Towards Nature-Inspired Modularization of Artificial Neural Networks via Static and Dynamic Weights

Alfons Schuster[1] and Daniel Berrar[2]

[1] Waseda University, School of International Liberal Studies, Tokyo, Japan
[2] Tokyo Institute of Technology, Yokohama, Japan
a.schuster@aoni.waseda.jp, berrar.d.aa@m.titech.ac.jp

Abstract. Many conventional artificial neural network (ANN) models are designed for one application domain only. The work presented in this paper describes ANN models that operate with a higher economy by sharing neurons across domains. The use of two different types of weights—*static* weights and *dynamic* weights—is a fundamental feature of the presented models. Results from a comprehensive series of experiments provide evidence for the meaningfulness of the proposed models.

1 Introduction

Artificial neural network studies usually involve a training session and a testing session. The training session typically begins with a random weight assignment to the network. These initial weights are then modified during the learning process. The learning process uses a training set to modify the weight configuration according to some predefined criteria. Whenever these criteria are met, the training process halts. Eventually, these procedures result in a network that is optimized for a specific, domain-related, problem-solving task. Crucially, the final weight configuration remains fixed, unless the network needs retraining for some reason. Thus, the conventional training process determines an ANN for a particular task. Another way to state this is that:

(i) Conventional ANN learning adapts a network to the input data.

Our previous work investigated the economy of such an approach and introduced ANN models deviating from the conventional approach. For instance, [Schuster, 2003b] introduced the idea of data-space transformation. This approach also begins with a random weight assignment to a network. Importantly, this initial weight assignment remains *unchanged* in the learning process, while some adaptation occurs on the input data side via a data-space transformation procedure. This transformation approach can be summarized as follows:

© Springer-Verlag Berlin Heidelberg 2014

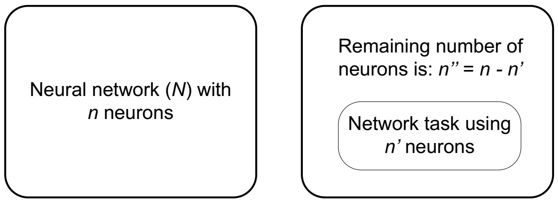

Neural network (N) with n neurons
Remaining number of neurons is: $n'' = n - n'$
Network task using n' neurons

Fig. 1. Problem of limited neuron supply

(*ii*) The data-space transformation learning approach adapts input data to a network.

Our earlier work [Schuster, 2003a] was also driven by the problem illustrated in Figure 1. The figure illustrates a neural system (N) with an arbitrary number (n) of neurons. The architecture of this network is irrelevant, and neither does it matter whether the figure refers to a biological system or an artificial system. Imagine, however, that the number (n) of neurons is large enough for the network to learn multiple tasks (e.g., by allocating n_1 neurons for task T_1, n_2 neurons for task T_2, and so on). Also assume that the network has not learned anything yet, and that all neurons are freely available. Now imagine a learning task T_1 undertaken by this network. This learning task may involve n_1 of the original n neurons (thus, $n_1 \leq n$). A viewpoint where a network structure remains static for a particular application now suggests the following:

(*i*) The n_1 neurons occupied for learning task T_1 are no longer available for any other learning task.
(*ii*) The number of neurons n' remaining for other learning tasks is $n' = n - n_1$.

These two suggestions now lead to the following conclusion:

(*iii*) No matter how many neurons are initially available in the network N, eventually the network is going to run out of neurons. That is, unless there is an approach that works differently from the conventional approach where neurons remain static and belong to an architecture once this architecture has been generated for a particular application.

Here, we want to mention, briefly, that biological neurons are anything else than static network components. For instance, the human brain not only contains various types of neurons [Augustine et al., 2004, pp.1-28], it also contains regions [Spalding et al., 2013] where subpopulations of neurons are exchanged on a daily basis. Even individual neurons constantly undergo structural change [Schuemann et al., 2013]. The ANN research domain is, of course, informed about these issues. For instance, [Yao, 1993] describes dynamic topologies where neurons or weighted connections could be either added or pruned (e.g., during the learning phase) dynamically via evolutionary algorithms, while others investigates the stability of working memory [Alnajjar et al., 2013]. Our previous work

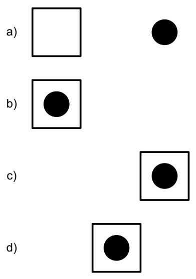


Fig. 2. Problem of the black dot and the white square

(e.g., [Schuster, 2003a]) relates to several aspects of these problems too. The aim in this paper is to extend this earlier work in two directions, namely: (*i*) to extend our earlier models to more complex multi-layer networks, and (*ii*) to develop models that are inspired by natural modularization. Please note, that a fundamental aspect of our work is to get a *first impression* about the overall quality and naturalness of our models. Of course, our models should stand up to current models in terms of classification performance, but they are not necessarily required to be networks with superior classification performance.

2 Network Economy

Initially, we want to describe two concepts that are relevant for the presented work: (*i*) sharing neurons and (*ii*) data-space transformation.

2.1 Sharing Neurones

This section first elaborates on the possible impact the sharing of neurons may have on a network structure.

As an introductory example, let the black dot and the white square in Figure 2 (a) be objects in a 2-dimensional space. Now, imagine the task of changing the spacial arrangement of these two objects such that the black dot ends up in the white square. Figure 2 illustrates several solutions to this task. In Figure 2 (b), the white square moves towards and encloses the black dot, while the black dot remains in its original position. Figure 2 (c) works the other way round: the black dot moves, and the white square remains in its initial position. Figure 2 (d) illustrates the case where both objects change their positions.

Let us now adopt the rather abstract point of view where the white square represents an ANN and the black dot a unit that provides (or pre-processes) some input data. From this point of view, we can imagine that in Figure 2 (b) the ANN configuration is modified (e.g., trained) in such a way that it accommodates the input data. Likewise, in Figure 2 (c) the reverse may happen: the input data (black dot) is transformed in 2-space such that it fits into the unchanged ANN (white square). It is easy to imagine that in Figure 2 (d), both entities (ANN

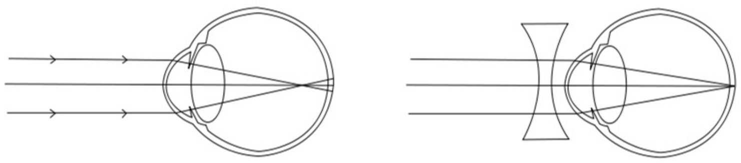


Fig. 3. Data-space transformation in nature

and input data unit) "join forces to get the job done". In a final effort, we may contemplate the possible nature of the objects involved. For instance, let the white square represent a biological brain and the black dot a biological eye. Now, Figure 2 (d) may illustrate the case where a brain and an eye work together in some way to achieve the task of (natural) visual object recognition. This example leads us to the idea of data-space transformation.

2.2 Data-Space Transformation

In order to understand the importance of data-space transformation, consider the case of shortsightedness in a biological eye (Figure 3)[1].

Shortsightedness is a defect in which the eye produces the focus at the wrong place, namely in front of the retina when accommodation is relaxed. The problem can be alleviated through the use of corrective lenses, which, essentially, help focusing the light onto the right place on the retina. A trivial, albeit important point here is the following: shortsightedness is not a problem of the brain—it is a problem of the eye. The brain is working fine, but the data that it receives from the eye is not optimal. In relation to the presented work, it is important to note that the neural systems in the brain and in the eye remain largely unchanged in this process. What undergoes change is the *data*, which is light in this example. Before the light enters the eye, it is altered by the lens in front of the eye. Shortsightedness is only one example, and many similar real-world examples could be cited. Just think of a hearing aid where sound is amplified and modulated, or a cardiac pacemaker, which uses synthetically produced impulses for heart rate control. In both cases, the wider neural system of the respective organs processes pre-transformed signals. Our earlier work has been motivated by this kind of natural data-space transformations. For instance, [Schuster, 2003b], tested the value of the data-space transformation approach for perceptron-style learning. Basically, a perceptron is a single neuron system that can solve linearly separable classification tasks [Mehrotra et al., 1997].

Figure 4 (a), where black dots represent objects of *Class 1*, and lined circles objects of *Class 2*, illustrates such a classification task. Typically, the perceptron training algorithm starts with a random weight assignment. In the context of the

[1] This figure is licensed under Creative Commons ShareAlike 1.0.

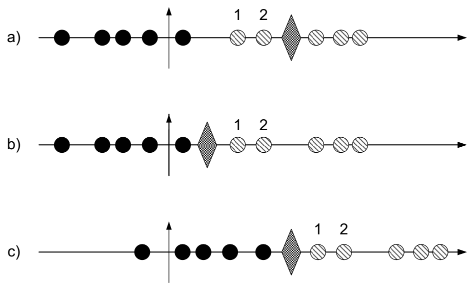


Fig. 4. Solutions to a one-dimensional linearly separable classification task

one-dimensional task at hand, such a random weight assignment represents an arbitrary dividing point on the x-axis. Let the dotted diamond in Figure 4 (a) be such an initial random dividing point. Figure 4 (a) illustrates that this initial assignment does not separate all objects into their corresponding class, because object 1 and object 2 in Figure 4 (a) are misclassified (they are located on the "wrong" side of the diamond). In order to achieve the correct classification of all objects, the perceptron training algorithm alters the initial weight assignment in a number of successive steps. This process usually continues until, either all objects are classified correctly, or until some predefined stopping criterion is met. Geometrically, the process of continuously altering the weight configuration is equivalent to an organized movement of the initial dividing point on the x-axis. In the current example, the training process may have moved the initial dividing point into the position illustrated in Figure 4 (b). This position represents a solution to the problem because all objects of *Class 1* are now on one side of the dividing point, and all objects of *Class 2* are on the other side of this point. Figure 4 (c) illustrates an alternative (data-space transformation) solution to the problem. In this solution, the initial weight configuration (the initial position of the dividing point) remains *unchanged*. Instead, the learning algorithm alters, in a coordinated way, the location of the objects on the x-axis. Figure 4 (c) illustrates that this change is similar to an offset along the positive direction on the x-axis. Note the similarity between this example and the example on short-sightedness! Our earlier work formalized the data-space transformation process for the perceptron case. Our motivation here is (*i*) to extend our earlier models towards more complex, multi-layer networks, and (*ii*) to pay attention to the issue of neurons that are shared across multiple domains or processing units.

3 Experiments

Overall, we carried out five *basic* studies (referred to as Study-1, Study-2, ..., to Study-5), and one comparative study in total. Study-1 involves a conventional ANN. Study-2, Study-3, and Study-4, apply the idea of static weights and dynamic weights, while Study-5 considers the value of the presented models in the

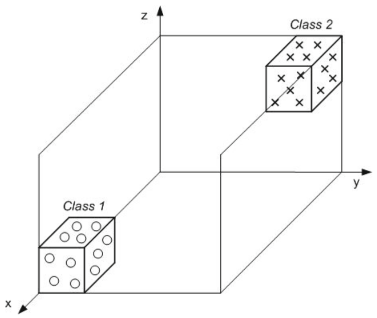


Fig. 5. Data used in Study-1 to Study-5

wider context of natural information processing. Study-1 to Study-5 apply our ideas on simple problems. The goal of the comparative study is to find out how the proposed models perform in more complex, real-life situations.

Each network study considers the typical ANN issues related to network topology, network training, and performance evaluation. For instance, the choice of network topology is often influenced by the domain data. Network training involves decision-making related to the choice of learning algorithm, learning rates, and stopping criteria, while network evaluation considers the performance of a network. Here, the network topology changes in each study. In terms of learning algorithm, all studies use the backpropagation learning algorithm, or variations of it. (Please note that, arguably, the backpropagation learning algorithm is still one of the most common learning algorithms for iterative weight adaption [Werbos, 1974] and variants thereof [Bishop, 1996].) The procedures for network training and network performance evaluation are essentially the same across all studies. For instance, the training phase in each basic study involves 100 iterations. Each iteration presents all records in a training set (here, 100 records) to the network. A network quality assessment procedure records the number of correctly classified records in the training set and in a test set (also 100 records). The test set contains records that were not involved in the training phase of the network. The performance on the test set, therefore, is an indicator for how well the network performs on previously unknown records. The forthcoming text refers to the complete set of processes just mentioned as a *session*. Finally, based on uniform random sampling from the domains shown in Figure 5, we generated synthetic training sets and test sets. Each set contains 100 records (50 from *Class 1*, and 50 from a *Class 2*). Each class is defined by three attributes (*Class 1*: $4 \leq x \leq 5$, $0 \leq y \leq 1$, $0 \leq z \leq 1$; *Class 2*: $0 \leq x \leq 1$, $4 \leq y \leq 5$, $4 \leq z \leq 5$). The two classes are linearly separable.

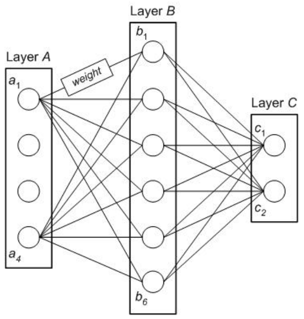


Fig. 6. Study-1 (Model-1). This "conventional" network uses "conventional" weights between all layers.

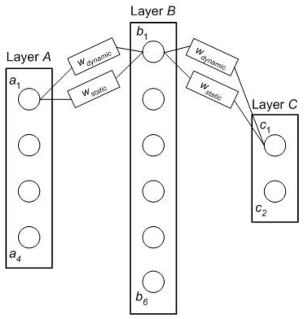


Fig. 7. Study-2 (Model-2). Every weight consists of a dynamic weight component and a static weight component. An update only changes dynamic weights.

Table 1. Results for Study-1 to Study-4 (averaged over 100 sessions)

	Training Set			**Testing Set**		
Study	**Total**	**Class-1**	**Class-2**	**Total**	**Class-1**	**Class-2**
Study-1 (Model-1)	99.5%	100.0%	99.0%	99.5%	100.0%	99.0%
Study-2 (Model-2)	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Study-3 (Model-3)	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Study-4 (Model-4)	99.5%	99.0%	100.0%	99.5%	99.0%	100.0%

3.1 Study-1: Network Model-1

Figure 6 illustrates the topology of the (conventional) ANN in Study-1. Throughout this text, this model is referred to as Model-1. Note that all forthcoming basic studies use this network as a point of reference. The network in Figure 6 has 4 neurons in the input layer (A), 6 neurons in the hidden layer (B), and 2 neurons in the output layer (C). We can assume that the network in Figure 6 is fully connected. Each neuron in the input layer is connected to each neuron in the hidden layer, and each neuron in the hidden layer is connected to each neuron in the output layer. For simplicity, Figure 6 only shows connections coming from two input layer neurons (a_1, and a_4). In ANN terminology, a connection between two neurons is called a *weight*. A weight represents the degree or strength of association between two neurons. Figure 6 illustrates the weight of one connection, that between neurons a_1 and b_1, as a rectangular box. The manipulation and evaluation of all weights in a network are a fundamental element in our studies.

Results of Study-1 (Model-1). Table 1 holds the results for Study-1 to Study-4. The table indicates a high score across all Study-1 outcomes (training set, testing set, total score, and individual scores by class). Remember that a single session trains an individual network on 100 records. After training, the network is evaluated on the training set (100 records) but also on a test set (100 records). A single session, therefore, evaluates 200 records. Thus, 100 sessions require the evaluation of a total of 100 × 200 = 20,000 records. Overall, however, the outcome for Study-1 is not too surprising. In fact, we would expect such a performance for such a simple domain. However, please remember that in this work, the basic studies are deliberately kept trivial.

3.2 Study-2: Network Model-2

Figure 7 illustrates the first alternative model, referred to as Model-2. The architecture of the network in Figure 7 is similar to that of the conventional network in Figure 6 (both networks are fully connected, with 4 neurons in the input layer, 6 in the hidden layer, and 2 in the output layer). However, Figure 7 also emphasizes the main characteristic of Model-2. Every single connection between two neurons (e.g., that between neurons a_1 and b_1, or that between neurons b_1 and c_1) is composed of two different types of weight. The first type of weight is a *static* weight, whereas the second type of weight is a *dynamic* weight. Initially, a network receives a random allocation of weight values. This allocation includes static weights as well as dynamic weights. Network learning, however, only updates the dynamic weights in the network. The static weights remain unaltered in the training phase. In operation (e.g., during a classification task), however, the network combines both weights by summing up their individual values. (Please note that we understand that other methods of combination, such as multiplication or average, for instance, are possible. At this stage, however, we consider other alternatives to be outside the scope of the presented work.) For illustration purposes, consider the dynamic and static weights connecting neurons a_1 and b_1 in Figure 7. We can express the initial random weight allocation for the combined, dynamic, and static weights as follows:

$$w^0(a_1, b_1) = w_d^0(a_1, b_1) + w_s^0(a_1, b_1), \tag{1}$$

where the combined weight $w^0(a_1, b_1)$ is the sum of the dynamic and static weight. At a particular point in the network learning process, say, after iteration i, the combined weight $w^i(a_1, b_1)$ between neurons a_1 and b_1 equals:

$$w^i(a_1, b_1) = w_d^i(a_1, b_1) + w_s^0(a_1, b_1). \tag{2}$$

Further, if network learning completes after n iterations, then the altered combined weight $w^n(a_1, b_1)$ between neurons a_1 and b_1 consists of the sum of the changed dynamic weight $w_d^n(a_1, b_1)$ and the unchanged static weight $w_s^0(a_1, b_1)$:

$$w^n(a_1, b_1) = w_d^n(a_1, b_1) + w_s^0(a_1, b_1). \tag{3}$$

Note, that the static weight $w_s^0(a_1, b_1)$ remains unchanged throughout the training process. During operation, the network uses the final, combined weight value $w^n(a_1, b_1)$ for classification.[2]

Results of Study-2 (Model-2). Table 1 indicates that over 100 sessions, Model-2 achieved a classification performance of 100% accross all categories. Although this outcome is (slightly) better than that achieved by the conventional model in Study-1, we consider both studies to perform equally well (e.g., at other occasions, Study-1 also achieved such a perfect score). In comparison to Study-1, however, where we may expect a good outcome, for Study-2 this outcome may be a bit more surprising. For instance, we mentioned earlier that the network parameters are the same in all studies. Because of this, we may feel that a more complex network (more weights) may need more training iterations to learn the data. We may also feel that there could be minor changes because of rounding issues when the algorithm processes a larger number of numerical values. This work is not concerned about such issues at this moment. Instead, we move on to the next experimental investigation.

3.3 Study-3: Network Model-3

The networks in Study-3, as well as those in the forthcoming Study-4, are hybrids between the networks described in Study-1 and in Study-2. In a typical Study-3 network (Figure 8) a connection from the input layer to the hidden layer consists of a dynamic weight component and a static weight component. Connections between the hidden and the output layer consist of conventional weights only. The learning algorithm for such a network only updates: (*i*) conventional weights, and (*ii*) dynamic weights. All static weights in the network remain unchanged.

[2] Please note that, strictly speaking, an approach that uses the simple addition of static weights and dynamic weights for the production of an aggregate weight value is (mathematically) equivalent to a standard implementation of the backpropagation algorithm. Nevertheless, in the context of the current work there are important differences. For example, although the sum of a dynamic weight and a static weight may be mathematically equal to a single (standard backpropagation) weight value, in physical terms there are really two different weights in the network. A network including static weights and dynamic weights, therefore, has significantly more weights. This, however, can affect network parameters such as learning rate, or number of iterations, for example. Indeed, the deviations in network performance we observe in our work may be explained by these differences. Another important issue is network robustness. Our previous work on network robustness clearly suggests that networks with a mixture of static weights and dynamic weights should be less prone to errors such as, for example, weight loss or noisy data. Further, it is also important to recall that the simple addition of a static weight and a dynamic weight is one of many options to combine weights and to integrate these weights in a network. Lastly, we want to mention that our work is a conceptual contribution to the field of ANNs. It is for later sections (and, perhaps, for further work), however, to explain, more clearly, the potential this conceptual contribution may provide.

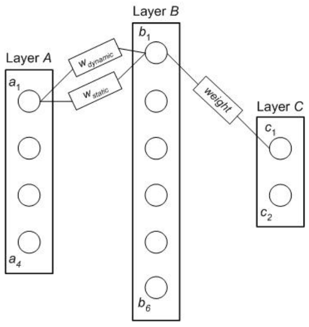


Fig. 8. Study-3 (Model-3)

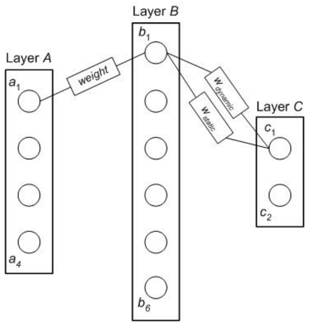


Fig. 9. Study-4 (Model-4)

Results of Study-3 (Model-3). It is easy to see that Table 1 indicates a perfect score (100%) accross all categories for Study-3 (Model-3).

3.4 Study-4: Network Model-4

Figure 9 illustrates a typical Study-4 (Model-4) network. In such a network, the space between the input layer and the hidden layer contains only conventional weights. By contrast, the hidden layer and the output layer are connected by weights with a dynamic weight component and a static weight component. The learning algorithm takes care of these weights in the familiar way: it updates conventional weights and dynamic weights, but leaves all static weights unchanged.

Results of Study-4 (Model-4). Table 1 shows that although Study-4 (Model-4) does not achieve a perfect score across all categories, overall, its scores are comparable to those achieved in all other studies. Study-4 concludes the first set of experimental studies. In summary, we find that the different models perform more or less the same. In our experiments, on average, there is no significant difference in terms of classification performance. At first glance, this outcome may not be particularly impressive, because the classification problem is rather trivial. This impression may change, however, when we consider our models in a wider, more natural context.

3.5 Comparative Studies

The basic studies Study-1 to Study-4 form a unit in this work. This section, therefore, has the goal to compare the quality of the proposed models on a more complex, real-life dataset. We have chosen the Iris Plant dataset, one of the best known datasets found in the pattern recognition literature. The Iris Plant dataset contains 150 records [Blake and Merz, 1998]. Each record contains four attributes: the lengths and widths of sepals and petals of one of three iris classes (Iris-setosa, Iris-versicolor, and Iris-virginica). Each class contains 50 records. One class is linearly separable from the other two classes, but the latter two

Table 2. Comparative results. Study-1 to Study-4 on the Iris Plant dataset.

	Training Set				Testing Set			
Study	**Total**	**Class-1**	**Class-2**	**Class-3**	**Total**	**Class-1**	**Class-2**	**Class-3**
Iris-Study-1	90.8%	100.0%	73.1%	99.5%	89.8%	99.9%	72.1%	97.4%
Iris-Study-2	89.9%	100.0%	70.0%	99.6%	88.6%	99.8%	68.7%	97.4%
Iris-Study-3	89.6%	100.0%	69.2%	99.6%	88.1%	99.9%	67.3%	97.0%
Iris-Study-4	89.8%	100.0%	69.9%	99.5%	89.0%	99.8%	69.1%	97.9%

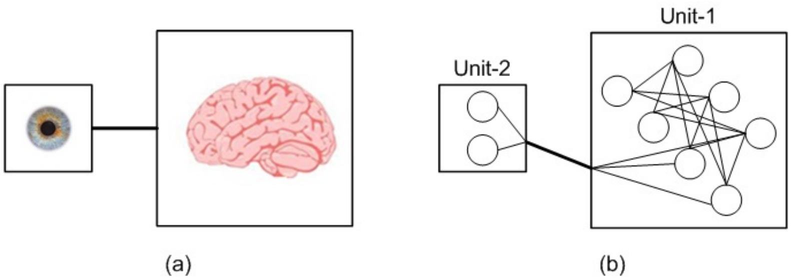


Fig. 10. (a) Eye-brain reference system. (b) Partitioning into subunits.

classes are not linearly separable. Our goal is to replicate Study-1 to Study-4 on the Iris Plant dataset. In terms of data processing, we decide on a network (5 input neurons, 6 hidden layer neurons, and 3 output neurons), use a random procedure to divide the dataset in a training set and a testing set of equal size (75 records, with 25 records per class), then train and test the network. The process is repeated over 100 sessions and the results are averaged over these 100 sessions. Table 2 summarizes the outcomes for the various studies. Note that in order to differentiate the comparative studies involving the Iris Plant dateset from the basic studies, we use the label Iris-Study-X (X standing for 1 to 4).

Results of Comparative Studies. Iris-Study-1 in Table 2 is the starting point for our discussion. Remember, the network in Iris-Study-1 includes only conventional weights. From Table 2, we can see that the Iris-Study-1 network learns the linearly separable class (Class-1) very well, and also that the network struggles a bit to clearly distinguish the two classes that are not linearly separable (Class-2, and Class-3). This outcome is typical for a Iris Plant dataset analysis. Actually, for our purposes, we do not need to go deeper into the individual numbers. What we need to do is to find out how Iris-Study-2 to Iris-Study-4 (the networks with a mixture of conventional, static, and dynamic weights) compare to Iris-Study-1. From this point of view, Table 2 informs us that the proposed models stand up very well to the conventional learning approach. Of course, this is a very important and satisfying outcome for our work.

4 Towards Nature-Inspired Modularization

The aim in this section is to describe how our models may lead to a feature we can observe in many natural environments, namely, the partitioning of a larger system into multiple, interrelated components. For illustration purposes, we employ a simplified view of a biological eye as a reference system Figure 10 (a). For instance, although, anatomically, the retina is a part of the brain, it is meaningful to make a separation between the eye and the brain as shown in Figure 10 (a).

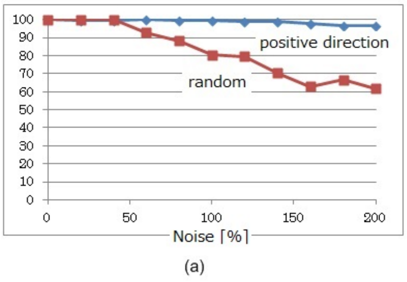
In reality, the eye receives light stimuli, preprocesses them, and propagates them further to the brain where higher-level processing takes place. The units or modules are connected and interact to execute a particular function and to give rise to the global behavior of the system. In a wider context, Figure 10 (a) also addresses the fundamental design principles of *high coherence* and *low coupling.* High coherence means that a unit has a high degree of identity (i.e., the elements within a unit should relate strongly and closely to this unit). Low coupling, on the other hand, means that a unit is not connected to too many other units. Loosely speaking, in Figure 10 (a), the eye and the brain have their own specific jobs to do. The two units communicate (ideally) minimally, which may require or lead to a particular type of communication infrastructure [Wedeen et al., 2012].

With the help of Figure 10 (b), we are now ready to describe how our models may contribute to the current discussion. Because, although the models in our earlier studies provided interesting results, we did not really address the question: "What is the real benefit of using a mixture of conventional, static, and dynamic weights in a network?" So, let us assume that Unit-1 and Unit-2 in Figure 10 (b) together form one system (e.g., a larger neural network, similar to the eye-brain system in Figure 10 (a)). Consider Unit-1 as a higher-level network component, and Unit-2 as one responsible for somewhat lower-level tasks. These assumptions are meaningful, because, for instance, although it is possible to assume that the learning of the eye-brain system is done entirely by the brain, there is evidence that the eye is able to accomplish some learning tasks on its own (e.g., [Maturana et al., 1959], or [Brighton and Selina, 2003, pp.135-137]). These learning tasks, however, are at a lower level of complexity than the higher cognitive tasks involved in the "seeing" process. Our motivation now is to use our models for the modeling of these different types of learning. So, let us hypothesize a Unit-2 (eye unit) and a Unit-1 (brain unit) each containing all three types of neurons (conventional, static, and dynamic). Let us further hypothesize a required change or adaptation (e.g., changing the focus of the eye). In this case, it should predominantly be the task of Unit-2 (eye) to respond to these changes. Unit-1 (brain) should receive minimal feedback about Unit-2's accomplishment (success/failure) related to this task. But then, if the weights in Unit-2 change in the adaptation process, we want to know the following: "If Unit-2 carries out some learning tasks, which may involve a change of its weights (dynamic and conventional), how does this affect the performance of the overall system consisting of all Unit-1 and Unit-2 weights?"

4.1 Study-5: Robust Dynamic Adaptation

From a higher level view, this section is interested in network sensitivity, which may be seen as an instance of robustness. We are interested in the possibility of network adaptation, which requires a change in the network, without disturbing the whole network at large. Study-5, therefore, specifically explores how a change in the dynamic weights in a network affects overall network performance. The rationale is as follows: if there is a certain range in which the dynamic weights, which are only a subset of all network weights, can be altered without affecting the overall network performance too much, then it may be possible to use these weight changes for other learning tasks. The procedures in Study-5 are as follows. The basic network topology in a Study-5 run is that of a Study-2 network. All connections in the network include a dynamic as well as a static weight component. The training process updates dynamic weights only (static weights remain unchanged). We evaluate the network on the training set and a test set, and typically observe that the network usually achieves about 100% accuracy on both sets. Next, we proceed as follows. We keep the training set and the test set. We also keep the static weights as they are, but we induce some noise into every single dynamic weight in the network. We then evaluate the network again by using the (same) training set and the (same) test set on the network with the noisy dynamic weights. We induce noise in two different ways. One approach induces noise in a random fashion. For example, if a dynamic weight has the value x, then this procedure generates the new value $x\prime = x \pm \alpha x$, where α is a real number from $[0, 2]$. The sign $\pm$ is determined randomly (with equal probability for $+$ and $-$). For instance, if $x = 0.5$ initially, and $\alpha = 0.2$, then the new noisy weight value may be either $x = 0.51$ or $x = 0.49$. The second approach also alters all dynamic weights. The second approach, however, either increases all dynamic weights or it decreases all of them. For example, if $x = 0.5$ and $\alpha = 0.2$, then the updated dynamic weight value is $x\prime = 0.51$ in the first case, and $x\prime = 0.49$ in the second case. In contrast to the first approach where the dynamic weights can change in both directions (positive and negative), the weights can change in only one direction in the second approach (all weights are either increased or decreased, but it cannot happen that some weights increase while others decrease). We refer to the second approach as coordinated weight change.

Figure 11 (a) illustrates the outcome for two Study-5 tests. Figure 11 (a) shows that if the dynamic weights change randomly (positive and negative direction), then the network performance degrades at a noise level of approximately 60%. By contrast, when the dynamic weights change in a coordinated way (here, in the positive direction), then the network is remarkably resilient to these changes. We completed various other Study-5 tests (data not shown), and in all tests, we observed characteristic curves similar to the curves depicted in Figure 11 (a). Figure 11 (b) illustrates one example in which we think we can use these findings. Let us assume that the neuron n_x in Figure 11 (b) is located in Unit-2. Further, assume that it is not part of the network that is responsible for the two-class classification task. Instead, assume that the neuron and its output



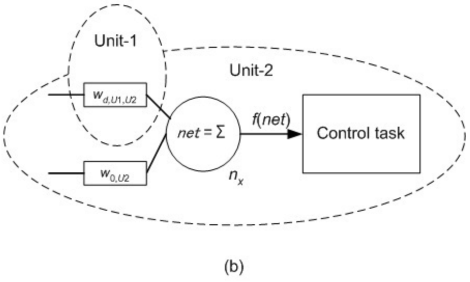


Fig. 11. (a) Network robustness. (b) Weight utilization accross domains.

$f(net)$ are involved in some control task (e.g., for adapting a corrective lens in an eye). For simplicity, neuron n_x has two weighted inputs only. Weight $w_{d,U1,U2}$ is a dynamic weight located in Unit-2, but connected to Unit-1. On the other hand, weight $w_{0,U2}$, which may be a conventional weight (or a static weight component), solely exists in Unit-2 to assist the control task. We can think about the involvement of both weights in the control task as follows: weight $w_{d,U1,U2}$, which is shared between two units, has some range in which it can be altered without affecting the two-class classification task too much, whereas weight $w_{0,U2}$ can change more freely. This example indicates an intuitive way in which neurons can cooperate across domains, as they often do in nature. The example also captures the rationale of our work, namely to propose alternatives to conventional ANN models that are simple, perform well, are robust, and nature-inspired.

5 Summary

Our work was motivated by the simplicity and beauty we find in abundance in many natural networks (not only neural networks). Defining features of such networks are often a naturally occurring compartmentalization, as well as a degree of redundancy in terms of the elementary components defining such systems (e.g., neurons and their interconnections). Our work seeks to mimic and utilize these features in various ways. By using a mixture of conventional weights, static weights, and dynamic weights, we could devise simple ANNs that mimic such modular (redundant) architectures. We developed and tested our models in simple environments and then transported them to more complex, more realistic, and well-known problems. We found that our models performed well in any of these environments. Actually, we did not observe any significant differences between our models and the conventional ANN architectures we investigated. Our work also suggested how our models may work across domains, for instance by supporting several individual units in a larger network. We also found that our models are remarkably robust to noise. Of course, we do not claim that the proposed dynamic weights and static weights have their equivalent in natural systems. We would like to mention, however, that in natural systems there are indeed different types of neurons [O'Shea, 2013]. Interneurons, for instance, participate in the local aspects of a neural circuit, while projection neurons extend to more distant targets [Augustine et al., 2004, pp.1-28]. We may also look at static weights as a type of weight that in a way stabilizes a network, which is a feature we have observed in our earlier work on robustness [Schuster, 2008]. In summary, using both static weights and dynamic weights might be an interesting approach to design more robust networks that, at the same time, can be adapted across domains. Our experimental results are very promising and indicate several new avenues our work could take.

References

[Alnajjar et al., 2013] Alnajjar, F., Yamashita, Y., Tani, J.: The hierarchical and functional connectivity of higher-order cognitive mechanisms: neurorobotic model to investigate the stability and flexibility of working memory. Frontiers in Neurorobotics (2013), doi:10.3389/fnbot.2013.00002

[Augustine et al., 2004] Augustine, G., Fitzpatrick, D., Purves, D.: Neuroscience, 3rd edn. Sinauer Associates Inc., U.S. (2004)

[Bishop, 1996] Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (1996)

[Blake and Merz, 1998] Blake, C., Merz, C.: Uci repository of machine learning databases, University of California, Irvine, Dept. of information and computer sciences (1998), `http://www.ics.uci.edu/`

[Brighton and Selina, 2003] Brighton, H., Selina, H.: Introducing Artificial Intelligence. Icon Books Ltd. (2003)

[Maturana et al., 1959] Maturana, H., Mcculloch, W., Pitts, W.: What the frog's eye tells the frog's brain. Proceedings of the Institute of Radio Engineers 47(11), 1940–1951 (1959)

[Mehrotra et al., 1997] Mehrotra, K., Monan, C., Ranka, S.: Elements of Artificial Neural Networks. The MIT Press (1997)

[O'Shea, 2013] O'Shea, M.: Instant expert: The human brain. New Scientist 218(2911), 28 (2013); Instant Expert Series (8 pages)

[Schuemann et al., 2013] Schuemann, A., Klawiter, A., Bonhoeffer, T., Wierenga, C.: Structural plasticity of gabaergic axons is regulated by network activity and gabaa receptor activation. Frontiers in Neural Circuits (2013), doi:10.3389/fncir.2013.00113

[Schuster, 2003a] Schuster, A.: Application of artificial neural networks to multiple tasks. In: Proceedings of 2nd International Conference on Decision Support for Telecommunications and Information Society, DSTIS 2003, Warsaw, Poland (2003a)

[Schuster, 2003b] Schuster, A.: A world according to artificial neural networks. Journal of Telecommunications and Information Technology 3, 102–107 (2003b)

[Schuster, 2008] Schuster, A.: Robustness in nature as a design principle for artificial intelligence. In: Schuster, A. (ed.) Robust Intelligent Systems, pp. 165–188. Springer (2008)

[Spalding et al., 2013] Spalding, K., Bergmann, O., Alkass, K., Bernard, S., Salehpour, M., Huttner, H., Boström, E., Westerlund, I., Vial, C., Buchholz, B., Possnert, G., Mash, D., Druid, H., Frisén, J.: Dynamics of hippocampal neurogenesis in adult humans. Cell 153(6), 1219–1227 (2013)

[Wedeen et al., 2012] Wedeen, J., Rosene, D., Wang, R., Dai, G., Mortazavi, F., Hagmann, P., Kaas, J., Tseng, W.: The geometric structure of the brain fiber pathways. Science 335(6076), 1628–1634 (2012)

[Werbos, 1974] Werbos, P.: Beyond regression: new tools for prediction and analysis in the behavioral sciences. PhD Thesis, Harvard University, Cambridge, MA, USA (1974)

[Yao, 1993] Yao, X.: Evolutionary artificial neural network. International Journal of Neural Systems 4(3), 203–220 (1993)