... Perhaps it is the similarity of the tasks we learn that enables us to learn so much with so little experience.

A child trained from birth on a single, isolated, complex task—and nothing else—would be unlikely to learn it. If you were trained from birth to play tennis—and nothing else—you would probably not learn tennis. Instead, you learn to play tennis in a world that asks you to learn many other things. You also learn to walk, to run, to jump, to exercise, to grasp, to throw, to swing, to recognize objects, to predict trajectories, to rest, to talk, to study, to practice, etc. These tasks are not the same—running in tennis is different from running on a track—yet they are related. Perhaps the similarities between the thousands of tasks you learn are what enable you to learn any one of them, including tennis, given so little training data.

An artificial neural network trained tabula rasa on a single, isolated, complex task is unlikely to learn it. A net with a 1000 by 1000 pixel input retina will be unlikely to learn to recognize complex objects in the real world given the number of training patterns and amount of training time we can afford. It may be better to embed the net in an environment that simultaneously tasks it to learn many related tasks. If these tasks can share computed subfeatures, the net may find that it is easier to learn the tasks together than to learn them in isolation. Thus if we simultaneously train the net to recognize objects and object outlines, shapes, edges, regions, subregions, textures, reflections, highlights, shadows, text, orientation, size, distance, etc., it may be better able to learn to recognize complex objects in the real world. This is Multitask Connectionist Learning.

Multitask Learning (MTL) is predicated on the notion that tasks can serve as mutual sources of inductive

bias[1] for each other. MTL is one particular kind of inductive bias. It uses the information contained in the training signal of related tasks to bias the learner to hypotheses that benefit multiple tasks. One does not usually think of training data as a bias; but when the training data contains the teaching signal for more than one task, it is easy to see that, from the point of view of any one task, the other tasks' training signal may serve as bias. For this multitask bias to exist, the inductive learner must also be biased (typically via a simplicity bias) to prefer hypotheses that have utility across multiple tasks.

AN EXAMPLE OF MULTITASK CONNECTIONIST LEARNING

Consider the following four boolean functions defined on 8 binary inputs:

```
Task 1  =    B1   OR   Parity(B2-B8)
Task 2  =   -B1   OR   Parity(B2-B8)
Task 3  =    B1  AND   Parity(B2-B8)
Task 4  =   -B1  AND   Parity(B2-B8)
```

where Bi represents the ith bit, "–" is logical negation, and Parity(B2–B8) is the parity of bits 2–8.

These tasks are related in several ways: 1) all are defined on the same inputs (eight binary bits); 2) each is defined using a common computed subfeature (Parity(B2-B8); and 3) on those inputs where Task 1 must compute Parity(B2-B8), Task 2 does not need to compute parity, and vice versa. (Tasks 3 and 4 are related similarly.)

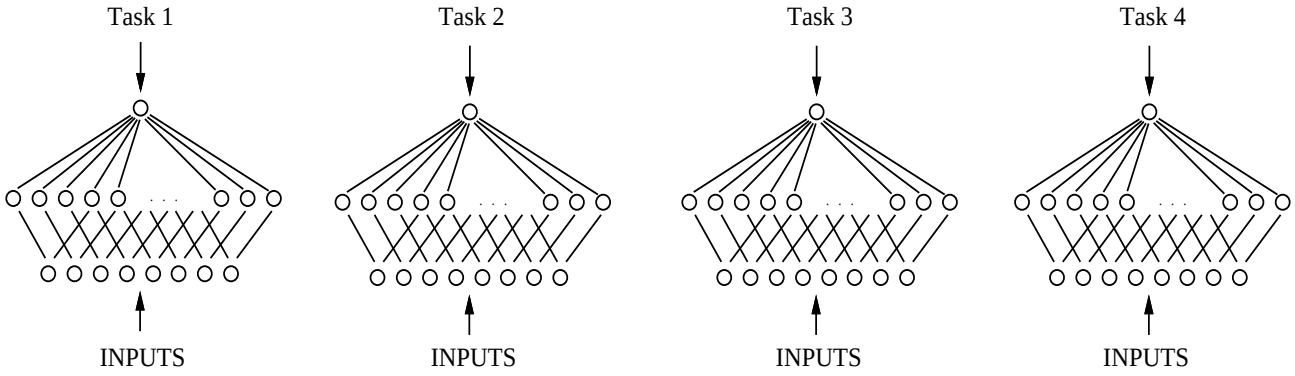

Figure 1: Single Task Learning (STL) of Four Functions Defined on the Same Inputs

Figure 1 shows the backprop nets one would typically train on these tasks. Since the tasks are independent, a separate net is used for each task. This is single task learning (STL). It is also possible, as shown in Figure 2, to use a single net to learn the four tasks at the same time. This is multitask learning (MTL).

The four graphs in Figure 3 show the generalization performance for single and multitask nets trained on Tasks 1–4. Each graph contains three generalization performance curves, one for that task trained in isolation, one for that task trained paired with another task (i.e., MTL with two tasks), and one for that task when all four tasks are trained together (i.e., MTL with four tasks). To simplify the presentation of the two-task multitask data, we present curves only for Tasks 1 and 2 trained together and for Tasks 3 and 4 trained together. Other task pairings yield similar results.

All runs used fully connected feed-forward nets with 8 input units, 160 hidden units, and 1–4 outputs. Nets were trained with backpropagation using MITRE's Aspirin/MIGRAINES 5.0 with learning rate = 0.1,

[1]Inductive bias is anything that causes an inductive learner to prefer some hypotheses over other hypotheses. Bias-free learning is impossible; in fact, much of the power of an inductive learner follows directly from the power of its inductive bias [7].

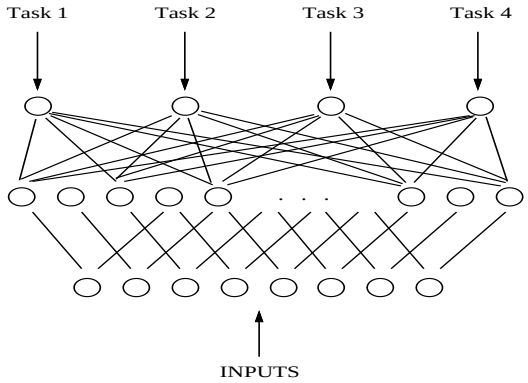

Figure 2: Multitask Learning (MTL) of Four Functions Defined on the Same Inputs

momentum = 0.9, and per-pattern updating. Training sets contained 64 patterns selected randomly from the 256 possible patterns. The remaining 192 patterns were used as a cross-validation hold-out set to measure generalization performance. Training sets were kept small to make generalization challenging. Each plot is the average of 20 runs.

The first graph in Figure 3 shows the average generalization performance during training for Task 1. The three curves represent the performance for Task 1 when trained alone, when trained with Task 2, and when trained with Tasks 2, 3, and 4. The poorest performance occurs when Task 1 is trained by itself (i.e., single task connectionist learning). Better generalization performance on Task 1 is achieved when a single net is trained on both Tasks 1 and 2 at the same time. And even better generalization performance is achieved by training all four tasks together on a single net. Similar results are found for the other tasks: for each task, better performance is achieved if that task is trained on a net that is trained simultaneously on the other tasks. (Note that better generalization performance is obtained without the need for additional training passes. The MTL run is computationally more efficient than the four STL runs it replaces.) To check the results presented graphically in Figure 3, an analysis of variance (ANOVA) of the peak generalization performances from each of the 20 training runs was also performed. This analysis confirms that MTL statistically outperforms STL at the .05 level.

Why is each task learned better if it is trained in the context of a net learning other related tasks at the same time? The MTL nets have sufficient capacity to learn each task independently—nets with less than 20 hidden units are adequate for these tasks. Other experiments confirm that backprop can separate unrelated tasks given sufficient net capacity and that excess capacity is not injurious to generalization on these tasks. We use large nets to minimize capacity effects when different *numbers* of tasks are trained on a net. Specifically, by using excess capacity we can be sure that the improved generalization performance is *not* due simply to multiple tasks constraining [2] the internal representation the hidden layer must learn—it must be a bias effect that arises from a beneficial summing of training signals in the aggregate gradient.

We have run two experiments to verify that the improved generalization performance of MTL is the result of a multitask inductive bias by ruling out two effects that do not depend on the tasks being related. The first effect is that adding noise to neural net learning sometimes improves generalization performance [6]. To the extent that the tasks are *uncorrelated*, their contribution to the aggregate gradient will appear as noise to other tasks and this might improve generalization. To test for this effect we trained nets on the tasks using random additional tasks. For example, we trained a four output net on Task 1 and three different random tasks. (Each random task was held constant during training—the random tasks are true functions.) Figure 4 shows the generalization performance for Task 1 multitask trained with three random functions. The performance for Task 1 when it is trained on nets simultaneously trained on random functions is similar to the performance for Task 1 trained alone. (When the other tasks are multitasked with random functions their performance drops slightly.) We conclude that it is unlikely that the MTL effect is due to the

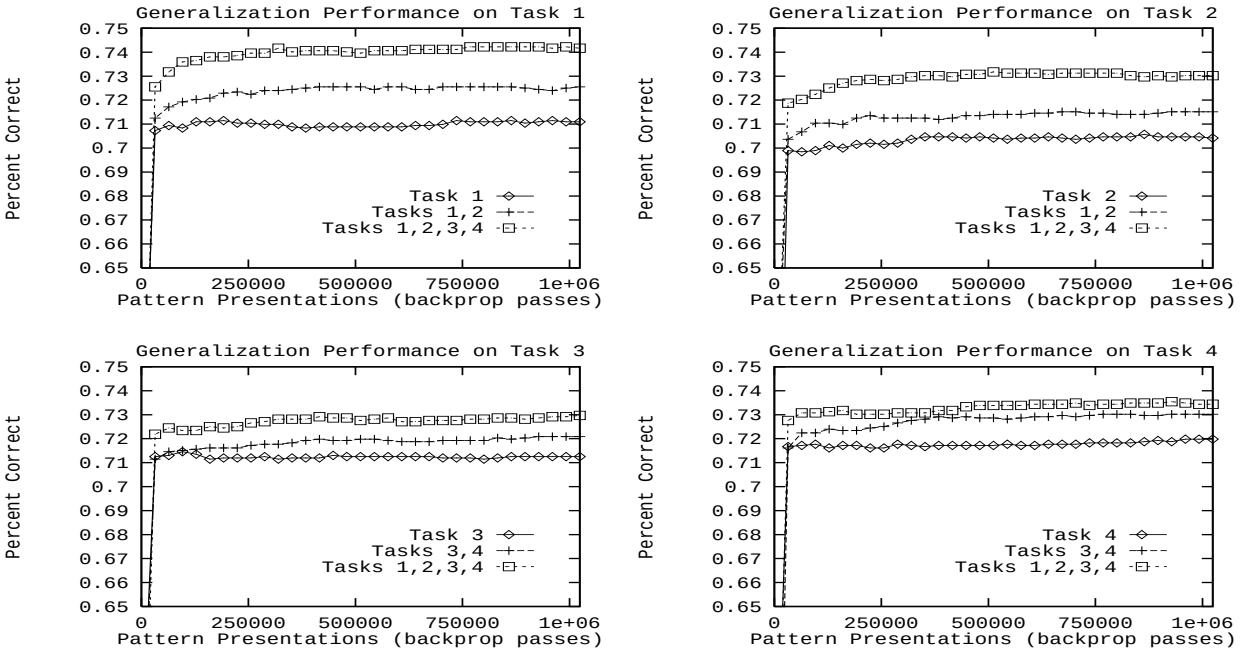

Figure 3: Generalization Performance of Single Task Learning and Multitask Learning on Tasks 1–4

uncorrelatedness of the multiple tasks serving as a beneficial source of noise. (Nets with 20 hidden units are able to train to 100 percent accuracy on one task and three random tasks, so net capacity is not artificially reducing performance with random functions.)

The second effect tested for is the possibility that adding tasks just changes the weight update dynamics to somehow favor nets with more tasks. To test for this we trained nets on "replicated" tasks. For example, a single net was trained on four copies of Task 1; each of its four outputs received exactly the same training signal. In this degenerate form of MTL each task correlates perfectly with the others and contributes no additional information and thus cannot serve as a source of knowledge-based inductive bias. Figure 4 also shows the performance of nets trained this way. (The performance for one output selected at random is shown.) Surprisingly, performance is better than for Task 1 trained in isolation. This improvement is gained without any additional information being given to the system. We can think of four explanations: 1) the extra copies increase the effective learning rate on the weights from the input layer to the hidden layer because their backpropagated error signals add; 2) the copies provide a better signal-to-noise ratio early in training when error signals are backpropagated through initially random weights because there are four times as many random weights from the output layer to the hidden layer; 3) having more weights connected to the same signal increases the odds that one of the weights will be close to a "good" value; and 4) the multiple connections to the hidden layer allow different hidden layer predictions to be averaged and thus act as a mechanism for combining multiple experts [8]. Work is currently underway to determine what mechanism(s) accounts for the observed performance increase. Fortunately, the replication effect is too small to explain all the MTL performance increase observed on Tasks 1–4 even if we assume it is somehow able to confer the same magnitude advantage for tasks as different as Tasks 1–4.

Is the MTL bias effect we observe large enough to be interesting? Probably. First, performance increased as the number of related tasks increased. In many domains it is surprisingly easy to add more tasks. The accumulated benefit of MTL might be significant in these domains. Second, generalization on small parity functions is hard. (Most studies learn parity by training on the complete set of patterns.) The fact that we see improved *generalization* on tasks that compute parity internally (but are not trained on any explicit

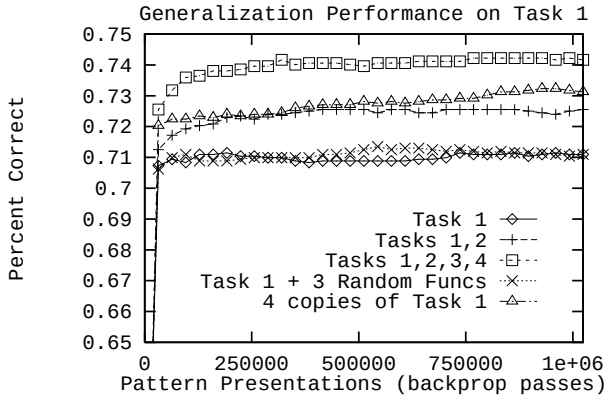

Figure 4: Performance of MTL Variants on Task 1

information about the parity function itself) is remarkable. Third, the networks have not been "told" that the tasks share a common computable subfeature and they differ significantly in how they use this subfeature. This represents a challenging case for MTL because the relationship between the tasks is buried deeply. Yet improvement occurs.

POTENTIAL DISADVANTAGES OF MTL

The advantages of MTL stem from the possibility that, if the tasks are related, information from the teaching signal of some task(s) might aid learning other tasks. But what are the potential disadvantages of using one net to learn several tasks?

1. Because the MTL net is learning several potentially competing tasks at the same time, the gradients computed from the error signal for each task may interfere. Fortunately, experiments suggest that backprop is able to separate uncorrelated tasks. The MTL aggregate gradients may still be flatter because they average several different error sources, but often they will point in directions beneficial to several tasks, i.e., towards more generally useful subfeatures.

2. The MTL net may begin to overtrain on different tasks at different times. If overtraining is prevented by halting training when the cross-validation generalization performance begins to drop, there will not be one place to stop training the MTL net to achieve peak generalization on all tasks. This is not really a problem, though. MTL requires that tasks are trained *in the context of other tasks,* not that learning produce a single net to use for all tasks. MTL can produce separate nets for each task, each representing the point where generalization performance peaked on that task. (We did this already for the ANOVA analysis.)

3. If the learning parameters (learning rate, momentum, etc.) must be set differently for different tasks, there may be no one setting appropriate to all the tasks. Where tasks require tweaking the learning parameters it may be impossible to train the tasks at one time. This may limit MTL's use in some domains.

HOW MTL CAN IMPROVE GENERALIZATION

There are several mechanisms by which MTL can improve generalization. MTL can provide a data amplification effect, can allow tasks to "eavesdrop" on features discovered for other tasks, and can bias the

net towards representations that might be overlooked or unlearnable if the tasks are learned separately.

The data amplification effect is best explained using Tasks 1–4. Both Task 1 and 2 need to compute the parity subfeature on half of the patterns, but Task 1 needs parity only on those patterns for which Task 2 does not, and vice versa. Thus the teaching signal for Task 1 provides information about parity half of the time and the teaching signal for Task 2 provides information about parity the other half of the time. A net trained on one of these tasks receives information about the parity subfeature only about half the time, whereas a net trained on both Tasks 1 and 2 gets information about the parity subfeature on every training pattern. *Both nets see exactly the same training patterns, but the multitask net is getting more information with each training pattern.* This can increase the effective sample size for a computed subfeature. To any one task, the other tasks are providing an inductive bias, i.e., a preference for hypotheses that more accurately compute common subfeatures. (This effect does not explain all the MTL performance observed on Tasks 1–4 because performance increases if Tasks 1 and 2 are trained with Tasks 3 and 4, yet Tasks 1 and 2 already "require" the parity subfeature for all patterns.)

"Eavesdropping" will be a common benefit of MTL. If tasks are related, it is likely that computed subfeatures needed by some tasks will sometimes be useful to other tasks. These subfeatures may not be learnable by the tasks that eavesdrop on them if they are trained separately, but might prove valuable sources of information when they are learned for other tasks.

MTL will sometimes bias nets to learn internal representations they would not learn if trained on tasks in isolation. For example, there are many representations nets can use for numeric quantities. If tasks require different properties of the representation (e.g., less sensitivity to noise like a Gray code or easy multiplication by powers of two like binary coding), these needs can influence the representation learned. Thus MTL may lead to the development of sophisticated representations that better capture the underlying structure of the domain [5] because they satisfy the needs of many tasks.

WHERE MULTIPLE TASKS COME FROM

Tasks 1–4 were carefully devised to be related in specific ways. Where will the additional tasks MTL requires come from for real-world problems? Often the real world gives us related tasks to learn at the same time if one avoids reductionism. For example, the Calendar Apprentice System (CAP) [3] learns to predict the location, time of day, day of the week, and duration of the meetings it schedules. These tasks are related; they are functions of the same data and share some common subfeatures. CAP currently learns these tasks independently. The MTL approach is to induce a single model for all four tasks at the same time.

It is not essential that a domain "naturally" present multiple related tasks to use MTL. MTL is a means of adding domain-specific inductive bias to an inductive system via additional teaching signal. For many domains it is surprisingly easy to devise additional related tasks. These might be semi-modular pieces of the larger task (e.g., tasking a robot to predict the trajectory of a ball when the full task requires the robot to catch the ball) or might be tasks that appear to require similar abilities as the given task(s). Acquiring the additional training signal is not trivial, but it is probably easier than asking domain experts to provide other forms of domain-specific inductive bias: domain experts excel at using their knowledge, but usually find codifying their knowledge difficult. MTL may be a convenient "pipe" through which to add domain-specific inductive bias. Note that MTL does not require that the training signals be available at run time. The training signals are needed only during training because they are outputs—not inputs—of the MTL net. Thus one is free to specify any feature or task that might be useful even if there will be no operational procedure to compute it later.

RELATED WORK

It is common to train artificial neural nets with many outputs. For example, in classification tasks it is common to use one output for each class. But using one net for a few strongly related tasks is also not new. The classic NETtalk [12] application uses one net to learn both phonemes and their stresses. *NETtalk is an*

example of multitask connectionist learning. The multitask approach was natural for NETtalk because the goal was to learn to control a synthesizer that needed both phoneme and stress commands at the same time. (There is evidence that STL NETtalk performs far worse than MTL NETtalk [4].)

Work has been done to demonstrate that what is learned on one task can be used as a bias to aid other tasks learned later [10][11][13]. The key difficulties with these serial transfer approaches are: 1) it is difficult to scale the methods to long sequences of tasks; 2) the sequence in which the tasks will be learned must be defined manually; 3) learning tasks in sequence makes it harder for the net to devise complex representations satisfying the needs of many tasks; 4) if all the tasks are too hard to learn in isolation, sequential transfer will fail because it lacks the combined inductive bias at the most critical stage of learning: when it is first starting to develop an internal representation.

This work is most similar to Abu-Mostafa's catalytic hints [1][14]. We hope to advance the catalytic hints work in several ways:

1. by showing that tasks can be related in more diverse ways than hints yet still yield beneficial bias
2. by further exploring what mechanisms allow multiple tasks to serve as mutual sources of inductive bias
3. by demonstrating that creating new related tasks may be an efficient way of providing domain-specific inductive bias to learning systems

The empirical results for Tasks 1–4 begin to address these: Tasks 1–4 are not easily viewed as providing hints via constraints for each other, yet generalization performance increases when they are trained together.

SUMMARY

Acquiring domain-specific inductive bias is subject to the knowledge acquisition bottleneck. We suggest that one kind of domain-specific inductive bias is readily available if one avoids modularization, and that more can be acquired by collecting the teaching signal for additional tasks drawn from those domains. We describe several mechanisms by which this teaching signal can serve as an inductive bias when combined with a bias that prefers hypotheses benefiting several tasks. We provide an empirical demonstration that shows that even when the similarity between tasks is difficult to learn and recognize, MTL can improve generalization. We further demonstrate that this improvement is not easily attributed to capacity/constraint effects or to other effects not resulting from a mutual inductive bias due to similarity of the tasks.

We are currently applying MTL to other learning methods. For example, we have developed a multitask decision tree induction algorithm that automatically controls the strength of the multitask bias to prevent the decision trees from being so influenced by other tasks that they perform poorly on the main task(s). We are studying MTL in connectionist learning first, however, because there is an established methodology for training artificial neural networks with multiple outputs.

Precisely what is required for multitask connectionism to work is not known. We may have to devise new ways to help nets recognize common substructure and to "coerce" them to share representations for this substructure. We need to devise methods for detecting "eavesdropping" and for determining when new (or modified) internal representations have been learned. We need to understand better how tasks should be related for MTL to succeed. We need to demonstrate empirically that MTL can provide a strong enough inductive bias to be useful in complex tasks. (Work is currently underway to apply multitask connectionism to autonomous land-vehicle driving in ALVINN [9].) We conjecture that as the complexity of the domain increases, so too will the opportunity for creating additional related tasks. If this is true, MTL may provide a practical method of acquiring domain-specific inductive bias that scales naturally with the complexity of the task.

Acknowledgements

Thanks to Wray Buntine, Lonnie Chrisman, Jeff Jackson, Tom Mitchell, Jordan Pollack, Herb Simon, and Sebastian Thrun for their help in refining the ideas presented here.

This research was sponsored in part by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] Y.S. Abu-Mostafa, "Learning From Hints in Neural Networks," *Journal of Complexity* **6**:2, pp. 192–198, 1989.

[2] Y.S. Abu-Mostafa, "Hints and the VC Dimension," *Neural Computation,* **5**:2, 1993.

[3] L. Dent, J. Boticario, J. McDermott, T. Mitchell, and D. Zabowski, "A Personal Learning Apprentice," *Proceedings of 1992 National Conference on Artificial Intelligence*, 1992.

[4] T.G. Dietterich, H. Hild, and G. Bakiri, "A Comparative Study of ID3 and Backpropagation for English Text-to-speech Mapping," *Proceedings of the Seventh International Conference on Artificial Intelligence,* pp. 24–31, 1990.

[5] G.E. Hinton, "Learning Distributed Representations of Concepts," *Proceedings of the Eight International Conference of The Cognitive Science Society,* pp. 1–12, 1986.

[6] L. Holmstrom, and P. Koistinen, "Using Additive Noise in Back-propagation Training," *IEEE Transactions on Neural Networks.* **3**(1), pp. 24–38, 1992.

[7] T.M. Mitchell, "The Need for Biases in Learning Generalizations," Rutgers University: *CBM-TR-117,* 1980.

[8] M.P. Perrone, and L.N. Cooper, "When Networks Disagree: Ensemble Methods for Hybird Neural Networks," to appear in *Neural Networks for Speech and Image Processing,* 1993.

[9] D.A. Pomerleau, "Neural Network Perception for Mobile Robot Guidance," Carnegie Mellon University: *CMU-CS-92-115,* 1992.

[10] L.Y. Pratt, J. Mostow, and C.A. Kamm, "Direct Transfer of Learned Information Among Neural Networks," *Proceedings of AAAI-91,* 1991.

[11] L.Y. Pratt, "Non-literal Transfer Among Neural Network Learners," Colorado School of Mines: *MCS-92-04,* 1992.

[12] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: A Parallel Network that Learns to Read Aloud," John Hopkins: *JHU/EECS-86/01,* 1986.

[13] N.E. Sharkey and A.J.C. Sharkey, "Adaptive Generalisation and the Transfer of Knowledge," University of Exeter: *R257,* 1992.

[14] S.C. Suddarth and A.D.C. Holden, "Symbolic-neural Systems and the Use of Hints for Developing Complex Systems," *International Journal of Max-Machine Studies* **35**:3, pp. 291–311, 1991.

[15] A. Waibel, H. Sawai and K. Shikano, "Modularity and Scaling in Large Phonemic Neural Networks," *IEEE Transactions on Acoustics, Speech and Signal Processing,* **37**(12), pp. 1888–98, 1989.