Meaningful Representations Prevent Catastrophic Interference

Jordi Bieger[a] Ida Sprinkhuizen-Kuyper[ab] Iris van Rooij[ab]

[a] *Radboud Universiteit Nijmegen*
[b] *Donders Institute for Brain, Cognition and Behaviour*

Abstract

Artificial Neural Networks (ANNs) attempt to mimic human neural networks in order to perform tasks. In order to do this, tasks need to be represented in ways that the network understands. In ANNs these representations are often arbitrary, whereas in humans it seems that these representations are often meaningful. This article shows how using more meaningful representations in ANNs can be very beneficial. We demonstrate that by using our Static Meaningful Representation Learning (SMRL) technique, ANNs can avoid the problem of catastrophic interference when sequentially learning multiple simple tasks. We also discuss how our approach overcomes known limitations of other techniques for dealing with catastrophic interference.

1 Introduction

For decades researchers have been trying to recreate intelligence in computers. One important approach is to try to imitate what we know about human and animal intelligence. It has been known for long that our brains are networks of interconnected neurons. Apparently, these neural networks allow us to think, learn, act and perceive as we do. Inspired by this knowledge, *artificial neural networks (ANNs)* were created [22]. In this article, we focus particularly on a subclass called *multi-layer perceptrons (MLPs)*. MLPs are ANNs with multiple layers of neurons and no circular connection paths. Such networks can learn to perform a given task, defined as an input-output mapping, by training the networks on exemplar input-output pairs.

In neural networks, both input and output are vectors of numbers, which can be taken to represent something. The relation between the input vector and that which it represents is often fairly arbitrary [4, 20]. Usually this is not a problem, because the networks can learn even using these arbitrary representations. Yet, research has shown that using more meaningful representations can increase network performance [6]. Furthermore ANNs with arbitrary representations struggled with sequentially learning multiple tasks since they were first used [19]. When an ANN is first trained on one task and then on another, it will forget everything it ever knew about the first task. This phenomenon is called *catastrophic interference*. In this article we will show that by using meaningful task representations, this problem may be avoided.

Section 2 will give some background information about catastrophic interference (Section 2.1), transfer learning (Section 2.2) and task representation (Section 2.3). Section 3 will introduce our *Static Meaningful Representation Learning (SMRL)* technique. In Section 4 we will describe the experiments we carried out to show the merit of meaningful representations and Section 5 discusses the results. Section 6 contains the conclusion of the article.

2 Background

2.1 Catastrophic Interference

Catastrophic interference or “catastrophic forgetting” is the disruptive effect that learning something new has on existing knowledge. Both intuition and research [6] suggest that while some interference may occur between different tasks, humans do not suffer from *catastrophic* interference (i.e. we do not nearly forget *everything*).

An ANN's knowledge is represented by the weights between its nodes. When the network is training on some task, these weights are adjusted. Then, when it tries to learn something new, it will again start to change those weights. It will not, however, take into account that it might want to leave those weights intact in case it is ever required to perform the first task again. It would appear that artificial neural networks, which are abstracted models of the brain, have failed to model the characteristic of human neural networks that allows them to avoid catastrophic interference. Building an ANN that does not suffer from catastrophic interference might therefore provide insight into how this mechanism could work in humans.

The most common way to avoid catastrophic interference in ANNs is to interleave training on the new information with training on the existing knowledge. There are two problems with this: it is not how human learning works and it is terribly inefficient, because it fails to utilize all of the previously acquired knowledge and requires starting the learning process from scratch every time something new has to be learned. A lot of research has nevertheless focussed on simulating the interleaving of datasets [2, 21].

When MLPs are trained using back propagation they tend to use the entire network to perform the learned task. It is generally assumed that humans have specialized brain areas that are only used for certain tasks [7, 12]. French [6] proposed a method called "node sharpening" that aims to reduce catastrophic interference by coaxing the neural network into using only a couple of neurons to match each input pattern. For each input pattern a number of hidden nodes was selected that should be used to match that particular input. Ideally, each input pattern would select a different set of hidden nodes, which means that learning to match one pattern did not interfere with the learning of another, because different nodes and thus different weights were used. Although node sharpening reduces catastrophic interference, it also reduces the network's performance and its ability to generalize. Also, the method stops working when two or more patterns elect to use the same hidden nodes. This happens when those patterns are too similar to each other or when the number of hidden nodes to choose from is too small.

Catastrophic interference is related to the plasticity-stability problem in models of memory, which states that they should be "simultaneously sensitive to, but not radically disrupted by, new input" [6]. Grossberg and Carpenter developed *Adaptive Resonance Theory (ART)* specifically to deal with this issue [10]. ART networks deal well with catastrophic interference, but they dynamically add new neurons, which is not generally regarded as biologically plausible and they are very complex, especially when they have to be adapted in order to support supervised learning. Section 3 introduces a much simpler method that we will use to demonstrate the power of meaningful task representations by avoiding catastrophic interference when learning multiple tasks.

2.2 Transfer Learning

Another field related to learning multiple tasks is the field of *transfer learning*. Our brains enable us to efficiently learn new things during our entire lives. People can often correctly generalize from only one example [1]. It is believed that this ability is facilitated by the fact that our brains already contain so much relevant knowledge about (most) new tasks. The idea behind transfer learning and the *Machine Life-Long Learning (ML3)* framework is that a learning system should take advantage of the knowledge it already possesses and use it as an *inductive bias* [15] when learning new tasks [27]. It should also be able to continue learning for the rest of its 'life'.

There are basically two distinct approaches to knowledge transfer: *representational transfer* and *functional transfer* [25]. Both are used to guide the learning of a new task by using the structure of already acquired knowledge. With representational transfer the initialization of the weights in the network for learning a new task is biased by the existing knowledge of the system rather than random (see for instance [16, 17]). The main advantage of this paradigm is that storing representational knowledge requires little memory. A disadvantage can be that accuracy can decline over time, because the neural network representations are often not perfect.

In the functional transfer paradigm, existing knowledge is used to pressure the new network to share a similar encoding [3, 23]. The easiest way to do this is to just remember all training examples of previous tasks and use them in addition to the train set for a new task when learning something new by interleaving them into the new training set. Storing all training examples takes up a lot of space however. An alternative might be to store neural networks for all of the previously learned tasks and use them to (re-)generate training examples when required. This is similar to the pseudo-pattern based approaches that were mentioned in the previous subsection. This can save storage space, but becomes inaccurate if an input vector from the train set for the new task is not valid for one of the already known tasks.

The learning paradigm introduced in Section 3 makes great use of existing knowledge in the network and facilitates very efficient storage of task knowledge (just one meaningful representation vector per task).

2.3 Representation

ANNs are used to perform tasks, solve problems, find patterns, etc. There are neural networks that learn aspects of languages, play the saxophone or play card games [4, 20]. The inputs for the last task could for instance be the cards that the system can see as well as information about the actions of the other players. The output might be that the network lays a card or makes some sort of bid. These are not things that neural networks, or software systems in general, are capable of. The inputs for the real-life task need to be translated to something that the ANN can understand and the output needs to be translated to something that makes sense in the real world.

Thinking of a good representation for these inputs and outputs is in general a hard problem, so in most cases fairly arbitrary representations are used [4, 20]. Neural networks generally work fine with these arbitrary representations, presumably because they do not have the prior knowledge to make use of more meaningful ones. It has been shown though, that representing similar real situations with vectors that are close to each other and dissimilar situations with orthogonal vectors can increase performance [6]. One might say that the representations in such an approach are more meaningful, because at the very least they preserve certain similarities and differences between situations in the domain.

Creating useful representations is not only a problem for task status inputs however. Even though most people are capable of performing more than one task, they do not always know what to do in each situation. This could happen for instance if one is dealt a hand of cards without knowing the game, or when sitting down at a chess board without knowing which variant is played. In these cases you need to be told what to do; extra inputs are necessary. This is done using so-called *action words*, which are represented in the brain in a way that is relevant to the task that they describe [9, 11, 13, 18]. The question then is: how to represent which task to perform in neural networks?

To the best of our knowledge, this question has hardly been researched. Turney [28] gives an account of how to recognize and manage what he calls *contextual features*, but does not talk about how they might be specified. Silver and Poirier [24] use extra input nodes to represent the tasks it knows (and will learn). One *representation node* is used for every task that the network knows. Every node is turned off, except for the one associated with the task that should currently be performed. This approach requires that N representation nodes are used, where N is the number of tasks that the network should be able to learn.

Another, more distributed approach, might only require $\boldsymbol{log}(N)$ representation nodes (rounded up). So if the network needs to perform 7 different tasks, only 3 input nodes are necessary. Each task representation is given by the binary number signifying how many tasks were learned before it. For the first task (binary `000`) all the nodes are turned off, for the third (binary `010`) the second node is on and for the fourth (binary `011`) only the first node is off, etc. (In binary, 0's and 1's are usually used as symbols, but in our experiments we will use -1 and $+1$.)

Both of these methods assign fairly arbitrary representation vectors to each task. They do not even contain any information about the tasks themselves, but rather about the order in which the tasks were learned. In fact, these *representation vectors* are not so much *representing* tasks as they are *identifying* them.

2.4 Parametric bias

One way to obtain meaningful task representations, is to treat the representation nodes as regular input nodes and train their activation values using the back propagation training algorithm. The model proposed by Tani et al. [26] is called RNNPB (Recurrent Neural Network with Parametric Bias) and can learn to predict multiple time series. It accomplishes this by adding some *parametric bias (PB)* nodes to the input of the network. In the training phase, the network learns all the required time series in an interleaved fashion while determining the PB values for each of them by simply using the back propagation training algorithm. When the network is required to reproduce a certain time series, it should be fed the corresponding PB values in addition to the regular input. Tani et al. developed these nodes with the goal of mimicking the functionality of mirror neurons in our brains, which in turn have been linked to action representation [11]. We will show that PB nodes can also be used for learning meaningful task representations without interleaving.

Algorithm 1 Static Meaningful Representation Learning (SMRL)

Learn first task T_1 (*Initial Knowledge Acquisition*)
- Construct an MLP for learning one of the required tasks
- Add a number of *meaningful representation nodes (MRNs)* that will represent a task
- Train the entire network (including MRNs) on task T_1
- Store the values of the MRNs along with T_1
- Fix all the weights in the network

For all new tasks T_x do (*Novelty Learning*)
- Reset the values of the MRNs
- Train the MRN values to represent T_x
- Store the values of the MRNs along with T_x

Perform a learned task T_x (*Knowledge Application*)
- Clamp the representation values stored with T_x to the MRNs
- Clamp the desired task input values to the other input nodes
- Read the output values of the network

The name "parametric bias", chosen by Tani et al. is suitable, because PB nodes do in some sense alter the biases of the nodes in the first non-input layer of the network. A node's bias can be thought of as the weight between that node and an imaginary node that is always on, regardless the input. One could also conceive of the bias as representing the part of the neuron that is always there, that does not change with the task input. That is why using PB nodes can be thought of as adjusting the biases of the connected nodes: once a task is chosen, their values do not change. In other words: the network is parameterized by the values of the PB nodes so that it can perform different tasks by adjusting these biases.

3 Static Meaningful Representation Learning

In this section we describe the approach that we use to show that using meaningful representations can enable the sequential learning of multiple tasks without suffering from catastrophic interference. The basic idea is that when the weights of a network cannot change, the network cannot forget the things it already knows. Normally, no new tasks can be learned in such static networks, but with the *Static Meaningful Representation Learning (SMRL)* approach we propose a method that allows new tasks to be learned by representing them in the context of the existing knowledge in the network.

According to Chalmers and Hofstadter [5] "representations are the fruits of perception" and perception cannot be separated from learning and cognition. They also suggest that learning happens mostly by making analogies between what we already know and the thing we are trying to learn. The way things are perceived depends on the knowledge we have of it and vice-versa. It may very well be the case then, that perceptions and by extension representations, are learned in tandem with task content. This is reflected in the Static Meaningful Representation Learning method.

There are three stages in the life of SMRL-trained networks: the *Initial Knowledge Acquisition* phase where the weights are plastic and the network can attain its initial knowledge, the *Novelty Learning* phase where the network learns new tasks without forgetting that initial knowledge and the *Knowledge Application* phase where the network applies its knowledge to perform tasks. In the novelty learning phase the network is static (i.e. the weights do not change anymore) and new tasks are learned by training the representation (parametric bias) nodes. Because every task has its own meaningful representation vector (MRV), learning of one task does not interfere with learning other new tasks.

Since catastrophic interference will not occur when using SMRL, the only question left is whether it is possible to learn representations for new tasks that are capable of changing the network's behavior in such a way that these new tasks can be performed. This clearly depends both on the initial knowledge and on the new task(s) and the relation between them. To keep things simple, the initial knowledge of the networks in our experiments will just be the knowledge of one task in the task set. That task is given an initially meaningless (random) representation vector which is trained, along with the rest of the network, on the task. After initial training, the connection weights of the network are fixed. Next, the network attempts to learn representations for new tasks without changing the connection weights. We can think of these fixed connection weights as context knowledge provided by the first task. The specific implementation of the

Inputs		NONE	AND	NIF	1ST	JUST2	2ND	XOR	OR	NOR	IFF	¬2ND	¬JUST2	¬1ST	IF	NAND	ALL
-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+
-	+	-	-	-	-	+	+	+	+	-	-	-	-	+	+	+	+
+	-	-	-	+	+	-	-	+	+	-	-	+	+	-	-	+	+
+	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
binary		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
#		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table 1: Here the 16 tasks with two Boolean inputs and one Boolean output are listed. The tasks are given names for easy reference, some of which are fairly well known (e.g. AND, IF and XOR). For each of the four possible combinations of inputs, the target output for each task is listed. Inputs and target outputs are all either $+1$ or -1. The tasks can also be numbered from 0 to 15 when their outputs are first translated to binary by changing $-$'s into 0's and $+$'s into 1's and concatenating the four target outputs of the task.

SMRL framework that is used in this paper, is summarized in Algorithm 1.

In the rest of this article we will give a proof of concept suggesting that the use of such meaningful task representations can enable sequential learning even in these handicapped networks (where weights are static).

4 Experiments

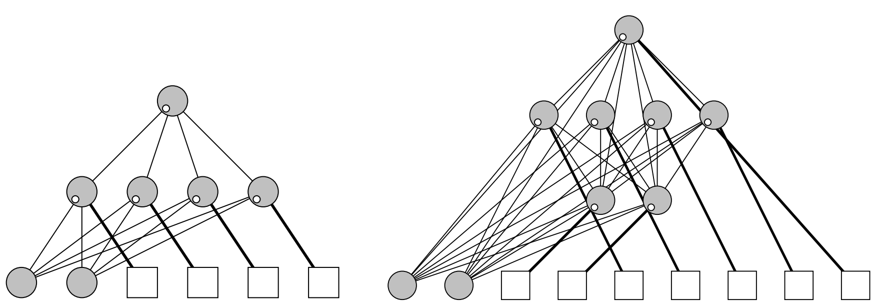
In order to explore whether using meaningful representations could be used to learn new tasks in static networks, we elected to use a simple task domain: all tasks with two Boolean inputs and one Boolean output which are fully specified, which means that the training set contains every possible input combination. These tasks include logical operations such as AND, OR and XOR and are described in Table 1.

The learning of a task representation is considered successful if the network behaves correctly for that task, meaning that it outputs a positive value when the target is $+1$ and a negative value if the target is -1.

One potential problem with the SRML algorithm is that the network can easily learn to ignore the weights originating from the MRNs. This problem is solved when those weights are fixed instead of trained and the MRV consists of the biases of the nodes it is connected with. To keep in line with the idea of action words and communicating what the task is, we have extracted these values back to the input layer by putting them into PB nodes which each have one weight to the non-input layer that is forever fixed at a value of 1 (see Figure 1(a)). We call these networks *Explicit Parametric Bias (EPB) networks*.

Differences in the number of hidden layers and MRNs are tested. Since having more MRNs means that the hidden nodes can be manipulated more flexibly, it is expected that having more MRNs will result in better performance. Adding more layers to the network makes the transformation from input into output more gradual and it could be that the MRV can manipulate network behavior better at an earlier stage of this transformation.

Finally, three ways of connecting the neurons in the network are tested (see Figure 1(b)). In the default



(a) (b)

Figure 1: (a) In an EPB network there are exactly as many PB nodes as hidden nodes in the first hidden layer and those hidden nodes and PB nodes form pairs. The weights between nodes in a pair are fixed to 1 (the bold lines) while the other weights are fixed to 0 (i.e. non-existent). (b) An EPB network with multiple layer spanning weights (MLSW). In the MLSMW configuration only the bold weights are allowed to span multiple layers.

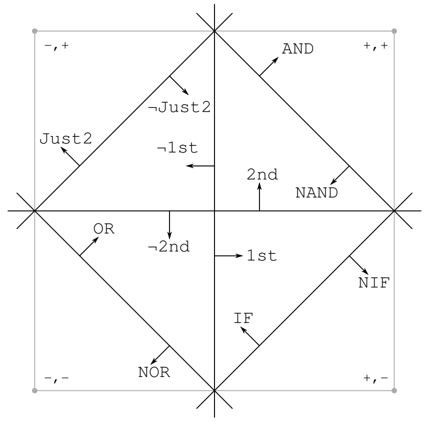


Figure 2: Visualization of all linearly separable tasks defined in Table 1 (i.e. all tasks except `NONE`, `ALL`, `XOR` and `IFF`). The corners of the square represent the different inputs. The black lines separate the input space into a part where the output should be positive (the side where the arrow points) and a part where it should be negative.

configuration neurons are only connected to every neuron in the adjacent layers. In the second configuration we attempted to given the MRNs more power over the output node(s) by allowing connections from a neuron to every other neuron that was nearer to the output layer (instead of just to neurons in the next layer). The third configuration is a hybrid between the first and the second: most neurons are only connected to neurons in adjacent layers, but the MRNs connect to every non-input node. We will refer to these configurations as *default*, *multi-layer spanning weights (MLSW)* and *multi-layer spanning meaningful weights (MLSMW)*.

All of the used networks are trained using the back propagation training algorithm using the squared error function, a variable learn rate and a momentum parameter [8, 22]. Each experiment consisted of 20 trials in which every combination between initial and novel tasks was tested.

5 Results

The relation between the nature of the initial knowledge and each new task greatly affects the chance that the new task can be learned. We evaluate the effects of task difficuly and similarity on this performance as well as the benefits of choosing a good first task.

A task's difficulty changes with the output's dependency on the inputs. For `ALL` and `NONE` there is no dependence, because regardless of the inputs, the output is always the same. For `XOR` and `IFF` the relation between input and output is non-monotonic. As opposed to the other tasks, these two can only be learned in a network with a hidden layer [14]. All other tasks have an intermediate difficulty. It makes some intuitive sense that a network with complex or difficult initial knowledge should be able to learn simpler tasks, whereas for networks with simpler initial knowledge it might be hard to learn the intricacies of the harder tasks. This is also what we see in the performance of most of the networks. Knowing which task is the hardest can help to guide the search for good initial knowledge in more complex task domains.

Similarity is only defined for the tasks with intermediate difficulty and also looks at how the inputs affect the output. Each input can affect the output positvely, negatively or not at all. These effects are visualized by the arrows in Figure 2, where their horizontal direction determines the effect of the first input and their vertical direction the effect of the second. Each line divides the input space into a part where the target output should be positive and a part where it should be negative. The arrows point in the direction of the positive part. Tasks where these effects have the same sign are called *parallel* (e.g. for both `AND` and `OR` both inputs have a positive effect on the output). Two tasks are *similar* if they have the effect of one input in common and if the other input does not have an effect on one of them (e.g. `AND` and `1ST`). It seems intuitive that it is easier to learn a task if you already know something similar and that is indeed what we see in most networks. This measure also has a lot of practical merit, because in most domains it only makes sense to learn related tasks – e.g. a language learning system only needs to be able to learn actual languages, which all have some common properties.

While it would be ideal if the network could always learn every other new task, regardless of its initial

Weights	Hidden Nodes	Representation Nodes	Difficulty	Parallel	Similarity	Prodigy	Overall
default	2	2	73	100	68	51	32
default	4	4	87	100	80	75	42
MLSW	2	3	84	100	69	70	38
MLSW	4	5	91	100	73	85	43
MLSW	4+4	9	95	100	80	91	47
MLSMW	2	3	84	100	74	74	41
MLSMW	4	5	93	100	89	89	53
MLSMW	4+4	9	96	100	92	92	59

Table 2: The results of the experiments. Succes percentages of learning the second task. Difficulty: combinations where the initial task is more difficult than the second; Parallel: all task pairs for which the inputs affect the output in exactly the same way; Similar: all task pairs for which the inputs affect the output in roughly the same way; Prodigy: combinations between the best initial knowledge and all tasks; Overall: all task combinations

knowledge, this is not realistic. It is therefore imperative to find good initial knowledge. An important characteristic of a good network architecture is that it can take advantage of such knowledge. The prodigy measure therefore only looks at the performance of the network with the best initial knowledge (almost always `XOR`, presumably because it is one of the two hardest tasks).

The results of the experiments are summarized in Table 2. The numbers describe the percentage of the time that a new task could be learned in the context of existing knowledge of another task. Table 2 shows clearly that increasing the number of MRNs greatly increases performance. Allowing weights to span multiple layers, especially only the weights connected to the MRNs, increased performance as well.

6 Conclusion

Standard multi-layer perceptrons (MLP) suffer from catastrophic interference. We showed that explicit parametric bias (EPB) networks trained with static meaningful representation learning (SMRL) are able to learn a second Boolean function of two inputs without forgetting the first function learned. The learning success depends on the difficulty and the similarity of the tasks. Everything else being equal, learning is most successful if the first task that is learned is relatively more difficult than the second task. Also, everything else being equal, learning is most successful if the first task that is learned is similar to the second task.

Our results depend on the tasks being represented in a way that is intrinsically 'meaningful' for the network. These representations seem to play a role similar to that of the role of action words in the human brain. Our results highlight the benefits of taking into account knowledge of human cognitive brain functioning, and the design of artficial neural networks to simulate intelligent behavior, in this case learning multiple tasks.

References

[1] W.-K. Ahn and W. F. Brewer. Psychological studies of explanation-based learning. *Investigating Explanation-Based Learning*, 1993.

[2] B. Ans, S. Rousset, R. M. French, and S. Musca. Preventing catastrophic interference in multiple-sequence learning using coupled reverberating Elman networks. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, pages 71–76. Erlbaum, 2002.

[3] R. A. Caruana. *Multitask Learning*. PhD thesis, Carnegie Mellon University, 1997.

[4] M. J. Castro, F. P. T, and F. Casacuberta. MLP emulation of N-gram models as a first step to connectionist language modeling. In *Proceedings of the ICANN '99*, pages 910–915, 1999.

[5] D. Chalmers, R. French, and D. Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology, 1991.

[6] R. M. French. Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. In *Trends in Cognitive Sciences*, pages 128–135, 1999.

[7] M. S. Gazzaniga, R. B. Ivry, and G. R. Mangun. *Cognitive Neuroscience: The Biology of the Mind*. W. W. Norton & Company, 2002.

[8] M. T. Hagan, H. B. Demuth, and M. Beale. *Neural Network Design*. PWS Publishing Co., Boston, MA, USA, 1996.

[9] O. Hauk, I. Johnsrude, and F. Pulvermüller. Somatotopic representation of action words in human motor and premotor cortex. *Neuron*, 41(2):301–307, January 2004.

[10] L. G. Heins and D. R. Tauritz. Adaptive resonance theory (ART): An introduction. Technical Report 95–35, Leiden University, 1995.

[11] E. Kohler, C. Keysers, M. A. Umiltà, L. Fogassi, V. Gallese, and G. Rizzolatti. Hearing sounds, understanding actions: action representation in mirror neurons. *Science*, 297(5582):846–848, August 2002.

[12] B. Kolb and I. Q. Whishaw. *Fundamentals of Human Neuropsychology*. Worth Publishers, 2008.

[13] A. Martin and L. L. Chao. Semantic memory and the brain: structure and processes. *Curr Opin Neurobiol*, 11(2):194–201, April 2001.

[14] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, Massachusetts, 1969.

[15] T. M. Mitchell. The need for biases in learning generalizations, 1980.

[16] L. Y. Pratt. Non-literal transfer among neural network learners. In *Mammone, R. (Ed.), Artificial Neural Networks for Speech and Vision*, pages 92–04. Chapman and Hall, 1992.

[17] L. Y. Pratt, S. J. Hanson, C. L. Giles, and J. D. Cowan. Discriminability-based transfer between neural networks. In *Advances in Neural Information Processing Systems 5*, pages 204–211. Morgan Kaufmann, 1993.

[18] F. Pulvermüller. Words in the brain's language. *Behavioral and Brain Sciences*, 22:253–336, 1999.

[19] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97:285–308, 1990.

[20] A. Röbel. Neural network modeling of speech and music signals. In *NIPS*, pages 779–785, 1996.

[21] A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7:123–146, 1995.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *MIT Press Computational Models Of Cognition And Perception Series*, pages 318–362, 1986.

[23] D. L. Silver and R. E. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. In *Connection Science Special Issue: Transfer in Inductive Systems*, pages 277–294, 1996.

[24] D. L. Silver and R. Poirier. Context-sensitive MTL networks for machine lifelong learning. In D. Wilson and G. Sutcliffe, editors, *FLAIRS Conference*, pages 628–. AAAI Press, 2007.

[25] D. L. Silver and R. Poirier. Requirements for machine lifelong learning. In *IWINAC '07: Proceedings of the 2nd international work-conference on The Interplay Between Natural and Artificial Computation, Part I*, pages 313–319, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] J. Tani, M. Ito, and Y. Sugita. Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB. *Neural Networks*, 17(8-9):1273–1289, 2004.

[27] S. Thrun. Lifelong learning: A case study. Technical Report CMU-CS-95-208, Computer Science Department, Pittsburgh, PA, 1995.

[28] P. D. Turney. The identification of context-sensitive features: A formal definition of context for concept learning. *CoRR*, cs.LG/0212038, 2002.