BACHELOR THESIS
ARTIFICIAL INTELLIGENCE

Radboud University

Unsupervised Learning of Compression in the Semantic Pointer Architecture

Author:

Sebastian Pack
S1037899

Supervisor:

Dr. S. Thill
Artificial Intelligence
serge.thill@donders.ru.nl

Second reader:

Dr. L.P.J. Selen
Sensorimotor Neurosciencec
luc.selen@donders.ru.nl

3 February 2023

Abstract

The Semantic Pointer Architecture for spiking neural networks has proven to be a biologically plausible model of cognition that can be practically applied for many tasks while having some advantages over deep neural networks. However, compression of information which serves as the backbone of this architecture, has previously only been learned in biologically implausible ways. Since semantic pointers are meant to form clusters, we hypothesize unsupervised learning of compression to be possible by using a clustering metric as a learning signal. To make learning easier, it is first explored how to minimize the negative impact of circular convolution, a common operation in this architecture, on separability of data. For this, many vectors are drawn from normal distributions with different parameters and then convolved with the same dataset. The silhouette score is taken after convolution to find where convolution had the worst effect on the score. This is also done with vectors generated by Nengo. For an experiment testing the unsupervised learning hypothesis, two simple networks are created in Nengo that take a linearly separable dataset and compress it in multiple stages. The output is sampled, clustered using k-means and the silhouette score is inverted to be fed as an error signal. The convolution experiment shows that Nengo's vectors are not optimal to preserve separability when convolving with data. Unsupervised learning of compression failed. It is hypothesized that this approach might work if local learning signals were derived instead of providing one global signal for every population.

Contents

1. Introduction

1.1 Spiking Neural Networks

While Deep Neural Networks (DNNs) have been finding great success in many practical applications, there are still some major unaddressed issues that users and stakeholders must face. One such issue is the extreme amount of power consumption taken up by training DNNs on massive datasets (Canziani et al., 2016). This poses both an environmental and financial burden. Furthermore, a lack of biological plausibility exists in these networks that inhibit their potential to aid in neuroscience. No temporal dynamics are modelled in DNNs, where the propagation of signal from input to output are treated as one time step. Activation of neurons in DNNs are real-valued instead of binary as in the brain where a neuron only spikes or does not spike. And lastly, backpropagation is almost always used to learn connection weights, whose biological plausibility is questionable (Mazzoni et al., 1991; Stork, 1989). Another great issue is the lack of explainability regarding the inner mechanisms and choices of DNNs. This is especially problematic for AI systems in sectors where decisions impact security or health of people involved, such as air traffic control (Pack, 2022) or healthcare (Sunarti et al., 2021). It also makes improvement of AI systems hard as there is little possibility to inspect which aspect of the network requires improvement.

Spiking neural networks (SNNs) are a promising remedy for these shortcomings of DNNs. Firstly, the spiking neurons in SNNs have been shown to have much lower power consumption (Rueckauer et al., 2017) while holding more computational power (Maass, 1996) than their real-valued counterparts. SNNs also provide temporal dynamics by modelling synapses with differing delay times. That means signals take some time to propagate through the network as they do in the brain. Instead of backpropagation, SNNs also usually use learning rules determined to be more biologically plausible (Caporale & Dan, 2008; Voelker, 2015). Finally, explainability can be at least partially addressed in the form of the Semantic Pointer Architecture (Eliasmith, 2013) explained in Section 1.2.

Compared to the real values held by nodes in a DNN, neurons in an SNN have binary activation like neurons in the brain. The information is then contained in the rate of these spike trains. When going between numerical representations and spiking patterns we speak of encoding to and decoding from spiking patterns. The simplest example of this would be a single number encoded in a single neuron. One possible encoding of this number is for the neuron to spike at a rate proportional to this number, i.e., a higher number would result in a faster firing rate. The more neurons we choose to represent this number, the more we can factor out neuronal noise when decoding. This can be scaled up to vectors of any dimension by simply partitioning a population of neurons into parts that each represent one of the dimensions.

1.2 Semantic Pointers

In the Semantic Pointer Architecture (SPA), there are populations of neurons, also called ensembles, that encode the values of high dimensional vectors. These vectors, referred to as semantic pointers, are highly compressed representations of concepts. They are formed by taking representations of even greater dimensionality and compressing it into a lower dimensionality by passing it through multiple layers of neurons, which Eliasmith (2013) compares to the visual stream in the brain. Take for example a picture of a dog. At a resolution of 128×128 and three color channels, flattening this picture would result in a vector of almost 50,000 dimensions. In the SPA, this information passes through many ensembles that each serve to abstract and thus compress the information into less and less dimensions. At some point we have a neural ensemble encoding a semantic pointer, usually chosen around a dimensionality of 64, which represents an abstraction of the visual features of the dog. Pictures of other dogs should result in similar semantic pointers. The manifold holding all semantic pointers of visual dog features can then be considered as the compressed internal visual representation of not just one dog but the concept of a dog itself. For that reason, the space spanned by semantic pointers is also called the conceptual space.

A strength of semantic pointers is that we (and SNNs) can mathematically manipulate them in meaningful ways. The most common computations done with semantic pointers are addition, binding, unbinding and inversion. When taking a sum of pointers, each concept is represented simultaneously. This operation alone cannot result in pointers where multiple properties are matched with separate subjects. For example, the semantic pointers for the concepts black, brown, cat and dog can be summed to represent all concepts in one semantic pointer. However, it would hold no information about which of the subjects have which of the properties, in this example meaning we cannot determine which of the animals is black and which is brown. This problem in a more general sense is also commonly known as the binding problem (Treisman, 1996). The binding operation is introduced to address this issue. When binding two pointers, the resulting pointer will be a new one unlike the original two, at least in terms of the dot product. By binding with the inverse of one of the two pointers, they can be unbound again. This way, a network can be queried about its internal representations. In the following example, it is queried which of the subjects is brown, where $\circledast$ denotes binding and $brown^{-1}$ is the inverse of the semantic pointer for the concept “brown”.

$$brown^{-1} \circledast (cat \circledast black + dog \circledast brown) \approx dog$$

This ties back to the explainability advantage of the SPA mentioned in Section 1.1. We can take any vector encoded in a neural ensemble and take it apart into its components by unbinding. Any semantic pointer can also be compared to a vocabulary of known semantic pointers to find out which concept was represented at any point and time in the network. This way, we can see

the inner workings of an SNN with this architecture more clearly than we could in a DNN, where we can only interpret the output.

Another use of the binding operator arises from the fact that semantic pointers can hold not only compressed content but also context. For example, when performing multimodal integration, the pointer holding the compressed representation of the modal content can be "tagged" as coming from that mode by being bound to a contextual pointer. Eliasmith's (2013) demonstrates this idea by defining the following pointer for perceptual features of a robin:

$$robinPercept = visual \circledast robVis + auditory \circledast robAud + tactile \circledast robTact + \cdots$$

Here, the mode tag (e.g. $visual$) exists to give context to the pointer to the content (e.g. $robVis$) being bound to it. This is useful to break a concept down to its components, such as making a network output its internal visual representation of a concept. These tag vectors are usually randomly generated, such as in the Nengo python library (Bekolay et al., 2014) also used in this paper.

A final thing to note is how this binding operator is mathematically implemented. While there exist several implementations, in the SPA circular convolution is used. This is a mathematical operation between two vectors where first a Fourier Transform is applied on both vectors. Then, element-wise multiplication is performed on the resulting vectors. Finally, the Inverse Fourier Transform is applied to acquire the final vector.

To summarize, we can encode high dimensional vectors in spiking neurons by having spiking rates proportional to the vector's values. We use these vectors to represent highly compressed and abstracted concepts. The vectors serve as manipulatable symbols that can be bound and added to represent many concepts at once and their relations to each other. Multimodal integration can be explained as a special case of this where we bind compressed content pointers to "modal tag" pointers and then taking the sum.

1.3 Prior Research

This section serves to point out some of the important prior work done on SNNs and the SPA. Understanding what has been achieved so far places this thesis into its proper context.

Spaun is an exemplary network built to showcase the SPA (Eliasmith, 2013; Stewart et al., 2012). It proved that using the SPA, one can build a single network that can perform multiple different tasks, which is unlike the usually highly specialized DNN applications. It did so by having action selection neurons control the flow of information based on which task was queried, which happened via visual input. In this network the synaptic weights between the

neural ensembles that compress information into semantic pointers are analytically derived. The reader is left with the open question of how these weights could be derived through learning.

Efforts to combine the practical advantages of backpropagation and biological realism of SNNs led to a network by Hunsberger & Eliasmith (2015). Backpropagation relies on differentiation of signals in the network. This poses a problem when trying to use backpropagation in SNNs as spike signals are not differentiable. To get around this issue, a DNN was first trained on image classification using backpropagation. Then, that network's parameters were transferred over to an SNN. This proved to be successful by achieving the highest result on the CIFAR-10 dataset at the time.

One year later, Lee et al. (2016) managed to essentially eliminate the need for transferring of weights from DNNs to SNNs to use backpropagation. Instead, the problem of non-differentiable spikes was addressed by transforming them into a mostly continuous signal. This transformation was done numerically outside of the network. Their results were also able to compete with the best performing DNNs.

The SPA has also been successfully applied to neuromorphic hardware (Hampo et al., 2020), a kind of hardware that physically models the spiking neurons to avoid slow and costly software simulation. This way, power consumption can be drastically decreased, as mentioned in Section 1.1. An associative memory was built utilizing semantic pointers to learn and represent concepts in the memory. This was learned in a supervised manner where its outputs were compared to the desired outputs. The power consumption benefits allowed this to be used in a real exploring robot.

1.4. Gap and Research Question

An important observation is that none of the applications mentioned learned the computations necessary to compress information into semantic pointers in biologically plausible ways. The first example avoided learning in the SNN entirely by transferring weights from a DNN. The second network required computations outside of the network to transform the spike signals and used the biologically questionable backpropagation algorithm. Supervised learning was used in the third example by computing an error based on the given desired output, which must be provided externally. Of course, all of this research has been focused on useful behavior of SNNs where it makes sense to leave out this not well-researched detail. But, since each of these networks operate based on semantic pointers, the aspect of learning how to form these pointers is also essential. Without it, none of them could exhibit the useful behavior. We argue that understanding how compression can be learned in a biologically realistic manner is just as important as researching SNN applications. Modelling this learning process means leveraging one of the most essential advantages of SNNs and the SPA; the fact that they provide a bridge

between realistic brain models that help us understand cognition and AI systems that are useful in real world applications.

In order to address this gap in SNN research, we want to find an unsupervised learning method for compression. The Prescribed Error Sensitivity (PES) rule provides for a learning rule usable in SNNs that react to error signals (Voelker, 2015). These error signals are usually derived from external information for supervised learning, as it was in the associative memory example mentioned in the last section (Hampo et al., 2020). However, if we derived some error signal without using any external information, we could use this rule for unsupervised learning as well. Since we want to learn compression, we need an error measure based on how well the network is performing compression based only on internal information.

In Section 1.2, we mentioned that concrete examples of the same concept should produce semantic pointers similar to each other. Semantically different concepts should instead have greater distance between them (Eliasmith, 2013). In other words, a desired property of encoded vectors after compression is that the compressed vectors should form well-defined clusters, where each cluster is one concept. Thus, a measure of cluster quality should also capture some of the quality of compression. Since this is a relatively straightforward computation that does not require supervision, we argue that using such a metric to learn compression is more biologically plausible than the learning methods used in the prior work mentioned in the last section. In order to investigate whether this learning is possible, we run an experiment on two SNNs that use a cluster quality measure as an error signal for the PES rule to learn compression. For this proof of concept, the score will be computed externally instead of using neurons but, based on the complexity of tasks already solvable by SNNs, we believe this should be possible internally as well.

Before this, it is important to recall the binding operation explained in Section 1.1. Being done in the frequency domain through transformations, this is a highly non-linear operation. Therefore it is not easy to predict the impact this has on clustered data. We want to make sure that shifts in the distribution of semantic pointers caused by binding minimally interfere with the network's efforts to cluster the pointers. The only aspect of the binding operations we can control, given that we do not use a different mathematical implementation, is the vector we bind with. In our networks, binding only happens in one part of the network when a tag vector gets bound to a semantic pointer. We thus run statistical experiments to determine a tag vector that least interferes with the separability of the semantic pointers bound to it.

To reiterate, we have two questions: Can we characterize tag vectors that decrease cluster quality when convolved with clustered data in order to avoid them? Furthermore, can we learn information compression in SNNs through unsupervised learning on a cluster quality metric of semantic pointers?

Mode	1	2	3	4
STD	0.3	0.1	0.5	0.45

Table 2.1: Standard deviations used for each mode in the synthetic dataset

Mean	-10,000	-1,000	-500	0	500	1,000	10,000

Table 2.2: Means used in the tag generation

STD	20	500

Table 2.3: Standard deviations used for tag generation

2. Methods

We search for tag vectors that minimally interfere with separability after convolution. In another experiment, we feed a cluster quality measure to learn compression in an SNN. For both experiments, the cluster quality measure used is the silhouette score, as it provides a simple singular metric that captures how dense each cluster is and how far apart they are. Instead of relying on real class labels in these computations, k-means clustering with $k = 2$ is used to find labels in an unsupervised manner.

2.1 Tag Experiment

Firstly, it is investigated which effects circular convolution with differently distributed tag vectors has on the separability on a dataset. A two-class toy multimodal dataset is created, which could be thought of as the perceptual features of two different animals like the robin example in Section 1.2. It contains 100 eight-dimensional vectors per four modes. 50 vectors are chosen to be in class c_1 and 50 in class c_2. For each mode separately, there are four dimension indices picked at random to belong to set A, the rest belonging to set B. When a pair of four vectors in c_1 is generated, values are drawn from a normal distribution with a mean of 1 for dimensions in A and a mean of -1 for dimensions in B. The standard deviation is unique for each mode to introduce some statistical differences between each one (see Table 2.1). For vectors in class c_2, the means are simply swapped. This means that in each dimension where one class contains high values, the other will contain lower ones, making the dataset form two well-defined clusters.

Tag vectors are generated as described by Algorithm 5.1. One tag is generated for each of the four modes. Relating back to the example in Section 1.2, these would be the vectors $visual, auditory,$ etc. The tags are generated following gaussian distributions. Seven different means and two standard deviations are considered for this, leading to 14 sets of tags (see Table

2.2 and Table 2.3). Each data vector is bound to the tag corresponding to its mode. Then, for each multimodal datapoint, the modes are added together. This means each datapoint goes from a dimensionality of eight features by four modes to just eight features. In the robin example, this step would be the summation to obtain the vector $robinPercept$. A cluster quality measure is then computed for the resulting 100 vectors. This process is repeated for each of the 14 sets of tags to inspect which performed best. This entire procedure is in turn repeated for 1,000 trials, each time with a newly generated dataset and tags. This is done to make sure results are robust against noise of any individual trial. This algorithm is also run on a set of tags generated by the default semantic pointer generation in Nengo, again 1,000 times. We can then inspect which mean and standard deviation for tag generation performed best on average and how Nengo's generation compares to each of these.

2.2 Unsupervised Learning Experiment

For the unsupervised learning experiment, we generate a six-dimensional dataset in a way that ensures linear separability between the two classes c_1 and c_2. This dataset is unimodal to provide the simplest circumstances for our experiment. Again, half of the dimensions are picked at random to form set A, with the remaining three dimensions forming set B. Each time a vector is generated, it is randomly decided which class it belongs to. If it belongs to c_1, values in the dimensions in set A are generated uniformly in the interval $[0.4, 1]$. Values for the dimensions in set B are uniformly generated in the interval $[-1, -0.4]$. If the datapoint belongs to c_2, the intervals are swapped so that A holds the smaller values and B holds the larger ones. Linear separability is ensured as there is clearly no overlap in the interval of possible values. Furthermore, the interval of $(-0.4, 0.4)$ is left completely unpopulated so that even with noisy neural representations of these vectors there is still no overlap.

For the implementation of our networks, we chose the Nengo python library, as this library was made specifically with the SPA in mind (Bekolay et al., 2014). The structure of Network N_1 can be seen in Figure 2.1. All neurons in this network were chosen to use the Nengo standard and widely popular Leaky-Integrate-And-Fire-Neurons. The stimulus node shows an input for 2 seconds until moving on to the next. This input is received by Ensemble 1, which is an ensemble of 1,500 neurons representing a six-dimensional vector. The decision of using 1,500 neurons is a somewhat arbitrary safe bet based on the observation that 50 neurons per dimension will give a reasonably stable representation of a vector (Eliasmith, 2013). We chose a number much higher to ensure that results are mostly independent from neuronal noise, without using Nengo's Direct mode which would use no neural approximations whatsoever. Information is then passed to a second neural ensemble. This ensemble represents only three dimensions and has half as many neurons, to serve as a compression stage towards the semantic pointer. What follows is a Nengo "State" object, which represents a semantic pointer using two neural populations. Here, the default parameters were used for neuron count, which uses the

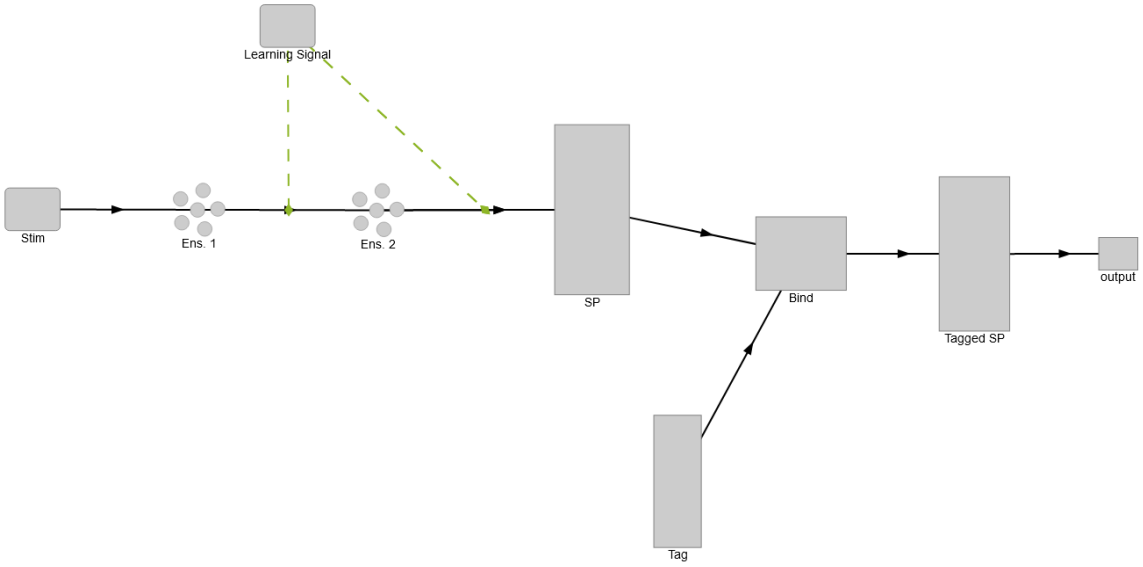

Figure 2.1: Layout of Network N_1

aforementioned 50 neurons per dimension rule. We have set the dimensionality of all semantic pointers in this network to 16. Note that this means the semantic pointer has more dimensions than the deep semantics, i.e. the input stimulus, which would normally not be the case. This is further discussed in Section 4.2. The semantic pointer is now bound through circular convolution with a tag vector that was randomly generated at the beginning. The output node merely exists to pass the tagged semantic pointers to the function computing the clustering metric, which is implemented outside of the network. We realize this takes away from the biological realism, but the focus here lies on whether our learning method is possible in the first place, regardless of implementation. To get a detailed look into how each computation in the network affects the cluster quality of the data, the vectors encoded by each ensemble along the way are sampled for further analysis.

The metric used here is identical to the one in the tag experiment, using silhouette scores on k-means partitions. The tagged semantic pointer is saved 1 millisecond before each time a new input is shown to the network. The cluster metric is then computed over the last 10 sampled pointers. To achieve an error signal from this, the Silhouette Score is subtracted from 1, its theoretical maximum. This is then fed to the connections between the first two neural ensembles, and from the second ensemble to the semantic pointer.

Also tested is a network N_2 with the same structure as in Figure 2.1, modified to include a third ensemble “Ensemble 3” after Ensemble 2, which serves as another compression layer to see if more layers are beneficial for the learning process. The connections to and from this ensemble naturally also receive the error signal. This ensemble has 500 neurons and represents only a one-dimensional value.

We also implement an alternative learning method. Since there are now synaptic weights between several different neural ensembles updated using the same error signal, there is a chance that one ensemble's learning may interfere with the others. For example, Ensemble 1 might respond to an increase in error actually caused by Ensemble 2. To address this potential issue, in the alternative approach, the error signal is only routed to the connections between one ensemble pair at a time. While this is happening, the other connections receive an error of 0. Every 40 seconds, the error signal is then routed to the next connection. We will refer to this approach as cyclic learning.

In summary, two networks are trained in two ways each. N_1 is as depicted in Figure 2.1, while N_2 has one additional ensemble after Ensemble 2. First, all learning connections receive the same error signal, which is comprised of a negative cluster score. Then, the connections are randomly initialized and trained again with the learning signal cycling between the connections to minimize learning interference.

3. Results

3.1 Tag Experiment

1,000 trials of circular convolution were done between the first dataset and tags spread across several means. This was repeated for a standard deviation of 500 and 20 and tags generated by Nengo. For each trial, the best performing tag mean was recorded, visible in Figure 5.1 in the appendix. As can be clearly seen, the greatest separability after convolution with a tag was most often achieved when the values in that tag were centered around a mean of zero. This difference is even more pronounced when standard deviation of the tags is lower. In Figure 5.2 we see silhouette scores for a mean of 1,000, which confirm that choosing a mean far from zero leads to much worse separability.

For each of the 1,000 trials per tag variance, the highest silhouette score among the seven tag means was also recorded, along with the 1,000 silhouette scores on Nengo's built in semantic pointer generation. With large standard deviation in the tags, the silhouette scores are on average larger and variance is smaller (see Figure 5.3). While convolution with tags generated by Nengo yielded similar separation on average, there are clearly farther outliers in the scores compared to both tag generation methods, when considering the best tag of each trial. In some rare cases, convolution with tags generated by Nengo can even result in the silhouette score dropping to around 0.35.

These results show that performing circular convolution of a dataset with one vector can lead to loss of separability for a poorly chosen vector. We characterize a poor vector as having a mean

far from zero and too little variance. Results also suggest that there is room for improvement of Nengo's semantic pointer generation when they are intended to be used as tags.

3.2 Unsupervised Learning Experiment

Two SNNs were given linearly separable data and trained to compress this data into semantic pointers. One network N_1 had two layers of compression while the other network N_2 had three. The unsupervised learning used an inverted silhouette score based on a k-means clustering of the semantic pointers sampled at regular intervals. Each network was trained once with all learning connections active at once and another time with the learning signal cycling between the connections every 40 seconds. The learning signal for N_2 with cyclic learning can be found in Figure 5.5. Unfortunately, during the 1800 simulated seconds there is no visible decrease of this error signal. The learning signals for the other conditions are extremely similar and thus omitted. For Figures 5.6 to 5.9, silhouette scores were computed for the decoded vectors of each ensemble. Each score was obtained by clustering over a window of 120 seconds, each containing 60 sampled vectors decoded from each population. The populations clearly fluctuate in the silhouette scores of their vectors, but unfortunately they show no clear upwards trend.

It is worth noting that in both runs of N_2, the silhouette scores show a significant increase after compression into Ensemble 3. Since this difference exists even in the beginning where weights are still random, this could not be a positive effect from the unsupervised learning. One might conclude from this that the silhouette score taken after k-means clustering is positively biased towards lower dimensional data. This would however not explain why Ensemble 1 has consistently higher scores than Ensemble 2 and Ensemble 3, despite encoding the highest dimensional vectors of the three, ignoring the small timeframe at around 1200 seconds in Figure 5.8. Seeing the inconclusive evidence and scope of this thesis, follow-up research would have to be done to find the cause of this effect.

Another interesting effect can be seen in Table 5.1, where for each ensemble silhouette scores are taken over all decoded vectors sampled over the entire training period. Compared to simultaneous learning, when cycling the learning connections there is a much larger silhouette score for the semantic pointers encoded in "SP". This happens in both N_1 and N_2. Since this effect is only observed when clustering all pointers over the entire training period but not when taking only the last 120 pointers, this could not mean that the learning resulted in better separability. The fact that this effect is not observable with simultaneous learning also rules out that the cause is only related to clustering with more datapoints, since the same number of samples is taken for both learning regimes. A possible cause could be that simultaneous learning causes the conceptual spaces to shift more dramatically than cyclic learning. This shift might cause that, for example, points belonging to class c_1 in the first half of training end up in the space inhabited by points of c_2 in the second half, leading to bad cluster quality only when

considering the whole timeline. This is only a theory as results here are also too inconclusive to make a strong statement.

In conclusion, we did not achieve unsupervised learning of compression based on a clustering score assigned to the final compressed vectors.

4. Discussion

The thesis was meant to fill the gap of a missing biologically plausible learning method for compression to form semantic pointers in SNNs by learning on a cluster metric. It also investigated effects of binding on cluster quality. It was found that when binding with semantic pointers generated by Nengo, in some cases separation that was present in the input is almost completely lost. A theoretical existence of a better tag generation method than Nengo's generation was shown. Trying to learn compression by means of unsupervised learning on a clustering measure failed to reduce loss within 1,800 seconds. An unknown effect caused cluster scores to increase after the second compression layer in network N_2. Scores computed over all samples were much higher in the untagged semantic pointer when learning connections were cycled over time.

4.1. Limitations of the Tag Experiment

The findings related to tag quality come with some major limitations. One of the limiting factors is the synthesized dataset used for the experiment. It contains two normally distributed classes and different modalities are only distinguished by different standard deviations. Of course, one could never expect to find these conditions with any data measured from the real world. Datasets that follow non-gaussian distributions or have noise and outliers could lead to very different results where Nengo's semantic pointer generation might fare better.

Furthermore, the results visible in Figure 5.3 did show farther outliers and variance for Nengo's tags compared to our method, but this comes with a caveat. These results are taken from the best performing tag mean of each of the trials, which was zero in most cases but not all. Figure 5.4 shows that taking only the results of the tags generated with a mean of zero, the tags no longer perform significantly better than Nengo's tags, at least for 1,000 trials. Thus, this experiment does not provide a better method for generating SPA tags, but instead shows that one theoretically exists. After all, the scores in Figure 5.3 are computed on the same datasets as the scores for Nengo. That means the outliers in Nengo's scores cannot be solely caused by the datasets in those trials, but have to be caused by Nengo's generation. What Figure 5.2 additionally highlights is that tag generation can be much worse when not being careful with the choice of distribution, meaning that preservation of separation after circular convolution cannot be taken for granted.

It is important to stress is that preservation of separation is just one desired property of tags and conceptual spaces. The measures we used here to quantify performance of the tags and networks is naturally far from perfect by only capturing this single aspect. Even regarding this aspect, only one specific measure and clustering algorithm were used. That was done to adhere to the scope of the thesis, especially considering the additional simulation time that would have resulted from computing multiple clusterings and quality measures during the simulation. Using other methods to determine separability could lead to entirely different results and our findings do not exclude the possibility that Nengo's tags have other desirable properties that our tags lack.

What the statistical test for the tags also does not consider is that in an SNN, there will be neuronal noise and more importantly, populations are optimized for the specific range of numbers they should represent in their encoded vectors. Essentially, neurons have a biologically dictated maximum firing rate (Wang et al., 2016) and a minimum firing rate where they become slow enough to simply not be useful anymore. When building an SNN in Nengo, one must choose which numbers those corresponding maximum and minimum rates should represent. The greater the numerical range, the greater the fluctuations will be in the encoded vectors due to neuronal noise. When computing circular convolution, if one of the vectors has high variance, the resulting vector will also span a larger numerical range, meaning that this would have to be accounted for in an SNN. Therefore, there is a possibility that the resulting increased effect of noise would nullify the findings seen in Figure 5.2 for real SNN scenarios.

4.2. Decompression in our Networks

The SNN experiments were done on a low dimensional dataset to prevent simulation time from becoming infeasible for this project, as Nengo runs on the CPU and can take several minutes of real time for a second of simulated time. That leaves us with the problem that, after compression of this low dimensional input, the semantic pointer would be even lower dimensional, in this case having just one dimension. A useful property we would lose by doing this is that high dimensional spaces make it easier to separate data (Gorban et al., 2016). That is why we decided to decompress to 16 dimensions before the ensembles named "SP", making the resulting semantic pointer higher dimensional than the input. Since semantic pointers are meant to be compressed representations of input, they are usually lower dimensional than the input. We argue that breaking this convention here is fine, as the primary focus of the experiment lied on learning the compression itself, which the networks still had to perform to first go from six dimensions down to three dimensions (or one dimension in N_2). The aim was to test this under the easiest conditions first, which is why a decompression after the compression was present to leverage benefits of higher dimensional spaces.

4.3. Outlook

We have found properties of tags to avoid in the SPA which are useful to know when generating them manually. Better tag generation than what Nengo provides is shown to exist, though not provided. The lack of biologically plausible learning methods for compression in SNNs was unfortunately not eliminated as our method failed. Despite this, we can learn from these results and conduct further research on this idea. We hypothesize the reason for our method's failure to be that a global cluster quality measure taken at the last ensemble is not local enough to the ensembles earlier in the networks. The cyclic learning method was meant to address this issue partially by preventing the PES rule from changing weights of one connection based on a change in error caused by another. However, it is hard to say whether an improvement in compression between the early ensembles would even still be captured in the final silhouette score taken after all the non-linear computations leading up to the last ensembles. One could instead feed to each connection a clustering measure taken from the population after that connection. For example, the connection between Ensemble 1 and Ensemble 2 would receive a score measured from Ensemble 2, which is unaffected from the non-linear computations later in the network. This would give a much clearer error signal to the learning rule and might make it possible to learn compression in this way after all.

5. Appendix

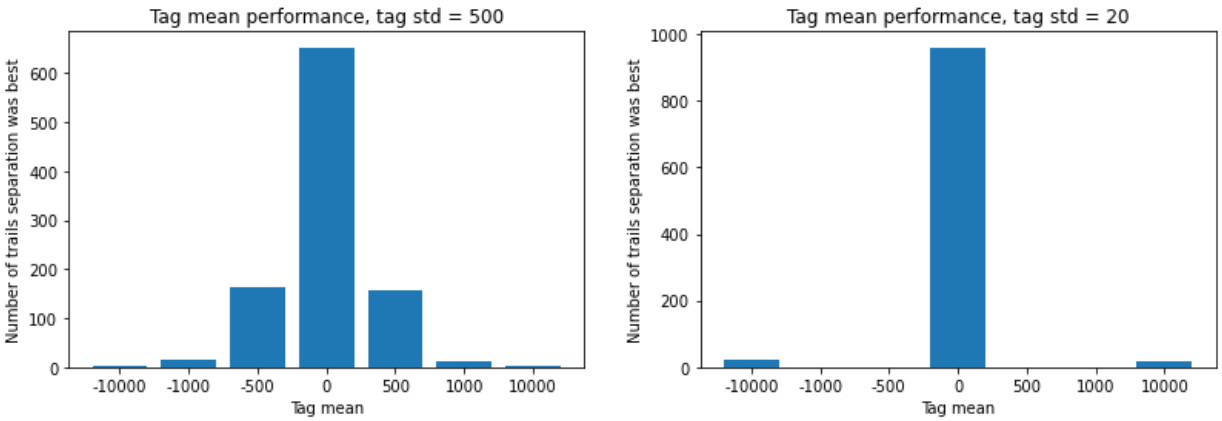

Figure 5.1: Recording of the tag means that achieved highest silhouette score after convolution for each of the 1,000 trials.

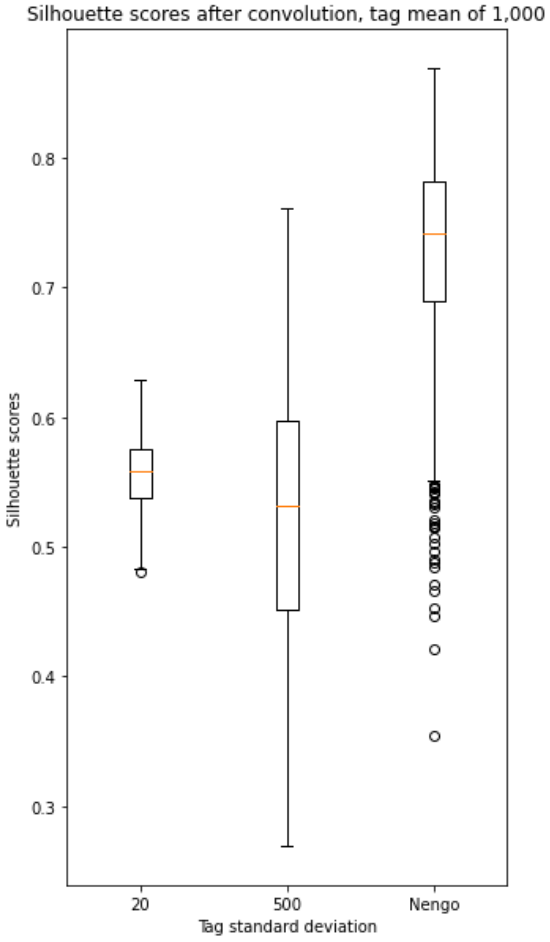

Figure 5.2: Silhouette scores after convolution with a tag mean of 1,000.

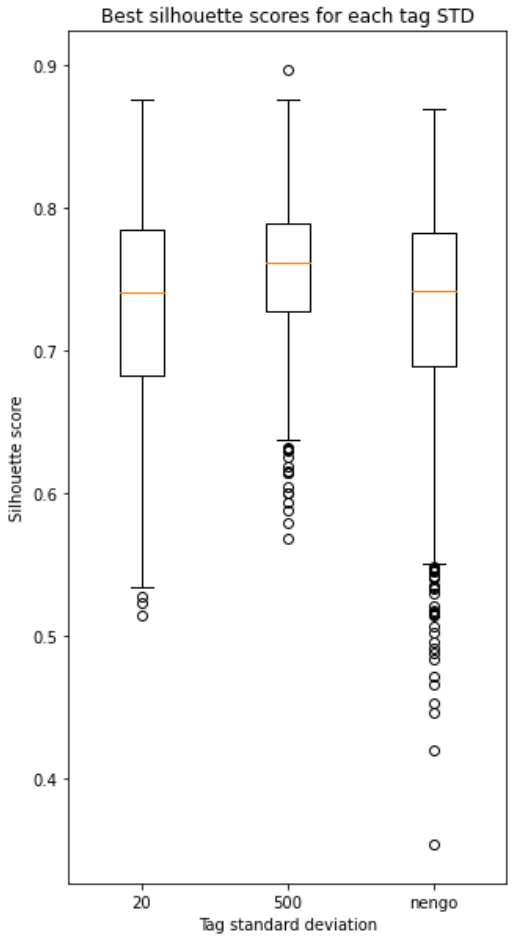

Figure 5.3: Highest silhouette scores out of each of the 1,000 trials per standard deviation used in tag generation.

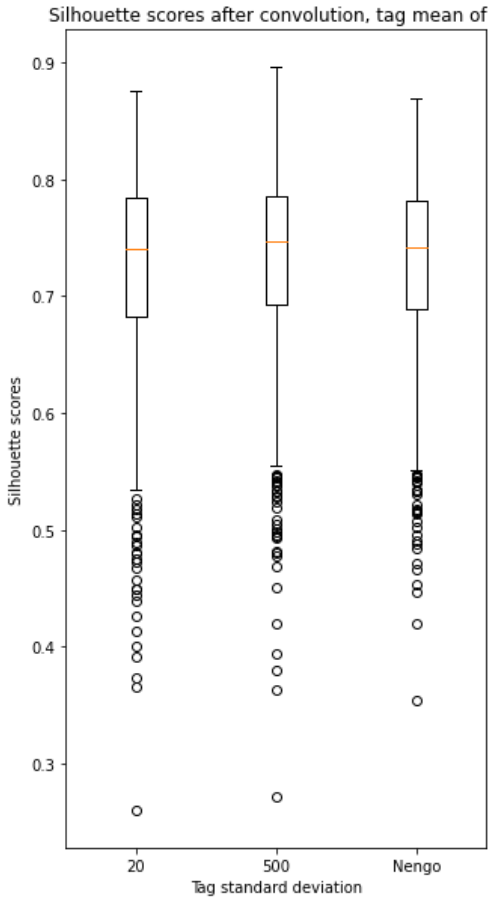

Figure 5.4: Silhouette scores after convolution for a tag mean of zero.

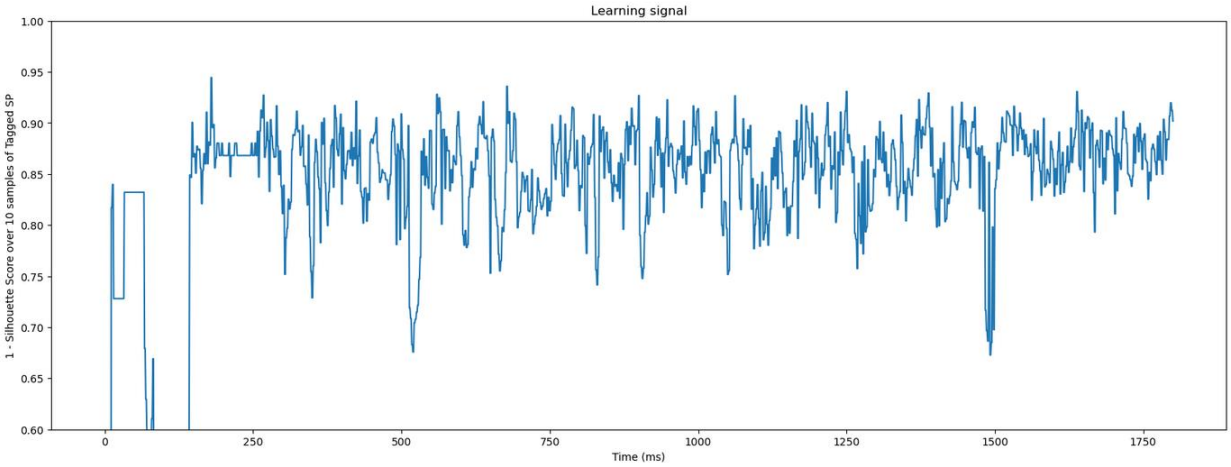

Figure 5.5: Training loss (silhouette score subtracted from its optimum) for N_2 with simultaneous learning.

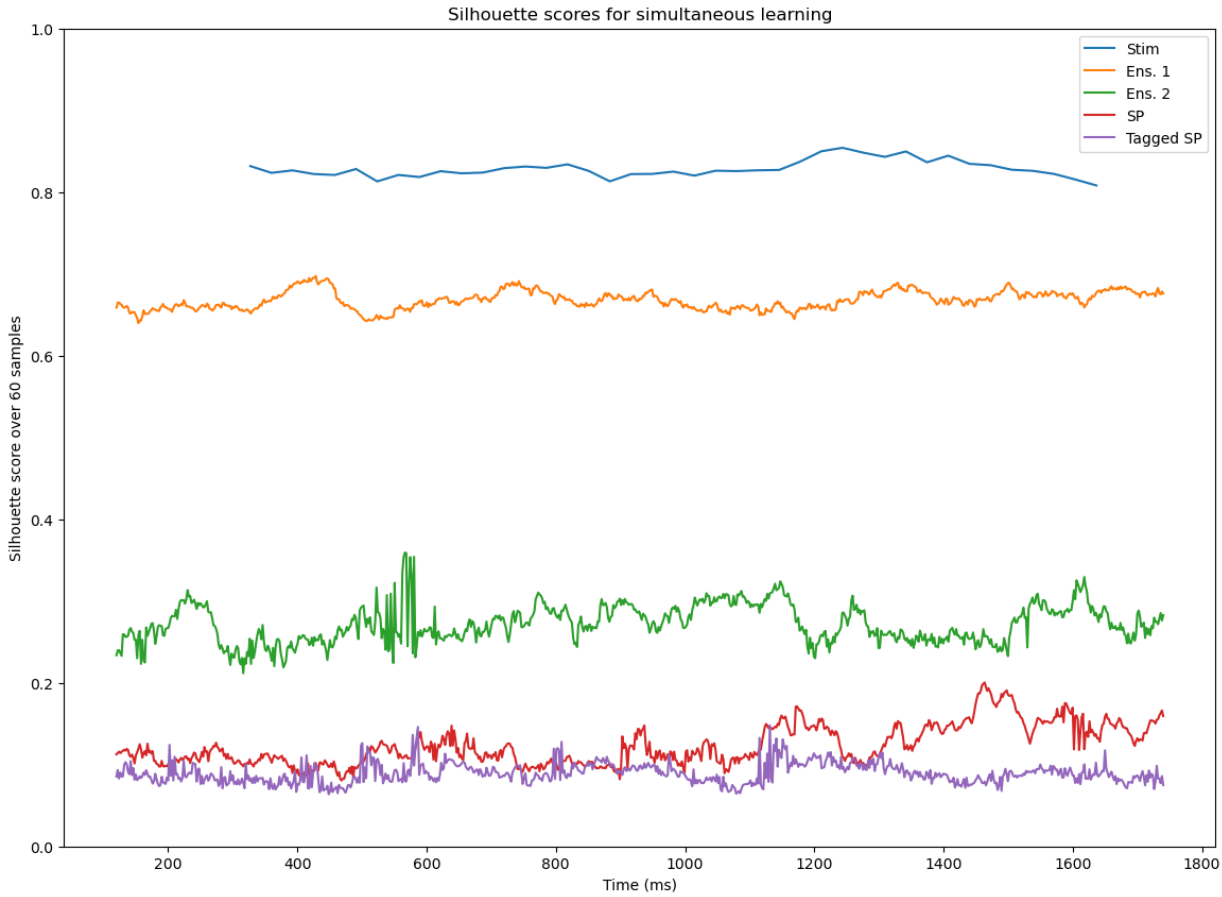

Figure 5.6: Silhouette scores for N_1 with simultaneous learning.

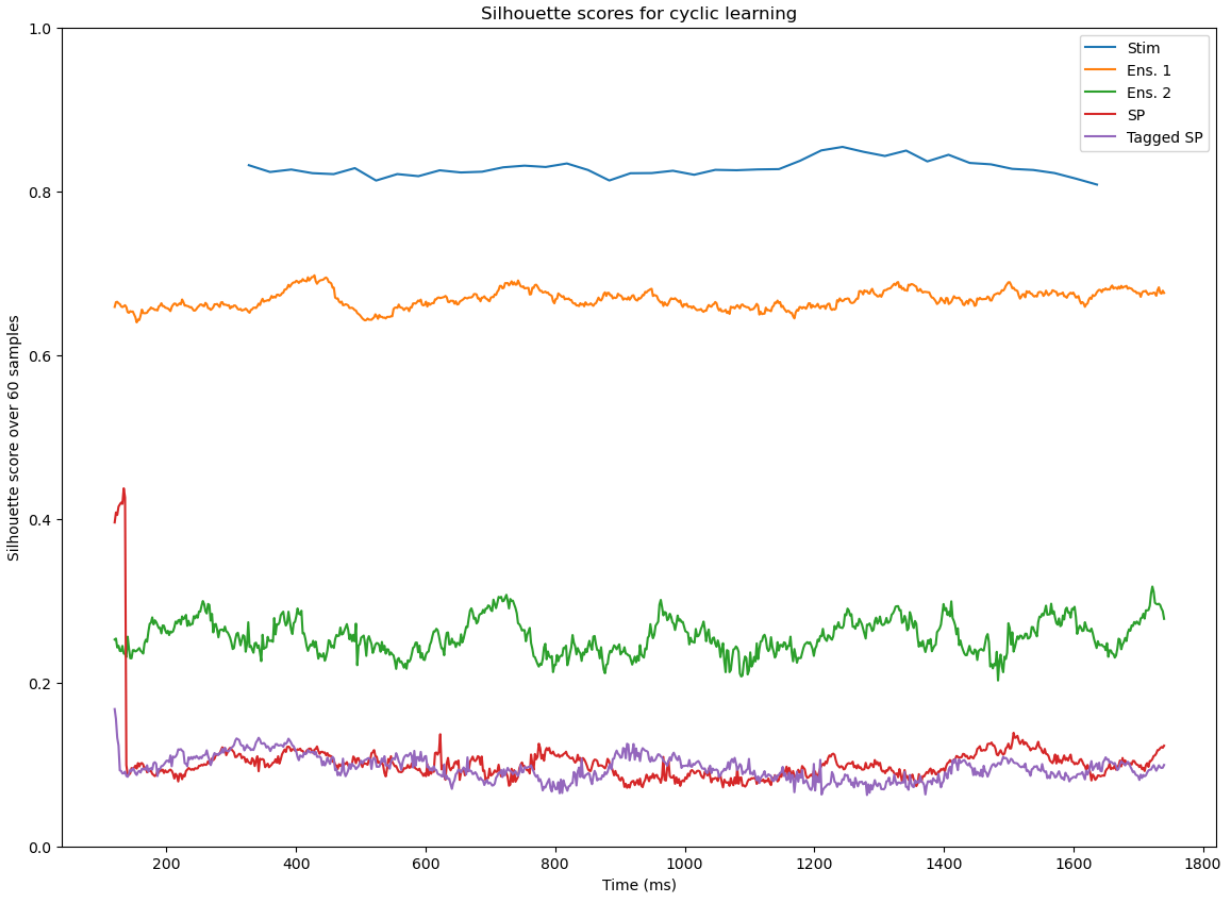

Figure 5.7: Silhouette scores for N_1 with cyclic learning.

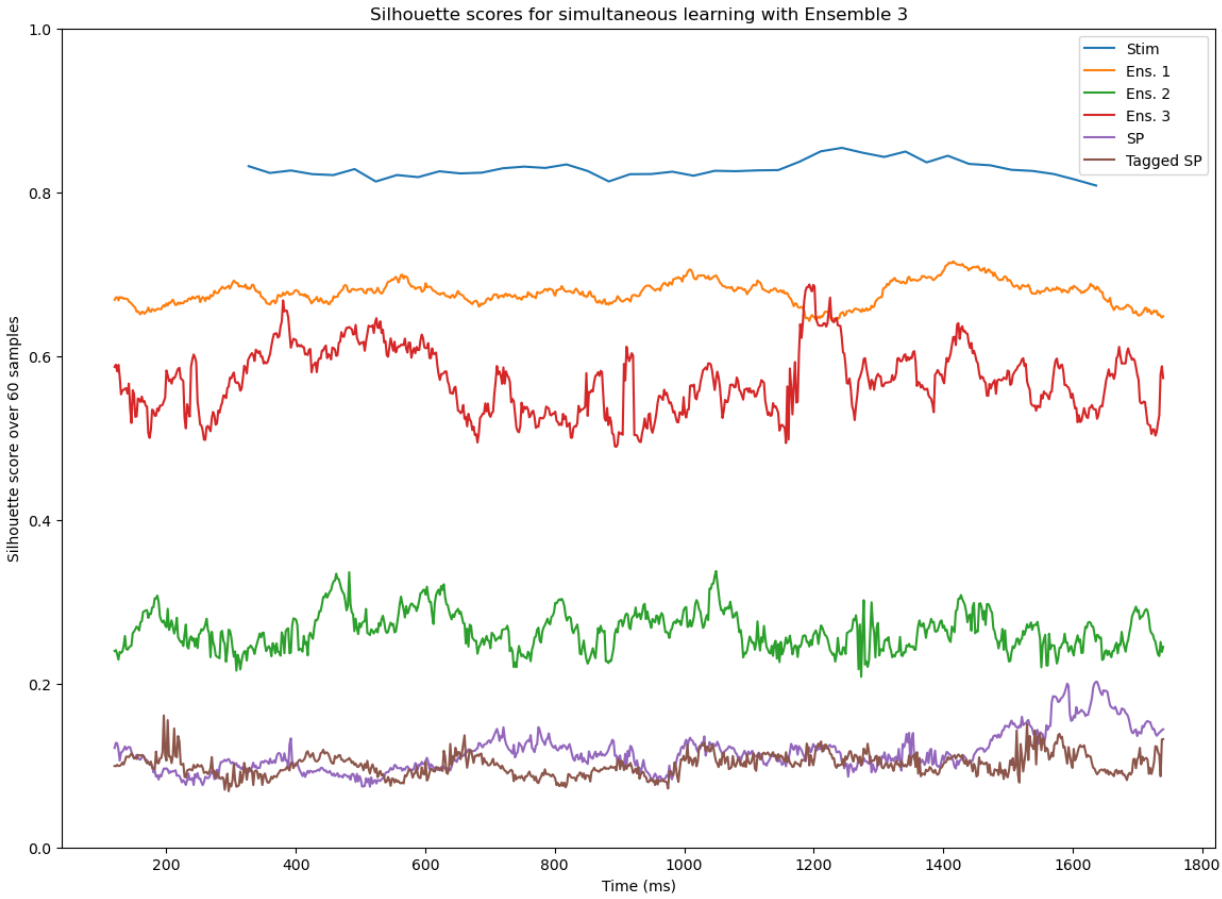

Figure 5.8: Silhouette scores for N_2 with simultaneous learning.

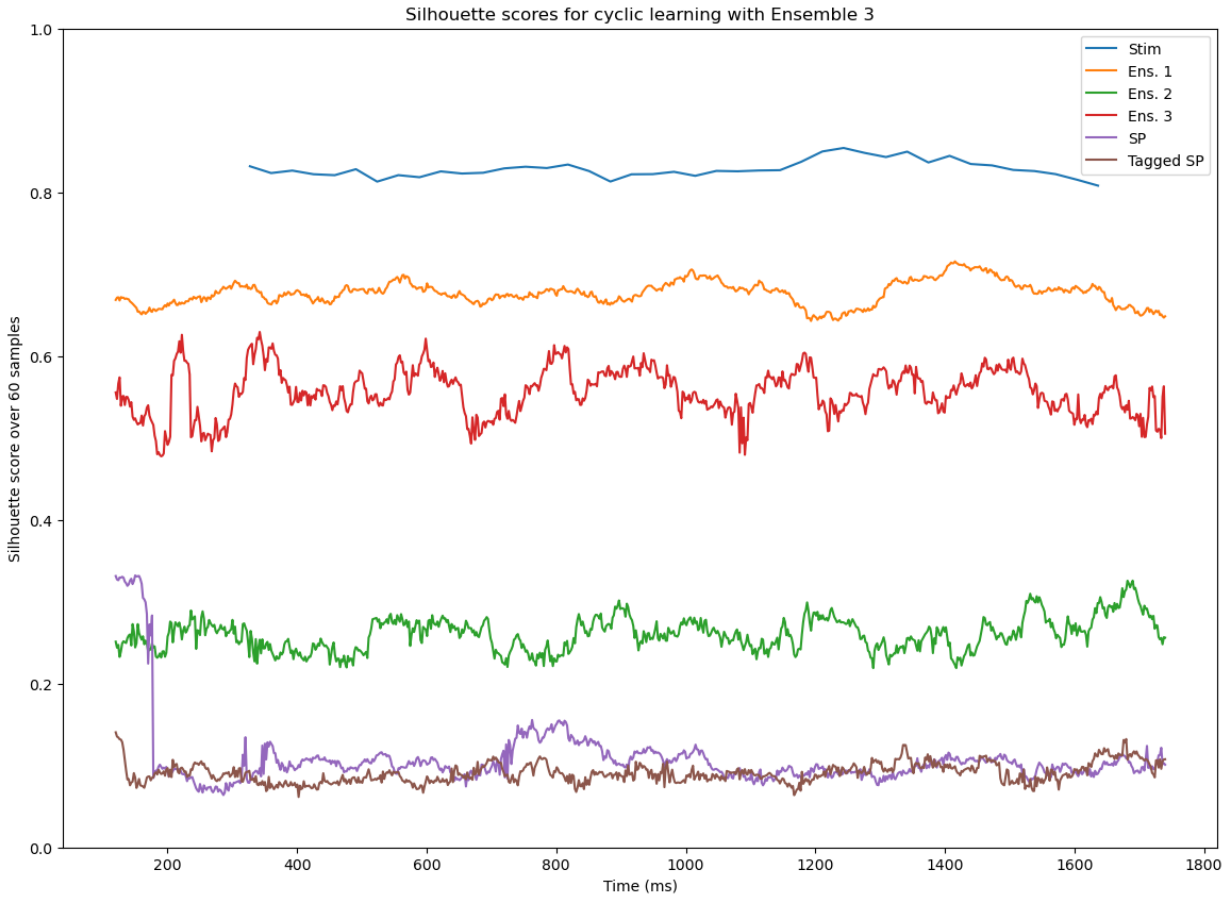

Figure 5.9: Silhouette scores for N_2 with cyclic learning.

Node	N_1 simultaneous	N_1 cyclic	N_2 simultaneous	N_2 cyclic
Stim	0.83	0.83	0.83	0.83
Ens. 1	0.76	0.67	0.67	0.67
Ens. 2	0.83	0.83	0.82	0.82
Ens. 3	N/A	N/A	0.53	0.82
SP	0.09	0.42	0.10	0.40
Tagged SP	0.07	0.08	0.09	0.07

Table 5.1: Silhouette scores taken for each node over all sampled decoded vectors.

```
repeat 1_000 times:
    for std in stds do:
        for mean in means do:

            for mode in 1,...,4 do:
                tag[mode] = gaussian(mean, std)
                for x[mode, i] in dataset do:
                    y[mode, i] = circular_convolution(x[mode, i], tag[mode])

            for i in 1,...,100 do:
                z[i] = sum from m=1 to 4: y[m, i]

            score = silhouette_score(z)
```

Algorithm 5.1: Tag generation algorithm, see Table 2.1 and Table 2.2 for “means” and “stds”

6. Bibliography

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T., Rasmussen, D., Choo, X., Voelker, A., & Eliasmith, C. (2014). Nengo: A Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, *7*. https://doi.org/10.3389/fninf.2013.00048

Canziani, A., Paszke, A., & Culurciello, E. (2016). An Analysis of Deep Neural Network Models for Practical Applications. *CoRR*, *abs/1605.07678*. http://arxiv.org/abs/1605.07678

Caporale, N., & Dan, Y. (2008). Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annual Review of Neuroscience*, *31*(1), 25–46. https://doi.org/10.1146/annurev.neuro.31.060407.125639

Eliasmith, C. (2013). *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199794546.001.0001

Gorban, A. N., Tyukin, I. Y., & Romanenko, I. (2016). The Blessing of Dimensionality: Separation Theorems in the Thermodynamic Limit**The work is partially supported by Innovate UK, Technology Strategy Board, Knowledge Transfer Partnership grant KTP009890. *IFAC-PapersOnLine*, *49*(24), 64–69. https://doi.org/10.1016/j.ifacol.2016.10.755

Hampo, M., Fan, D., Jenkins, T., DeMange, A., Westberg, S., Bihl, T., & Taha, T. (2020). Associative Memory in Spiking Neural Network Form Implemented on Neuromorphic Hardware. *International Conference on Neuromorphic Systems 2020*. https://doi.org/10.1145/3407197.3407602

Hunsberger, E., & Eliasmith, C. (2015). *Spiking Deep Networks with LIF Neurons*. arXiv. https://doi.org/10.48550/ARXIV.1510.08829

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training Deep Spiking Neural Networks Using Backpropagation. *Frontiers in Neuroscience*, *10*. https://doi.org/10.3389/fnins.2016.00508

Maass, W. (1996). Noisy Spiking Neurons with Temporal Coding have more Computational Power than Sigmoidal Neurons. In M. C. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in Neural Information Processing Systems* (Vol. 9). MIT Press. https://proceedings.neurips.cc/paper/1996/file/f93882cbd8fc7fb794c1011d63be6fb6-Paper.pdf

Mazzoni, P., Andersen, R. A., & Jordan, M. I. (1991). A More Biologically Plausible Learning Rule Than Backpropagation Applied to a Network Model of Cortical Area 7a. *Cerebral Cortex*, *1*(4), 293–307. https://doi.org/10.1093/cercor/1.4.293

Pack, S. (2022). *The Current State of AI in Air Traffic Control: A Categorization and Review*. Essay for course "AI in the Connected World" (SOW-BKI333), Radboud University. https://brightspace.ru.nl/d2l/common/viewFile.d2lfile/Database/NDU5NzQyOQ/Uses%20of%20AI%20in%20Air%20Traffic%20Control%20v2.pdf?ou=302357

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., & Liu, S.-C. (2017). Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. *Frontiers in Neuroscience*, *11*. https://doi.org/10.3389/fnins.2017.00682

Stewart, T., Choo, F.-X., & Eliasmith, C. (2012). *Spaun: A perception-cognition-action model using spiking neurons*. *34*(34).

Stork. (1989). Is backpropagation biologically plausible? *International 1989 Joint Conference on Neural Networks*, 241–246 vol.2. https://doi.org/10.1109/IJCNN.1989.118705

Sunarti, S., Rahman, F. F., Naufal, M., Risky, M., Febriyanto, K., & Masnina, R. (2021). Artificial intelligence in healthcare: Opportunities and risk for future. *Gaceta Sanitaria*, *35*, S67–S70. https://doi.org/10.1016/j.gaceta.2020.12.019

Treisman, A. (1996). The binding problem. *Current Opinion in Neurobiology*, *6*(2), 171–178. https://doi.org/10.1016/S0959-4388(96)80070-5

Voelker, A. R. (2015). A solution to the dynamics of the prescribed error sensitivity learning rule. *Waterloo: Centre for Theoretical Neuroscience*.

Wang, B., Ke, W., Guang, J., Chen, G., Yin, L., Deng, S., He, Q., Liu, Y., He, T., Zheng, R., Jiang, Y., Zhang, X., Li, T., Luan, G., Lu, H. D., Zhang, M., Zhang, X., & Shu, Y. (2016). Firing Frequency Maxima of Fast-Spiking Neurons in Human, Monkey, and Mouse Neocortex. *Frontiers in Cellular Neuroscience, 10*. https://doi.org/10.3389/fncel.2016.00239