Training to a Neural Net's Inherent Bias

Steven Gutstein, Olac Fuentes & Eric Freudenthal
Computer Science Department
University of Texas at El Paso

Abstract

A neural net with multiple output nodes is capable of distinguishing among a set of related input classes even in the absence of training. It can do so with an accuracy that is markedly better than random guessing. This is because each class will tend to activate a different set of output nodes. We refer to this tendency as the net's 'inherent' bias.

Ascertaining a net's inherent bias may be thought of as learning the net. One may learn the net either instead of training it, or prior to training it. Furthermore, one only needs a small number of samples from each input class in order to reliably learn the net.

If a net has been previously trained on a different, related set of classes, then ascertaining the inherent bias is a form of knowledge transfer. When such a net is trained to respond in accordance with its inherent bias, one may obtain substantially higher accuracies than is provided by nets trained in the standard fashion.

Furthermore, when using a deep net, we were able to obtain such improvements while only allowing the top layer of the net to train. This layer contained only about 5.7% of the net's free parameters.

Introduction

The term 'bias', when used with respect to inductive learning refers not only to the hypotheses that a learner is capable of expressing, but also to the manner in which it will search through the set of possible hypotheses for one which matches its experiences. The internal parameters of a neural net will give it preferred associations between various outputs and classes of inputs. This is what we refer to as the net's inherent bias. Ascertaining these associations can be thought of as learning the net. If these associations are one-to-one, the neural net will be able to discriminate among those classes with an accuracy far better than a purely random guesser. This advantage can translate into improved learning with smaller training sets.

Obviously, not all biases are created equal for all tasks. Yet, one general bias that humans seem to have is that similar tasks employ similar solutions. It is generally accepted that people use knowledge acquired from previously learned tasks to master new ones. This transfer enables us to acquire new concepts both quickly and accurately based on very few examples, because we have already learned to distinguish between relevant and irrelevant features.

A neural net which has already been actively trained on one set of tasks should have acquired a bias that will help it with a similar set of tasks. Its inherent bias will have the full advantage of knowledge gained while learning the prior set of tasks.

Techniques to more effectively train a learner by using relevant, previously acquired knowledge are known as 'knowledge transfer' techniques. Some examples of training methods that employ knowledge transfer include Discriminability-Based Transfer (Pratt 1993), Multi-Task Learning (Caruana 1997a), Explanation Based Neural Nets (Thrun 1996), Knowledge Based Cascade Correlation (Schultz and Rivest 2000) and Inductive Bias Learning (Baxter 2000). Our experiments draw most heavily from Caruana's Multi-Task Learning and Baxter's Inductive Bias Learning.

Background & Related Work

Knowledge Transfer

To the best of our knowledge, training a neural net using its inherent bias has not previously been studied. This technique makes the most sense when the inherent bias is relevant to the problem(s) at hand. The best way to ensure this is to use a net that has already been trained for a similar set of tasks.

Discriminability Based Transfer An obvious problem in transferring knowledge with neural nets is deciding which portions of a net are germane to a new task and which are not. One of the early attempts to identify these relevant portions is discriminability-based transfer (DBT), which was first introduced by Pratt (Pratt 1993). Pratt took a standard feed-forward neural net (i.e. 1 input layer, 1 hidden layer and 1 output layer) and trained it to perform, what she called, a 'source' task, which was the source of the knowledge to be transferred. Then, she used some of the learned weights to seed a new net with the same architecture. This net was then trained in a new, related task, which she called the 'target' task.

However, direct transfer of all the original weights (i.e.

literal transfer) was found to be counter-productive. The reason, given by Pratt, is that in general, only a subset of nodes from a previously trained net will be relevant to a new set of tasks, no matter how similar the two sets may be. Yet, weights for a trained node are more likely to be large, and thus resistant to retraining. This results in a net with both a reduced capacity and poor bias.

In order to identify which nodes aided and which nodes hindered learning, Pratt looked at the response of each hidden node to the training set for the new class. Nodes that gave relevant output for the new task(s) (as measured by mutual information) had their input weights transferred. Those that did not had their input weights randomly reset. All input weights for nodes of the output layers were reset.

This approach is best suited for cases when retention of prior knowledge is not important. Since, ultimately, we want to create a technique suitable for sequential learning, we do not want to reinitialize large portions of our net. This makes Pratt's technique of using mutual information less immediately attractive for us. However, we expect that in future work we will use a modified form of DBT to improve our technique.

At its heart, the method of knowledge transfer, that we use, is essentially literal transfer. We believe we avoid Pratt's problem of nodes that are both hard to train and irrelevant to the new task(s) by training our net with Multi-Task Learning. This provides an internal representation that is large and robust enough to have a sufficient number nodes that need only minimal training for the new task(s).

Multi-Task Learning Multi-Task Learning (i.e. MTL), which was introduced by Caruana (Caruana 1997a), simultaneously trains a neural net to perform many tasks. The main insight of this technique is that when a net is simultaneously trained to perform several related tasks, its performance for each task is enhanced. Although Caruana suggests 5 main mechanisms for this enhancement, the mechanism that is most important for us is 'Representation Bias' (Caruana 1997b).

According to Caruana, finding an internal representation for shared features that generalizes well between tasks, biases the net toward learning an internal representation that will generalize well within each task. Our use of knowledge transfer involves first using MTL to train a net with a good internal representation for several related tasks. So, when we present the net with a new set of similar tasks, it already has a good internal representation for them.

Inductive Bias Learning Use of a neural net to inductively learn an appropriate bias for a set of related tasks was employed by Baxter & Bartlett (Baxter and Bartlett 1998). This is the most similar to our approach, since it depends upon the neural net to inductively learn an internal representation for each member of a set of classes from a subset of those classes. Then, it uses the resultant encoding to provide the solution to classify input.

Baxter and Bartlett focused on learning a distance measure for their internal representation. They used this distance measure to perform 1-NN classification on a database of machine printed Japanese kanji, which they obtained from the CEDAR group at the State University of New York at Buffalo. This data set contained 90,918 segmented, machine printed Kanji characters, which could be grouped into 3,018 classes (i.e. there were about 30 samples of each kanji). They trained their net to recognize 400 of these classes and to identify the other 2,618 as 'not among the known 400'. Then, using 1-NN classification, they managed to achieve a misclassification error of 7.5% on a test set containing the other 2,618 character classes.

This experiment is very similar to ours, yet there are some notable differences. The main one being that their net was initially exposed to images of the 'new' classes and trained to recognize them as different than any member of the set of classes that were being learned. Our net was never exposed to the 'new' classes until we attempted to learn to recognize them. Additionally, we have a much smaller set of tasks from the common domain that are used for transfer. Furthermore, we examine transfer learning with drastically smaller training sets for the 'new' classes.

Learning a Comparator Function A variation on learning a distance function is to learn a comparator function. Rather than determine to which class a given input belongs, a comparator function looks at two specific inputs and decides whether they are members of the same class.

A recent use of a comparator function was demonstrated by Chopra et al. (Chopra, Hadsell, and LeCun 2005). In this work, a siamese net (Bromley et al. 1993) was trained on a relatively small number of faces to recognize whether a given pair of faces were from the same person. This technique was then able to correctly label pairs of faces, which came from people not seen during training, as being same or different. The fact that this technique works lends credence to the idea of using a net's inherent bias to improve learning.

Representational and Functional Knowledge Transfer The techniques that have been described rely upon the creation of an internal representation of raw data to enhance a net's ability to learn multiple tasks. However, they represent two fundamentally different methods of knowledge transfer - representational and functional (Silver and Mercer 1996). In representational transfer, as Pratt used, one finds the subsets of an existing internal representation which are useful for the new task being learned. In functional transfer, as was used by each of Caruana, Baxter and Chopra, one trains in such a way as to require learning a single, robust internal representation, which is suitable for several tasks.

The technique shown here will be seen primarily as a representational transfer technique. However, functional transfer is used to create an initial representation that is robust enough to be transferred using a literal transfer technique.

Convolutional Neural Nets

Our experiments were performed using a convolutional neural net (CNN). This architecture was chosen for several reasons. It has been successfully used for the specific problem we are employing to demonstrate the advantages of using a neural net's inherent bias (i.e. character recognition) (LeCun et al. 1999) and it has a highly modular structure. This

type of architecture should readily lend itself to knowledge transfer techniques.

CNN's are comprised of several layers and are, therefore, considered 'deep' nets. The nodes of each layer are organized into several feature maps. The nodes composing a given feature map will all share weights and common receptive fields. Each such map may, therefore, be viewed as acting to detect a given feature, wherever it occurs. Furthermore, at higher layers of the net, feature maps may be regarded as identifying the combinations of various low-level features that compose more complex higher-level features. The presence of feature maps as an architectural component of CNN's makes these nets an attractive candidate for knowledge transfer, since these maps represent discrete, localizable detectors for specific features that distinguish among the various classes.

There are two standard types of layers found in a CNN: convolutional and sub-sampling layers. In the convolutional layers (C-layers) of the net, each feature map is constructed by calculating the convolution of one or more small learned kernels over a subset of feature maps in the previous layer, or over the original input, when it constitutes the previous layer.

Although a single feature map may be connected to many feature maps of the prior level, it is connected to each by an individual learned kernel. This results in a map that will reflect the presence of a particular local feature, or local combination of features, wherever they occur in maps of the prior layer. It is the convolution of small kernels that gives CNNs an architecturally based bias for translational invariance. These kernels are also useful for problems with strong local correlations.

In the sub-sampling layers (S-layers), each feature map is connected to exactly one feature map of the prior layer. A kernel of the sub-sampling layers is not convolved over the corresponding feature map of the prior layer. Instead, the input feature map is divided into contiguous non-overlapping tiles, which are the size of the kernel. Each sub-sampling kernel contains two learnable parameters:

1. a multiplicative parameter, which multiplies the sum of the units in a given tile, and
2. an additive parameter, which is used as a bias.

This gives CNNs a decreased sensitivity to minor rotations and distortions of an image, which helps make them robust with respect to unimportant variations.

Because the architecture of CNN's forces the early layers to act as low-level feature extractors, these nets should make good use of MTL, since different tasks may require recognition of the same features located in different parts of an image for different classes. Furthermore, as the upper layers of the net are required to encode more instances from a set of classes (e.g. specific characters from a set of all characters), they should be learning internal representations of those classes, which could provide an effective mechanism for dimensionality reduction.

With this in mind, it is convenient to view a CNN as possessing two halves - a lower half, which acts as a feature extractor and an upper half, which combines the features to

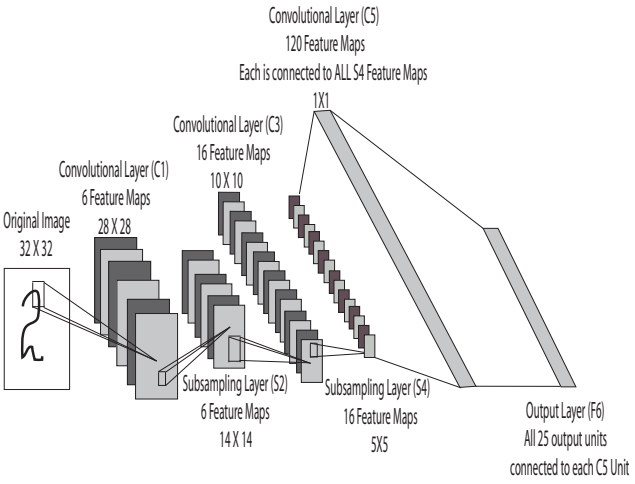

Figure 1: Architecture of our net, which is a slightly modified version of LeNet5. It should be noted that the feature maps in the C5 & F6 (i.e. Output Layer) are 1 node $\times$ 1 node. So, they could with equal accuracy be considered as traditional nodes in a non-weight sharing feed-forward neural net.

produce a reduced dimension representation of the input image. Once a CNN has been trained to recognize a number of specific classes from a set of related classes (i.e. characters, faces etc.) it should be possible to train to recognize other related classes by only training weights in the upper layers of the net. These upper layers will, in essence, be solving a problem with significantly reduced dimensionality. This will now give three main advantages:

1. Significantly faster training time, because the dimensionality of the problem domain has decreased
2. Better generalized accuracy with small training sets, since learning obtained from previous training sets is retained.
3. Smaller expected difference between the expected errors of the training set and testing set, due to the decreased net capacity.

Our experiments used a variation of the LeNet5 style architecture (LeCun et al. 1999), which has already been successfully used for Optical Character Recognition (i.e. OCR). A diagram of the architecture we used is shown in Figure 1.

Experiments

All our experiments that used knowledge transfer to enhance the benefits of training to a net's inherent bias, started with a net that had been trained to recognize the 25 characters A-F,H,I,P,R-W,Y,a,b,d,e,g,h,q,r and t.

The data set from which we obtained samples of these characters was the NIST Special Database 19, which contains 62 classes of handwritten characters corresponding to '0'-'9', 'A'-'Z' and 'a'-'z'. Our choice of which subset of characters to train upon was governed mainly by the number of samples of each character and the desire to avoid training

on either the letter 'O' or 'o', since we felt both were too similar to the number '0'.

The net was trained to respond to each character with a unique pattern of the output nodes either firing or not firing. These patterns were thought of as vectors in a 25 dimensional space and will be referred to as 'target' vectors. The specific target vector for each character class was determined stochastically. Each node had a 50% chance of either firing or not firing for the target vector of a particular class. Input images were then classified based upon to which target vector they were closest in a Euclidean sense.

Once the net was initialized, either with random values or, as described above - with values obtained by training it on a set of related tasks, it was introduced to samples of the classes '0'-'9'.

To estimate the inherent bias of the net with respect to this set, we simply calculated the average output of the net for each class. The desired output (i.e. target vector) for each class would be determined by calculating the sign function of each of the average output vectors. These were then used to classify characters belonging to the classes '0'-'9'. This process may be thought of as 'learning' the net.

For example, if we had been using only 3 output nodes and the average output of the *n* samples in our training set of the character '7' was [0.6, -0.8, 0.9], the target vector for '7' would have been [1.0,-1.0,1.0]. We felt this was a reasonable estimation of the the net's inherent bias for character '7'.

The estimation of the net's inherent bias did not change any of its parameters. So, no training took place. However, for lack of a better term, the set of characters used to establish the desired outputs will be referred to as a 'training' set, whether or not they were subsequently used for actual training.

All sets of experiments were run 5 times each, on multiple training sets of 1, 5 , 10, 20, 40, 80 & 160 samples per class, using different training, validation and testing sets for each run. In each run with training, the net was allowed to train for 400 epochs, although it would converge well before that. The version of the net corresponding to the epoch that performed best on a validation set of 100 samples per class (i.e. 1,000 characters) was then given a new set of 100 samples per class to categorize. Unless otherwise stated, results shown will be the average accuracies obtained on these testing sets of 1,000 characters for the 5 runs performed with each training set size.

Results

Initially, we examine the consequences of relying upon a net's inherent bias without any training. The results are shown in Figure 2, where we examine 3 separate cases.

The first case is a benchmark case, which is a randomly initialized net whose inherent bias is ignored. As expected, without any training, its performance is no better than random guessing.

The second case is a net with randomly initialized weights, but whose inherent bias is taken into account. Here, it may be seen, that the net's average accuracy, though not at the level of practical usefulness, is far better than random.

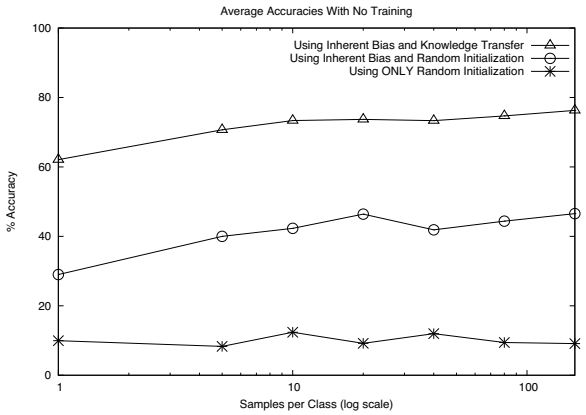

Figure 2: Average accuracies in discrimination amongst the characters '0'-'9', for a net with random initialization and ignoring its inherent bias, a net with random initialization using its inherent bias and a net employing knowledge transfer along with its inherent bias. The x-axis shows the samples per class used to estimate the nets' inherent bias. For the first net, this is meaningless other than to ensure average accuracy comparisons all involve the same test sets.

The third case is a net, which has been biased by learning its internal parameters by training for a related set of tasks and whose internal biases were taken into account. This net has a level of accuracy that is beginning to approach that needed for practical purposes.

It should be reiterated that use of the term 'training set' when describing a neural net that uses its inherent bias without training means that this set was used to help determine the internal bias of the net for each class. However, none of the parameters of the net were adjusted in so doing. The 'training' set was used to determine the net's existing behavior, not to train it for a specific, desired behavior. This is what is meant by 'learning the net'.

Figure 3 shows the minimum, average and maximum accuracies we obtained over 5 test sets for the experiment shown in Figure 2 by a line marked with triangles (i.e. the use of inherent bias and knowledge transfer without task-specific training).

In any realistic experiment, a neural net would be allowed to train. So, we next allowed the randomly initialized neural net, which did not make use of its inherent bias to train for 400 epochs. Then, we found the version of the net which had the highest accuracy on its validation set and found its accuracy on a new testing set. In Figure 4, we compare the performance of this net to a net which relied solely upon knowledge transfer and its inherent bias to discriminate among the images in the test set.

The most notable feature of Figure 4 is that when the training set has only 1 sample per class, using only knowledge transfer and the net's inherent bias without any task specific training, provides better accuracy than using only

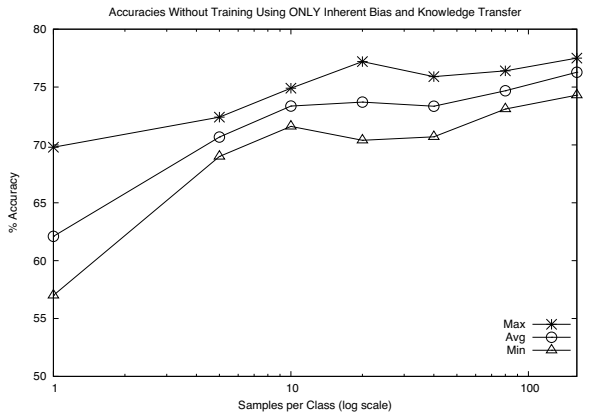

Figure 3: Minimum, Average and Maximum accuracies achieved for a neural net employing knowledge transfer along with its inherent bias to discriminate among characters '0'-'9'.

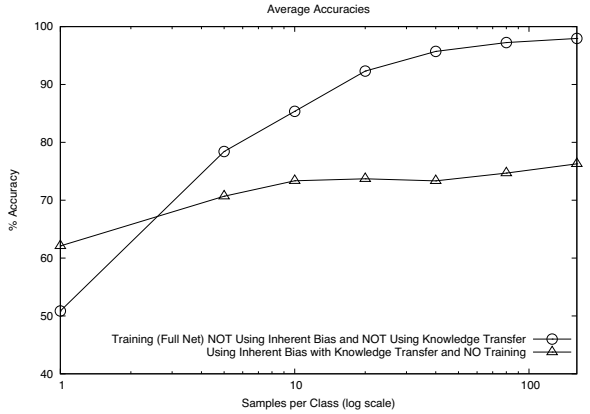

Figure 4: Accuracies achieved for a neural net employing knowledge transfer along with its inherent bias, but without training to discriminate among characters '0'-'9' vs. a neural net, which was trained from scratch and did not make use of either its inherent bias or knowledge transfer.

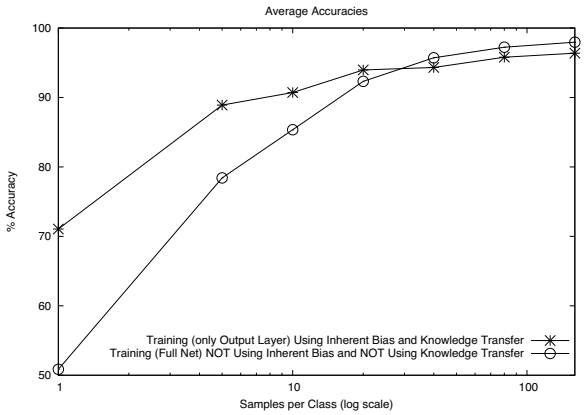

Figure 5: Accuracies achieved for a neural net employing knowledge transfer along with its inherent bias, and with training only the output layer (F6) to discriminate among characters '0'-'9' vs. a neural net, which was trained from scratch and did not make use of either its inherent bias or knowledge transfer. The output layer contains only 5.7% of the net's free parameters.

training. However, when the net is allowed to train, it does not need many samples to surpass a net that relies purely on knowledge transfer and its inherent bias and does not train.

Next, we decided to use a net's inherent bias and knowledge transfer, in conjunction with actively training it for the new tasks. In order to minimize catastrophic forgetting (i.e. losing the ability to perform previously acquired tasks) we only allowed the top layer of the net (i.e. F6 - the output layer shown in Figure 1) to train. The results of this are shown in Figure 5. In the future, we plan to investigate the degree to which our technique is prone to catastrophic forgetting and ways in which catastrophic forgetting may be minimized.

Here it can be seen that the advantage of using knowledge transfer in conjunction with the net's inherent bias is more pronounced and longer lasting than before. When the pure training technique does start outperforming, it requires a larger training set and the improvement is less pronounced. One additional difference, which is not apparent in Figure 5, is that by only training the top layer of the net, we are only training about 5.7% of the net's parameters.

The relative behavior of these two techniques seems analogous to the way an adult achieves proficiency in a new language far more quickly than a newborn, however, over time the adult will have an accent, but the newborn will not.

Finally, one last set of experiments was performed, wherein the entire prior trained net trained without reinitialization of any weights. Its performance is compared with a net that was trained from scratch in Figure 6.

Here, with extra capacity, the previously trained net maintains its advantage over training from scratch for slightly longer and eventually shows the same asymptotic behavior

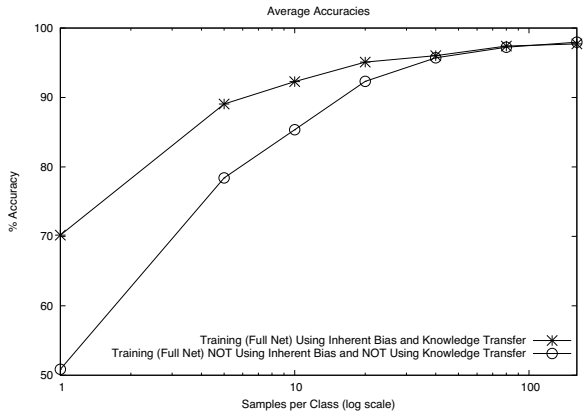

Figure 6: Average accuracies achieved in classifying the characters '0'-'9' for a 'pure transfer with full training' learning method and a 'pure training' learning method

as the training sets become larger.

Conclusions

The results shown are an initial attempt to use a neural net's inherent bias to master a set of tasks. This technique works best when used in conjunction with knowledge transfer techniques. These advantages are greatest with very small training sets. We would expect the benefits to become more apparent for nets that have more knowledge to transfer and that are required to discriminate among a larger number of classes.

Most interestingly, these results hint at the potential for learning new tasks without any additional training. Perhaps, if the previously learned set of characters were larger, had a different distribution of features or were guided to a better internal representation it would be possible to achieve near usable accuracies for new characters based purely upon the net's inherent responses to them.

Future work encompasses exploring ways of more accurately characterizing the net's inherent biases, training and designing the net to develop a more robust internal representation, along with learning the probability distribution of the internal distribution of learned classes to help characterize and learn new classes.

Additionally, preliminary experiments indicate that when a randomly initialized net is trained according to its inherent bias, its asymptotic accuracy is less than that of a randomly initialized net trained to have each output node responding to a single input class. This implies that some target outputs are superior to others. Future experiments will be directed towards finding productive ways to make use of this fact.

The potential for using this technique for sequential learning with either minimal or no catastrophic forgetting needs to be explored, along with the possibility of designing a net that knows when to train itself. A net which sees an input unlike any it has seen before, as judged by the output, possibly could remember the unusual output and recognize when it encounters something sufficiently similar. The results we've obtained using knowledge transfer in conjunction with a net's inherent bias and no training (cf Figure 2) are very promising in this regard.

Acknowledgements

This research was supported by the Center for Defense Systems Research and NSF Grant CNS-0454189.The views expressed here are those of the authors and should not be interpreted as representing official policies or endorsements of these agencies.

References

Baxter, J., and Bartlett, P. 1998. The canonical distortion measure in feature space and 1-NN classification. In *Advances in Neural Information Processing Systems*, volume 10, 245–251.

Baxter, J. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research* 12:149–198.

Bromley, J.; Guyon, I.; LeCun, Y.; Sackinger, E.; and Shah, R. 1993. Signature verification using a siamese time delay neural network. In *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann.

Caruana, R. 1997a. Multitask learning. *Machine Learning* 28(1):41–75.

Caruana, R. 1997b. *Multitask Learning*. Ph.D. Dissertation in Computer Science, Carnegie-Mellon University.

Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Proc. of Computer Vision and Pattern Recognition Conference*. IEEE Press.

LeCun, Y.; Haffner, P.; Bottou, L.; and Bengio, Y. 1999. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, 319–346. Springer.

Pratt, L. Y. 1993. Discriminability-based transfer between neural networks. In *Advances in Neural Information Processing Systems*, volume 5, 204–211. Morgan Kaufmann, San Mateo, CA.

Schultz, T., and Rivest, F. 2000. Knowledge-based cascade corellation. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks 2000*, volume 5, V641–V646.

Silver, D., and Mercer, R. 1996. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems* 8(2):277–294.

Thrun, S. 1996. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, volume 8, 640–646.