VICTORIA UNIVERSITY OF WELLINGTON

# Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

# Hierarchical Clustering with ART Neural Networks

G. Bartfai

Technical Report CS-TR-94/1
January 1994

**Abstract**

This paper introduces the concept of a modular neural network structure, which is capable of clustering input patterns through unsupervised learning, and representing a self-consistent hierarchy of clusters at several levels of specificity. In particular, we use the ART neural network as a building block, and name our architecture *SMART* (for *Self-consistent Modular ART*). We also show some experimental results for "proof-of-concept" using the ARTMAP network, that can be seen as an implementation of a two-level SMART network.

**Publishing Information**

This paper is to appear in *Proceedings of IEEE World Conference on Computational Intelligence, 1994* (WCCI'94), Orlando, Florida.

### Author Information

Guszti Bartfai is with the Department of Computer Science, and his e-mail address is `guszti@comp.vuw.ac.nz`

# 1 Introduction

The ability to learn about the environment without a teacher has long been considered an important characteristic of intelligent systems. Unsupervised learning can be found both at sensory-level of mammals, and at higher, cognitive levels of humans. Therefore, it has been an important topic in neural network research (e.g. [9], [6], [5]) for decades. Unsupervised learning networks typically contain two layers of neurons. One is the *input* layer, which is connected fully to the *output* (or *competitive*) layer. When an input is presented to the network, the output node whose connection weight vector is the closest to the current input vector (in some distance metrics) will be the only active node (winner-take-all competition), which will then be allowed to learn, i.e. modify its connection weights from each input neuron. After repeated exposure to the input environment, the weight vectors of the output layer arrange themselves such that their density will represent the probability distribution of the input samples.

However, many unsupervised learning problems that intelligent systems have to face are more complex than "first-order" statistical problems. Often, the environment exhibits some hierarchical class structure (e.g. classification of animals). In such cases, a single layer of competitive learning network will not be able to learn the complex structure of the environment. Moreover, the kinds of clusters the network discovers will very much depend on the number of output nodes in the network. Fewer nodes will result in the network learning fewer, more general classes; with more nodes, however, the network would find a number of smaller, more specific classes. As the number of nodes in such networks is normally fixed, they are not able to represent, therefore learn, a hierarchy of classes that might be present in the input environment. There have been attempts to address this problem. For example, Rumelhart et al. [9] suggest that a hierarchical structure can be formed if a network has several "inhibitory clusters" in each layer working in parallel, and each of them "sees" the output of previous layers only. They, however, left it to the reader to verify the feasibility of this method.

In this paper, we describe our attempts to find a neural network architecture that is capable of developing self-consistent, hierarchical internal representation through self-organisation. We have chosen the Adaptive Resonance Theory (ART) neural networks [2] as building blocks for our experimental system. ART networks have the ability to create new output nodes (i.e. categories) dynamically, and do not suffer from the problem of forgetting previously learned categories if the environment changes. In other words, it offers a unique solution to the well-known *stability-plasticity* dilemma that designers of autonomous intelligent systems have to face. They too, however, can only develop categories of inputs at a given level of specificity, which depends on a global parameter called *vigilance*. Here we claim that a combination of ART networks can be used to develop a consistent internal representation that explicitly represents a class hierarchy. We name this class of modular ART systems *SMART* networks (for *Self-consistent Modular ART*). As a "proof-of-concept" example for a two-level SMART network, we use the ARTMAP network [4] in an auto-associative mode.

We briefly summarise the main features of ART neural networks in section 2, discussing the distinctive features of ARTMAP network in section 2.1. The particular way the ARTMAP network is used in hierarchical clustering is introduced in section 3. Experimental results are shown in section 4, and conclusions are drawn in section 5.

# 2 ART neural networks

The ART architectures are neural networks that develop stable recognition codes in real time by self-organisation, in response to arbitrary sequences of input patterns. They were designed to solve the stability-plasticity dilemma that every intelligent machine learning system has to face. Namely, how to keep learning new things without forgetting previously learned information. A basic ART module has three layers[1]: the *input* layer ($F0$), the *comparison* layer ($F1$), and the *recognition* layer ($F2$), with $N$, $N$ and $M$ neurons, respectively (see module $ART_a$ or $ART_b$ on figure 1). The F1 and F2 layers interact with each other through bottom-up and top-down connections, whose weights are modified when the network learns. At each presentation of a new non-zero binary input pattern $\mathbf{a}$ ($a_i \in \{0,1\}, i = 1,2,...,N$), the network attempts to classify it into one of the existing categories based on its similarity to the stored prototype of each category, represented by individual nodes in the F2 layer. More precisely, for each node $j$ in the F2 layer, $B_j = \sum_{i=1}^{N} b_{ij} a_i$ is calculated, where $b_{ij}$ is the strength of the bottom-up connection between F1 node $i$ and F2 node $j$. Then the top-down template $\mathbf{T_j}$ ($T_{ji} \in \{0,1\}, i = 1,2,...,N$) of the F2 node with the highest $B_j$ is compared to the current input pattern at F1. If $|\mathbf{a} \cap \mathbf{T_j}||\mathbf{a}|^{-1} \geq p$ (where $p$ is a system parameter called *vigilance* ($0 \leq p \leq 1$), and $|\cdot|$ is the norm operator, $|\mathbf{x}| \equiv \sum_{i=1}^{N} x_i$) then input $\mathbf{a}$ is classified to F2 node $j$ and both the bottom-up and top-down connection weights are adjusted for that node: $b_{ij}^{new} = c_i(\beta + |\mathbf{c}|)^{-1}$ and $T_{ji}^{new} = c_i$, respectively, where $c_i$ ($i = 1,2,...N$) is the output of the $i$th node in the F1 layer, and $\beta > 0$ is constant. Otherwise, i.e. when the stored prototype ($\mathbf{T_j}$) does not match the input sufficiently, the winning F2 node is reset for the period of presentation of the current input. Then another F2 node (or category) will be selected, whose prototype will be matched against the input. This "hypothesis-testing" cycle is repeated until either the network finds a stored category whose prototype matches the input closely enough, or allocates a new F2 node, and then learning takes place as described above. After an initial period of self-stabilisation, the network will directly (i.e. without search) access the prototype of one of the categories it finds in a given training set. The higher the vigilance level, the more, finer categories will be developed.

With an ART network, therefore, we can find classes in an input set at a given level of specificity. If we want a more general view of the input set, we need to train an ART network with low vigilance; for a more detailed view, we need to retrain the network with high vigilance, or simply increase vigilance during training. In any case, however, the network will only be able to "see" an input set at a given level of specificity at any given time. There is no relationship between any pairs of category prototypes except that they are in the same network and compete with each other. With only one level of competitive layer, the network is clearly not capable of representing knowledge in a more structured way. Humans, however, are able to structure their knowledge, and therefore, must have a better way of representing that knowledge internally.

We propose a modular neural network structure (*SMART*), in which there are several ART modules operating in parallel at different levels of vigilance. The ART modules have to be connected in a way that would enable the *SMART* network to represent class hierarchies in a self-consistent way (hence the name *SMART*, for *Self-consistent Modular ART*).

In the following, we show how this concept can be implemented directly with an ARTMAP network to give a two-level class hierarchy, and then show some experimental results.

## 2.1 The ARTMAP network

The ARTMAP network can be considered as a two-level SMART network, although the Carpenter et al. had different goals in mind when proposing the ARTMAP architecture

[1]There are both binary (ART1) and continuous (ART2) versions of ART. From now on, I will use ART to refer to ART1, unless otherwise stated.

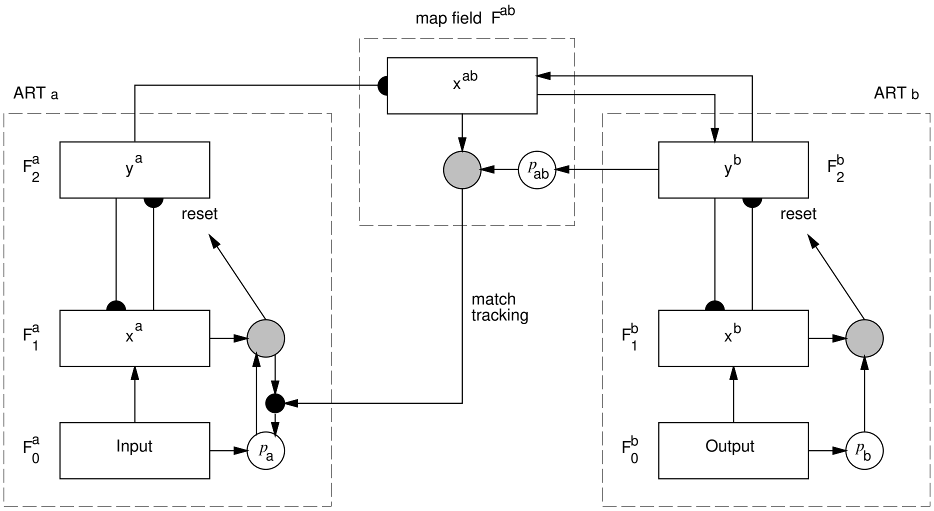


Figure 1: ARTMAP network architecture

in [4][2].

The architecture of the ARTMAP network can be seen in figure 1.[3] It consists of two ART modules that are linked together through an "inter-ART" associative memory, called *map field* ($F^{ab}$). Module $ART_a$ learns to categorise the binary input patterns presented at layer $F_0^a$ (with vigilance $p_a$), while module $ART_b$ develops categories of the target patterns presented at layer $F_0^b$ (with vigilance $p_b$). They do not work independently though. More specifically, when an input pattern is presented, the $ART_a$ module will attempt to make a prediction of the category to which the current target belongs through the map field. If the prediction is right, both modules will learn their respective inputs by modifying the prototypes of the corresponding categories. Otherwise, a process called *match-tracking* starts, whereby the vigilance level of the $ART_a$ module is raised by the minimal amount needed to cause mismatch with the current input, which will subsequently trigger a search for a new, better category. This process continues until the the network either finds an $ART_a$ category that predicts the category of the current target correctly, or creates a new $F_2^a$ node and a corresponding link in the map field, which will learn the current input-target pair correctly. The $ART_a$ vigilance is then allowed to return to its resting level ($p_a$).

# 3 Learning two-level class hierarchies with ARTMAP

The above mechanism works for arbitrary sequences of input/target pairs of binary vectors, so the network can learn an arbitrary binary mapping. Carpenter & al. in [4], however, only used the ARTMAP network where the target patterns were recognition categories of their corresponding input. The target patterns, therefore, belonged to mutually exclusive classes, and the $ART_b$ module was simply used to indicate which target (or recognition category) was presented, i.e. it was not forced to do any generalisations of its prototypes.

Since our goals were not to learn arbitrary mappings, but to see how a combination of two ART modules (which can be taken as an implementation of a two-level SMART network)

[2]Although the ideas introduced here can directly be applied to the continuous version of ARTMAP (Fuzzy ARTMAP, see [3]), we will apply them to binary patterns for simplicity.

[3]This figure is a simplified version of the one that appeared in [3].

can be trained to develop explicit representations for a hierarchy of classes present in the input space. So we used the ARTMAP network in an *unsupervised* way, by presenting the input patterns to *both* the $ART_a$ and $ART_b$ modules (auto-associative mode). With different settings of the two vigilance levels ($p_a$ and $p_b$), we can force the two ART modules to develop categories of the input patterns at different levels of specificity. As a result of the ARTMAP processing algorithm, the $ART_a$ module will attempt to make a prediction of the $ART_b$ category, which will be another *input* category. If the prediction is wrong, the $ART_a$ module will search for a better category via the match-tracking process.

In particular, we can let module $b$ develop fewer, more general categories (with a low level of $p_b$), and module $a$ to develop a greater number of more specific categories. In this case, by presenting an input pattern, the network will have access to both a specific and a general category such that the specific one is a subcategory of the more general one. Therefore, the two networks will form a self-consistent representation of a two-level hierarchy of clusters by several $ART_a$ clusters belonging (i.e. having explicit access) to the same, more general, $ART_b$ cluster. This is a feature that no single ART module is able to achieve, nor a pair of ART modules if they work independently.

## 4 Experimental results

We carried out experiments on the "zoo" machine learning benchmark database [8]. It is a simple database containing 101 instances of animals described with 18 attributes. Out of these attributes, we used the 15 boolean ones that indicate the presence or absence of certain features like "hair", "aquatic", "domestic" and so on. We also used the "number of legs" attribute, which is a set of 6 integers. We did not use the "type" attribute, as we did not want to "distort" the learned clusters with the additional information of predefined categories.[4] The patterns were presented to the ARTMAP network in *complement coding*. This was suggested in [4], which "has a useful role in searching for appropriate recognition codes in response to predictive feedback". However, complement coding was only introduced for binary patterns there. Since one of the features is not binary (see "legs" above), we had to encode that feature as well in a way that is compatible with complement coding. For this, we have used a coding scheme that we call *generalised complement coding*, which applies to arbitrary-sized sets.

According to generalised complement coding, a value $v_i$ of a set $\mathcal{S} = \{v_0, v_1, ..., v_{s-1}\}$ will be mapped to a binary vector ($\mathbf{c}$) of length $s$ such that

$$c_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad j = 0, 1, ..., s-1.$$

So, for example, if an attribute takes on the value 4 from the set of $\{0, 2, 4, 5, 6, 8\}$, then the corresponding binary vector will be 001000. In case of $s = 2$, we get back the original form of complement coding.

This way, we can keep the size of the (binary) input pattern to the network constant, thereby implementing a form of "binary normalisation" of the input. In the case of the database used in the above way, the input and target patterns were 36-element binary vectors, in which the number of 1's was always the same (16).

The simulations were carried out using a public domain neural network simulator program (`PlaNet`, see [7]) running under Unix and X-windows.

The network parameters were chosen to be similar to those presented in [4]. In particular, the initial values of the bottom-up weights in both ART modules were chosen in such a way

[4]The 18th attribute ("animal name") was simply used as a label for the individual instances.

| | $p = 0.1$ | | |
|---|---|---|---|
| *Attribute* | $C_1$ | $C_2$ | $C_3$ |
| hair | – | – | – |
| feathers | – | 0 | 0 |
| eggs | – | – | – |
| milk | – | 0 | – |
| airborne | – | – | 0 |
| aquatic | – | – | – |
| predator | – | – | – |
| toothed | – | – | – |
| backbone | – | – | 1 |
| breathes | 1 | – | – |
| venomous | – | – | 0 |
| fins | – | – | – |
| legs | – | – | – |
| tail | 1 | – | – |
| domestic | – | – | – |
| catsize | – | – | – |
| *size* | 24 | 20 | 57 |

| | $p = 0.4$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *Attribute* | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ |
| hair | – | – | 0 | 0 | – | – | 0 | 1 |
| feathers | – | 0 | – | 0 | – | 0 | 0 | 0 |
| eggs | – | 0 | 1 | – | 1 | 0 | 1 | 1 |
| milk | – | – | 0 | 0 | 0 | 1 | 0 | 1 |
| airborne | – | 0 | – | 0 | 1 | 0 | 0 | 0 |
| aquatic | – | 0 | – | – | 0 | – | 1 | 1 |
| predator | – | – | – | – | 0 | – | – | 1 |
| toothed | – | – | 0 | 1 | 0 | 1 | – | 0 |
| backbone | 1 | – | – | 1 | – | 1 | – | 1 |
| breathes | 1 | 1 | – | – | 1 | 1 | 0 | 1 |
| venomous | 0 | – | 0 | – | – | 0 | – | 0 |
| fins | 0 | 0 | 0 | – | 0 | – | – | 0 |
| legs | – | – | – | – | – | – | 0 | 4 |
| tail | 1 | 1 | – | – | – | – | – | 1 |
| domestic | 0 | – | 0 | 0 | – | – | – | 0 |
| catsize | 0 | – | – | – | 0 | – | 0 | 1 |
| *size* | 12 | 1 | 18 | 9 | 12 | 38 | 10 | 1 |

Table 1: Category prototypes created by an ART module at two different levels of vigilance.

that $F_2$ nodes become active in the order $j = 1, 2, ...$ and are small enough so the network selects an uncommitted node only if $|\mathbf{a} \cap \mathbf{T}_j| = 0$ for all committed nodes. Also, the $\beta$ parameter was taken to be sufficiently small that, among committed nodes, $T_j$ is determined by the size of $|\mathbf{a} \cap \mathbf{T}_j|$ relative to $|\mathbf{T}_j|$. The network was also used in *fast learning* mode, i.e. during learning, the connection weights were allowed to reach their asymptotic values while the current input was presented.

We also had to modify the ARTMAP match tracking algorithm slightly to let the network learn the correct mapping in certain situations where the original algorithm would not have learned it otherwise. This problem, that is claimed in [4] to never arise if complement coding is used, has been analysed further in [1].

Training proceeded by showing the input/input pairs repeatedly, until the network stabilised itself, i.e. each input read out its prototype directly, so no further learning took place.

Table 1 shows the prototypes of categories ($\mathbf{T}_j$ as column vectors, in a more readable format) that were created when a single ART module was trained fully on the training set,[5] at two different levels of vigilance ($p = 0.1$ and $0.4$). In the last row, we can also see the size of each category, i.e. the number of inputs that belong to that category. (Note that each category has some critical features, whose values have to match those of the input, and some "don't care"'s, too.) It is seen clearly that the network created more and finer (i.e. more specific) categories with the higher vigilance level.

In comparison, table 2 shows the category prototypes of both the $ART_a$ and $ART_b$ modules, with vigilance levels $p_a = 0.4$ and $p_b = 0.1$, respectively. The network was trained on exactly the same data set, in exactly the same order of input presentation as above. Columns $C_i$ $(i = 1, 2, 3)$ show the $ART_b$ category prototypes, and columns $C_{ij}$ contain the prototypes of $ART_a$ categories that belong to class $C_i$. First, we can observe that the $ART_b$ category

[5] Note that in this work, there was no test set, as the goal was not to measure the performance of the network, but to investigate its internal representation.

| *Attribute* | $C_1$ | $C_{11}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ | $C_{15}$ | $C_2$ | $C_{21}$ | $C_{22}$ | $C_{23}$ | $C_{24}$ | $C_3$ | $C_{31}$ | $C_{32}$ | $C_{33}$ | $C_{34}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| hair | – | – | – | – | 0 | 0 | – | 0 | – | 0 | 0 | – | – | – | 0 | 0 |
| feathers | – | – | 0 | – | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| eggs | – | – | 0 | – | 1 | 1 | – | 1 | 1 | – | 1 | – | 0 | 1 | 1 | 1 |
| milk | – | – | – | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – | 1 | – | 0 | 0 |
| airborne | – | – | 0 | – | 1 | 0 | – | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| aquatic | – | – | 0 | – | 0 | 0 | – | 1 | 0 | – | 0 | – | – | 1 | 1 | – |
| predator | – | – | – | – | – | 1 | – | – | – | 1 | 1 | – | – | – | – | – |
| toothed | – | – | – | – | 0 | 1 | – | – | 0 | 1 | 0 | – | 1 | – | 1 | – |
| backbone | – | 1 | – | 1 | 1 | 1 | – | – | – | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| breathes | 1 | 1 | 1 | 1 | 1 | 1 | – | – | 1 | – | 0 | – | 1 | – | 0 | 1 |
| venomous | – | 0 | – | 0 | 0 | 1 | – | – | – | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| fins | – | 0 | 0 | – | 0 | 0 | – | – | 0 | 0 | 0 | – | – | – | 1 | 0 |
| legs | – | – | – | 2 | 2 | 0 | – | – | – | 0 | 0 | – | – | – | 0 | – |
| tail | 1 | 1 | 1 | 1 | 1 | 1 | – | – | – | 1 | 0 | – | – | 1 | 1 | – |
| domestic | – | 0 | – | 0 | – | 0 | – | 0 | – | 0 | 0 | – | – | – | 0 | 0 |
| catsize | – | 0 | – | 1 | 0 | 0 | – | – | – | 0 | 0 | – | – | – | – | – |
| *size* | 24 | 7 | 1 | 6 | 9 | 1 | 20 | 8 | 10 | 1 | 1 | 57 | 38 | 2 | 11 | 6 |

Table 2: Category prototypes created by two ART modules connected in an "ARTMAP style".

prototypes are identical to those created by the single ART module above with $p = 0.1$. We can also see, on the other hand, that the $ART_a$ categories are different from those in table 1, and there are more of them. The table clearly shows that

- there is a clear hierarchy in the network in that a more general class is connected to several subclasses that inherit some features from their parent class as well as contain more features specific to that particular subclass,
- the $ART_b$ (or more general) module is forcing its categorisation onto the $ART_a$ (or more specific) module,
- the effect of this "forced categorisation" is an increased number of categories compared to the independent case.

We can also see that a feature has a specific value in an $ART_b$ prototype if and only if all its subclasses have that feature set to the same value.

### The "category fragmentation" phenomenon

From tables 1 and 2, we can also see that there is a significant variation in the sizes of $ART_a$ categories *within* a more general ($ART_b$) class. Moreover, in the majority of the experiments we carried out, the distribution of the sizes were such that there were a few large subclasses, and several small ones, most of them significantly smaller (less than half, or even smaller) than the large ones. Also, on average, the number of $ART_a$ categories were twice as many as those when the module learned independently with the same level of vigilance. The increments were made up by these relatively small subcategories. So what is the cause of this significant increase in the number of small categories (sometimes containing only one input)?

The explanation lies in the fact that the $ART_a$ module is learning the categorisation under an internal supervision "from above". The effect of this is that some categories that the $ART_a$ module would develop were it learning independently overlap with some $ART_b$ categories. Such regions were forced to split up, and created their own subcategories within their respective $ART_b$ categories. This fragmentation phenomenon is, therefore, a direct consequence of the "top-down" control structure of the network.

# 5 Conclusion

In this paper, we have proposed the concept of a modular neural network structure containing several ART network modules (SMART), which is capable of representing class hierarchies at several levels of specificity in a self-consistent way.

We have also shown some "proof-of-concept" experimental results when the ARTMAP neural network is used as a two-level SMART network.

We need to make further experiments (on other databases as well) to gain more insights into the nature of such hierarchical classifications using neural networks, and to see how a self-organising neural network can make explicit use of the created prototypes at different levels of generality. Among the questions we would like to see answers are:

- How can we make explicit use of the hierarchical representation stored in the network?
- What do we gain by increasing the number of levels?
- What would be a generic method for connecting two or more modules: ARTMAP-type, or other?
- How many different levels do we need?
- Can we create/delete modules and/or adjust their level of specificity dynamically?

We would also like to carry out more systematic investigation into the category fragmentation phenomenon reported here.

We have been encouraged by the experimental results, despite some apparent problems, and feel that we can provide some fruitful grounds for converging towards the common goal of traditional (symbolic) AI and neural network research: to understand the nature of "intelligence".

# Acknowledgements

I would like to thank Peter Andreae for his valuable comments and provoking questions throughout this work, and commenting on the draft of this paper.

This work was in part supported by Victoria University of Wellington (Research Grant No.RGNT/594/371).

# References

[1] G. Bartfai. The Match Tracking Problem of the ARTMAP Neural Network. Technical Report CS-TR-94/2, Department of Computer Science, Victoria University of Wellington, New Zealand, January 1994.

[2] G.A. Carpenter and S. Grossberg. A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[3] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, September 1992.

[4] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network. *Neural Networks*, 4:565–588, 1991.

[5] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26(23):4979–4984, December 1987.

[6] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[7] Yoshiro Miyata. *PlaNet, A Tool for Constructing, Running, and Looking into a PDP Network*, 1991.

[8] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases [machine-readable data repository]. Technical report, Department of Information and Computer Science, University of California, Irvine, CA, 1992.

[9] D.E. Rumelhart and J.L. McClelland, editors. *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, volume 1, chapter 5, pages 151–193. The MIT Press, 1986.