Latent Learning - What your net also learned

Steven Gutstein, Olac Fuentes and Eric Freudenthal

Abstract— **A neural net can learn to discriminate among a set of classes without explicitly training to do so. It does not even need exposure to any instances of those classes. The learning occurs while the net is being trained to discriminate among a set of related classes. This form of transfer learning is referred to as 'Latent Learning' by psychologists, because until specifically elicited, the knowledge remains latent. Evidence that latent learning has occurred lies in the existence of consistent, unique responses to the unseen classes. Standard supervised learning can improve the accuracy of those responses with exceedingly small sets of labeled images.**

In this paper, we use a convolutional neural net (CNN) to demonstrate not only a method of determining a net's latent responses, but also simple ways to optimize latent learning. Additionally, we take advantage of the fact that CNN's are deep nets in order to show how the latently learned accuracy of the CNN may be greatly improved by allowing only its output layer to train. We compare our results both to those obtained with standard backpropagation training of the CNN on small datasets without any transfer learning and to a related set of current published results.

I. Introduction

We have observed that one way for a neural net to employ transfer learning is through reuse of its learned internal representations. Internal representations that facilitate differentiation among one set of mutually exclusive classes may also contain latent responses for similar classes to which the net has not been exposed. These *latent responses* can be used to distinguish among the not yet seen classes. Because the responses are learned without any specific reinforcement and are not demonstrated until necessary, psychologists refer to the manner in which they are learned as 'latent learning'. Psychologists had observed latent learning in rats as early as 1930 [1].

We determine the latent response of a net to a new input class by observing its average response to a small number of labeled samples from that class. This average response determines what we will consider the net's latent response to be for that particular new class.

Determination of the latent responses is not training. Training is an iterative optimization process. Observation of a net's average responses to a small number of labeled samples from a new input class does not involve optimization and is not iterative (i.e. the average response is measured only one time). So, we claim it is possible for a net to learn to discriminate among new classes *without task-specific training*.

However, training does improve the performance of a net whose latent responses have been determined. Most of the benefit of training can be obtained by taking advantage of the net's architecture. In this paper, we use a deep net. When we train the net with its latently learned responses, we train only its top layer, since by doing so we obtain nearly all the benefit of training the entire net while only training a portion of the free parameters [2].

At the time of this work, Steven Gutstein, Olac Fuentes and Eric Freudenthal were with the Department of Computer Science, The University of Texas at El Paso, El Paso, Texas, USA (email:s.m.gutstein@gmail.com, {ofuentes, efreudenthal}@utep.edu).

As a result of our investigations, we were able to

- Demonstrate recognition accuracies achievable by pure latent learning without task specific training
- Demonstrate recognition accuracies achievable by latent learning followed by task specific training using only a very small training sets
- Demonstrate improvements in accuracy achievable by latent learning with respect to standard supervised learning for small datasets
- Determine initial means by which latent learning may be optimized

II. Related Work

A. Multi-Task Learning & Discriminability Based Transfer

Our work draws on several strands of previous research. The most fundamental way in which it does so is by learning to differentiate among several classes simultaneously. This is known as Multi-Task Learning (MTL), which was first introduced by Caruana [4]. The advantage of MTL is that when a net is being trained, simultaneously, in several sufficiently related tasks each task facilitates learning the other tasks. In this paper, we use MTL both for transfer learning within a given set of tasks, and to facilitate transfer learning from a set of source tasks to a different set of target tasks.

It has been observed that *literal transfer*, the reuse of a trained neural net for new tasks, actually tends to be detrimental for learning. The technique of Discriminability Based Transfer (DBT) was developed by Pratt et al. [5] in order to permit literal transfer. The key concept is that each node in a neural net acts as a decision plane. After training on a source task, some of these planes will be well positioned for a target task and will need only slight tuning, whereas others will not be well placed. Yet, because the input weights for trained nodes are large, relative to their initial random values, it is difficult to significantly move the decision planes of the less relevant nodes using backpropagation; this hinders training. Pratt's approach was to determine which nodes were not good discriminators for the new task and then to reinitialize their input weights and retrain only those nodes.

We have successfully used literal transfer without reinitializing any nodes. We believe we were able to do this because of our use of MTL with the source tasks. By initially

978-1-4244-9636-5/11/$26.00 ©2011 IEEE

learning several tasks at once, not only are more 'multi-purpose' decision planes created, but also they demarcate more closed regions in our decision space where instances of a new, related class may cluster. As a result, when literal transfer is used for the target tasks, it is more likely that a set of relevant decision planes exists..

B. Output Encoding Methods

Another strand of research that our work draws upon is the importance of a net's internal representations and output encodings for multi-class discrimination. Traditionally, neural nets have an output encoding scheme that can be characterized as 1-of-N encoding. With this encoding scheme, if a net is to differentiate among N classes, then it has N output nodes. Each node is supposed to fire for only a single class. This results in output codes where any two codes differ in only two of N bits. Because of these slight differences, the codes are also fairly sensitive to noise.

Bakiri & Dieterrich [3] first introduced Error Correcting Output Codes (ECOC) in order to create output codes that were robust to noise. In their paper, they concentrated on ensuring robustness by maximizing the differences between all pairs of output codes. Yet, this overlooks the fact that all classes are not identically distinct. For example, when considering alphanumeric characters, 'X' and 'K' look more alike than either one looks like a 'C' or an 'O'. It seems reasonable that these differences in similarity should be reflected in the output codes for each character.

In the late 90's LeCun et al. [6] used output codes that reflected class similarities and dissimilarities in a fairly straight-forward manner. Their experiments employed a convolutional neural net (CNN) to distinguish among handwritten characters. The output code assigned to each character corresponded to idealized images of each character drawn on a 7×12 bitmap.

Larochelle et al. [7] made use of well chosen output codes to even greater effect. They trained a learner to respond to a small number of class/target pairs and then were able to infer what the learner's latent response would be for a class to which it had never been exposed. The determination of these targets was made possible by the experimenters' ability to train their learner to produce outputs that were in some sense (not necessarily semantic) good descriptors for the input classes. When presented with a new class, the experimenters could assume that the output would be a descriptor that was consistent with the descriptors for previously learned classes.

One strategy to obtain such descriptors is to create a semantic encoding. A semantic encoding contains indicators for various semantic features (e.g. 'black', 'red', 'stripes', 'eats fish' etc. [9]). A class is then defined by the set of semantic features that define it. Now, just as with Larochelle et al. [7], it is possible to anticipate a learner's output for a new class. This strategy has been successfully used by Palatucci et al. [8] for neural decoding from functional MRI's and by Lampert et al. [9] for image labeling. The use of high level semantic features means that the classifier will gain useful latent responses to previously unseen classes, as long as those classes can be characterized by a unique subset of the semantic features already learned.

In our experiments we have neither a set of semantic features, nor a sufficient understanding of the encodings learned in order to predict latent responses to the new classes. Instead, we make use of a very small numbers of samples to *observe* latent responses. We also experiment with training portions of our net with those small samples in a manner consistent with the latently learned responses.

C. Convolutional Neural Net

The neural net architecture we used is a LeNet-5 style convolutional neural net [6], as shown in Figure 1. The advantage of this architecture is that it imposes biases that limit the net to hypotheses that are particularly useful for recognition of handwritten characters, which is the problem we used to investigate latent learning. The allowed hypotheses display:

1) Translational invariance
2) Moderate insensitivity to rotations
3) Moderate insensitivity to geometric distortions
4) Dependence upon small-scale, local correlations

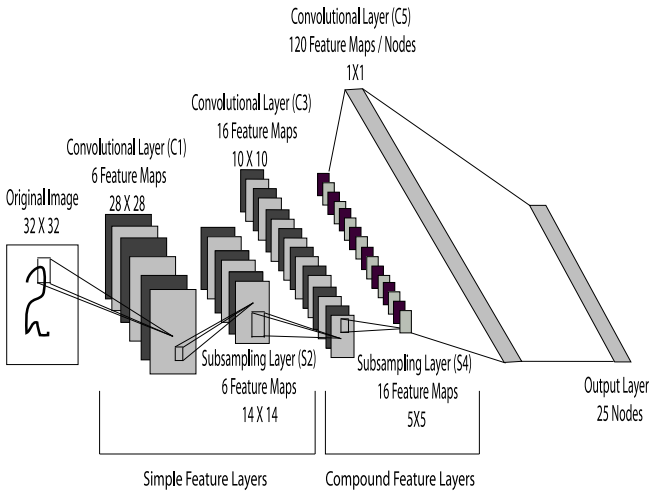

Fig. 1. Basic architecture of our nets, which is a slightly modified version of LeNet5 as used by LeCun et al. in [6] We used nets with 25, 100, 400, 1600 & 6400 nodes in the output layer.

One constraint that enforces these conditions is restriction of the inputs for each node to small, local regions. Rather than connecting a node to all the nodes of the layer beneath it, each node is only given connections to a small, adjacent grouping (i.e. receptive field) of nodes in the layer beneath it. This effectively limits each node to small-scale local correlations in that layer for use as significant features.

Furthermore, the layers of the net are divided into several feature maps (see Figure 1). Nodes in a given feature map have the same set of input weights. This is known as 'weight sharing'. Nodes with shared weights respond identically to the same set of features. Additionally, the spatial relationship between two nodes in a given feature map will be replicated with their receptive fields. A given node may have receptive

fields in several feature maps in the layer immediately below it. However, all nodes in the same feature map will have receptive fields with identical spatial relationships in the same set of feature maps. Therefore, each such map will detect and locate a given feature or combination of features from the previous layer no matter where it appears in the maps of the previous layer. This provides the translational invariance. A more detailed description of CNN's may be found in LeCun et al. [6]

III. Optimizing Latent Learning - Experiments & Results

A. Creating the Source Net for Latent Learning

All of our source nets were 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 dataset 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'. We used 32×32 pixel binary images of the characters. Our choice of which subset of characters to train upon was governed mainly by the number of available samples for each character.

The source nets were initially trained to respond to each of the 25 letters with a unique response pattern. The specific response for each character class was chosen randomly. Each node had a 50% chance of being set to fire in response to a particular class. Input images were classified based upon to which response pattern their output was closest in a Euclidean sense.

We experimented with nets having 25, 100, 400, 1600 & 6400 output nodes. The idea was to double the expected Euclidean distance between any two of our randomly determined responses. Since the expected increase in distance grows as the square root of the number of output nodes, we used a series of quadrupling numbers, starting with 25 output nodes because we had 25 source classes. In retrospect, a more significant set of output nodes would have been determined by the degree to which they compressed the original input data. Our original images were 32×32 pixels, which corresponds to 1024 bits. This means our output encodings provide compression ratios of approximately .025, 0.1, 0.4, 1.6 and 6.4, respectively.

For each number of output nodes, we used 5 randomly initialized nets, each with a unique set of responses for the source tasks with each of the above mentioned numbers of output nodes.

B. Determining Latent Responses and Their Accuracy

Once we had a trained source net, the next step was to determine what its latent responses were to the target set of characters and whether they could serve as indicative responses. The character set we chose for the target tasks was the set of handwritten numerals - '0' - '9'. The source classes did not include either the letter 'O' or 'o', since both are too similar to the number '0' for our purposes. Similarly, we avoided the letter 'l' (lower case 'L'), since it is also too similar to the number '1'.

For the target tasks, we used training sets with 1, 5, 10, 20, & 40 samples per number. It should be restated that the term 'training' set is something of a misnomer here. As previously discussed, one uses a training set to iteratively adjust the behavior of a learner by changing its internal parameters. For pure latent learning, we only use the training set to observe our net's latent responses. So, at this point, the training set for the target tasks is being used more as an observation set.

In order to determine a net's latent response for a given class, we calculated the average response of each output node to that class. The latent response was determined by calculating the sign function (i.e. $sgn(x)$) of each of the average outputs. For example, if we had been using only 3 output nodes and the average output of the *n* samples in our training set for the character '7' was [0.6, -0.8, 0.9], the latent response for '7' would have been estimated to be [1.0,-1.0,1.0]. Clearly, this procedure is not iterative and does not effect the net's response to input in any way. This is the basis of our claim that the net has not been trained for the new set of tasks, even though a 'training' set is used to estimate the net's latent responses.

C. Determining the Optimal Degree of Source Task Training

To determine how many training epochs on the source tasks would lead to optimal latent learning for the target set of tasks, we measured the accuracy with which the target tasks were latently learned after each source task training epoch. We also experimented with varying the number of output nodes and numbers of samples per class in the target classes' training (i.e. observation) set, which it should be recalled was not used for training the net, but rather for learning the net's existing set of latent responses. The test sets for the target classes had 100 samples per class (i.e. 1000 samples).

In the graphs of Figure 2, we show the results for these experiments. Each data point represents the average accuracy achieved by a group of 5 nets with different random initializations and trained indicative responses for the source tasks. Each net was tested with 5 different pairs of training and testing sets for the for the target tasks. So, each data point shown is an average of 25 trials.

Figure 2 illustrates how the accuracy of latent learning evolves with training epochs on the set of source tasks. At 0 epochs, when the net has only been randomly initialized, its latent, though not learned, responses still serve to discriminate among the digits (i.e. target tasks) with accuracies far better than purely random guessing provides. However, the accuracy of the latent responses to the target classes increases sharply after one training epoch on the source tasks, where latent learning begins. Surprisingly, this represents the peak accuracy of the latently learned responses. We had expected the peak accuracy with the target tasks would be more closely related to when the source tasks achieved their peak accuracy. It would be interesting to see if presenting the source training data to the net in a different random order for each training epoch would change this situation and allow further improvement in latent learning with additional training epochs. It would also be interesting to see how

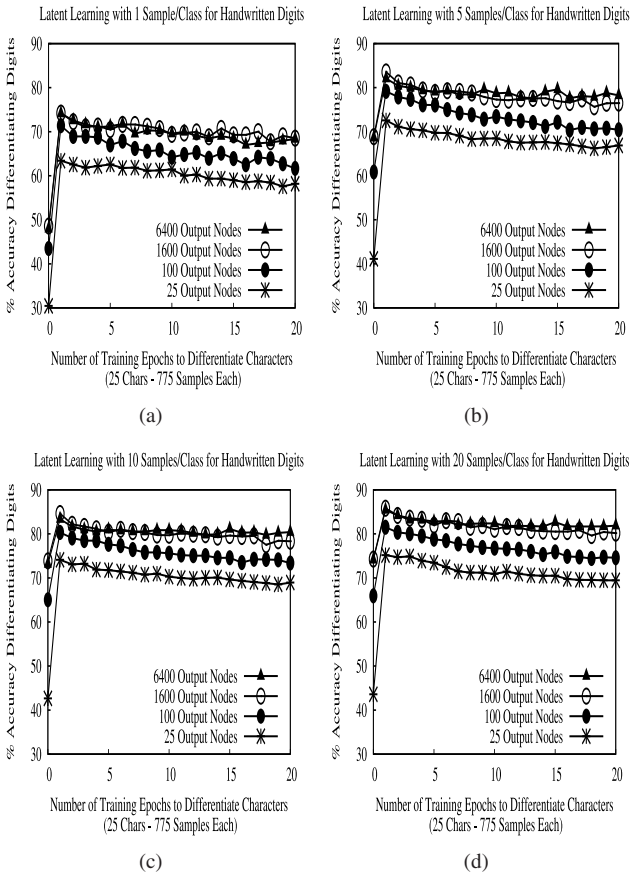

Fig. 2. Average accuracies achieved for discriminating among the digits '0'-'9' with latent learning after each training epoch over a set of 25 letters. Latently learned inherent responses for the digits were determined with 'observation' sets ranging from 1 sample/class to 20 samples/class. At no time during source task training was any net exposed to a member of the target task set. Since the target task results are for 'pure latent learning' neither were the nets ever trained to give a specific response to any digit. All averages are over 25 trials. The results for 400 output nodes were between those for 100 and 1600 output nodes. We have omitted them in order to make the graphs easier to read.

(and if) the latent responses changed after each source task training epoch.

Finally, there seems to be little difference between the accuracies achieved with 1600 output nodes and those achieved with 6400 output nodes. This seems to suggest that, for our problem, there is little to no benefit to be gained by having the output nodes occupy a greater dimensional space than the input data.

D. Comparisons Between Standard Training, Pure Latent Learning & Latent Learning with Partial Training

Once we had established a likely optimal number of source task training epochs (i.e. 1), the next issue was to compare how well a net that was near optimized for latent learning would perform when compared to a net that was trained using a standard backpropagation approach. It had been determined in earlier work [2] that almost all the benefit from allowing a net possessing some latent learning to also train could be obtained by limiting training to the top layer of the net. We refer to this as 'latent learning with partial training', since only part of the net experiences training. In 'pure latent learning' the net does not undergo any training in the target tasks for reasons that have been explained earlier.

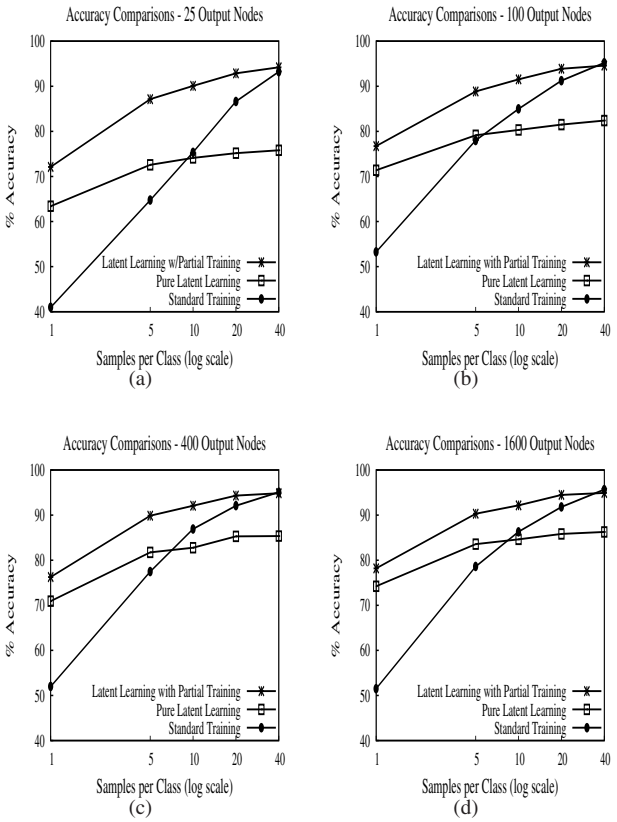

Fig. 3. Average accuracies achieved for discriminating among the digits '0'-'9'. The graphs shown compare results for a net that employs standard backpropagation training without any transfer learning, a net that employs 'pure latent learning' without any training specifically on the target tasks, and a net that uses latent learning with partial training, where the top layer of the net is trained to enhance the latently learned responses. All data points represent an average over 25 trials. The results for 6400 output nodes have been omitted, since as seen in Figure 2, they do not differ significantly from the results for 1600 output nodes.

The graphs shown in Figure 3 demonstrate that for small sets of labeled samples, pure latent learning provides markedly better results than standard backpropagation training without any transfer learning. This is in spite of the fact that pure latent learning does not involve any task-specific training. If one allows the top layer of a net to train in addition to latently learning, then this effect is more pronounced and remains for larger sets of labeled samples. In earlier work [2], it was shown with non-optimized nets (i.e. one where the number of training epochs on the source tasks was not chosen to maximize latent learning) that a partially trained net achieves asymptotic accuracies slightly less than those achieved by a net that has undergone standard backpropagation training, as the training set size grows. It was also shown with non-optimized latent learning that if a net that has benefited from latent learning is allowed to train

in its entirety (i.e. not just the top layer), then its accuracies for large training sets are indistinguishable from a net that has used standard backpropagation training.

E. Detailed Results & Comparison with Related Work

To give a more precise idea of the accuracies achievable with latent learning, we have tabulated some of our results in Tables I & II.

The results shown in Tables I & II compare well to results obtained by other researchers. In 2009, Bergstra and Bengio [10] introduced a method that involves not only a new activation function designed to better model the response function of actual neurons, but also uses a pretraining phase to optimize the initial conditions of the net before training begins. This use of a pretraining phase, where the net is trained to optimize a loss function is an example of transfer learning. Although the main focus of their paper was not to examine how to get high recognition accuracies from small labeled training sets, they chose to demonstrate the efficacy of their technique by reporting the accuracy of their net when used to differentiate among the handwritten digits '0' - '9' with a training set of only 100 samples. This corresponds to our use of 10 samples per class, assuming a uniform distribution of classes. The accuracies they achieved ranged from **81.0%** to **81.6%**, depending upon the number of pretraining epochs employed. They did not observe the early overtraining for transfer learning that we did.

Our results for pure latent learning and latent learning with partial training are shown in Tables I & II. The average results achieved with pure latent learning for 10 samples per class and 100 (or more) output nodes are similar to those obtained by Bergstra and Bengio; this is in spite of the fact that pure latent learning does not entail any training on the target tasks. Furthermore, when the net is allowed to partially train consistently with its latently learned responses, the performance improves significantly. However, as shown in Figure 3, a part of the reason for our relatively high accuracies lies in our choice of a CNN architecture. At 10 samples per class, a CNN that uses only standard backpropagation training achieves results slightly better than pure latent learning, which performs comparably to Bergstra & Bengio's technique.

In Tables I & II, we give minimum, average and maximum accuracies achieved in order to give the reader a sense of the error bars that should be attached to the reported average accuracies. As expected, we have the most variation when our training set has only 1 sample per class. At best, when training with only 1 sample per class we achieved 87% accuracy in discriminating among the handwritten digits, '0' - '9'.

Our average results have a slight tendency to be skewed towards our best results. We believe this is because poor results tend to be caused by unusually bad samples in the training set. For a sample to be unusually bad for training, it merely has to be unusual. Conversely, good results are largely due to good samples in the training sets. For a sample to be good for training, it must be, in some sense, a fairly usual exemplar of its class. Because it's more likely for 'usual' rather than 'unusual' samples to be randomly selected for a training set, good training sets are slightly more likely than poor ones.

TABLE I

MIN,AVERAGE,MAX % ACCURACIES ACHIEVED WITH PURE LATENT LEARNING (AVERAGE ACCURACIES IN BOLD)

Output Nodes	Samples per Handwritten Digit			
	1	5	10	20
25	50.9	67.2	63.8	69.2
	63.4	**72.6**	**74.1**	**75.2**
	72.0	78.9	78.9	79.7
100	59.3	74.0	76.6	77.3
	71.4	**79.1**	**80.3**	**81.5**
	78.6	83.1	84.1	87.0
400	58.6	78.4	77.5	82.1
	70.9	**81.7**	**82.8**	**85.3**
	80.1	86.9	85.5	88.0
1600	52.5	80.2	82.4	83.3
	74.2	**83.6**	**84.6**	**85.8**
	83.2	87.6	88.7	89.8

TABLE II

MIN,AVERAGE,MAX % ACCURACIES ACHIEVED WITH LATENT LEARNING AND PARTIAL TRAINING (AVERAGE ACCURACIES IN BOLD)

Output Nodes	Samples per Handwritten Digit			
	1	5	10	20
25	57.8	84.0	86.1	90.2
	72.1	**87.2**	**90.1**	**92.9**
	78.8	89.6	93.0	95.2
100	66.2	86.8	88.8	92.2
	76.7	**88.8**	**91.5**	**93.9**
	84.0	91.7	94.1	96.0
400	66.8	87.6	88.7	92.1
	76.3	**89.9**	**92.1**	**94.3**
	84.3	92.6	94.1	96.6
1600	62.6	87.1	88.4	92.2
	78.2	**90.3**	**92.2**	**94.5**
	87.0	92.3	95.1	96.4

IV. CONCLUSIONS AND FUTURE DIRECTIONS

Latent learning can enable a net to differentiate among a set of classes with surprisingly high accuracy, even in the absence of task specific training. If, however, for a deep net one does train with just the output layer, then these results may be significantly improved. Optimal latent learning seems to occur after just a single epoch of training on the source tasks. More work needs to be done to understand why we observed this and whether obvious techniques, such as randomly reordering the members of the source task training set would help delay overtraining on the source tasks.

Other preliminary results have shown that when we do allow partial training on the target tasks, there is a degradation of performance on the source tasks. Although this degradation is not severe enough to be labeled 'catastrophic', neither is it ignorable. If latent learning is to serve as one tool to help achieve lifelong learning, then this problem will have to be overcome, or even more significantly reversed. One should hope that learning the target tasks would provide a positive transfer back to the source tasks.

Finally, latent learning should be a key element in enabling a net to recognize *on its own* when it encounters a new class and to decide when to train itself to recognize that new class.

REFERENCES

[1] E. C. Tolman and C. H. Honzik, "'Insight' in rats," *University of California Publications in Psychology*, 1930.

[2] S.Gutstein, O. Fuentes and E. Freudenthal, "Training to a Neural Net's Inherent Bias," *Twenty-Second International Florida Artificial Intelligence Research Society Conference*, pp. 45-50, 2009.

[3] G. Bakiri and T. Dietterich, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263-286, 2000.

[4] R. Caruana, "Multitask Learning: A Knowledge-Based Source of Inductive Bias," *Proceedings of the Tenth International Conference on Machine Learning*, pp. 41-48, 1993.

[5] L. Y. Pratt, S. J. Hanson, C. L. Giles and J. D. Cowan, "Discriminability-Based Transfer between Neural Networks," *Advances in Neural Information Processing Systems*, no. 5, pp. 204-211, 1993.

[6] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition,"*Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[7] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks," *IEEE Proceedings of the 23rd national conference on Artificial intelligence*, vol. 2, pp. 646-651, 2008.

[8] M. Palatucci, D. Pomerleau, G. Hinton and T, Mitchell, "Zero-Shot Learning with Semantic Output Codes,"*Neural Information Processing Systems*, Dec. 2009.

[9] C. Lampert, H. Nickisch and S. Harmeling, "Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer," *CVPR*, 2009.

[10] J. Bergstra and Y. Bengio, "Slow, decorrelated features for pretraining complex cell-like networks," *Advances in Neural Information Processing Systems 22*, 2009.