UNIVERSITY OF GRONINGEN
HUMAN-MACHINE COMMUNICATION
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Fine-Tuning Input Selection while Learning Associative Memories in a Spiking Neural Network

Master's Thesis

by

Daan Sijbring

Primary Supervisor:	Dr. J. P. Borst
Secondary Supervisor	Prof. dr. N. A. Taatgen

Abstract

This research aims at designing a spiking neural network capable of learning associative memories. An important effect in associative recognition tasks is the fan effect – increased error rates and reaction times and decreased neural activity for memory items which are associated with other memory items. Our goal is to investigate whether the fan effect occurs in the results of the network, and how influencing the network's encoder selectivity through different Voja rule adaptations best approaches some expected manifestations of the fan effect. Learning the associative memory in a spiking neural network occurs through a combination of supervised Prescribed Error Sensitivity (PES) learning in connection weights and unsupervised Vector Oja (Voja) learning focusing on input selectivity. Measurements will involve the network's accuracy of representation after training, and its predicted neural activity for certain inputs based on the trained input selectivity that is caused by the Voja rule (or adaptation thereof). Results will show that all examined Voja adaptations more closely exhibit the fan effect than the original Voja rule, and shows how further subtleties in the individual rules influence memory performance. The research ultimately presents ideas on how to further any input selectivity rules in order to better approach the fan effect, and reflects on what the newly studied Voja adaptations can contribute to associative memory learning in different applications.

Contents

List of symbols

Chapter 1
Introduction

Part of human intelligence is due to our ability to make connections between pieces of information we are presented with. This is called *associative memory*, and it presents a compelling topic of research both in cognitive psychology and cognitive modeling. Certainly, if associative memory plays an important role in our intelligent capabilities, it would be beneficial to study the cognitive dynamics of how we form these associations, and to see if they can be replicated. From a neuroscientific perspective, associating memory items is a well-studied mechanism of repetitively forming and strengthening connections between (groups of) neurons associated with memory activations over time. In addition, it is a strong advocate for the concept of neural plasticity - the brain undergoing physical changes from a low-level perspective of individual neurons to widespread changes in the brain's structure associated with reorganization of the cortex.

The psychological approach concerns itself with which and how circumstances influence the forming and reproducing of these associations, and has, more specifically, yielded interesting observations related to the influence of additional information that is in some way related to a particular memory item. Anderson (1974) has found that when more information is associated with a memory item, the retrieval action of that memory item can be more prone to errors and take longer due to interference with the associated interference (for an extended review: Anderson and Reder 1999). This effect is known as the *fan effect*, where the fan indicates the number of items in memory the current item is associated with. The effect is most commonly demonstrated in the recognition of facts presented together (e.g. subjects in a sentence or individual words combined into pairs). The fan effect is found in a variety of different tasks, such as face recognition (Anderson and Paulson 1978) and alpha-arithmetic fact learning (Zbrodoff 1995), and its effects have aided in understanding results from studies that, for example, investigate working memory (Cantor and Engle 1993) or the effects of aging on memory (Radvansky et al. 1996). Section 1.1 will discuss the fan effect in-depth, along with some results from varying research to illustrate the influence of fan on accuracy, reaction time (RT) and neural activity when recalling previously studied pairs of memory items.

While the fan has been extensively studied in psychological task settings where results are studied and exhibit the known effects, it is yet unclear how associations between memory items are handled and formed on a neurological level. This study focuses on modeling the learning of associative memories in biologically-plausible structures that will aim to give an indication as to how associated information is treated, and how it influences learning. In addition, the role of input selectivity in this process will be investigated, and the effects of various learning rules concerned with input selection will be explored.

1.1 Fan effect

With the focus of this research concentrating on the emergent behaviour of memory when distinct memory items are associated with each other, the goal behaviour has to be identified first. Naturally, we look to human performance and behaviour when presented with associated information, which, as is mentioned briefly before, results in the well-studied fan effect. Anderson's pioneering work (1974) into this phenomenon consisted of recognition memory tests in which various facts about certain people in certain locations had to be studied (e.g. "the sailor in the park" or "the businessman in the theatre"). The people and locations in the studied facts could occur in one fact or more than one (e.g. "the sailor in the park" and "the sailor in the theatre"). Subjects were then asked to recognize facts about a person and location that either they had studied (target probe) or were novel combinations of people and locations presented earlier in separate facts (foil probes). The findings were that reaction times and error rates increase as the fan increases. Foil probes also added difficulty, increasing the reaction times and error rates as well. The research concludes that the more is known about a memory item, the longer it will take to retrieve that item from memory, and the more errors will be made.

In line with earlier research into the fan effect utilizing recognition tasks (Anderson 1974, Anderson and Reder 1999, Wickelgren and Corbett 1977), Borst et al. conducted research investigating the processing stages of associative recognition memory by letting subjects study word pairs (2016). Figure 1.1 shows the behavioral results for this research, showing the fan effect of increased reaction times and error rates for Fan 2 word pairs in comparison with Fan 1 word pairs. During testing, magnetoencephalography (MEG) measurements were taken from the participants. A multi-variate machine-learning classifier was used in order to analyze the MEG data and to identify various processing stages (Chan et al. 2010, Pereira et al. 2009). These processing stages included an encoding stage, an associative retrieval stage, a decision stage and a response stage - in line with Borst's earlier-developed and researched ACT-R model and EEG model (Borst and Anderson 2015). Fan was ex-

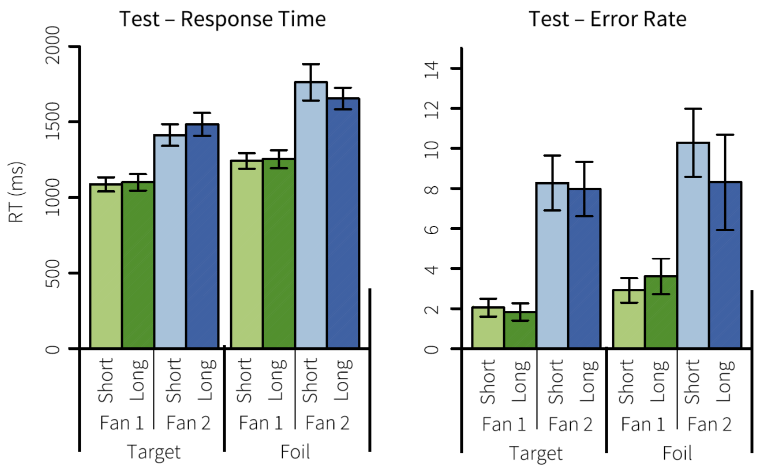


Figure 1.1: Reaction time (RT) in milliseconds an error rates (%) for Fan 1 and Fan 2 word pairs from Borst et al. (2016). Word pairs are either constructed of two 4 or 5 letter words (short) or of two 7 or 8 letter words (long). Target probes are studied word pairs and foil probes are novel combinations of single words learned previously in other word pairs, where the position of the word in the word pair was kept the same across the experiment. *Adapted with permission from Borst et al. (2016).*

pected to influence the retrieval stage, and Figure 1.2 shows the difference in estimated neural activity during this identified retrieval stage. Analyzed MEG data indicated that neural activity was increased for the recollection of Fan 1 word pairs in comparison with Fan 2 word pairs.

In conclusion, the three measurable characteristics of the fan effect this research can focus on are increased response time, increased error rates and decreased neural activity for word pairs with a higher fan.

1.2 Associative memory & neural networks

Interest in employing brain-inspired structures to learn associative memories has been first raised by results from Hopfield networks that are trained on associated information (Hopfield 1982). The focus of this research was the emergent capabilities of large collections of computational units (neurons). The dynamics of the individual neurons were inspired by electrochemical properties of actual neurons and their connections (synapses). Resulting capabilities involved the convergence

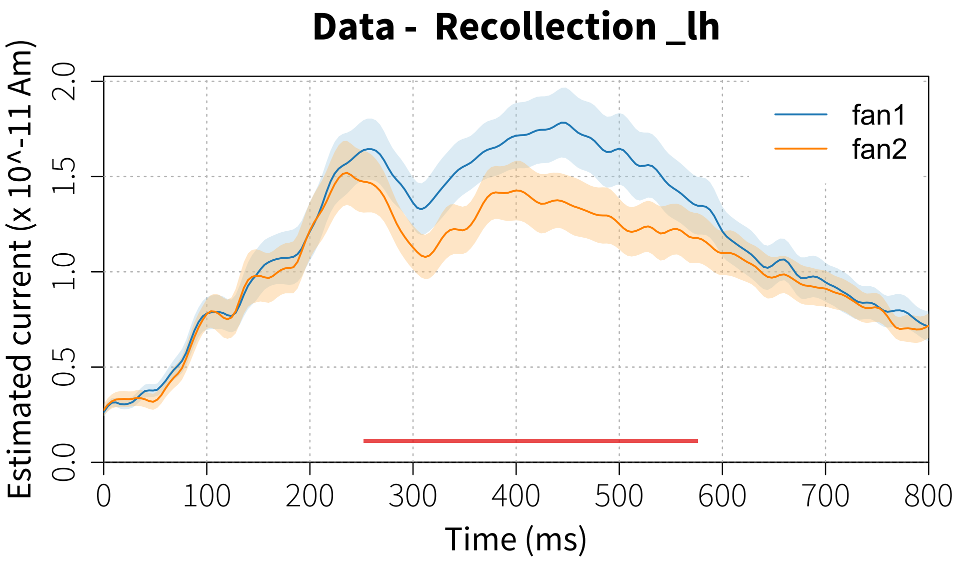


Figure 1.2: Estimated current for Fan 1 and Fan 2 word pairs from MEG data. This data has been analyzed using a multi-variate machine-learning classifier able to identify distinct processing stages. The horizontal red line above the x-axis indicates the manifestation of the retrieval stage, the processing stage that is likely to be associated with the fan effect. *Unpublished data from Borst et al. (2016), printed with permission.*

of the network to the individual items presented to the network, but also demonstrated the more intricate behavior of pattern completion. That is, if a fact is known to the network (e.g. about a shopping list: ”sugar, flour and milk”), a subpart of that fact (e.g. ”milk”) is sufficient for the memory system to retrieve the entire fact. Additional capabilities include family recognition, categorization and error correction (Hopfield 1982). The individual neurons in a Hopfield network were only capable of taking on binary states, which determined if information would be passed along. Nowadays, artificial neural networks employ more complex workings in their constituting neurons, but the early Hopfield networks did prove that their design was a very capable candidate in modeling associative memory. In addition, Hopfield’s work started interest in even more low-level neural modeling of associative memory, revealing more and more about specific memory capabilities of single neurons or small groups of neurons (Quian 2012, Gorban et al. 2019).

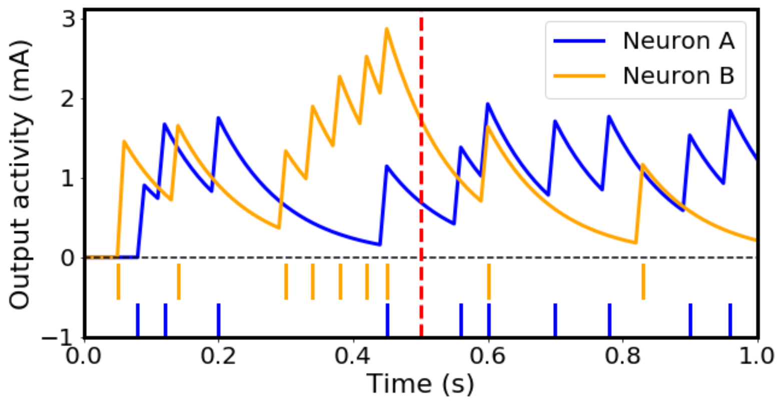


Figure 1.3: Filtered activity from arbitrary spike trains from two neurons. Both spike trains contain 10 spikes, of which 9 are randomly distributed between 0 and 1 second and 1 is set at t = 0.45. The relative spiking trains are indicated by upright lines below the *x* axis. The dotted red line at t = 0.5 shows an example of difference in activity of the two neurons 50 ms after their last spike. Despite both neurons last spiking at t = 0.45, there is a large difference in activity at t = 0.5 due to the spiking history of each neuron.

In order to more closely mimic biological dynamics of the brain while modeling learning, spiking neural networks can be used. These are different from artificial neural networks in the way that they receive and pass along information - instead of communicating a distinct value, the neurons mimic an biological neuron and communicate through spike trains. Various well-studied neural models can be chosen to model the accumulation of information in a neuron, and how it passes this down along its connections. Figure 1.3 illustrates how the spike trains generated by the neurons in spiking neural networks can be convolved into values indicating neural activity that, by means of the convolution, take the history of the spiking behaviour of each neuron into account. This causes communication in the spiking neural network to add another dimension to information that is being communicated, namely time.

The motivation for adhering to biological plausibility in designing and using a spiking neural network to model learning is two-fold (Stewart 2012). First, when one wants to model a psychological task in order to understand how the human brain works, the behaviour produced by your simulation should correspond to observed human performance as well as *the way* it reached that behaviour. For example, when a spiking neural network simulation of a production system was constrained with the relevant biological properties of the neurons in the involved brain regions, the model innately yielded the well-studied 50 millisecond cognitive cycle (Stewart et al. 2010). Secondly, adhering to biologically plausible aspects of neurons can explore novel types of algorithms. With the neurons in the simulation *approximating* a function, instead of the designer explicitly specifying it, an exact replication of the intended algorithm is often not achieved. This had led certain families of functions to be excluded from explaining certain psychological phenomena (due to timing constraints or the number of neurons), and has also shed light on other families of functions more suitable for the model (Stewart and Eliasmith 2009).

A popular conceptualization of spiking neural network dynamics suitable for modeling memory can be found in the Neural Engineering Framework (NEF; Eliasmith and Anderson 2002). This is a method for constructing neural simulations that incorporates information representation, processing and the corresponding neural dynamics. This is the framework that will be used to design the spiking neural network studied in this research as well, and a more detailed description of its aspects will be presented in Section 2.2.

When adhering to the design constraint of biological plausibility in the structure of cognitive models employing spiking neural networks, neural dynamics involved in learning are innately constrained as well. These constraints apply to what information can be used by the changing connections in the network. Specifically, the information has to be locally available to individual neurons and/or network layers. The NEF typically approaches this constraint by the combination of two local learning rules; the supervised Prescribed Error Sensitivity (PES; MacNeil and Eliasmith 2011) learning rule and the unsupervised Vector Oja (Voja) learning rule - both of which are discussed extensively in Section 2.4. The changes PES imposes on a network strengthen associations between neurons, while the Voja updates concern input selectivity of the neurons. Previous research has shown the success of combining the supervised learning rule with the unsupervised selection of encoders in learning associations in NEF-based simulations (Knight et al. 2016, Aubin et al. 2016, Aubin 2018).

However, the memory items learned in these simulations are constructed with optimal separability - an aspect that enables Voja to distinguish between neurons throughout the input range more easily. Section 2.1 will go into how information can be optimally separated, but in short it holds that the information items do not overlap in representation (often using numeric vectors). This ensures the most chance to discern between distinct inputs, but the cognitive task in question here, associative recognition, relies on the associative strength between certain memory items. Specifically, this associative recognition task that is used in this research is word pair learning. Sets of two words are presented at the same time (e.g. ”WHEEL & FILE”), and its words can either occur in another word pair (e.g. "WHEEL & RACK") or only occur once. It is impossible here to hold optimal separability between all word pairs when associations (i.e. presentation in the same word pair, overlap between word pairs due to shared words) between the memory items bind them uniquely - of course, associations will not be the same for all items.

The shared information violates the optimal separability between memory items and, as will be shown in this research, troubles input selectivity in the go-to combination of PES and Voja in spiking neural networks. That is why, in this thesis, several novel adaptations of the Voja learning rule will be discussed and tested on word pair learning task where the word pairs will either have no associations (Fan 1) or have one association (Fan 2) with another word pair. Several aspects of the fan effect will be isolated, and learning rules causing input selectivity will be tested with regard to how they model effects observed in the fan effect in humans.

1.3 Thesis outline

This thesis will employ the theory from the Neural Engineering Framework in order to design a spiking neural network capable of learning word pairs, aiming at replicating the fan effect. Next, encoder selection learning rules will be investigated, adapted and tested for their ability to more closely approach the behaviours associated with the fan effect. These effects include lower reaction times, lower accuracy and lower neural activity as the fan increases.

The structure of this thesis will be as follows; first, the methods section will discuss the NEF in depth, along with Semantic Pointers that are employed to represent memory items in relations to each other. Also, biologically plausible learning rules will be explained. Next, results from the simulation will be presented and compared to the effects found in human performance on associative memory tests. Lastly, an in-depth discussion regarding the implications of this research will be presented, along with various ideas on how to further this research.

Chapter 2

Methods

When constructing any neural network, certain design considerations have to be taken into account and decided upon beforehand. These, naturally concern the structure of the network but also decisions regarding the representation of information, transforming this information into activity and deducing meaning from the network's output. For this research, the theory and structure for designing the SNN will be the Neural Engineering Framework (NEF). This theory was briefly introduced in Chapter 1), and will be described in detail in Section 2.2. Additionally, the representation and handling of information will be based on Chris Eliasmith's Semantic Pointer Architecture. This is an architecture that offers dynamics for a theory of cognition while taking biological constraints and plausibility into account (Eliasmith 2013).

The creators of the NEF and SPA have previously combined these two into a read-to-use toolbox called Nengo, a Python package that allows the user to build, test and deploy spiking neural networks (Bekolay et al. 2014). This is a popular option for simulating a biologically plausible version of a neural network to study learning, for example. However, in this research, a custom implementation of a spiking neural network is designed, also adhering to the principles of the NEF and while handling information in line with the SPA. The motivation for using a custom solution here is transparency in designing the network to our needs, and later on lets us integrate and experiment with novel learning rules more easily. Yet, some tools from the Nengo toolkit will be used in order to avoid solving some computationally heavy problems, such as optimal separability between memory items and linear optimization of decoders - vectors used to translate neural activity back to information (elaborated upon in Section 2.1). The rest of this chapter will assess the design of the spiking neural network in depth, as well as explain the dynamics and motivation behind learning rules that are used in this simulation.

2.1 Semantic Pointer Architecture

However, before the dynamics and design of our spiking neural network can be discussed, we first need to look at how information will be represented and handled. Our final simulation will be tasked with learning word pairs, but in order for our network to understand the input, we cannot simply use words. Instead, we need to transform them into some numerical form. Additionally, the associative information of two connotated words has to be reliably represented in the information we feed the network as well. The Semantic Pointer Architecture (SPA; Eliasmith 2013) offers a solution for both our problems, and is designed to be used in combination with the NEF. The SPA offers a wide range of functions applicable for cognitive modelling, but in this research we are merely interested in the generation, manipulation and use of Semantic Pointers (SPs). As Eliasmith (2013) states in his semantic pointer hypothesis, they enable for higher-level cognitive functions in biological systems, and are characterized by: "...neural representations that carry partial semantic content and are composable into the representational structures necessary to support complex cognition" (p. 78). SPs can vary in implementation and form, but we will focus on multidimensional vector representations of words.

2.1.1 Vector operations

The SPA employs various operations in order to handle the vector representations of information; *superposition*, *binding* and *unbinding* (Plate 1995). These processes allow for the combining of separate concepts, and opens up possibilities for the NEF to relate information between vectors because they are mathematically similar. With superposition, a new numerical vector can be formed by the simple addition of two previously generated concept vectors, creating a new concept:

$$\overrightarrow{u} = \overrightarrow{v} + \overrightarrow{w} \tag{2.1}$$

This concept is simply the sum of the two components, and thus will relate back to its components equally. Binding two vectors involves a mathematical operation called *circular convolution* ($\circledast$). This operation combines multiplications of each element in both 'original' vectors into a resulting new vector, like so:

$$\overrightarrow{u} = \overrightarrow{v} \circledast \overrightarrow{w} \tag{2.2}$$

where

$$u_i = \sum_{j=1}^{D} v_j w_{i-j} \mod D \tag{2.3}$$

and D is the dimensionality of the vector. The resulting vector is dissimilar to both component vectors, but it is possible to reverse the process that created the convolved vector. This is the last operation (unbinding), and can be achieved by convolving the resulting vector with the approximate inverse of one of the components:

$$\overrightarrow{v} = \overrightarrow{u} \circledast \overrightarrow{w}^{+} \tag{2.4}$$

where the inverse of a vector is approximated by the reordering of the vector components:

$$\overrightarrow{w}^{+} = (w_1, w_D, w_{D-1}, ..., w_2)^T \tag{2.5}$$

Additionally, the operation of circular convolution is, just like many mathematical operations, associative, commutative and distributive:

Associative Components of the operation can be swapped without changing the result: $\overrightarrow{v} \circledast \overrightarrow{w} = \overrightarrow{w} \circledast \overrightarrow{v}$

Commutative Components of the operation can be grouped in any combination without changing the result: $(\overrightarrow{v} \circledast \overrightarrow{w}) + \overrightarrow{u} = \overrightarrow{v} \circledast (\overrightarrow{w} \circledast \overrightarrow{u})$

Distributive The result of a convolution between a vector and a superposition of two vectors is the same as the superposition of two individual convolutions with each component: $\overrightarrow{u} \circledast (\overrightarrow{v} + \overrightarrow{w}) = \overrightarrow{u} \circledast \overrightarrow{v} + \overrightarrow{u} \circledast \overrightarrow{w}$

Applied to an example of learning about objects and colors as concepts, we can create a scene of two objects that are convolved with a color: $\overrightarrow{v} = BAG \circledast GREEN + HOUSE \circledast YELLOW$. Next, we can extract information about the house, for example, by unbinding that vector:

$$\begin{gathered}\overrightarrow{v} \circledast HOUSE^{+} = BAG \circledast GREEN \circledast HOUSE^{+} + ...\\ HOUSE \circledast YELLOW \circledast HOUSE^{+} \approx YELLOW\end{gathered}$$

These operations allow for concepts that are bound by commonality, which can be discovered and extracted. This commonality results in some abstract form of meaning that is attached to a concept of, for example, a house. In our implementation, we have limited the SPs we create to simply use addition of two words, but the operations of binding and unbinding could be used as well in order to relate meaning to the position of individual words in word pairs.

2.1.2 Vocabularies

The Python implementation of SPA offers a structure especially designed to create, store and manipulate these vectors - a vocabulary. When generating new SPs, the vocabulary automatically takes similarity between vectors into account. New vectors that are generated for new SPs are aimed to be as distinct as possible from the vectors of already existing SPs. It essentially looks for free space in the hypersphere formed by the dimensions of the vocabulary, and generates new vectors that fit into the unpopulated areas. This means that the amount of SPs that can be stored without overlapping too much depends on the dimensionality of the vocabulary. As a similarity metric, the dot products between all SPs in the vocabulary are calculated. When vectors are normalized to 1, which is the default for these vocabularies, the dot product between vectors indicates the angle between them.

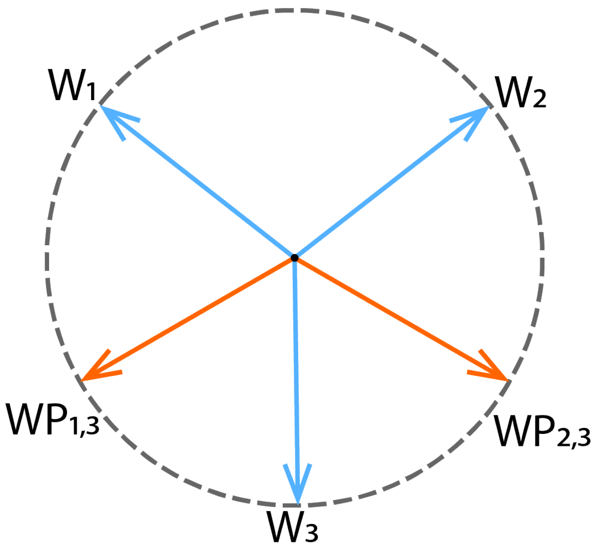


Figure 2.1: Illustration of distribution of three word (W_1, W_2, W_3) and two word pair ($W_{1,3}$, $W_{2,3}$) vectors in two-dimensional space. While the aim is to optimally separate word vectors generated by the vocabulary in the available hyperspace, this is not exactly achieved. Thus, the angles θ_{W_1,W_2}, θ_{W_1,W_3} and θ_{W_2,W_3} seem to be at an equidistant 120° but this optimal angle will just be approximated. Note that, because the *calculated* word pair vector is a normalized addition of two individual word vectors, they are exactly between the constituting word vectors ($\theta_{W_1,WP_{1,3}} = \theta_{WP_{1,3},W_3}$ and $\theta_{W_3,WP_{2,3}} = \theta_{WP_{2,3},W_2}$).

In our research, the vocabulary is initialized with all unique words appearing in the word pairs that are to be learned by the network. The similarity constraint of the SPA vocabulary tries to ensure a minimum angle distance between vectors that populate the vocabulary. Next, word pairs are constructed by adding the two individual word vectors making up the pair, and normalizing these resulting word pair vectors. They are also added to the vocabulary but will not be bound by the similarity constraint, since they are supposed to be a result from the individual words. Accordingly, the addition of two individual word vectors causes the resulting normalised word pair vector to have an equal angle from each constituting word vector. See Figure 2.1 for a simplified two-dimensional visualization of the vocabulary design.

2.2 Neural Engineering Framework

As is mentioned before, the Neural Engineering Framework (NEF; Eliasmith and Anderson 2002) formalizes spiking neural networks that represent and process information in a manner that is biologically plausible. The main aspects of the NEF that are important for the scope of this research are the encoding and decoding concepts. They are concerned with how information is received into the neuron and extracted from that neuron, respectively. These concepts are discussed in the following subsections.

2.2.1 Encoding

What is meant with *encoding* is the representation of some numeric vector ($x(t)$) in a neural population (called ensemble). This information is represented in the ensemble by means of the spiking activity produced by the neuron, through which a distinction is made between the representation (activity) and the represented value (input vector). Each neuron i is assigned an encoding vector (e_i) which states the *preferred direction* of that neuron, an intercept (x_i^{int}) which determines the input value for which the neuron starts firing, a *gain* parameter (α_i) stating the responsiveness of the neuron and a *bias* parameter (β_i) which indicates background current.

The gain and bias parameters for a neuron describe how a neuron fires for a certain input vector, once it's responding to that vector. The encoding vector, along with the neuron's intercept, determine the input vector for which that neuron will fire most strongly. This design is in line with the common neuroscientific concept of neurons adhering to certain *tuning curves* - a neuron responds more strongly to certain 'preferred' conditions or stimuli. The combination of all neurons in an ensemble that have preferred directions (encoders) that are uniformly distributed cause

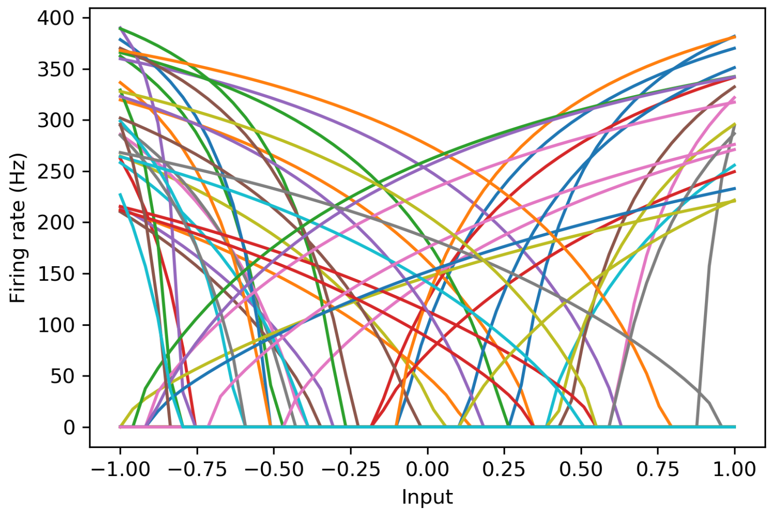


Figure 2.2: Example of tuning curves of an ensemble. The ensemble contains 50 neurons with uniformly distributed encoders, indicating a preferred direction that determines the response of that neuron for a range of inputs. The intercept here is also randomly distributed between -1 and 1, and states the input value for which the neuron starts firing. Additionally, the form of the tuning curve for a neuron is determined by the bias and gain parameters.

the ensemble to respond to a pre-defined range of inputs using different subsets of neurons that are geared towards that input. Figure 2.2 illustrates how the distribution of encoders and intercepts causes neurons in an ensemble to respond with different firing rates to various inputs. For example, for high positive input values, a certain subset of neurons respond very strongly. Alternatively, for a negative value towards -1, another subset of neurons is geared to respond strongly. However, the whole range of inputs (in this case, -1 to 1) is covered by some subset of the neurons in the ensemble. Figure 2.2 also illustrates the effect of the encoders (preferred directions) and intercepts for all individual neurons - the neuron starts firing when the input has reached the intercept value of that neuron, and will either fire for higher or lower values compared to its intercept, depending on the neuron's encoder. This already highlights the importance of the design choice of what distribution should be selected for the intercepts in an ensemble, as they play an important role in the *input selectivity* of that ensemble - which is the main focal point of this research. Section 2.2.2 will later investigate this influence (especially in the context of memory effects) on how a spiking neural network can learn and exhibit certain behaviours.

In the NEF, the encoder vector applies a linear function to the input to achieve a certain input current for the neuron according to:

$$J_i(x(t)) = \alpha_i e_i \cdot x(t) + \beta_i \tag{2.6}$$

The resulting input current (J_i) is passed through the neural model dictating the neuron's non-linearity, in order to achieve a spike train (s_i):

$$s_i(x(t)) = G_i[J_i(x(t))] \tag{2.7}$$

where function $G_i[\cdot]$ states the neuron's nonlinearity. This can be any neural model, ranging from sigmoidal neurons to Izhikevich neurons to models that are more biologically inspired. The standard neuron model, and the one we're using in this research, is the leaky integrate-and-fire (LIF) neuron model. In this model, the neuron builds up potential gained from excitations due to input it corresponds to, whilst also continuously 'leaking' a constant current of its potential. Once this membrane potential reaches a set threshold, the neuron fires. Finally, a low-pass filter ($h(t)$) modeling the exponential decay of the neuron's post-synaptic current is applied to the spiking pattern, resulting in the neuron's filtered activity (a_i).

2.2.2 Encoder intercepts

With the knowledge of what inputs we are presenting to the network (Section 2.1), we can make a design choice that ensures the network will respond to this input accordingly. That is, we can tune the *input selectivity* of the network by choosing the intercepts for the neurons individually, assign them a common predefined value, or give the ensemble of neurons a distribution of intercepts. Figure 2.2 shows how the intercept, together with the encoder, determines the input range that causes a neuron to fire. Now, for learning purposes explained in Section 2.4, it can be beneficial to limit the range of inputs a neuron can respond to. In the case of a two-dimensional example, we can initialize the encoders of each neuron in an ensemble to random values normalised to the unit circle (see Figure 2.3). The place of the encoder on the unit circle for a neuron determines the preferred 'direction' in which the neuron will respond. More importantly here, the intercept determines the *receptive field* for that neuron; the range of inputs around the neuron's encoder for which the neuron will fire. A neuron's firing rate increases towards the centre of this receptive field, and decreases when inputs are approaching the borders of its receptive field. These borders are bound by the intercept value. When the angle between encoder

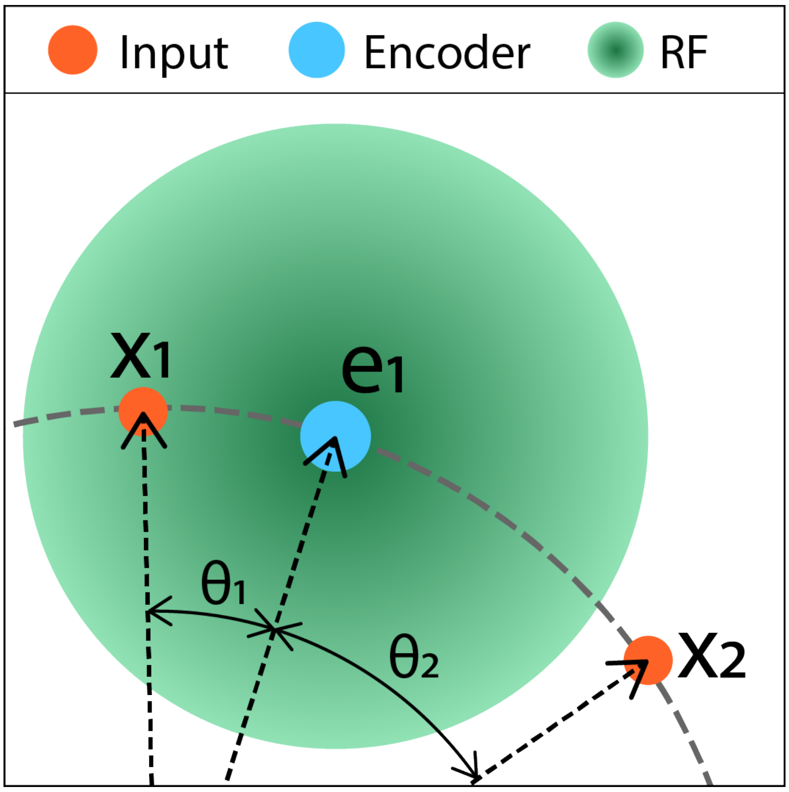


Figure 2.3: A section of the unit circle, with encoder and input positions denoted on the circle, $\theta_1 = e_1 \cdot x_1$ and $\theta_2 = e_1 \cdot x_2$. The radius of the receptive field (RF) is equal to the neuron's intercept x_1^{int}. This illustrates input selectivity, where angles between encoder and input positions determine the neuron's firing rate in line with Equation 2.6. The firing rate is maximal when the $e \cdot x = 1$ (i.e. identical position), decreases towards the edge of the receptive field, and is 0 when $e \cdot x > x_1^{int}$.

position and input vector exceeds this threshold, the neuron will not respond to the input. Adapting the intercept can put restrictions on the input range a neuron can respond to but, as Section 2.4 will show, a differentiation between neurons will be of importance when implementing the Vector Oja (Voja) learning rule.

2.2.3 Decoding

Decoding is reversing the filtered activity of neurons in an ensemble back to the value that is being represented by the ensemble. Essentially it is a translation from activity to a numeric value that can be related back to the vector that was fed to the ensemble. This is a process that is important for communicating information from one ensemble to the next, and, ultimately, for the observers of the network for receiving information from the network that it communicated. For each neuron *i*, there is a decoding vector d_i that maximally approximates the value being represented. When we know the input vector $x(t)$ for an ensemble, and the function we desire it to compute (f, which can simply be an identity function), finding its optimal decoding vector reduces to a least squares optimization problem:

$$d_i = \int \Big\| x - f(x) \Big\| dx \tag{2.8}$$

Often, the goal will be to compute multiple functions in series or employ different aspects of connected neurons in neural networks in order to approximate more complicated problems. Rather than focusing on decoding output, the filtered activity is passed from the initial ensemble to another one by using a weight matrix (w). This weight matrix is composed of the decoders of neurons (i) in the first ensemble and the encoders of neurons (j) in the receiving ensemble:

$$w_{ij} = d_i \cdot e_j \tag{2.9}$$

The fully-connected weight matrix described above allows for bypassing the decoding stage, but implicitly involves decoders and encoders in passing along activity. These weights towards the receiving ensemble are combined with the filtered activity from the first ensemble to calculate the input current to the receiving ensemble directly:

$$J_j(t) = \sum_i w_{ij} a_i(t) + \beta_j \tag{2.10}$$

It has to be noted here that, in this case, we have chosen to incorporate the gain parameter (α) of the neurons (j) in the incoming ensemble during the calculation of the encoder. Another possibility is to move this to the calculation of the weights, where the dot product of the incoming neuron's (*j*) encoder and gain parameter should be used.

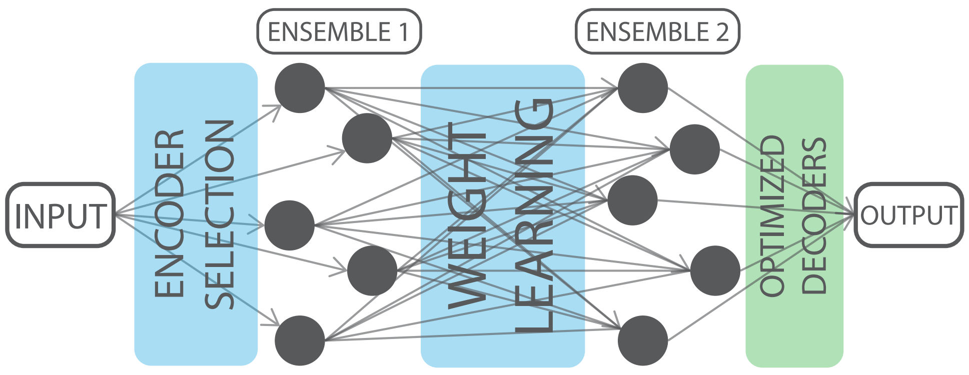


Figure 2.4: Layout of the network used in this research for learning. Encoder selection is applied to the encoders of the first ensemble in order to adapt to the inputs presented. These affect the excitation of the neurons in the first ensemble. These neurons connect to the second ensemble through a fully connected weight matrix, to which weight learning is applied. In order to inspect these both forms of learning without interference, linearly optimized decoders are used for the last ensemble. These decoders are used to translate neural activity in the second ensemble back to the numerical output.

2.3 Network design

Our network has a rather straightforward design - two ensembles, one receiving the input and the other representing the output. We can directly present the SPA-generated inputs to the first ensemble. This ensemble's role is to pass along the received input through its neurons, and (as is explained in Section 2.4) plays an important role in encoder selection - which neural paths are triggered by certain inputs. The first ensemble translates the input into its neural activity response and is linked to the second ensemble trough a fully connected weight matrix. This fully connected weight matrix represents a connection for every possible combination of neurons in the first and second ensemble. The values represent its connection strength, and the sign indicates if it is an inhibitory or excitatory connection. Next, the second (and last) ensemble's neural activity represents the output of the network. Its linearly optimized decoders translate this activity back to a multidimensional numerical vector that is (if learning occurred successfully) equivalent to the input vector. Figure 2.4 illustrates the network design, along with where online learning is implemented. The ensemble's initial encoding vectors were distributed randomly along the unit circle.

2.4 Learning

After designing the network, we need to focus on how the constructed network will be able to learn from the presented information. As mentioned in Chapter 1, learning in this research will be online and biologically plausible. That is, for supervised purposes, we can only use locally computed errors from the calculated responses within the network itself. Learning can be implemented in two fashions; locally on the ensemble, or in the connections between two ensembles. With ensemble learning, the learning rules can either apply to the ensemble's encoders or to the decoders. Here, the ensemble exhibits the learned function on its own. The information communicated between ensembles in the network has to decoded and encoded again in each connection in order to apply the learning rule. A more direct solution is to utilize weighted connections between ensembles to transfer activity without any decoded value. Section 2.2.3 explains how these weights incorporate the outgoing ensemble's encoding vector and the incoming ensemble's decoding vector. With weight learning, the weight matrix itself is updated every time step, and the changes made relate back to its incorporated encoding and decoding vectors. The decoding and encoding step are bypassed, while learning can ensue.

All learning rules discussed later on can be applied as both ensemble learning or weight learning. This research employs learning in a fully connected weight matrix between the first and second ensemble. However, due to the design of the neural network eventually learning the word pairs, we have the need for an encoding ensemble. This ensemble receives the input directly, so activity is not coming in from weights. This leads to the learning in this research being two-fold; encoder learning in the first ensemble, and weight learning between the first and second ensemble. Finally, the network's output has to be decoded in order to relate it back to the input. We could also learn these decoders online but, since we are learning the actual behaviour between the first and second ensemble in the network, we have no way to discern between the contribution of the actual weight learning and the subsequent decoder learning. This is why the decoders at the end of the network are linearly optimized according to Equation 2.8. This ensures that the change in learned behaviour in the network is completely due to the weight learning and encoder selection.

2.4.1 Encoder selection

Before we get into *how* encoder learning and selection works, it is important to look at *why* it is necessary. As mentioned in Section 2.2.1, a neuron's encoding vector (e_i) influences the preferred direction of this neuron. This becomes clear from Equation 2.6, which takes the dot product ($\cdot$) of the input vector and the encoder vector into the calculation of the input current for that neuron. With encoders and inputs typically being normalized in these networks, this means that the smaller the angle between an encoder vector and the input it's receiving, the more it will respond. This is also illustrated and explained by the receptive field in Figure 2.3, which also showed the effect of the intercept - the size of the neuron's receptive field.

Depending on the purpose of your neural network, we can change the distribution of an ensemble's input selectivity, or learn relevant positions for the encoders - increasing the ensemble's selectivity. When the inputs into the network are continuous, it might be useful to distribute the ensemble's encoder randomly and uniformly around the hyperspace the network is receptive to. In this way, the responsive behaviour of the ensemble's neurons can cover the range of possible inputs. However, in some simulations the range of inputs is set and known. This is the case in our simulation, and we can benefit from this by letting the encoders *learn* the relevant positions. This not only causes a more selective approach, but we can also benefit from the design of the semantic pointers we created. This is possible because word pairs that are relevant to each other, are closer in distance in the hyperspace of inputs than non-related word pairs. Next, learning rules are discussed that are geared towards increasing input selectivity. First, the Voja learning rule will be explained. This will form the basis for the other learning rules, which are adaptations of Voja's concept.

2.4.1.1 Voja

Vector Oja (Voja) is an extension of the well-known Oja learning rule, a mathematical formulation of Hebbian learning (Oja 1989). It is developed by some of the pioneers of the NEF, and their main motivation was to formalize Hebbian learning for spiking neural network simulations (Voelker et al. 2014). The Voja learning rule uses the difference between the input (x) and the current encoder position (e_i) as an error term, combines this with the neural activity the neuron produces (a_i) and factors in a learning rate (η):

$$\Delta e_i = \eta a_i (x - e_i) \tag{2.11}$$

Like many single update learning rules, it aims to minimize the error term over time, which is the angle between encoder position and input here. With this angle narrowing, the neural activity of the neuron in question rises. This, in turn, positively influences the change in encoder position again, resulting in a vicious cycle of the encoder becoming equal to the input it is presented with. The rate with which this is happening is governed by the Voja learning rate (η).

It is, however, not the case that *all* neurons in an ensemble that is presented with an input will have their encoders geared towards this particular input. As mentioned before, the response range of a neuron is modelled by its intercept, and the Voja rule efficiently demonstrates the need for a properly-chosen intercept in

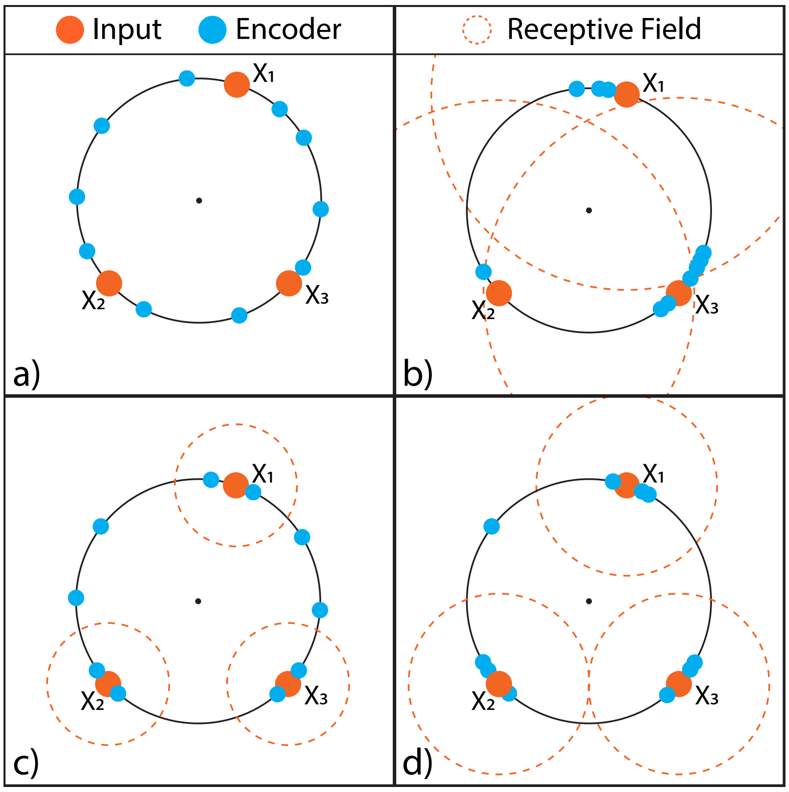


Figure 2.5: Influence of intercept on distribution of encoder position after convergence with Voja in a two-dimensional ensemble with ten neurons. The ensemble is presented with three unequally spaced inputs for an arbitrary duration in an order of $[X_1, X_2, X_3]$. **a)** Starting positions for the neuron's encoders in the ensemble. **b)** The intercept is too low, causing the receptive fields to be large. Presented inputs are competing for encoders, causing the encoders to favor the last presented input (X_3). **c)** The intercept is too high. There is no competition for encoders between inputs, but not all neurons are utilized. **d)** Properly-chosen intercepts corresponding to the highest dot product between all combinations of inputs. Almost all encoders are utilized and there is no competition between inputs.

Figure 2.5. When the intercepts of neurons in an ensemble are chosen to be relatively low, their receptive fields are very broad. This can be useful for some scenario's, and catastrophic in others. For example, it allows different inputs that are both within a receptive field of an encoder to continuously let the neuron 'compete' in selecting one of them. It also causes the order in which various inputs are presented to be of higher influence. When some input can be presented often at first, it can already 'claim' a great deal of the neurons in an ensemble to be specified towards that input. Another input that is presented later on, will not get the same reaction and has to compete to 'steal' other encoders from that first input. Alternatively, if the intercept is chosen to be too high, the receptive field of neurons shrinks. This causes for every input to have its own, distinct group of neurons to be drawn to itself. However, it might lead to a high proportion of neurons in an ensemble not responding to anything, because no input is presented in their receptive field in the hyperspace. In some situations, this might offer more freedom for novel inputs that are 'new' to the simulation, in others it will be a design flaw. As can be concluded from these different cases, choosing an intercept accordingly to the simulation one is running is of vital importance.

When the goal is to maximally utilize all neurons in an ensemble but to avoid inputs competing for encoders, a common practice is to choose an intercept equal to the largest dot product between pairs of inputs. This causes the receptive fields of neurons operating in that hyperspace to have minimal overlap. Ideally, inputs will be evenly spaced in the hyperspace which, theoretically, allows for using every neuron in the ensemble.

2.4.1.2 Voja2

The first adaptation of the Voja learning rules comes forth from current research performed by dr. Borst. It is yet unpublished and still in development, and will for now be referred to as Voja2. It is a combination of regular Voja and negative Voja, where the update moves encoders in the direction away from the presented input. The neuron's filtered activity is incorporated into this learning rule, but in a different way. Instead of simply integrating a neuron's filtered activity in the update rule itself, a shift factor (γ) is calculated. This factor determines the sign and magnitude of the encoder update using a threshold for a neuron's filtered activity:

$$\gamma = \frac{1}{a_i - \sigma} \tag{2.12}$$

where σ if the filtered activity threshold based on a threshold percentage ρ and a maximum filtered activity $\max(a)$. The parameter ρ has to be set beforehand, and

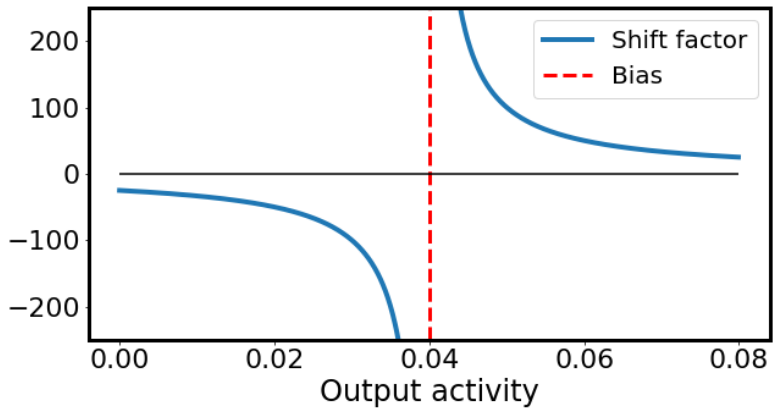


Figure 2.6: Shift factor for output activities between 0 and 0.08, the maximum neural activity a neuron has in this simulation. This maximum activity can be updated during simulation or can be set beforehand. The bias percentage (ρ) is at 50%, creating a threshold activity (σ) at $a_i = 0.04$.

determines the value of the activity threshold: $\sigma = \rho * \max(a)$. This creates a threshold in neural activity for which the encoder update should become positive. Figure 2.6 shows the effect of the shift factor for neural activities in the range of our simulation. The shift factor (γ) is combined with the standard Voja update rule from Equation 2.11, with the exception of the filtered activity:

$$\Delta e_i = \eta(x - e_i) * \gamma \tag{2.13}$$

yielding the Voja2 update rule. As can be seen in Figure 2.6, values for the shift factor are rather extreme when output activities approach the threshold activity, and decrease in both directions towards an asymptote at $\gamma = 0$. This causes the Voja2 rule to mainly affect encoders of neurons that fire around the threshold activity, which get relatively large updates. These updates cause the neuron's encoder positions to either move quickly away from or towards the presented input, respectively decreasing or increasing the neuron's output activity even more. Consequently, the aim of the Voja2 rule is to separate the neurons based on their activity. During the convergence of the neurons in an ensemble, a neurons reactivity will be of influence on whether the encoder position will converge to a certain input or move away from it. Moving away from a presented input a neuron is not responding to, can cause non-responding neurons to be more receptive to other inputs yet to be presented. The separation of responding and non-responding neurons therefore aims to utilize more of the encoder space and, in theory, could converge towards the presented input range more quickly.

2.4.1.3 Voja2-RF

An observation can be made when analysing Equation 2.13 and Figure 2.6 that has a fairly consequential implication for the updates being performed by the Voja2 rule. Filtered activity has, instead of being incorporated into the update rule directly, an influence on the magnitude of the shift factor. This aims at the desired goal of specifically separating responding and non-responding neurons to presented inputs. As a neuron's activity approaches either not or fully responding to the input, the shift factor approximates an asymptote at $\gamma = 0$. In fact, when a neuron is completely not responding to an input (i.e. it's encoder position is located *outside* the input's receptive field), the shifted factor still has a significant influence. For our use case, this is $\frac{1}{0-0.5*0.08} = \frac{1}{-0.04} = -25$, which is directly fed into the Voja2 update (Equation 2.12). Since all neurons with encoder positions outside of the input's receptive field will not fire for that input (according to Equation 2.6), their encoder positions will be constantly updated negatively, shifting them as far as possible from the presented input. The argument of Voja2 discussed above still holds, namely that the rule aims to maximally separate 'useful' (i.e. responding neurons) from the non-useful neurons. However, NEF's biological plausibility design approach makes a case for *only* altering neurons that receive a current input. Indeed, how can a neuron that is not even receptive to a certain input, be influenced by it?

The original Voja rule already solves this question by incorporating a neuron's filtered activity directly into its update rule. That is also the solution proposed here for negating the influence the shift factor poses on Voja2 updates for neuron's with encoder positions outside the presented input's receptive field, yielding the Voja2-RF (Voja2 Receptive Field) update rule:

$$\Delta e_i = \eta a_i (x - e_i) * \gamma \tag{2.14}$$

This simply addition into the update rule limits updates to the input's receptive field and, as is intended, satisfies the biologically plausible learning intent.

2.4.1.4 Voja2-RF-H

Another adaptation of the Voja2 learning rule that will be used in our simulation is different in the way the shift factor (γ) is calculated. Figure 2.6 shows the range of values shifted post can take in our network. This can differ greatly based on what maximum activity a spiking network's neurons can have. For example, a network

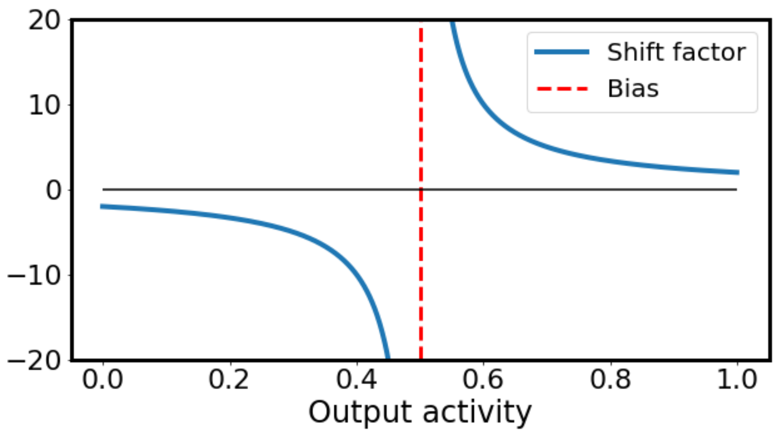


Figure 2.7: Shift factor for output activities between 0 and 1. The bias percentage (ρ) is set at 50% again, creating a threshold activity (σ) at $a_i = 0.5$.

in which the maximum filtered activity of one neuron is equal to 1 has less extreme values for the shift factor. Figure 2.7 shows the range of values possible for the shift factor in this case. For the case of non-responding neurons, the shift factor will still be -2 however ($\frac{1}{0-0.5*1} = \frac{1}{-0.5} = -2$). Another aspect of the shift factor distribution is its contribution to the eventual encoder update applied to the encoder positions. Around the threshold activity ($\gamma = \pm 100 - 200$) but also towards maximum and minimum possible activity values ($\gamma = \pm 25$) shift factor values are orders of magnitude higher than any other argument in the encoder update rule ($\eta = 0.05 - 0.00005$, $a_i = 0 - 0.08$, $(x - e_i) = 0 - 2$). The consequence is that the shift factor is by far the most important variable in the encoder update rule. In order to cancel this effect out, other functions are considered.

In essence, we need the shift factor to indicate if the encoder update rule applies a negative (-1) or positive (1) influence. This decision is based on the threshold value of filtered activity. A Heaviside step function (H) is especially suited for discerning values:

$$H(x) = \begin{matrix} -1 & : x < \sigma \\ 1 & : x \geqslant \sigma \end{matrix} \tag{2.15}$$

However, in essence, discerning between values based on a discrete threshold value is a discontinuous function. In this case, the discontinuity occurs at σ. In order to avoid this problem, a logistic approximation of the Heaviside can be used, slightly

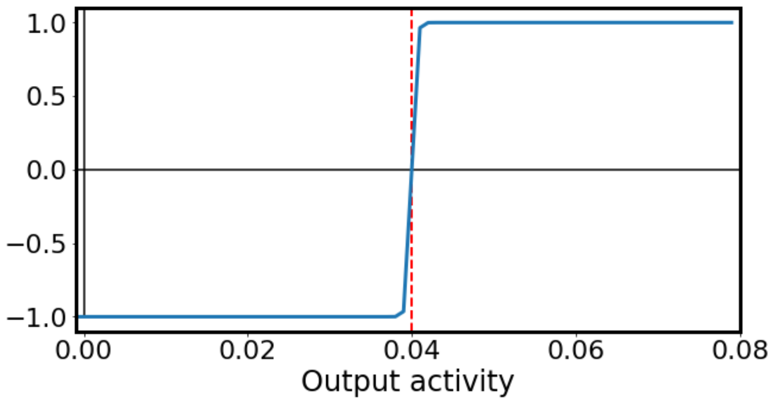


Figure 2.8: The altered logistic approximation of the Heaviside step function $H(x)$ as the new shift factor γ. The transition factor κ is set at 2000.

altered for this purpose to use a range of $< -1, 1 >$ instead of $< 0, 1 >$:

$$H(x) = \frac{2}{1 + e^{-2\kappa(x-\sigma)}} - 1 \tag{2.16}$$

where κ indicates a transition parameter for which a higher value results in a sharper transition. Figure 2.8 shows the transition for the new shift factor. The update rule for Voja2-RF-H is simply Equation 2.14 in which γ is given by Equation 2.16.

2.4.2 Weight learning

The last section described various ways of selecting neurons in an ensemble for the range of inputs that are presented to the network. When these neurons are pre-tuned to fire to certain inputs, it logically follows to let these neurons learn some function on that preferred input. All neurons in the ensemble that receives the input (*N*) are fully connected to all neurons in the next ensemble (*M*), in the network used in this research. This results in a *NxM* weight matrix filled with the connection weights for every possible connection between these two ensembles. Naturally, since each individual neuron in the first ensemble is connected to *all* neurons in the next ensemble, activation in the second ensemble is not in a one-on-one correspondence with the selection of neurons in the first ensemble. However, this information of selected neurons is communicated through the weight matrix and its connection weights are adapted accordingly. Next, the learning rule will be discussed that allows the weights to learn a function taking the information of which neurons fire into consideration.

Prescribed Error Sensitivity (PES; MacNeil and Eliasmith 2011) is developed through the research of some of the founders of the NEF and SPA architectures. Even though some learning rules could already be applied to learn associations online in a spiking neural network based on these architectures, the PES rule was motivated by the constraint of biological plausibility by using a locally computed error term. It can be applied to the decoders of an ensemble as well as to the connection weights between two ensembles. Earlier research has already shown the success of combining this supervised learning rule with the unsupervised selection of encoders in learning associations in NEF-based simulations (Knight et al. 2016, Aubin et al. 2016, Aubin 2018). The PES learning rule aims to reduce the difference between a neural population's output ($\hat{x}$) and the desired value, resulting in the error term (*E*). It can do this by adjusting the decoders (d_i) through using the following equation:

$$\Delta d_i = \eta E a_i \tag{2.17}$$

where η again indicates a learning rate. The error term can be used to apply a function to the learning:

$$E = f(x) - \hat{x} \tag{2.18}$$

or it can assume the identity function ($f(x) = x$) when this is not required. This update rule can be extended to the weight function by incorporating the gain ($\alpha_j, j \in N$) and encoder ($e_j, j \in M$) from the second ensemble as both play vital roles in how information received from the connections from the first ensemble causes activation in the second:

$$\Delta w_{ij} = \eta \alpha_j e_j E a_i \tag{2.19}$$

Here, the incorporation of the gain and encoder of the 'receiving' neuron is justified by how the weights (w_{ij}) are formed (Equation 2.9). In the calculation of the encoders of the second ensemble the gain of that neuron is already considered as is mentioned in Section 2.2.3.

Chapter 3

Results

3.1 Simulation parameters

A spiking neural network is constructed in Python employing the methods presented in Chapter 2. A list of 8 word pairs comprising 12 unique words is constructed by means of the Semantic Pointer Architecture (SPA) methods discussed in Section 2.1. Table 3.1 shows the word pairs and associated fan, where all word pairs are fan-complete (i.e. components of a word pair have an equal fan). Word pairs are converted into 32-dimensional numeric representational vectors which are subsequently normalized to 1. A 32-dimensional vocabulary is created using the Python implementation of the SPA, and is populated with all unique words. This creates 32-dimensional numerical vector representations for each word which are normalized to 1. These vectors are spaced apart in the hyperspace with a similarity constraint of 0.1, which was satisfied for each unique vector. Similarity between numerical vectors is calculated by using the dot product between the two. Since all vectors are normalized to 1, this results in the a value between -1 and 1 (respectively corresponding to angles of 180°and 0°), which can be translated back to the angle by calculating the inverse cosine. In the rest of this research, we will stick with the

Word 1	**Word 2**	**Fan**
CART	DOGS	1
CATS	BALL	1
BIKE	BEER	1
LAMP	FARM	1
LENS	RACK	2
LENS	FILE	2
WHEEL	RACK	2
WHEEL	FILE	2

Table 3.1: Table containing all 12 unique words, forming 8 word pairs to be studied. Word pairs are fan-complete.

dot product as a similarity measure. Word pair vectors are subsequently generated by superposition of the constituting word vectors, and the resulting vector is normalized to 1 again. This created small angles between Fan 2 word pairs (*M* = 0.41, *SD* = 0.01) and large angles between Fan 1 pairs (*M* = -0.13, *SD* = 0.08), indicating similarity for Fan 2 pairs (as was intended).

In training, each individual word pair is presented 15 times, for a duration of 1 second. After training, each word pair is tested 5 times, for a duration of 0.5 seconds. Each presentation in training and testing is followed by a pause of 0.5 second during which an input of 0 was presented. This is necessary for neural activities in the network to revert back to resting state in order for the network to in a 'neutral' state for receiving the next input. Word pair train and test order are randomized, but fixed across experiments, ensuring the same influence of order on every experiment.

The network itself consists of two ensembles of 1600 neurons each, which are fully connected through a 1600x1600 weight matrix. The weights are initialized at 0, and will be learned by using the PES learning rule. Due to the dimensionality of the input vectors, the network is configured to handle 32-dimensional information. Encoders are randomly distributed around the hypersphere, and the decoder of the second ensemble is linearly optimized using Equation 2.8 in order to retrieve output from that ensemble. The intercept for the first ensemble is uniformly chosen to be the highest dot product between unrelated word pair vectors (i.e. word pairs that share no constituting word, x_N^{int} = 0.09).

3.2 Baseline experiment

First, we will look at how the spiking neural network is capable of learning the associative memory of word pairs without any encoder selection. This sets a baseline for the rest of our experiments, since adding any Voja rule will solely focus on encoder changes and eventual changes in performance are due to encoder convergence (or divergence).

A spiking neural network in line with the parameters in Section 3.1 is trained on the 8 word pairs from Table 3.1. The only learning rule that will be applied is PES on the connection weights. The output vector retrieved from the second ensemble is compared to all word pairs in the vocabulary in order to determine the similarity. Again, the dot product is used, where the higher the dot product is, the more similar the output vector is to the word pair. Figure 3.1 shows average output similarities and standard deviations averaged over presented pair types (Fan 1 & Fan 2) for various PES learning rates. It becomes clear that the network has no issue in learning the distinction between the word pairs.

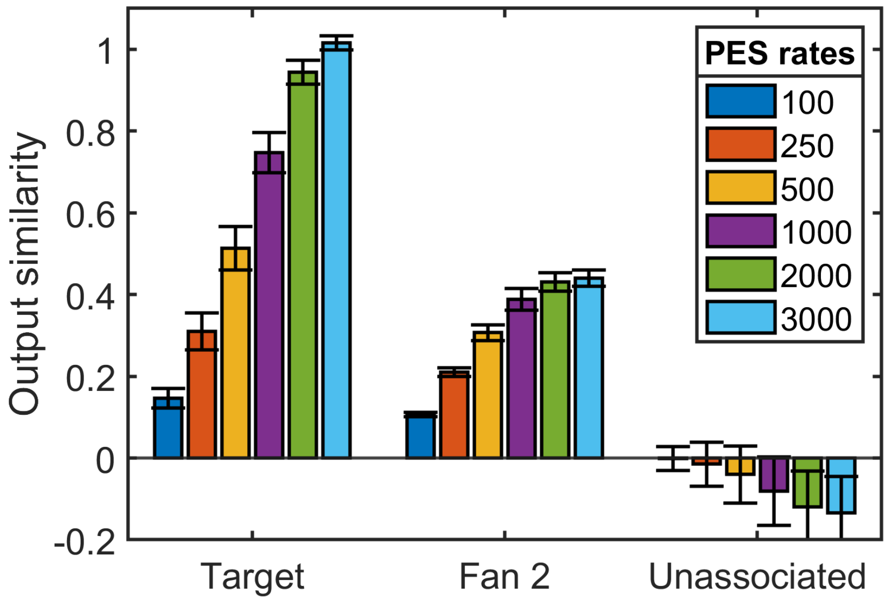


Figure 3.1: Output similarities for various incrementing PES learning rates. Output is taken from the last 100 milliseconds of presentation during testing. This is averaged and compared to the presented (target) pair (n = 40), associated Fan 2 word pairs (n = 60) and unassociated pairs (n = 220).

Figure 3.2 shows the results for a PES learning rate of 2000, split by what pair is presented. This is more interesting because, if we know that the network can learn our generated word pairs, we want to focus on how it handles Fan 2 pairs in comparison to Fan 1 pairs. It is clear that for both pairs, the network has correctly learned the right vector and outputs this. Fan 2 output similarities are slightly higher, due to the overlap in information. Fan 2 word pairs are more similar than the other, distinct Fan 1 pairs. This results in slightly more exposure (training) on the Fan 2 pairs, resulting in higher similarity. The Fan 2 column shows that the Fan 2 associated pairs are also correctly similar to the output. It would be strange if this was not the case, since if the output vector is similar to the Fan 2 pair that was input, the associated Fan 2 word pairs are similar because of the way we generated them and the similarity between them. However, what is more interesting is that the standard deviation of the Fan 2 target pair similarity is really low. This indicates that every Fan 2 word pair is distinctly and correctly learned. The network is thus

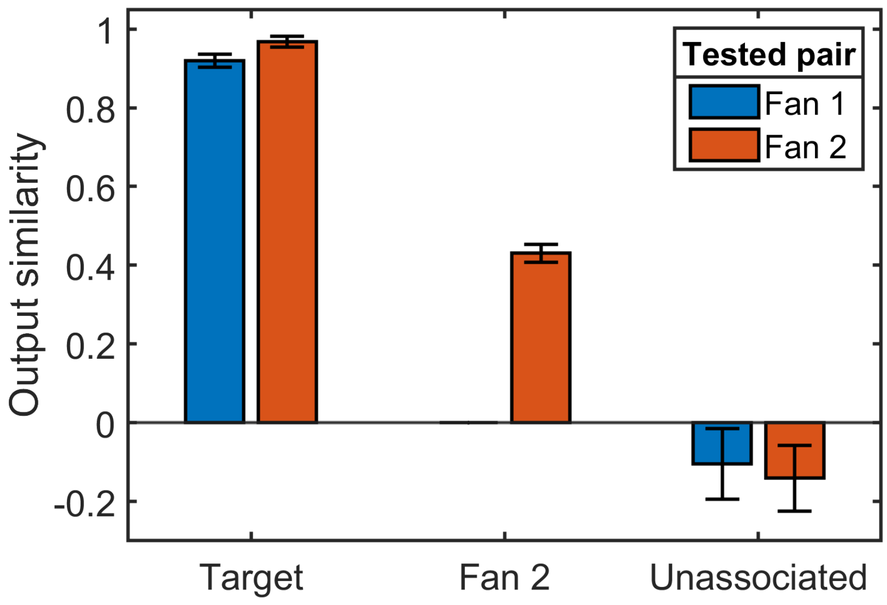


Figure 3.2: Averaged output similarity to word pairs, divided based on the fan of the tested pair. Outputs are again compared to target pairs (n = 20 per bar), associated Fan 2 pairs (n = 60) and unassociated pairs (n = 140 (Fan 1), n = 80 (Fan 2)).

able to deal with the overlap between information in a way that related information can still be distinctly learned. However, associated information should be harder to retrieve and, as the fan effect describes, results in a lower accuracy. Here, we see a slightly higher accuracy and no heightened difficulty. The following sections will focus on how encoder selection affect this behaviour, and how this can produce a different effect of associated information.

3.3 Voja

For the experiments regarding encoder selection, we have kept the PES learning rate at 1000. As Figure 3.1 indicates, this learning rate causes the network for this simulation to correctly learn the word pairs, but also leaves some room for improvement. In addition to showing associated memory effects, we are also interested in how encoder selection aids the overall performance of the network. When PES learning rates are high, the network already performs nearly perfect, and the network can not improve due to the influence of encoder selection.

With a constant PES learning rate, we can focus on what influence various Voja learning rates have. It is be important to note that one can vary the network's encoder's receptive fields as well, but in order to maximally utilize the hyperspace, this has been set at the highest dot product between unrelated word pairs (as described in Section 2.4.1). Figure 3.3 shows the averaged results over presented pair

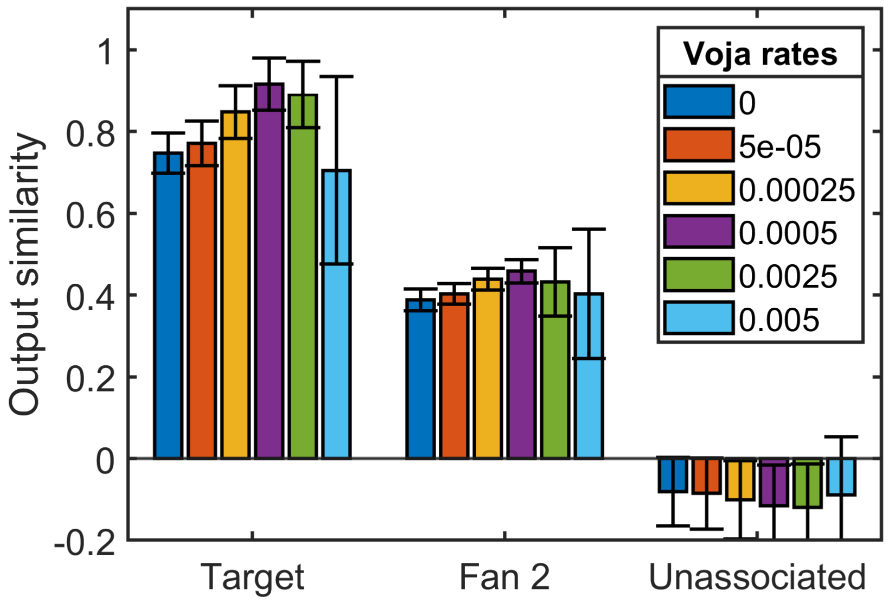


Figure 3.3: Output similarities for various incrementing Voja learning rates. Output is taken from the last 100 milliseconds of presentation during testing. This is averaged and compared to the presented (target) pair (n = 40), associated Fan 2 word pairs (n = 60) and unassociated pairs (n = 220).

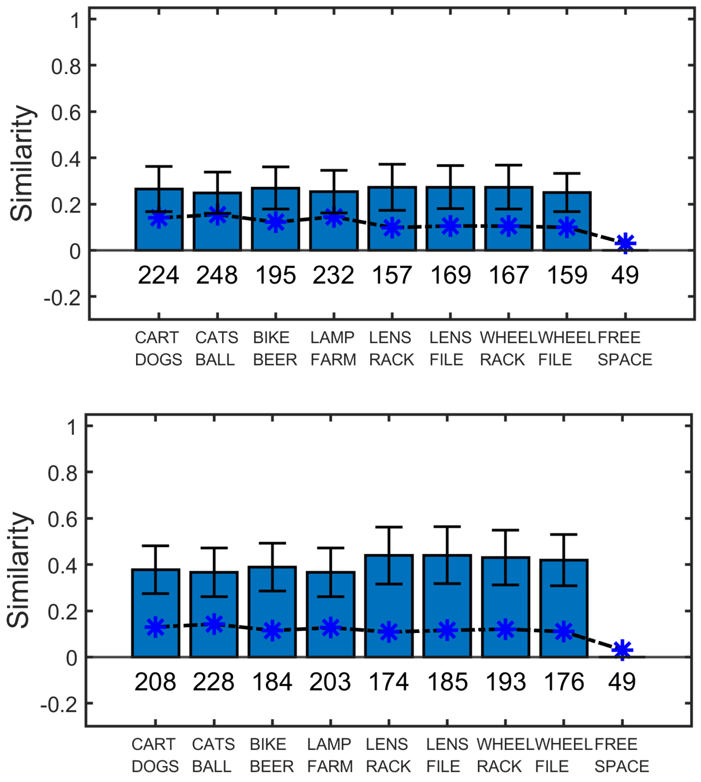


Figure 3.4: Similarity of encoder positions to word pairs before training (top) and after training (bottom). Lines through the bar and numbers under the x-axis respectively indicate percentage and amount of neuron's encoder positions that are within the receptive field of word pair vectors.

type for a range of Voja learning rates. The results show an increase in output similarity for when Voja is applied (at 0.0005 learning rate) versus when it is not applied (0 learning rate). In this case, the encoder selection has caused an encoder convergence towards the presented inputs, which has helped PES to learn the distinct word pairs better. This is illustrated as well in Figure 3.4, in which the start and end positions of the encoders are shown, along with their distribution across the receptive fields of the inputs. The lower graph shows that, after training, encoder positions have shifted towards the relative inputs (slightly). Also, the distribution has not changed, dividing all encoders equally between the presented inputs. The effect of more exposure for Fan 2 vectors shown in Figure 3.2 is visible as well, whereas they

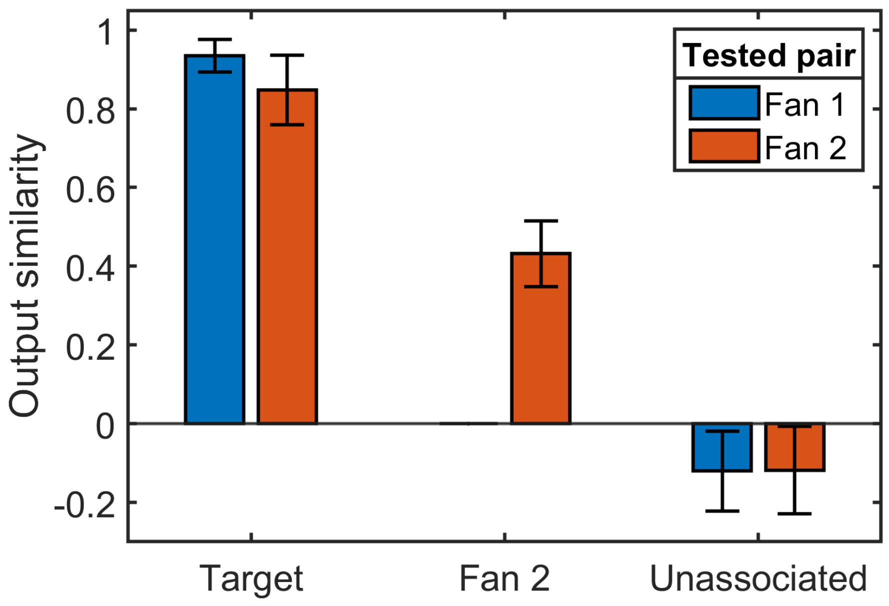
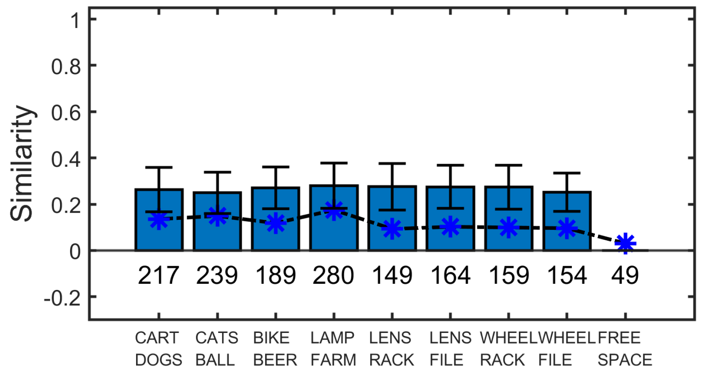


Figure 3.5: Output similarities for a Voja learning rate of 0.0025, split by fan of presented pair.

were first equal to Fan 1 vectors, they now have increased more.

Another interesting observation is the increased standard deviation for higher learning rates. In order to highlight where this variance comes from, Figure 3.5 splits the output similarities based on the presented pairs for a Voja learning rate of 0.0025. While the Fan 1 pairs that are presented, were correctly learned, there is some variation in performance for Fan 2 pairs. When looking at the encoder distributions in Figure 3.6, one can see that the encoder positions have been shifted to a particular word pair, namely $WHEEL \circledast RACK$. The overlapping receptive field between associated Fan 2 word pairs cause the inputs to 'compete' for encoder positions, as was already highlighted in Section 2.4.1.1 and Figure 2.5. The presentation order of inputs will be of influence to which positions the encoder positions have converged for the last time. This will also create a certain bias towards a word pair, which causes inaccuracies in retrieval in the testing phase, explaining the increased standard deviation for Fan 2 word pairs in Figure 3.5.

Now, another important observable difference between Figure 3.4 and 3.6 is that encoder similarities to presented word pair vectors has increased dramatically af-



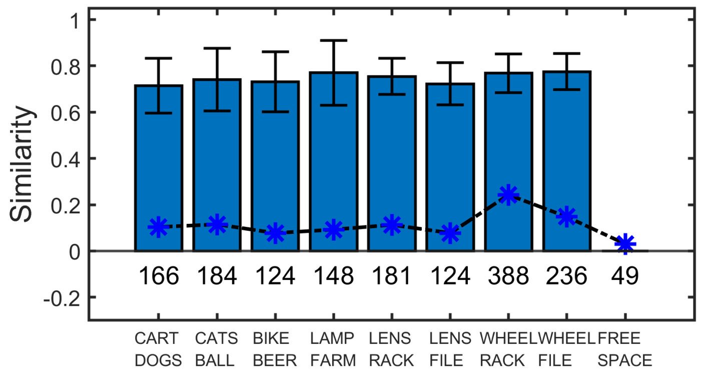


Figure 3.6: Similarity of encoder positions to word pairs before training (top) and after training (bottom) for a Voja learning rate of 0.0025.

ter training for the case with the higher Voja learning rate. Of course, this is the behaviour we seek for in encoder selection, but this increased change in encoder similarity has also created the bias for Fan 2 word pairs. The following section will describe how the changes made in Voja's first adaptation, Voja2, will influence this behaviour.

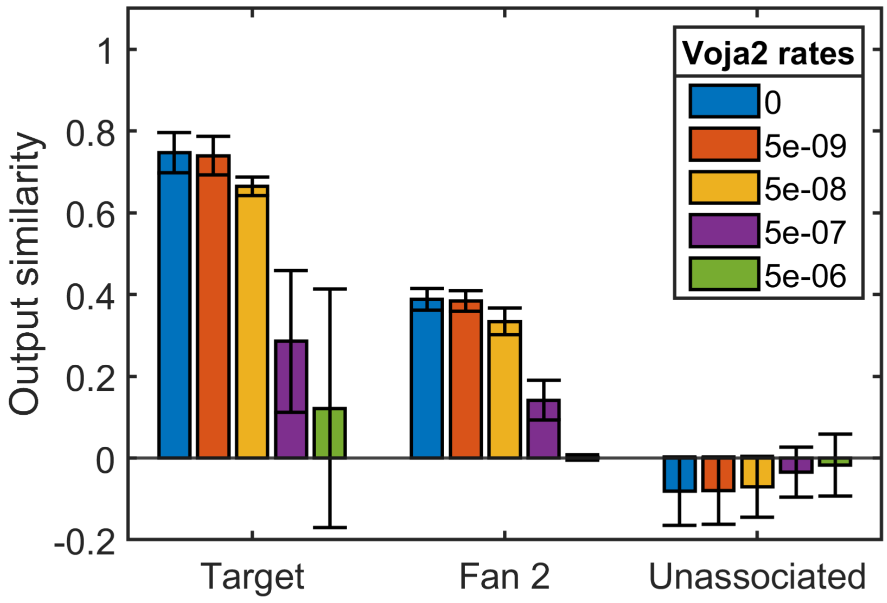
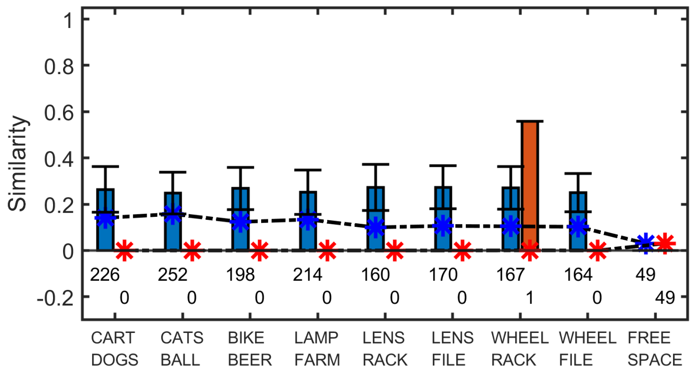


Figure 3.7: Output similarities for various incrementing Voja2 learning rates. Output is taken from the last 100 milliseconds of presentation during testing. This is averaged and compared to the presented (target) pair (n = 40), associated fan 2 word pairs (n = 60) and unassociated pairs (n = 220).

3.4 Voja2

The same comparison for learning rates has been conducted for Voja2. Figure 3.7 shows the results of output similarities for a range of learning rates. In this case, Voja2 is not enabling our network to improve from the performance when it was trained solely learning with PES. First, the results from a learning rate of 5e-08 shows us a decreased, but uniform performance (low standard deviation). Figure 3.8 shows the encoder positions pre-training (top) and post-training (bottom) for this learning rate, similar to Figure 3.6. Where Voja only applied positive encoder updates to the encoder positions, Voja2 (and further adaptations) also apply updates that move encoders away from the presented input. When looking at averaged encoder similarities to inputs, the converging and diverging encoder positions will bring so much variation into the overall average that this becomes a void measure. That is why a separation is created at the middle of the receptive field in order to



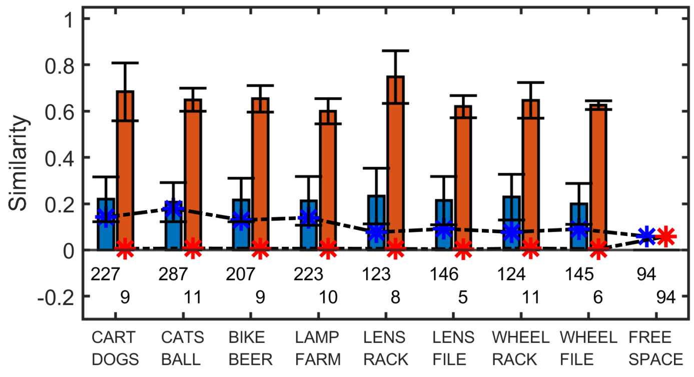


Figure 3.8: Similarity of encoder positions to word pairs before training (top) and after training (bottom) for a Voja2 learning rate of 5e-08. Encoders within receptive fields are split into a high receptive field (red bars) and a low receptive field (blue bars). The separation between the high and low receptive field is defined by (1-x_N^{int})/2, i.e. half of the receptive field.

divide the encoder positions in two different fields; a low receptive field and a high receptive field.

Figure 3.8 allows us to look at updates made due to the training to both fields. First, when we look at the high receptive field, it becomes clear that Voja2 has caused some responding neurons in a word pair's receptive field that are responding to that input to converge towards that pair's positions. However, amounts given below the x-axis indicate that these are very low numbers ($\sim$10 encoders converged towards each input). Second, the low receptive field indicates that the encoder positions within that lower receptive field have diverged from their respective inputs - in line with the negative influence Voja2 poses on the encoders. Third, the amount of neurons with encoder positions in the free space has increased two-fold, because the updates are now not limited to receptive fields anymore. This also allows for some shifting of neurons between receptive fields of unassociated inputs.

Ultimately, the cause for the lower performance of a Voja 2 learning rate of 5e-08 (in comparison to no encoder learning) can be ascribed to the high amount of neurons diverging from the inputs they did not respond to. Some encoders converged towards the word pairs but the amounts were low enough that their influence did not influence the total networks output.

Next, it is interesting to look at what happens to encoder positions for learning rates that caused output similarities to plummet in Figure 3.7. For example, Figure 3.9 shows the encoder position similarity before and after training for a Voja2

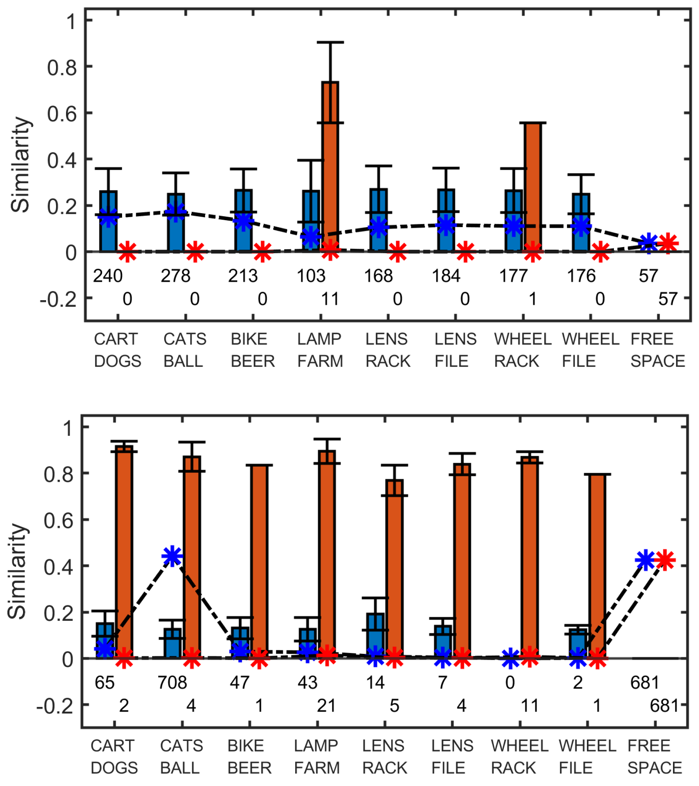


Figure 3.9: Similarity of encoder positions to word pairs before training (top) and after training (bottom) for a Voja2 learning rate of 5e-07.

learning rate of 5e-07.

Again, some encoders have converged to high similarities to word pairs, but the most clear observation is that most of the encoders have been pushed to certain positions not relevant to the word pairs. A large amount (681 out of 1600, 42.6%) have been pushed to free space by negative updates, and the other large part (708 out of 1600, 44.2%) is located in the lower receptive field of $CATS \circledast BALL$. This is due to the behaviour of pushing most of the encoder positions away, where there is no limit to where they are pushed. It is thus likely that the last presented word pair during training is one opposite from the position these encoders ended up in after training. Figures 3.8 and 3.9 have mainly shown that there is only a small portion of neurons that satisfy the threshold activity for Voja2, as there are only $\sim$10 neurons converging to each word pair.

As a solution, we can vary the threshold percentage (ρ) for output activity. This was originally at a value of 50%, separating activity on a bias of half of the maximum activity. Figure 3.10 shows the averaged output similarities for Voja2 learning rates

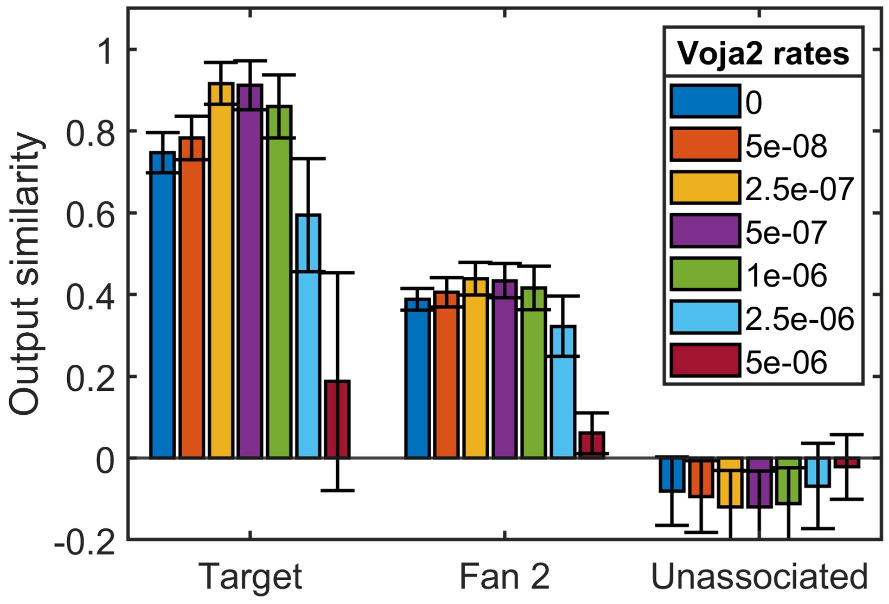


Figure 3.10: Output similarities for various incrementing Voja2 learning rates. Bias percentage (ρ) is set at 25%.

for a threshold percentage of 25%, i.e. where the threshold activity is at 25% of the maximum output activity. Here, we can see that Voja2 now does improve the network's performance before dramatically decreasing, indicating that the combination of converging/diverging encoder positions due to Voja2 can help learning. Figure 3.11 shows that, indeed, more encoder positions are converged towards word pair positions due to a lower activity threshold. Also, an additional factor is that non-responding neurons seem to have been pushed out of all receptive fields, into the free space of the hyperspace. Another observation to be made from this figure, is that average similarities for the low receptive fields have increased. Looking at the amounts of encoders present in these low receptive fields, these could be neuron's encoder positions that have not yet converged towards their respective word pairs,

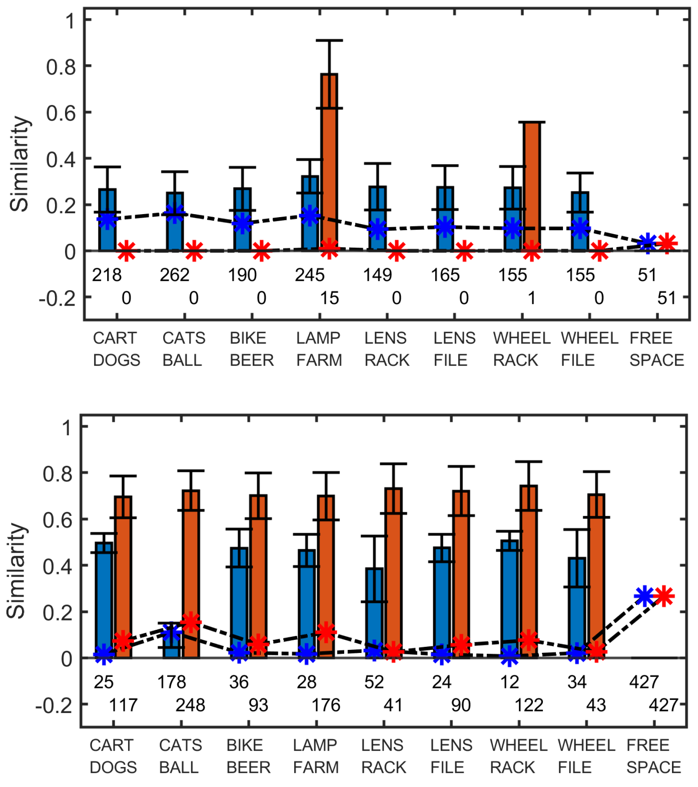


Figure 3.11: Similarity of encoder positions to word pairs before training (top) and after training (bottom) for a Voja2 learning rate of 5e-07. The bias percentage (ρ) is set at 25%.

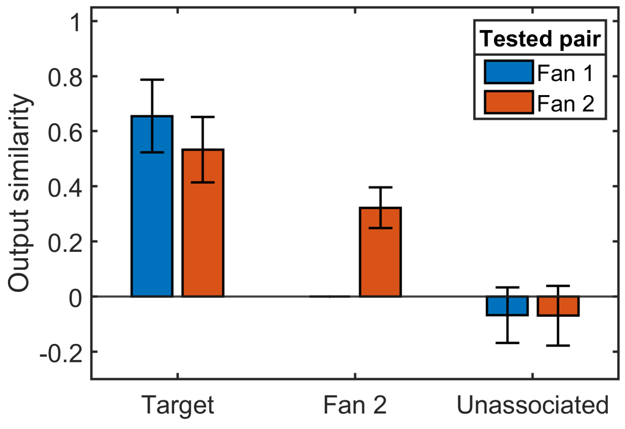


Figure 3.12: Output similarities split by presented pair type for a Voja2 learning rate of 2.5e-06, with a bias percentage (ρ) that is set at 25%.

and would have converged given more training time or a larger learning rate.

The last observation to be made from Figure 3.10 is that when learning rates increase beyond 5e-07, performance of the network decreases. As Figure 3.12 shows, performance not only decreases for Fan 2 word pairs, but also for Fan 1 pairs, which is not what the fan effect states. Figure 3.13 shows the encoder positions for this sim-

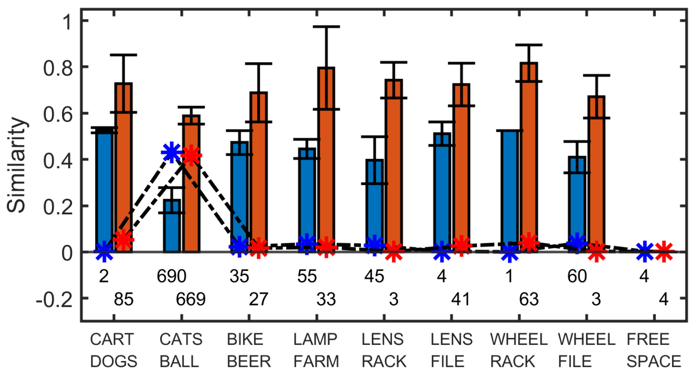


Figure 3.13: Encoder position similarity to inputs after training using a Voja2 learning rate of 2.5e-06 and a bias percentage (ρ) that is set at 25%.

ulation after training and illustrates how diverging encoder updates have caused a shift of almost all encoders towards one input. This input was most likely the furthest in hyperspace from the last presented input during training.

Appendix A shows the same graphs as are shown for a bias percentage of 25% in this section, but now for a bias percentage of 10%. This increases the influence on final performance that the negatively updated encoder positions had, as can be seen from the encoder similarity plots. This results in too little encoder positions converging to the word pairs that are presented. The following section goes into the effect of how this behaviour changes when the Voja2 encoder updates are limited to the receptive fields of presented inputs.

3.5 Voja2-RF

The adaptation of Voja2 Receptive Field (Voja2-RF) limits encoder updates to the receptive field of a presented input, in line with Equation 2.14. This rule is implemented in the spiking neural network as well, and the results from Figure 3.10 we just discussed in the previous section, will be compared to a similar situation using Voja2-RF. All other simulations performed in Section 3.4 are also performed using the Voja2-RF rule, but in the interest of readability, these results can be found in

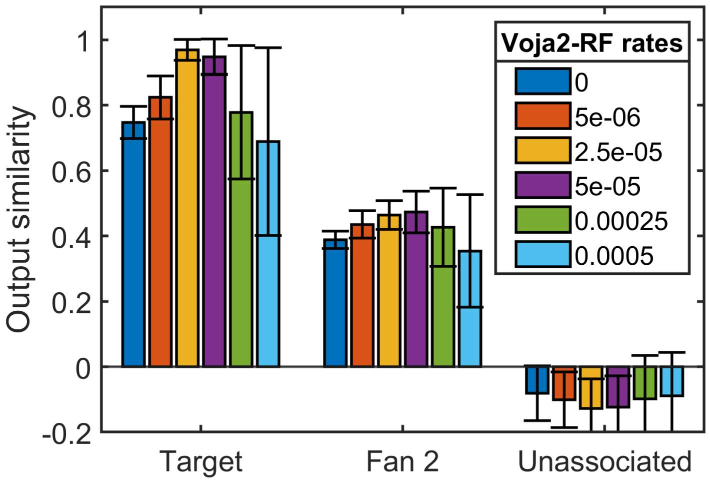


Figure 3.14: Output similarities for various incrementing Voja2-RF learning rates. Bias percentage (ρ) is set at 25%.

Appendix B. In this, and in the upcoming section, we will focus on the results of a simulation testing a range of learning rates using a threshold percentage (ρ) of 25%. Figure 3.14 shows the results for this simulation learning the encoders with Voja2-RF.

Similar to Voja2 at a bias percentage of 25%, the network's performance increases uniformly (for both Fan 1 and Fan 2 word pairs) as learning rate increases. The interesting difference between Figure 3.10 and Figure 3.14 occurs as learning rates increase beyond the point of uniform performance increase. Figure 3.15 shows the performance of the network for a learning rate of 2.5e-04, split by presented word pair. This situation shows no decreased performance for Fan 1 pairs, but a decrease in performance for Fan 2 pairs - in line with what the fan effect shows in human performance. Figure 3.16 shows how encoder positions in receptive fields of Fan 1 word pair inputs, after training, have converged to their respective Fan 1 word pairs. However, for Fan 2 word pair inputs there is some competition between encoders. The final positions of encoders for Fan 2 pairs have settled around the high receptive field of the pairs $WHEEL \circledast RACK$ and $LENS \circledast FILE$, two unassociated Fan 2 pairs. These encoder positions will contribute towards the correct recollection of these two pairs, while the other two Fan 2 pairs will be under represented in encoder space. However, these remaining pairs ($LENS \circledast RACK$ and $WHEEL \circledast FILE$)

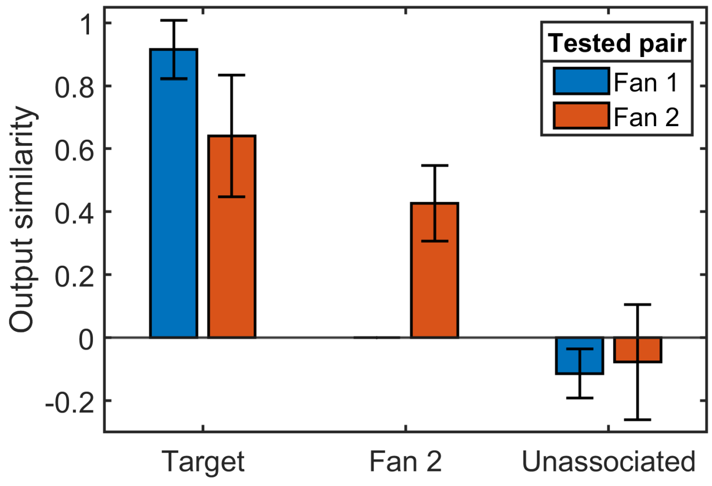


Figure 3.15: Output similarities split by presented word pair type for a learning rate of 2.5e-04 and a bias percentage of 25%.

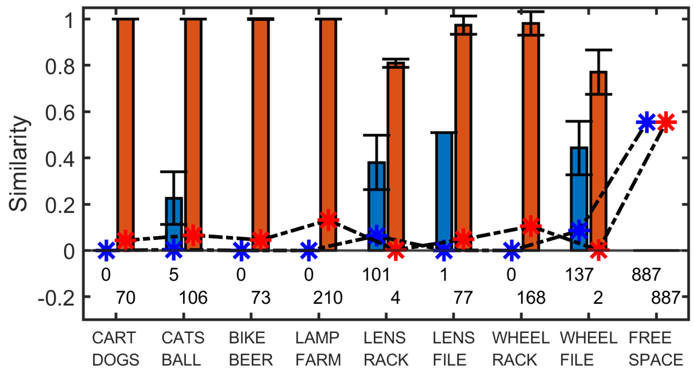


Figure 3.16: Encoder position similarity to inputs after training using a learning rate of 2.5e-04 and a bias percentage (ρ) that is set at 25%.

have relatively large amounts of encoders present in their low receptive field, which will still contribute to the performance of the recollection of these pairs.

Additionally, Figure 3.16 shows that more than half of the ensemble's encoder positions ended up in the hyperspace's free space (887 out of 1600, 55.4%). The Voja2-RF rule is designed to be limited to the input's receptive fields, and can thus not make updates beyond those receptive fields. However, it is possible that a neuron's activity causes for a diverging update, which is large enough to push the neuron's encoder position outside of the receptive field of the presented input. Once in free space, those neurons will not respond to any input anymore. This is however not a design flaw, but rather leaves the network with more adaptibility for eventual new inputs. Non-responding neurons are selected to be pushed out of the receptive fields of known inputs, after which they cannot be attracted anymore by the known inputs. As the performance of the network for Fan 1 word pairs shows, the amount of neuron's encoder positions converging to the inputs is still high enough to correctly learn these inputs.

Only the case of a bias percentage of 25% is discussed here. For the purpose of additional comparisons with other encoder selection rules, the same figures as those shown for a bias percentage of 25% are shown in Appendix B for bias percentages of 50% and 10%.

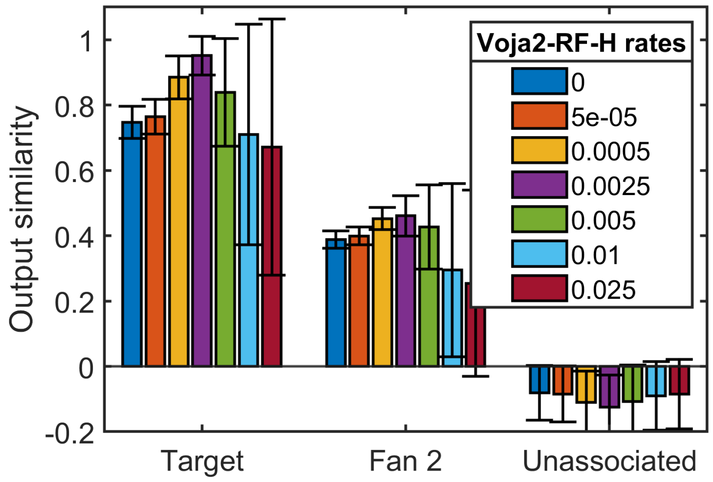


Figure 3.17: Output similarities for various incrementing Voja2-RF-H learning rates. Bias percentage (ρ) is set at 25%.

3.6 Voja2-RF-H

Section 2.4.1.4 motivated the adaptation of the current learning rule, Voja2-RF-H. Figure 2.6 shows the value range of the shift factor (γ) and, when analysing the learning rules used in the two previous sections (Equations 2.13 and 2.14), the influence this shift factor has is deemed to be too big. Voja2-RF-H aims to diminish this influence by calculating the shift factor according to Equation 2.16.

Again, all simulations performed for the previously studied learning rules are performed, and the case of a bias percentage of 25% will be discussed here. Figure 3.17 shows the results of the network for a range of learning rates. Similar to the results from Voja2-RF, Voja2-RF-H is able to uniformly increase the network's performance and to subsequently decrease the performance with high variation. This indicates a lower performance for (most likely) Fan 2 word pairs, as Figure 3.18 confirms for a learning rate of 0.005.

Also, the encoder positions after training in Figure 3.19 show a similar situation as Figure 3.16. Encoders in the receptive fields of Fan 1 pairs have converged to the respective Fan 1 word pairs, while there is 'competition' between encoders for the Fan 2 word pairs. A lower amount of encoders that ended up in the free space (499 out of 1600, 31.2%), which can be attributed to the lower influence of the newly calculated shift factor. Non-responding neurons are updated with a shift-factor of

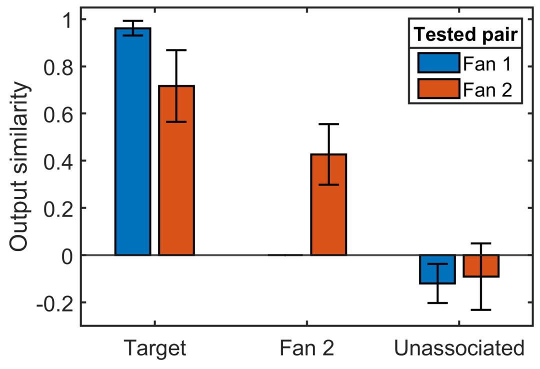


Figure 3.18: Output similarities split by presented word pair type for a learning rate of 0.005 and a bias percentage of 25%.

-1, instead of higher values of the asymptote of the shift factor used in Voja2 and Voja2-RF.

Similar simulations are performed for bias percentages of 50% and 10%, and the results can be viewed in Appendix C.

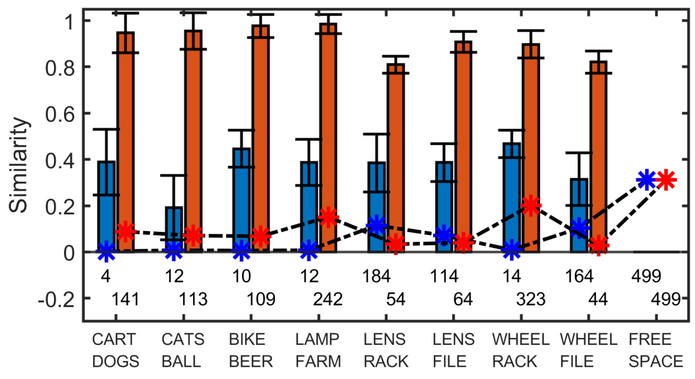


Figure 3.19: Encoder position similarity to inputs after training using a learning rate of 0.005 and a bias percentage (ρ) that is set at 25%.

3.7 Predicted neural activity

In addition to the accuracy, another manifestation of the fan effect in human associative memory is of interest for this research; estimated neural activity. Lower neural activity in brain regions associated with the retrieval stage as fan increases is a finding that is documented in Borst et al. (2016). This phenomenon is further being investigated, and it will be interesting to see how encoder learning rules influence predicted neural activity in our network.

We can make a predicted estimate regarding neural activity when an input is presented based on the values involved in the calculation of neural activity. The eventual neural activity in an ensemble is a filtered version of the individual neuron's spike trains, where the spike train is calculated from the neuron's non-linearity (Equation 2.7). In turn, the input for the spike train is the input current to each neuron, calculated by Equation 2.6, where values are involved that eventually influence the ensemble's neural activity. When looking at constant inputs, the only variable that changes due to encoder selection, is the encoder position itself. The gain and bias parameters both stay the same, regardless of encoder learning. This means that neural activity will depend solely on the encoder position, which is the value encoder learning changes. In addition, the number of neurons responding to a certain input will influence the ensemble's total neural activity as well, which is also a value the encoder learning changes. It will therefore be interesting to investigate the course of how predicted activity changes as encoder learning progresses, and what influence each encoder learning rule has on the predicted neural activity. Neural activity (a_P) for a fan will simply be predicted by:

$$a_P = \sum_{n=1}^{WP} N_n^{RF} \times \overline{s_n} \tag{3.1}$$

where WP is the total amount of word pairs for that fan, N_n^{RF} is the amount of neuron's encoder positions within the receptive field of input n, and $\overline{s_n}$ is the average similarity of encoder positions within the receptive field of input n to that input n. Note that encoder positions within the *high* receptive field of an input are used here, due to different learning rules adapting non-responding neurons in a different way. These non-responding neurons will be pushed to the low receptive field, and can either (due to learning rate or shift factor) be pushed out of the receptive field or just stay in. In order to isolate responding neurons that will have the most influence on the final predicted neural activity, the non-responding neurons will be excluded. Since the motivation for investigating predicted neural activity is to compare this to findings from the fan effect in humans, we are looking at network simulations that resulted in a lower accuracy for Fan 2 word pairs in comparison with Fan 1 word

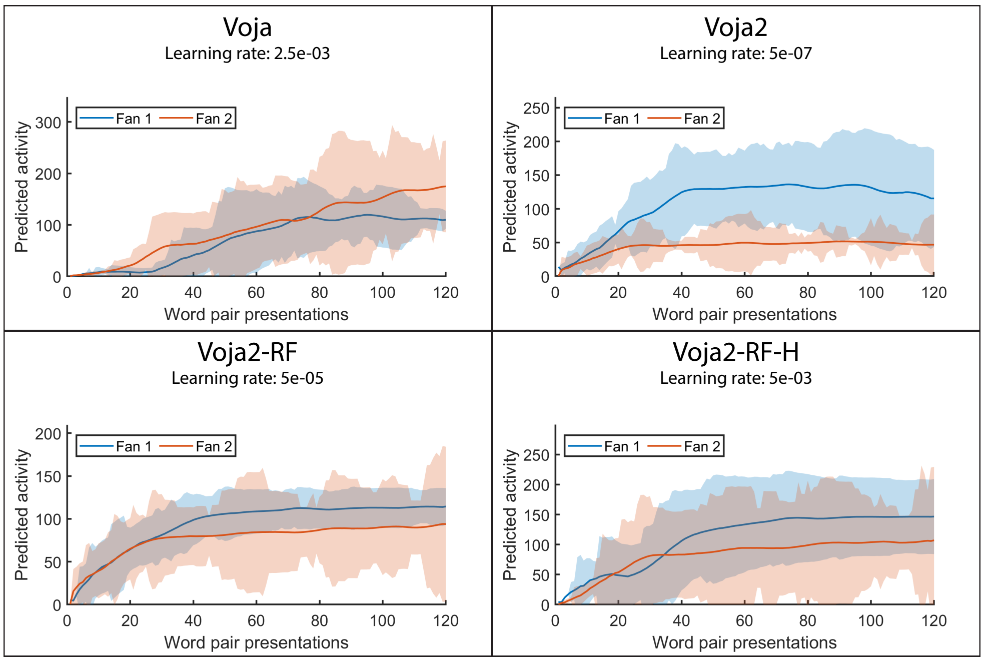


Figure 3.20: Changes in predicted neural activity during training using the four different encoder learning algorithms. Standard deviations are taken over the four different word pairs per fan type, and thus can differ greatly. Simulations where chosen where the encoder learning rule caused the network to have variance in its performance (i.e. prone to making mistakes, lower accuracy). The adaptations of the original Voja learning rule all employed a bias percentage of 25%. Predicted neural activity is calculated according to Equation 3.1.

pairs.

Figure 3.20 shows the resulting changes in predicted neural activity due to the various examined encoder learning rules. The sought-after behaviour of increased neural activity for Fan 1 word pairs seems to be apparent in all three adaptations of the original Voja rule, whereas the original Voja rule predicts higher neural activities for Fan 2 pairs.

Since adaptations of the original Voja rule using a bias percentage of 50% have not shown to improve the encoder selection, nor did they create variance between performance of Fan 1 pairs and Fan 2 pairs, neural activities were not predicted. A bias percentage of 10% has shown some performance in line with the expectancies of the fan effect, and predicted neural activities can be found in Appendix D.

Chapter 4
Discussion

While exploring a spiking neural network's capability of handling associated information, various learning rules have been investigated. Specifically, the interest lies on how the different encoder learning rules influence the network's results with regard to measures of the fan effect in humans – reduced accuracy and neural activity as fan increases. First of all, the baseline experiment showed that the PES weight learning rule was capable of discerning between all word pairs presented to the network. Indeed, a high enough learning rate proved all word pairs to be recognized correctly. However, the accuracy for Fan 2 word pairs increasing with learning rate did not indicate any behaviour similar to the fan effect. The approach then was to fine-tune the input selection of the first ensemble in the network using the Voja learning rule and three adaptations of this rule.

The results of the first encoder learning rule, Voja, showed that accuracy decreased for Fan 2 word pairs as the learning rate increased. Analysis of the encoder positions after training clarified this result – the overlapping receptive fields of Fan 2 word pair inputs (i.e. associated information) caused competition between inputs for encoders. Whatever the last presented input during training was, would have a higher amount of encoder positions converged to it, creating a bias for that input in the results. Additionally, predictions of neural activity show that Fan 2 word pairs are likely to cause more activation compared to Fan 1 word pairs – the reverse of the expected effect shown in human data. Thus, while the Voja learning rule did result in accuracy results corresponding to behavioural results, the neural activity it predicts is the reverse from what was found in human MEG data.

Voja2 has, next to converging capabilities, diverging capabilities for non-responding neurons. In the results from the network trained using this learning rule, it was shown that the network demonstrated both lower accuracy and lower predicted neural activity for Fan 2 word pairs. That is, when using an aptly chosen activity bias. However, where the accuracy decreased for Fan 2 word pairs, the accuracy also decreased for Fan 1 word pairs. One could say that mistakes are also made in Fan 1 word pairs, but it is an illogical result from increasing the learning rate.

Next, Voja2-RF limited the encoder updates to the receptive fields of inputs. This modification proved to increase the network's performance for Fan 1 word pairs while decreasing the accuracy for Fan 2 word pairs as learning rate increased. The limitation to receptive fields converged responding neurons to the input, while pushing the non-responding neurons just out of the receptive field. While this learning rule limits updates to the receptive field of an input, its final update before an encoder position is no longer in the receptive field can be large enough to just push the encoder's position out of reach. A fairly large percentage of the neuron's encoder positions had been diverged to free space in the network's operational hyperspace, while still resulting in excellent performance. Even though the updates this learning rule made go just beyond the receptive field, it does produce behaviour that is more biologically plausible than the Voja2 rule. Overall, this learning rule not only created the expected behaviour in accuracy scores, it also showed to have lower predicted neural activity for Fan 2 word pairs.

Finally, the Voja2-RF-H learning rule changes the manner in which the shift factor was calculated and aimed to reduce this factor's influence on the eventual encoder update. This learning rule employs a more logical calculation of the shift factor, simply indicating the direction of the encoder update without an additional magnitude influencing the update any further. In comparison to Voja2-RF, this learning rule shows the same behaviour, with the slight change that the number of encoders diverging to free space is lower. This can be attributed to the lower value of the shift factor, which reduces the extremity of the diverging encoder update before the encoder's position gets pushed out of the receptive field.

In summation, the designed spiking neural network shows effects of the fan effect similar to those observed in human data. In order to exhibit these effects, encoder learning has to be involved. Competition between inputs for encoders due to overlapping receptive fields for associated word pairs causes an expected decrease in recognition accuracy for these associated word pairs after learning. In order to show decreased neural activity, the learning rule has to diverge non-responding neuron's encoders from the receptive fields of all presented word pairs.

4.1 Implications

4.1.1 Encoder learning rules

The combination of Voja for encoder learning and PES for weight learning is a popular option often employed by networks designed according to the NEF. This research shows that Voja cannot explain all effects in human data in the case of associated information, and some alternatives that are investigated here can provide new options for simulations of memory handling such associated information.

First, the Voja2 learning rule applies updates to encoders with positions outside of the current input's receptive field. This provides more diverging capabilities to the learning rule, pushing non-responding neurons as far as possible from the current input. Results proved that this could result in extreme updates in encoder positions, and it is also hard to justify the biological plausibility of this behaviour. The aim in the design of a neuron responding to inputs in its receptive field was to limit neurons to a certain range of stimuli (a common concept in neurological modeling; Blais 2013). Adapting a neuron's encoder position when it can not possibly respond to the presented input anymore and thus has no relation to this input - as happens in Voja2 - is not in line with that earlier design decision which was based on biological plausibility. This was the motivation for Voja2-RF and, as can be seen in the results, resulted in measures that reflect the fan effect more accurately.

In line with this logic, one might also argue that the updates Voja2-RF and Voja2-RF-H make to push encoder positions just beyond the receptive field are not biologically plausible either. However, the updates stop just beyond the edge of the receptive field, and encoders are not adapted if their neurons do not respond anymore (i.e. its encoder positions lies outside of receptive field).

4.1.2 Cognitive modelling using SNNs

As mentioned above, the NEF poses the Voja-PES combination as a suitable candidate for biologically plausible learning. However, the results in this thesis have shown that this way of encoder selection does not exhibit the effect of lower neural activity for word pairs with a higher fan. The other Voja adaptations have demonstrated this effect, along with the desired accuracy scores (for Voja2-RF and Voja2-RF-H).

As mentioned in the introduction, in certain (ideal) situations, one is looking to design the inputs for a network in such a way that optimal separability can be maintained. This, in combination with aptly chosen intercepts, will offer perfect convergence to the presented inputs using Voja. The demonstrated Voja adaptations in this research can be proven useful for situations where the model does not allow for optimal separability, for example due to associations between inputs. In reality, every piece of information is innately associated with its environment it occurred in. Automatically, this extends to a broad range of cognitive tasks that involve any sort of memory, which could benefit from these novel rules in order to better learn associated information.

In addition, all modified Voja rules in this research have shown to discard non-responding neurons by moving their encoder positions into free space, while not diminishing performance. This offers a novel advantage, namely in the form of flexibility of the network. With neurons that are not responding to any of the inputs in the current input set, an input that is presented to a trained network still has a good opportunity to be learned. Encoders in the free space whose neurons will respond to this input can cause this input to be learned, instead of all neurons being converged to the input in their respective fields as happens with regular Voja.

4.1.3 Neurological findings

This network's weight learning due to the PES learning rule follows the Hebbian learning theory of "neurons that fire together, wire together" (MacNeil and Eliasmith 2011, Hebb 1949). This learning rule is biologically inspired, but the findings from this research have shown that its standard implementation with Voja does not exhibits the findings from Borst et al. into neural activity and associated information (2016). The correct neural activity predicted by all Voja adaptations tested in this research imply some form of selection based on how a neuron responds to an input that is important for *if* a neuron gets selected to fire any further. In this context, we can see the encoder learning rule as some pre-selection based on a neuron's readiness to respond. Additionally, the network's structure implies a layer of pre-selection neurons in the first ensemble, that are trained to pass along information based on their 'preference' (or not).

4.2 Recommendations

In this thesis' introduction, the fan effect was introduced and its effects in both behavioural and neurological measures were identified; lower accuracy, higher reaction times, and lower neural activity. The network currently investigated, however, was not capable of making any predictions regarding reaction times when recalling studied information. Every memory item was simply tested for 500 milliseconds, and the network's output activity over the last 100 milliseconds of presentation was averaged to compare to the input vector. With further research, two approaches could be applied in order to retrieve an indication of reaction time from the network. First, bins of 50-100 milliseconds during the complete presentation of a word pair during testing could be collected, and each of them averaged to be compared to the input. Subsequently, a threshold can be set and the similarity between the input vector and output bin can be compared with that threshold. This would yield a reaction time indication of in which time bin the response would be adequate to indicate a retrieval. Second, this approach can be taken over by an additional ensemble in the network that is tasked with constantly applying a threshold to the output of the second ensemble. This would offer a more accurate indication of response time of the network.

To further investigate encoder learning and input selectivity, one could experiment with further adaptations of the Voja learning rule. For example, the Voja2-RF-H rule assigns a factor indicating a direction in which the update should be performed, whereas the Voja2 rule used a magnitude of update as well that is calculated according to the neuron's activity. Since activity is influenced by the distance of the neuron's encoder position to the input, this assigned a high value for neurons that are around the middle of the receptive field, and this value decreased towards either the input or the edge of the receptive field. Applying a factor like this to the Voja2-RF-H rule could regulate the magnitude of encoder updates either around the edge of the receptive field or towards the input. The limiting of encoder updates around the receptive field will cause the updates to not push encoder positions beyond the receptive field.

Next, it would be interesting to study how different weight learning rules affect the network's performance related to associated information. Especially, if the encoder learning rules affect the performance after training with weight learning alone in the same way as in this experiment. For example, another weight learning rule often employed in NEF models is the Bienenstock-Cooper-Munro rule (Toyoizumi et al. 2005). This rule incorporates more information about spiking history and averaged postsynaptic activity into how weights should be updated, and is a more limited approach to Hebbian learning than PES (Bienenstock et al. 1982). Encoder learning will most likely benefit all forms of subsequent weight learning since it affects the neurons in the pre-ensemble to tune better to the input, but the interaction of the two rules can change the performance across various weight learning rules.

Section 2.1 discussed the operations involved in generating Semantic Pointers, and in the current simulation of the word pair learning experiment, convolution is not involved. It would be interesting to extend this simulation in order to study the influence of word position in a pair. Individual words ($LAMP$ or $BALL$) can first be convolved with their position in the word pair ($ITEM1$ or $ITEM2$) before superposition with each other, resulting in the full word pair ($LAMP \circledast ITEM1 + BALL \circledast ITEM2$). The operation of convolution results in a distinct vector not in relation to the constituting vectors ($LAMP$ and $ITEM1$) and causes words to have varying angles to each other. This experiment would illustrate how the design of a representation of (associated) information plays an important role in the interactions between encoder selection and input positions of associated information.

To investigate any potential behavioural advantages of Voja2-RF or Voja2-RF-H over the conventional Voja rule, another simulation could be conducted investigating the new properties created by the altered encoder learning rules discussed here. In this situation, Voja2-RF and Voja2-RF-H 'discard' a high percentage of the available neurons in the first ensemble and they are pushed towards free space, or the edge of the input's receptive field. A comparison between two networks could be designed, one learning the encoder positions with Voja and one with Voja2-RF/Voja2-RF-H. The networks would be designed equally, and are trained until encoders converge to their respective input. Subsequently, a new training set of inputs would be presented to both networks. The network trained with Voja2-RF/Voja2-RF-H would be expected to learn these without problem, since it's encoder learning rule has freed up non-responsive neurons to the inputs already presented. These neurons can be utilized to converge towards the newly presented inputs, whereas the network trained with Voja would have no 'available' neurons since all its receiving neurons have already converged to the first inputs.

This also raises an interesting question towards the rate of convergence. If neural activity is lower when retrieving memory items with a higher fan, it would be interesting how many times this associated information has to be presented for this effect to consolidate into neural behaviour. An experiment with word pair learning just like the one performed in Borst et al. (Borst et al. 2016) could be conducted, where the amount of presentations of Fan 2 (or higher) effects are varied between 1 - 10 times. One would assume that the higher the number of presentations, the better a fact can be retrieved, but it's interesting what the consequence is for our measures of the fan effect (accuracy and neural activity). Subsequently, the experimental paradigm can be repeated for the network designed for the experiment discussed in this research.

4.3 Conclusion

In summary, the results of this research have shown that the Voja2-RF and Voja2-RF-H rules are promising in explaining associative recognition data, in a behavioural aspect as well as a neurological one. Numerous simulations regarding associative recognition can be repeated using the investigated network, and its suggested additions and alterations. These additional simulations can further confirm behavioural and neurological predictions, with the first choice for an adjustment being to incorporate a prediction regarding reaction time.

Bibliography

Anderson, J. R.: 1974, Retrieval of propositional information from long-term memory, *Cognitive Psychology* **6**(4), 451 – 474.

Anderson, J. R. and Paulson, R.: 1978, Interference in memory for pictorial information, *Cognitive Psychology* **10**(2), 178 – 202.

Anderson, J. R. and Reder, L. M.: 1999, The fan effect: New results and new theories., *Journal of Experimental Psychology: General* **128**(2), 186 – 197.

Aubin, S.: 2018, Learning and leveraging neural memories. Master Thesis.

Aubin, S., Voelker, A. and Eliasmith, C.: 2016, Improving with practice: A neural model of mathematical development, *Topics in Cognitive Science* **9**.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., Choo, X., Voelker, A. and Eliasmith, C.: 2014, Nengo: a Python tool for building large-scale functional brain models, *Frontiers in Neuroinformatics* **7**(48), 1–13.

Bienenstock, E., Cooper, L. and Munro, P.: 1982, Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex, *Journal of Neuroscience* **2**(1), 32–48.

Blais, B.: 2013, *Receptive Field Modeling*, Springer New York, New York, NY, pp. 1–6.

Borst, J. and Anderson, J.: 2015, The discovery of processing stages: Analyzing EEG data with Hidden Semi-Markov Models, *NeuroImage* **108**, 60–73.

Borst, J. P., Ghuman, A. S. and Anderson, J. R.: 2016, Tracking cognitive processing stages with MEG: A spatio-temporal model of associative recognition in the brain, *NeuroImage* **141**, 416 – 430.

Cantor, J. and Engle, R. W.: 1993, Working-memory capacity as long-term memory activation: An individual-differences approach., *Journal of Experimental Psychology: Learning, Memory, and Cognition* **19**(5), 1101 – 1114.

Chan, A., Halgren, E., Marinkovic, K. and Cash, S.: 2010, Decoding word and category-specific spatiotemporal representations from MEG and EEG, *NeuroImage* **54**, 3028–39.

Eliasmith, C.: 2013, *How to Build a Brain : A Neural Architecture for Biological Cognition.*, Oxford Series on Cognitive Models and Architectures, Oxford University Press.

Eliasmith, C. and Anderson, C. H.: 2002, *Neural Engineering: Computational, Representation, and Dynamics in Neurobiological Systems*, MIT Press, Cambridge, MA, USA.

Gorban, A. N., Makarov, V. A. and Tyukin, I. Y.: 2019, The unreasonable effectiveness of small neural ensembles in high-dimensional brain, *Physics of Life Reviews* **29**, 55–88.

Hebb, D. O.: 1949, *The organization of behavior: A neuropsychological theory*, Wiley.

Hopfield, J. J.: 1982, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the United States of America* **79**(8), 2554–2558.

Knight, J., Voelker, A. R., Mundy, A., Eliasmith, C. and Furber, S.: 2016, Efficient SpiNNaker simulation of a heteroassociative memory using the neural engineering framework, *2016 International Joint Conference on Neural Networks (IJCNN)* pp. 5210–5217.

MacNeil, D. and Eliasmith, C.: 2011, Fine-tuning and the stability of recurrent neural networks, *PLOS ONE* **6**(9), 1–16.

Oja, E.: 1989, Neural networks, principal components, and subspaces, *International Journal of Neural Systems* **1**, 61–68.

Pereira, F., Mitchell, T. and Botvinick, M.: 2009, Machine learning classifiers and fMRI: A tutorial overview, *NeuroImage* **45**(1, Supplement 1), S199 – S209. Mathematics in Brain Imaging.

Plate, T.: 1995, Holographic reduced representations, *IEEE Transactions on Neural Networks* **6**, 623–41.

Quian, R.: 2012, Concept cells: the building blocks of declarative memory functions, *Nature reviews. Neuroscience* **13**, 587–97.

Radvansky, G. A., Zacks, R. T. and Hasher, L.: 1996, Fact retrieval in younger and older adults: The role of mental models., *Psychology and Aging* **11**(2), 258 – 271.

Stewart, T. C.: 2012, A technical overview of the neural engineering framework, Centre for Theoretical Neuroscience Techinal Report, University of Waterloo.

Stewart, T. C., Choo, X. and Eliasmith, C.: 2010, Dynamic behaviour of a spiking model of action selection in the basal ganglia, 10th International Conference on Cognitive Modeling.

Stewart, T. and Eliasmith, C.: 2009, Compositionality and biologically plausible models, *in* W. Hinzen, E. Machery and M. Werning (eds), *Oxford Handbook of Compositionality*, Oxford University Press.

Toyoizumi, T., Pfister, J.-P., Aihara, K. and Gerstner, W.: 2005, Generalized Bienenstock–Cooper–Munro rule for spiking neurons that maximizes information transmission, *Proceedings of the National Academy of Sciences* **102**(14), 5239–5244.

Voelker, A., Crawford, E. and Eliasmith, C.: 2014, Learning large-scale heteroassociative memories in spiking neurons, Centre for Theoretical Neuroscience, University of Waterloo.

Wickelgren, W. A. and Corbett, A.: 1977, Associative interference and retrieval dynamics in yes-no recall and recognition., *Journal of Experimental Psychology: Human Learning and Memory* **3**(2), 189–202.

Zbrodoff, N. J.: 1995, Why is 9+7 harder than 2+3? Strength and interference as explanations of the problem-size effect, *Memory & Cognition* **23**, 689–700.

Appendix A

Supplementary Voja2 results

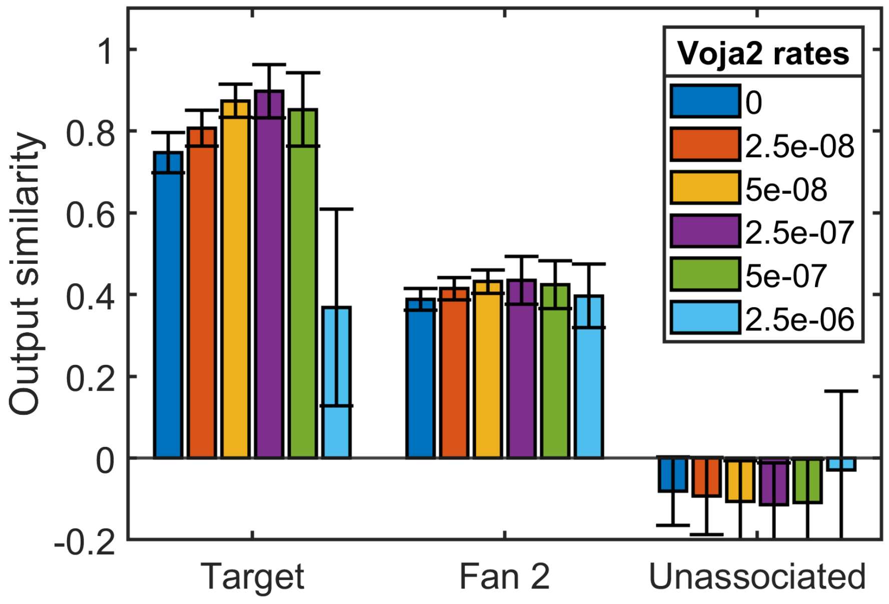


Figure A.1: Output similarities for various incrementing Voja2 learning rates. Bias percentage (ρ) is set at 10%.

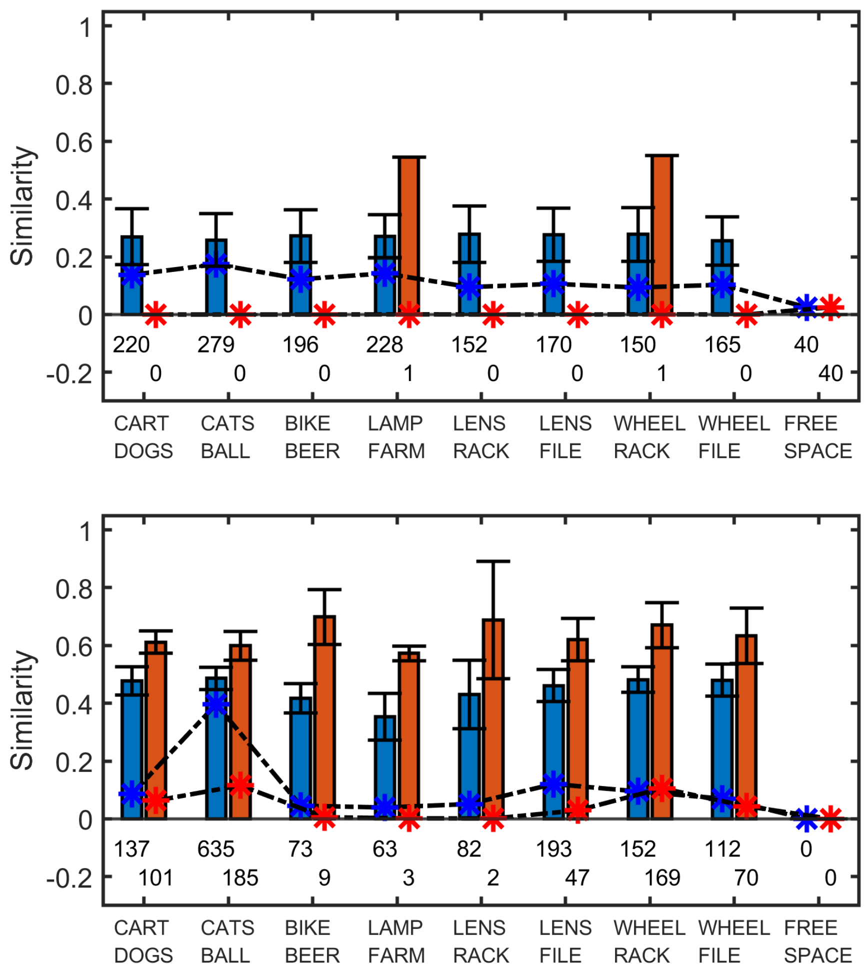


Figure A.2: Similarity of encoder positions to word pairs before training (top) and after training (bottom) for a Voja2 learning rate of 5e-07. The bias percentage (ρ) is set at 10%.

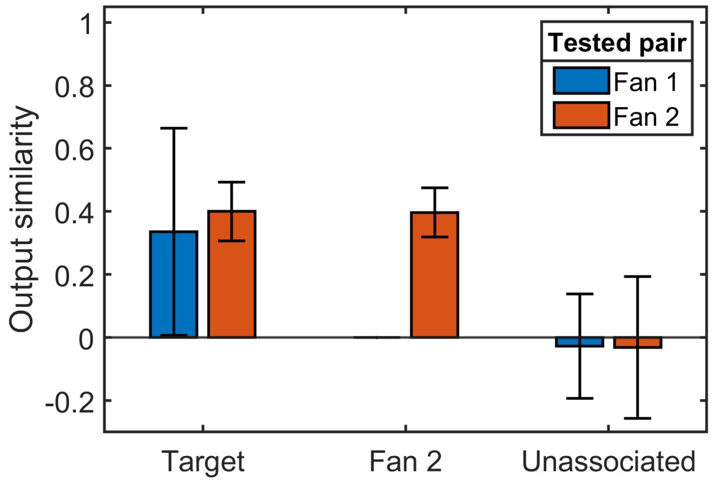


Figure A.3: Output similarities split by presented pair type for a Voja2 learning rate of 2.5e-06, with a bias percentage (ρ) that is set at 10%.

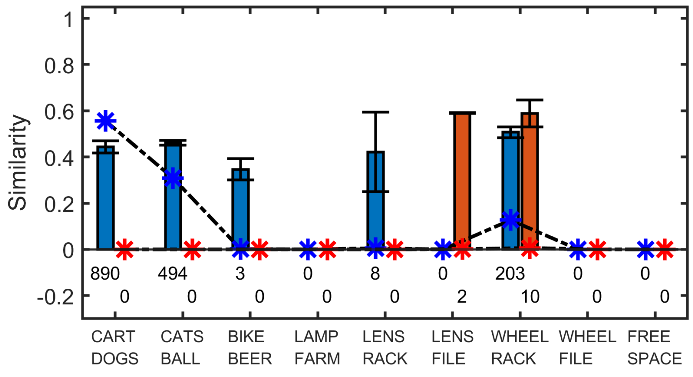


Figure A.4: Encoder position similarity to inputs after training using a learning rate of 2.5e-06 and a bias percentage (ρ) that is set at 10%.

Appendix B
Supplementary Voja2-RF results

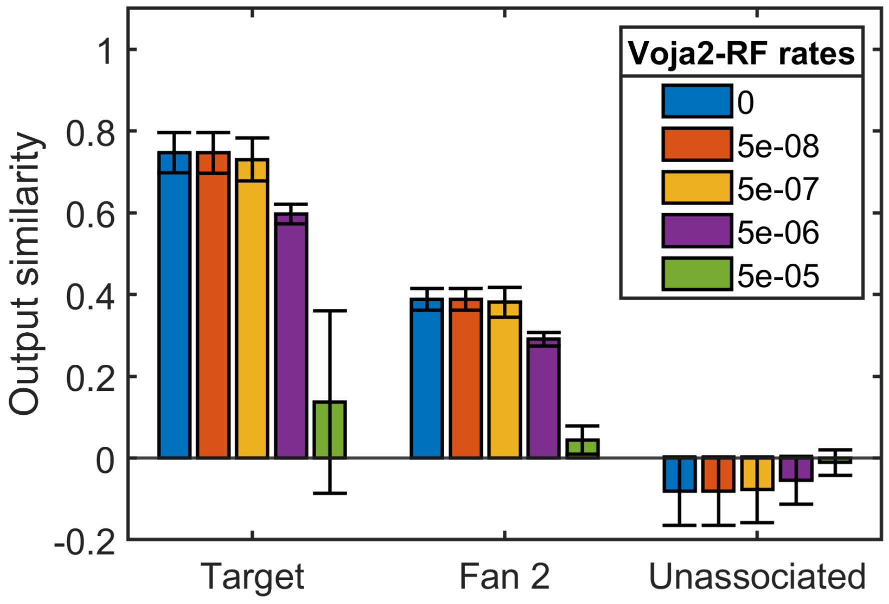


Figure B.1: Output similarities for various incrementing Voja2-RF learning rates. Bias percentage (ρ) is set at 50%.

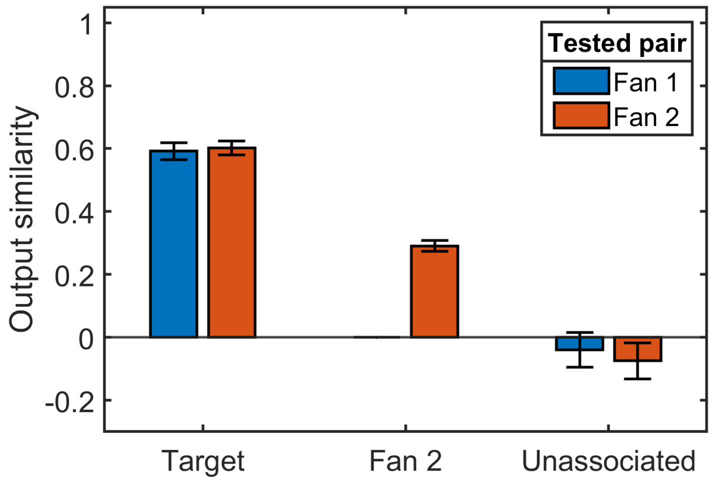


Figure B.2: Output similarities split by presented word pair type for a learning rate of 5e-06 and a bias percentage of 50%.

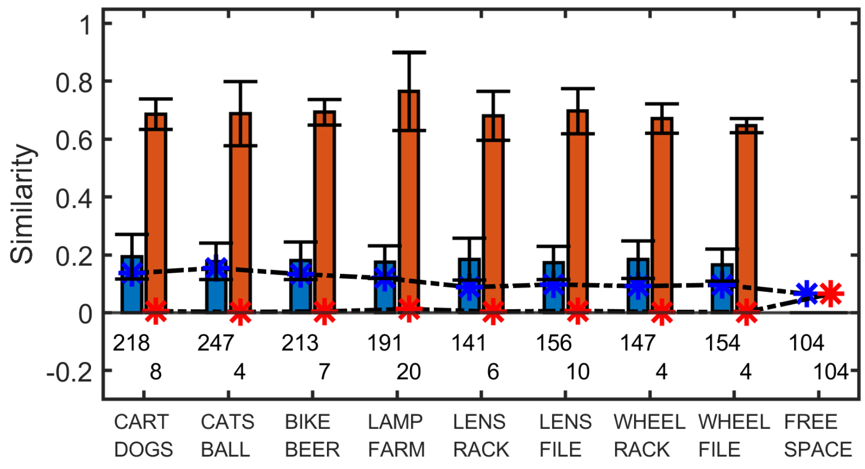


Figure B.3: Encoder position similarity to inputs after training using a learning rate of 5e-06 and a bias percentage (ρ) that is set at 50%.

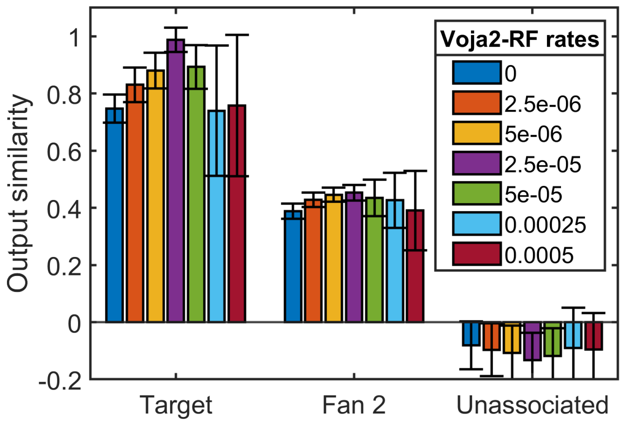


Figure B.4: Output similarities for various incrementing Voja2-RF learning rates. Bias percentage (ρ) is set at 10%.

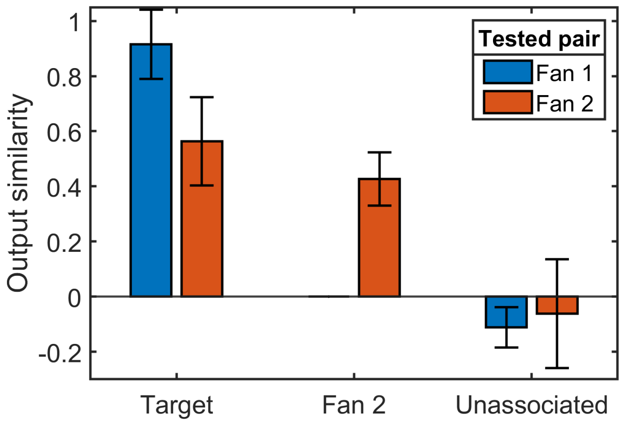


Figure B.5: Output similarities split by presented word pair type for a learning rate of 2.5e-04 and a bias percentage of 10%.

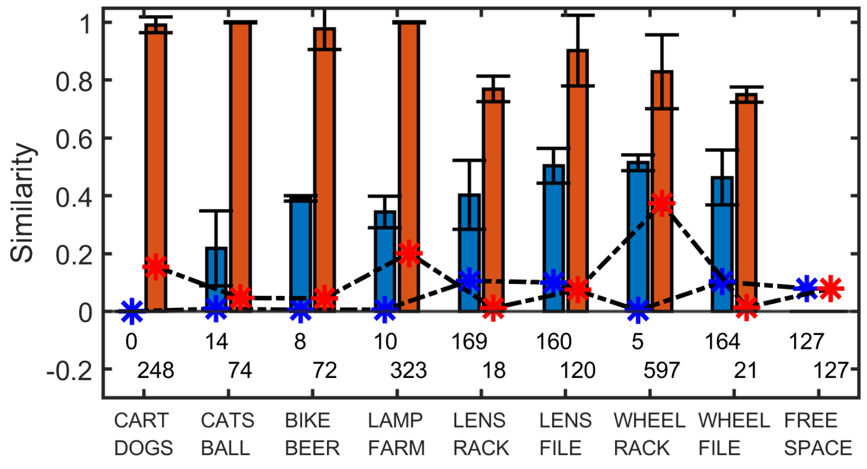


Figure B.6: Encoder position similarity to inputs after training using a learning rate of 2.5e-04 and a bias percentage (ρ) that is set at 10%.

Appendix C
Supplementary Voja2-RF-H results

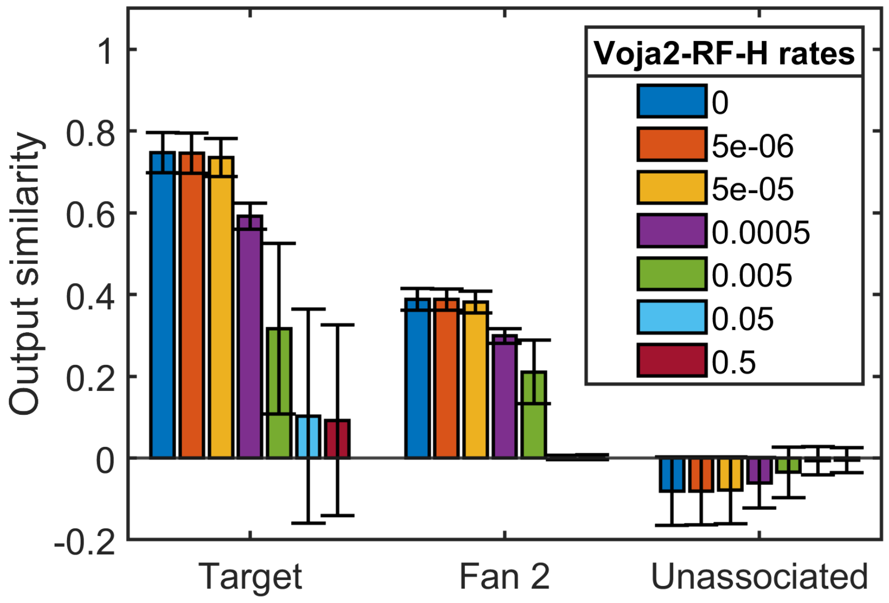


Figure C.1: Output similarities for various incrementing Voja2-RF-H learning rates. Bias percentage (ρ) is set at 50%.

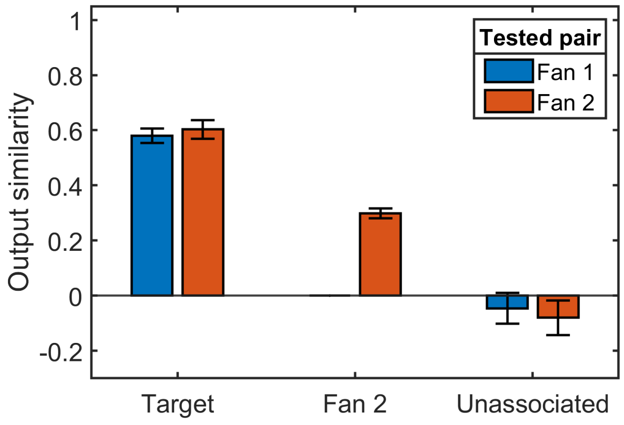


Figure C.2: Output similarities split by presented word pair type for a learning rate of 5e-04 and a bias percentage of 50%.

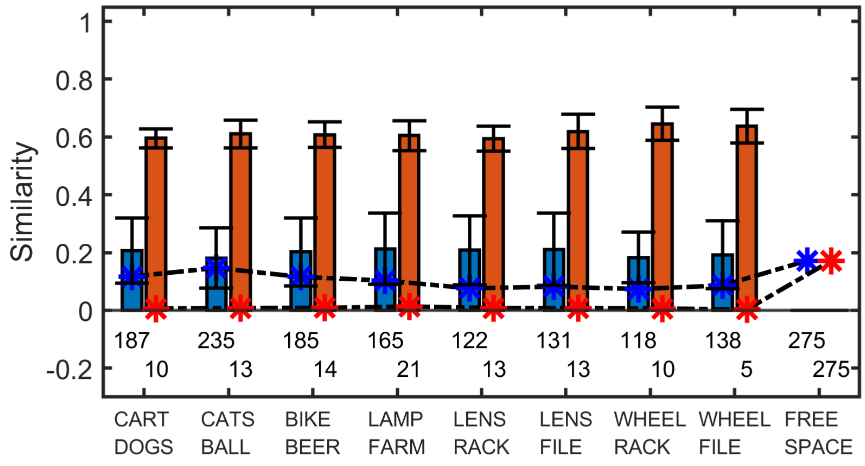


Figure C.3: Encoder position similarity to inputs after training using a learning rate of 5e-04 and a bias percentage (ρ) that is set at 50%.

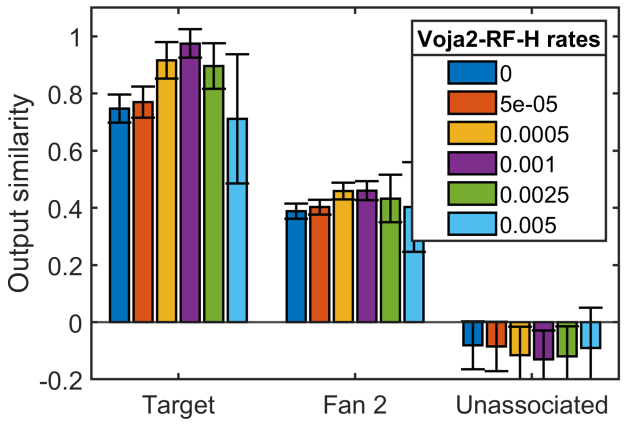


Figure C.4: Output similarities for various incrementing Voja2-RF-H learning rates. Bias percentage (ρ) is set at 10%.

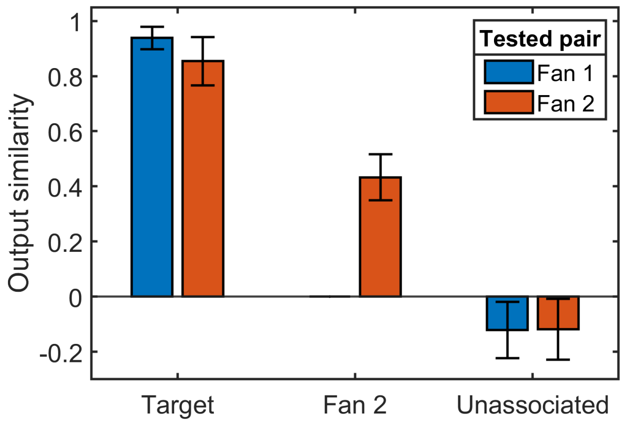


Figure C.5: Output similarities split by presented word pair type for a learning rate of 0.0025 and a bias percentage of 10%.

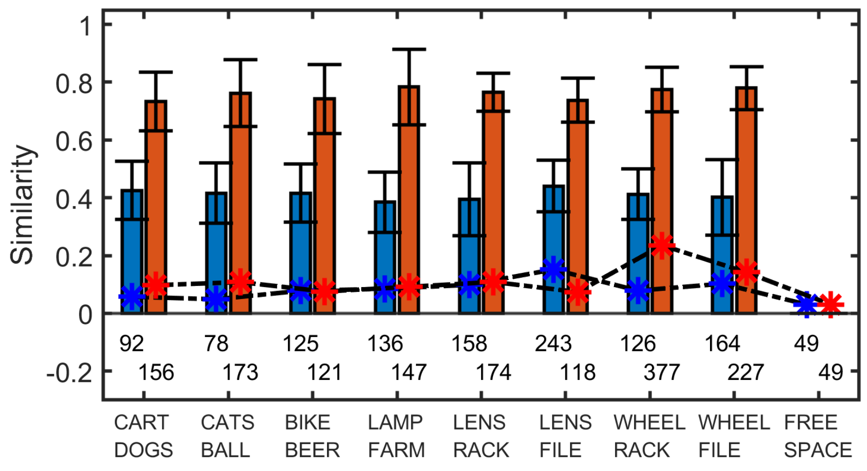


Figure C.6: Encoder position similarity to inputs after training using a learning rate of 0.0025 and a bias percentage (ρ) that is set at 10%.

Appendix D
Neural activity for a bias percentage of 10%

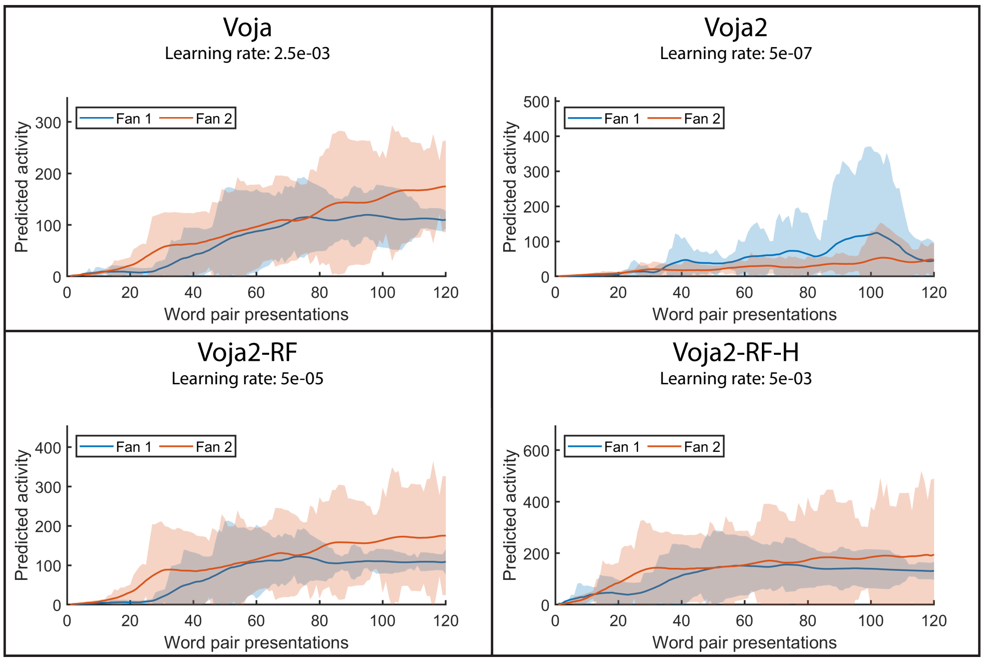


Figure D.1: Changes in predicted neural activity during training using the four different encoder learning algorithms. The adaptations of the original Voja learning rule all employed a bias percentage of 10q%. Predicted neural activity is calculated according to Equation 3.1.