university of groningen / faculty of science and engineering

Hard Clustering of ESN Dynamics Using Conceptors

Bachelor's Project Thesis

Joris Peters, s4001109, j.peters.13@student.rug.nl,
Supervisors: G.Pourcel & Prof Dr H. Jaeger

Abstract: For data analysis, explainability, and more, it can be of interest to identify groups and hierarchies within the activity dynamics of recurrent neural networks (RNNs). Using conceptors to represent RNN dynamics, this study aims to cluster the dynamics of Echo State Networks (ESNs), a variant of RNNs. Conceptors were computed from the ESN-response to phoneme recordings of the TIMIT dataset. These conceptors were then used to perform phoneme classification with evidences (supervised), clustering with an adaptation of K-means (unsupervised), and hierarchical clustering with average linkage (partly supervised). Conceptor-based phoneme classification reached a reasonable [In addition to classification, I show that conceptors may be used for clustering into groups and hierarchies.] accuracy and clustering produced groups and hierarchies that resemble existing phonetical taxonomies. I conclude that conceptors are well-suited to classifying and clustering ESN-dynamics and the time-series that induced these dynamics.

1 Introduction

Topic: Use conceptors to identify discrete symbols (clusters) within the dynamics of Echo State Networks (when driven with speech samples).

"Symbols lie at the root of intelligent action".

Biological and artificial cognitive agents benefit from internal representations of their environments; These representations form the playing fields to various cognitive functions like reasoning, creativity, attention, memory, and language. On the other hand, the usefulness of the resulting cognitive functions depends, to some degree, on the adequacy of the underlying representation. For instance, in recall tasks, a well-suited representation that chunks together related information (like adjacent numbers (Miller, 1956) or chess pieces (Chase and Simon, 1973)) can aid its short memorization. Likewise, in artificial agents, a good task representation can often simplify the task (...).

Traditional cognitive science frames the nature of cognitive representations as fundamentally symbolic. The hypothesis of Newell and Simon (1975) that humans, computers, and any other system capable of intelligence must be a physical symbol system, one that performs operations on symbols, has been widely adopted. Symbols are entities that represent other entities. Human speech may exemplify the symbolic nature of our cognitive representations. Phonemes are the mental representations of phones, the smallest still distinguishable units of speech sound. For example, the phoneme /t/ is a symbol of the English-proficient mind and the mental representation of the phones [t^{h}], [t], and [ɾ]*, each of which encompasses a subset of speech sounds. The phones that a phoneme represents are its *allophones*. Communication through speech is possible, among many other factors, for the speech representations of the communicators coincide; The representations of English speakers generally feature the same 44 phonemes representing the same set of phones with some variability across dialects. Note that phonemes are language specific; Every language uses a different set of symbols to represent its particular speech sounds. Thus, cognition seems to rely on symbolic representations.

Cognitive neuroscience attempts to explain how these cognitive symbols arise in sub-symbolic brains. From a sub-symbolic perspective, the brain is a net-

*Phonemes are broadly transcribed using a slanted bracket. Phones are narrowly transcribed by a square bracket. Phonic transcriptions will be written using the International Phonetic Alphabet (IPA).

work of neurons. When this dynamical system is stimulated, for example, by sensory inputs, patterns of neural firing result. These firing patterns may be considered symbols if they are associated with a specific entity. Symbol acquisition, then, is the process of reinforcing these firing patterns (through Hebbian learning) face to the repeated exposure to instances of the referenced entity. For example, when a child is first exposed to speech, every sound signal will trigger an arbitrary firing pattern. As the child repeatedly hears the phones [t$^{\mathrm{h}}$], [t], and [ɾ] in similar linguistic contexts, a single pattern of neural firing becomes associated with them. This pattern corresponds to the cognitive symbol, the phoneme, /t/. After symbol acquisition, the pattern is no longer tied to any specific instance but represents a whole subset among the possible speech sounds and sensory experiences in general). Notably, children learn to distinguish phonemes without external supervision; No explicit instructions are needed let alone possible since only later in development do phonemes become the building blocks of words and enable the understanding and production of language Maye and Gerken (2000). So, symbols correspond to sub-symbolic firing patterns acquired through unsupervised learning.

The field of artificial intelligence (AI) has sought to bridge between its symbolic and sub-symbolic representations. One such bridge is being built between Echo State Networks (ESNs) that are, like the human brain, networks of sub-symbolic units (neurons) and *conceptors* that offer a symbolic representation of the sub-symbolic firing patterns. I will proceed by introducing ESNs and then conceptors taking inspiration from Jaeger (2014). On the sub-symbolic end, ESNs are a type of recurrent neural network. As such, they implement a dynamical system; Their internal states are a function of time. This time-extended activity relies on cyclic connections to introduce memory into the system and enable the processing of time series like speech. However, ESNs distinguish themselves from other RNNs by their large size (many internal neurons), low density (low connections-to-neurons ratio), and random and untrained internal- and input weights (Yildiz, Jaeger, and Kiebel, 2012). Given these features, ESNs can have a capacity for performing a high-dimensional non-linear expansion of their input signals, that, under the right circumstances, uniquely *echo* of these inputs, paralleling how a brain may transform sensory inputs into characteristic neural firing patterns. Figure 1.1 depicts a typical ESN as it is driven by an input sequence u, possibly a speech sample, and elicits a high-dimensional response x, the sequence of internal states of the ESN's reservoir. An output layer is sometimes added to ESNs for (re)productive purposes but omitted here since only the internal states of the network will be of interest for the current study as will become apparent shortly.

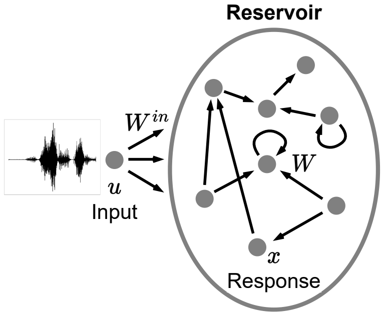

Figure 1.1: Sketch of an ESN driven with an input signal. State vector x is a high-dimensional response to the input signal u.

On the symbolic end, conceptors offer a symbolic representation of the sub-symbolic activity patterns within neural networks like ESNs. A conceptor is a positive semi-definite matrix that can be derived from any set of network states like the state sequence x. Their mathematical properties (Section 1.5.2) allow conceptors to, i.e., recognize, control, compare, combine, and logically evaluate patterns of network activities. Thereby, using conceptors, groups of ESN activities are considered as entities [**(possibly discrete and semantic)**] similar to how neural firing patterns of the brain were considered symbols by cognitive neuroscience. Thus, conceptors can be used as lenses on the dynamics of ESNs; lenses that open up a variety of symbolic mechanisms.

A common use-case of conceptors with ESNs has been supervised time series classification (Jaeger, 2014). An instance of this method will be outlined in Experiment 2.4, but the guiding idea is the following. Under the echo state property (ESP), an ESN responds uniquely to its inputs; i.e., there is a functional relationship between inputs and responses [**Why is it not bijective?**]. More concretely, with this functional relationship between inputs and responses, the responses can be seen as high-dimensional embedding of the input. Hence, any label assigned to a reservoir response is taken to equally apply to the input sequence that induced that response. To train the classifier, one captures the reservoir responses of each class using a conceptor. To classify, the new response is assigned the class whose conceptor indicates the highest similarity [**or correlation?**] to the previously seen responses of the class. Jaeger, Lukoševičius, Popovici, and Siewert (2007) demonstrated this method in speaker recognition on the *Japanese Vowels* dataset to a perfect accuracy, a result that promisingly exceeded several previous attempts. [Describe the advantages of this classification method.] Building on this experiment, Vlegels (2022) showed that competitive accuracies in the classification of non-stationary time-series may be achieved using several variations of this methods. Moreover, Chatterji (2022) adapted the method to time-series *recognition.* The difference between classification and recognition is that the former estimate the class of pre-segmented signals (one label per segment), whereas the latter estimates the class present within unsegmented signals (multiple labels per signal over time) (Lopes and Perdigao, 2011). Concretely, Chatterji (2022) demonstrated their method on the recognition of phonemes on the TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT), the dataset also used in the current study. Concluding, in the supervised tasks of time series classification and recognition, the classes of interest are given, but they may not always be.

Unsupervised learning tasks typically involve analyzing the structure of data without given classes or labels. For example, one may seek to identify important clusters (groups), hierarchies of clusters, axes, or relationships among a set of data points. In the following cases, conceptors were used to identify initially unknown structures or relationships among neural network state clouds for explainability and data analysis. Bricman, Jaeger, and van Rij-Tange (2022) used conceptors for neural network explainability. The developed technique, Nested State Cloud, extracts a knowledge graph from the latent space encodings of a set of 100 symbols (e.g., "apple"). Each symbol encompassed multiple instances, contexts in which they appear (e.g., "The apple fell."), all of which had different embeddings. Conceptors served as a compact representation of each symbol's embeddings and allowed them to be semantically related in the graph. Similarly, Mossakowski, Diaconescu, and Glauer (2019) used conceptors to represent and then organize classes of network states. This time, the symbols (i.e., classes) were Japanese speakers, and the network states were an ESN's responses to their speech samples. Using the fuzzy generalization of the Löwner ordering as a dissimilarity function on the speaker-specific conceptors, the symbols were organized in a tree by hierarchical agglomerative clustering. The authors mention the potential application in classification, but whether the resulting hierarchy truly coincided with any features of speakers like dialectal groups remained unknown. These studies analyze structures between classes of reservoir states, but this means that they, like the supervised classification algorithm, start from pre-established symbols. I shall refer to these as *above-symbol structures*, for they are at a level of abstraction above that of symbols.

1.1 Proposed solution

Reiterating, a symbol was defined as a pattern of network (ESN) activity that represents some entity. Such activity patterns, or symbols, can be represented, identified, compared, or induced using conceptors. In the previous Experiment, each positive conceptor represented the ESN activity patterns characteristic for a set of states. phoneme (the symbol) that, in turn, was the ESN. From these clusters, conceptors can be derived that reflect a new set of symbols.

To compute conceptor for any symbol, a set of network states (a state cloud) in which the pattern is present is needed. These states are responses to signal instances of an abstract class whose defining concept (the entity) is represented by the symbol.

When deriving a conceptor for any symbol from a set of states, it is preferable for that symbol to be the main commonality between the states. Geometrically, this translates to having lots of variance along principal components and low variance

forms the basis derive the conceptor When a set, or cluster, of network states

In the following two experiments, the generalized centroid-based hard clustering and HAC algorithms were adapted to cluster the ESN responses to the phonemic speech recordings. The general theme of both experiments was to (A) embed time series (the pre-processed phoneme recordings) as reverberations of an ESN (the response), (B) use conceptors or other centroids to represent these ESN reverberations and (C) group them into a set of meaningful clusters. Figure 1.2 depicts the steps from the original signals to ESN states to conceptors to clusters. Although the conceptors were subjected to the clustering algorithms, the tight relationship between signals, ESN responses, and conceptors presumably allowed the resulting clusters to generalize from conceptors to the other two representations. This generalization was exploited during cluster evaluation when comparing the found clusters against phonetic groups of the TIMIT speech samples.

1.2 Motivation

A significant challenge and the main aim of this study lies in unsupervised symbol identification [†]. The motivation for analyzing such *below-symbol structures* is the move toward more autonomous cognitive agency with applications in, for instance, computational creativity. Concretely, in the context of ESNs, the question arises of how to determine which network state instances should be included within each conceptor, that is, which patterns of network activity should be treated as the same symbol. It is relevant for several reasons. Firstly, unsupervised symbol identification is a step toward more general cognitive agency. Representations are often fed with a priori information on their task to improve performance; for instance, the dataset for training a speech recognition model may feature additional phonemic transcriptions. These will be used to form the model's task representation. However, there often is little basis to presume that, for any given task, the known representation is optimal. Moreover, for yet unsolved tasks, the inability to autonomously identify relevant symbols may form a ceiling to machine cognition. This need for autonomy is analogous to the earlier example of how children need to learn to distinguish phonemes due to, or rather in spite of, the lack of explicit instruction or supervision.

Secondly, computational creativity, in particular, may require the unsupervised generation of new symbols. According to Boden (2004), the most advanced type of creativity involves transforming an existing conceptual space. For example, some groundbreaking scientific discoveries may not be reachable via the mere exploration of existing theories (the current conceptual space) but may require a shift in the problem representation. Similarly, Douglas Hofstadter models creativity as an ability to form and alter symbols. Thus, without the capability to discover new symbolic representations, the creative agent might find itself quite limited, capable of optimizing , but unable to redefine the problem space itself. Regarding analogy-making, a core feature of human creativity, Hofstadter's Copycat model thereof relies on the fluidity of symbolic representations. Therefore, potential improvements in the available symbols could enhance analogical reasoning potential and, by extension, creative potential.

The identification of both below- and above-symbol symbol structures resembles classical clustering tasks. Clustering refers to the unsupervised task of organizing a data set into groups, or *clusters*, such that instances within the same cluster are more similar to each other than to those of other clusters. This similarity may be determined by a dissimilarity measure like the Euclidean distance. Clustering could play a significant role in finding below-symbol structures relevant to unsupervised symbol identification. By feeding the states of a neural network into a clustering algorithm, they can be grouped into clusters, each of which can then be treated as a symbol with the cluster members being the representations of that symbol. The analogy holds, for similarity is expected among states of the same symbol and dissimilarity between states of different symbols. Conceptors may serve to represent and evaluate the clusters. Conceptors can compactly represent and compare sets of network states and thereby act as centroids, i.e., exemplar-based representations of the symbols Bricman et al. (2022). Hence, the process of unsupervised symbol identification will be seen as a form of clustering, where each cluster corresponds to a symbol and is represented by a conceptor. Similarly, clustering algorithms seem equally applicable to the analysis of above-symbol structures by pre-grouping the network states by symbol according to a priori memberships.

A final motivation is to extend the use cases of conceptors to time series clustering. Reiterating, conceptor-based time series classification was achieved by classifying an ESN's responses to the

[†] Symbol identification is distinct from acquisition since no learning is involved.

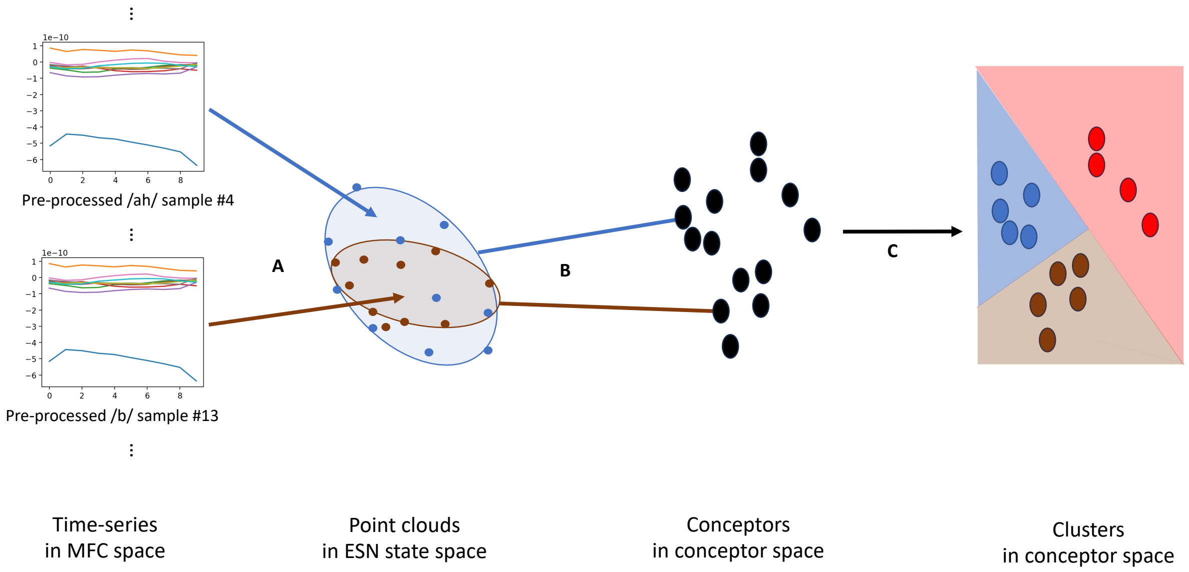

Figure 1.2: The steps from the MFC space of pre-processed speech signals to ESN state space by driving the ESN (A), followed by a transfer to conceptor space by computing conceptors (B), and finally the partitioning of the data into clusters (C). Note, while Experiment 2 aims to find partitions, Experiment 3 identifies hierarchies; thus, the right illustration would differ for Experiment 3. [Double check representation is truthful of apertures.]

time series. Similarly, clusters found among ESN dynamics may coincide with relevant clusters among the time series that underlie these dynamics. This relationship will prove useful in extrinsically evaluating the clusters. Moreover, the formalism of conceptors would thus be extended to time series clustering.

RQ Concluding, this study explores conceptor-based, unsupervised, options for identifying symbols within the sub-symbolic dynamics of ESNs (the responses to input time series). Therefore, it investigates the question of whether meaningful clusters and hierarchies can be identified among ESN state sequences. These structures are considered meaningful if they coincide with domain-relevant features of the corresponding inputs.

1.3 Justification of Experiments

I take inspiration from two classical hard clustering algorithms, K-means and hierarchical agglomerative clustering (HAC). Hard clustering assigns each data point to one, and only one, cluster, enforcing a clear separation between clusters. In contrast, soft clustering, like that of Bricman et al. (2022), assigns data points a level of membership for each cluster, allowing for overlap between clusters. I assumed, in the spirit of traditional symbolic representation theory, hard clustering to be the better fit for the endeavor of automated symbol identification using symbols to represent discrete, separated entities instead of multiple entities to varying degrees. ESN dynamics will be hard-clustered by adapting K-means and HAC, two well-known hard clustering algorithms with guaranteed convergence. K-means partitions data into a number of non-overlapping clusters and will be used to group below-symbol instances into symbolic groups. Importantly, traditional K-means can cluster N data points within $O(N)$ enabling its use on larger data sets. Meanwhile, HAC returns a hierarchy of clusters and will be used to analyze above-symbol structures among pre-existing symbols. HAC requires $O(N^2)$ with optimal implementation. Both algorithms have previously been used for time series clustering by [...]. However, while ESN state sequences are time series, it is unclear whether the previous methods are suited to the clustering of high-dimensional non-linear dynamics. Moreover, previous time series clustering algorithms have scaled badly with long, potentially infinite, time series, were often offline (fail to adapt as new data points), or struggled to handle variable length time series, a limitation mediocrely solved using Dynamic Time Warping. These shortcomings were equally menacing time series *classification* algorithms but were resolved by use of conceptors; the similarity in methods gives hope to similarly resolve them in conceptor-based time series *clustering.*

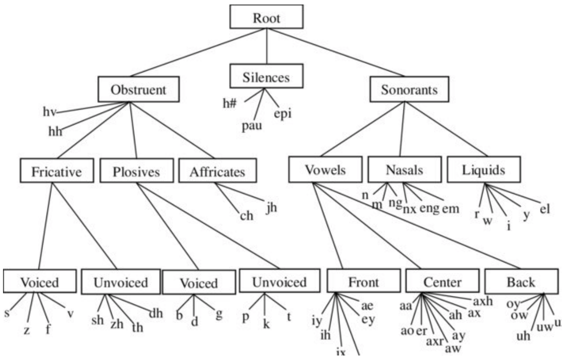

Figure 1.3: A taxonomy of all phones present in TIMIT Pfeifer and Balik (2011)

The developed methods will be demonstrated on phonemic speech recordings. As mentioned, Phonetics – the branch of linguistics concerned with speech – groups speech sounds into phones (sound units) and phones into phonemes (symbols); yet, further above-symbol organizations exist. For example, Figure 1.3 depicts a possible taxonomy of the phonemes present in the TIMIT corpus which contains clusters like vowels and nasals[‡].

The availability of these phonetic systems makes phoneme recordings a good domain for evaluating clustering algorithms. On the one hand, these phonemic classes and taxonomies can be used as ground truth clusterings to determine the performance of the developed clustering algorithms. On the other hand, the results from this study may provide an empirical verification or falsification of phonetic theory; whereas the groups and taxonomies of phones were largely developed based on their articulatory features, the following clustering Experiments will investigate whether these groups can be found on a purely acoustic basis. [Phonemes clustering literature on TIMIT (reference to methods to avoid

[‡] The TIMIT labels, also shown in Figure 1.3 and listed in Table A.1, are from IPA, although not written in Greek letters. The translation keys are given in the documentation.

redundancy)] [transition]

For clustering ESN responses to phoneme recordings, the ESN must be suited for the task. The following assumption affects the experimental design; Given an ESN, the more accurately a conceptor-based classifier can distinguish activity groups, the better these classes can be retrieved by an unsupervised clustering algorithm. The classification and clustering performances seem correlated because, for many hyperparameters like the aperture and spectral radius, a sweet spot seems to exist, where the ESN's signal-to-noise ratio is maximized and where the network is neither over- nor under-excited, that can be determined independent of the application in clustering or classification. Thus, I assume that tailoring the ESN to the TIMIT data and classification task will improve clustering on the same data. The hyperparameters were optimized in Experiment 1 on time series classification. Moreover, performing the classification task in Experiment 1 has the positive side-effect of demonstrating the application of conceptors to the phoneme classification on the TIMIT dataset. The following classification method builds on that of Jaeger (2014). To my knowledge, the currently highest accuracy of 78.4% for phoneme classification on TIMIT's test set was achieved using fixed-sized kernel logistic regression [Karsmakers, Pelckmans, Suykens, and hamme (2007) mentioned in Lopes and Perdigao (2011)].

1.4 Structure

Concluding, this study aims to identify groups (below-symbol) and hierarchies (below-symbol) among ESN states. The next section will provide a more formal introduction to ESNs, conceptors, and the adapted clustering algorithms. Experiment 1 will be a classification task used to tune the hyperparameters to be used in Experiments 2 and 3. In Experiment 2, K-means will be adapted to group the ESN responses to individual phoneme recordings. Experiment 3 will attempt to distill hierarchies of pre-grouped ESN responses.

1.5 Formal Definitions

The following sections on ESNs and conceptors were inspired by the detailed report Jaeger (2014). For an index of some of the mathematical notations used throughout the paper, please refer to Appendix B.

1.5.1 Echo State Networks

Let us formalize an example ESN like the one from Figure 1.1. Let N be the number of internal neurons. As mentioned, N will typically be large relative to the dimensionality d of the input. The input weight matrix $W^{in} \in \mathbb{R}^{N \times d}$, the bias vector $b \in \mathbb{R}^N$, and the internal weight matrix $W \in \mathbb{R}^{N \times N}$ are randomly initialized, the latter of which will typically contain many zeros to implement the internal sparsity of the reservoir. When driven by a discrete time series input u of length L, the ESN elicits a response, the internal state sequence x of dimensionality N and of the same length as the input. The ESN's update equation is that of classical discrete-time RNNs:

$$x(n+1) = \tanh(Wx(n) + W^{in}u(n+1) + b), \quad (1.1)$$

where $x(n)$ and $u(n)$ are the internal reservoir state- and input column vectors at time step n and $tanh$ is the hyperbolic tangent. Observe that any ESN state depends on the current input and the ESN's previous internal state. Indeed, the response x is a high-dimensional non-linear expansion of the input since $N >> d$ and tanh is applied. For later convenience, I define function r from the set of input sequences to the set of response sequences where $r(u)$ is the response that results from driving the ESN with u. The used ESN and the parameters will be clear from the context where r is used.

1.5.2 Conceptors

Definition and Intuition Given any sequence of network states $x = (x(1), ..., x(L))$ that may have arisen from running the above ESN [§], the conceptor matrix C computed from x minimizes the following loss function $\mathcal{L}$:

$$\mathcal{L}(C|x, \alpha) = \sum_{n=1}^{L} ||x(n) - Cx(n)||^2/L + \alpha^{-2}||C||^2$$
$$C = \arg\min_{C} \mathcal{L}(C|x, \alpha), \quad (1.2)$$

where $\alpha \geq 0$ is the conceptor's aperture (further explained below). The conceptor C that minimizes

[§] I consider *sequences* of network states for practicality, but they can be thought of as *sets* because their states need not necessarily be ordered; any ordinal information is lost during the computation of conceptors. Neither do the states need necessarily stem from the same ESN run.

$\mathcal{L}(C|x,\alpha)$ may be analytically computed via the following procedure:

1. Concatenate the states in x column-wise in an $N \times L$ collection matrix $X = [x(1)|...|x(L)]$.
2. Compute the correlation matrix $R = XX'/N \approx \mathrm{corr}(X)$.
3. Obtain the conceptor $C(R,\alpha) = R(R + \alpha^{-2}I)^{-1}$ with $N \times N$ identity matrix I.

To provide some intuition, conceptors can be considered "'fingerprint' of the activity of [a] network" over a period of time (Jaeger, 2014). This potential for conceptors to uniquely identify or fingerprint a sequence of states is reflected in its loss function. When minimizing $\mathcal{L}$, the term $||x(n) - Cx(n)||^2$ nudges C toward realizing an identity mapping for the states $x(n)$. However, the regularization term $a^{-2}||C||^2$ attaches a cost to the magnitude of its entries, limiting the numerical resources available to realize this identity mapping. Thus, the conceptor is drawn away from the identity matrix toward the zero matrix, especially on those axes that can account for little of the variance of x, i.e., where the minimization of $||x(n) - Cx(n)||^2$ is less beneficial. The amount of regularization applied to the conceptor depends on its aperture α. The higher the aperture, the closer the resulting conceptor maps all the members of x to themselves. The lower the aperture, the more selective the conceptor becomes, realizing a "good" identity mapping only along the important axes that explain most of the sequence's variance.

A geometric perspective might extend this intuition. The reservoir states being a point cloud in state space $(0,1)^N$, their conceptor can be represented as a hyperellipsoid of a similar shape as the point cloud but constrained to the unit circle of $\mathbb{R}^N$. The directions and lengths of the hyperellipsoid's axes are given by the singular vectors and values of the conceptor, respectively, and "represent the principle components of the state cloud" (Vlegels, 2022). However, the shape of the hyperellipsoid slightly deviates from that of the point cloud as the regularization *squashes* it along its shorter axes. With a low aperture, the hyperellipsoid would be very squashed, extended only along a few main axes. With a large aperture, the conceptor would be a little squashed and approach the unit hypersphere.

This squashing in space may be seen as a spatial compression as it systematically reduces the variance of the point cloud along its less important axes. Furthermore, conceptors are a temporal compression of ESN state sequences. Mapping sequences of vectors in state-space to conceptor-space turns variable-length objects into constant-sized objects. When the state sequence is a time series, this mapping removes the temporal dimension, whence it may be seen as a temporal compression. **[Is there a better way to describe this compression?]**

Conceptor operations The following section describes several useful operations on conceptors some of which illustrate their symbolic nature. [Elaborate]

Similarity function Conceptors may be used to compute the similarities of state sequences. If C_a and C_b be the conceptors derived from the two sequences of states to be compared, their similarity may be defined as:

$$\mathrm{Sim}(C_a, C_b) = \frac{|(S_a)^{1/2}(U_a)'(U_b)(S_b)^{1/2}|^2}{|\mathrm{diag}(S_a)||\mathrm{diag}(S_b)|}, \quad (1.3)$$

where $U_a S_a (U_a)'$ is the SVD of C_a and $U_b S_b (U_b)'$ is the SVD of C_b. It is a function of the squared cosine similarity of the conceptors that measures the angular alignment between all pairings of singular vectors of the two conceptors weighted by the corresponding singular values.

Aperture The aperture of a conceptor can be set during its computation or when given a precomputed conceptor C **[Is it okay to use the name 'C' to refer to other conceptors than this one. Or do I need to name every object differently e.g., C_2.]**, its aperture can still be adapted by any factor of $\gamma > 0$ using the aperture-adaptation function φ that returns the aperture-adapted conceptor C_{new}:

$$C_{new} = \varphi(C, \gamma) = C(C + \gamma^{-2}(I - C))^{-1} \quad (1.4)$$

Logical Operations on Conceptors Several logical operations have been meaningfully defined on conceptors. Given two conceptors C and B, we have the following definitions and semantics:

1. **Negation** ($\neg$)

$$\neg C := I - C \tag{1.5}$$

   It returns a conceptor that describes the linear subspace complementary to that of C.

2. **Conjunction** ($\wedge$)

$$C \wedge B := (P_{R(C)\cap R(B)}(C^{\dagger} + B^{\dagger} - I)P_{R(C)\cap R(B)})^{\dagger} \tag{1.6}$$

   See Appendix A.1 for details on the computation of $P_{R(C)\cap R(B)}$. It returns a conceptor that describes the intersection of the linear subspaces of C and B.

3. **Disjunction** ($\vee$)

$$C \vee B := \neg(\neg C \wedge \neg B), \tag{1.7}$$

   by De Morgan's law. It returns a conceptor that describes the union of the linear subspaces of C and B.

Although a simpler method for computing con- and disjunction exists, that method relies on the inversion of B and C and thus fails when B or C contain singular values of 0. Such singular values may occur in practice, for example, due to rounding or through the negation of conceptors with unit singular values. Therefore, the above method is recommended whenever the absence of null and unit singular values cannot be ensured.

1.5.3 Clustering Algorithms

Centroid-based parametric hard clustering Hard clustering aims to identify a set of K hard (distinct) clusters among its input data. As mentioned, the algorithm follows an iterative relocation scheme to find a set of clusters and centroids that minimize a dissimilarity function. Thus, it depends on the following three components:

HAC HAC is a clustering algorithm aimed at identifying a hierarchy of clusters among the input data. HAC works by initializing one cluster per data point and then iteratively merging the two closest clusters until a single cluster or a desired number of clusters is obtained. The result is a hierarchical structure of nested clusters commonly visualized as a dendrogram. Its implementation depends on three components:

1. A *set of points*, typically, in Euclidean space.

2. A *function that provides the distance between two points*, typically, the Euclidean distance between the points.

3. A *function that provides the distance between two clusters.* A common choice is the mean distance between all pairs of points (as defined above) in the two clusters. This variant of HAC is coined average linkage.

2 Methods and Results

2.1 Dataset

The TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) was chosen as the data source. It features diverse and phonetically annotated speech signals. TIMIT comprises 6300 sentence utterances. Each of the 630 US-based native-English speakers (from eight dialect regions and 30% of whom were female) read ten sentences: five phonetically-compact, three phonetically-diverse, and two dialect sentences. Each utterance comes with a phonetic transcription that indicates which of 64 phones is uttered at any time. Moreover, the corpus is pre-split into a training (73 % of the utterances) and a test set used as such in Experiment 1. Experiments 2 and 3 relied solely on the training set.

2.2 Pre-processing

For all experiments, the following pre-processing steps were performed. The utterances were segmented according to the phonetic transcriptions into $n = 241225$ segments ($n_{\text{TIMIT-train}} = 177080$ and $n_{\text{TIMIT-test}} = 64145$), each a vocalization of one phone (Figure 2.1). From each segment, the first $d = 13$ Mel Frequency Cepstral Coefficients (MFCCs) were extracted, consistent with previous literature Bromberg, Qian, Hou, Li, Ma, Matthews, Moreno-Daniel, Morris, Siniscalchi, Tsao, et al. (2007). This representation ought to isolate the information most relevant to speech analysis. To compute the MFCCs, the Librosa Python library (McFee, Raffel, Liang, Ellis, Mcvicar, Battenberg, and Nieto, 2015) was used with one MFCC vector computed every 1 ms from a 2.5 ms long sliding window.

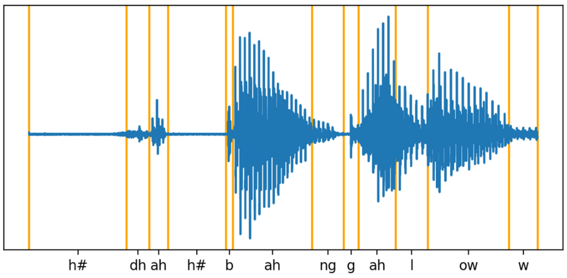

Figure 2.1: Example of the twelve first segments of one of the utterances.

The resulting time series were normalized in amplitude and time. First, the amplitudes varied strongly across the channels [the lowest mean amplitude of an MFCC channel ($\mu_1 = -593.2$) is 24.2 times lower than that of the second lowest channel ($\mu_4 = -22.4$)]. To grant each MFCC a similarly strong effect on the ESN dynamics under the identically distributed input weights, each channel was normalized to a range of $[-0.5, 0.5]$ across all samples. Second, to account for differences in utterance speeds, the series were normalized in time by fitting each channel with a cubic spline and sampling it at $L = 10$ temporally equidistant points.

Lastly, the phonetic labels (transcriptions) p_i $(i = 1, ..., c)$ were mapped from the original set of 61 phones to a subset of 39 phonemes P. Initially proposed by Lee and Hon (1989), this mapping amounts to folding stress-related variations and allophonic variations of phonemes (e.g., /em/ and /m/) into the same classes. The concrete mappings are given in appendix Table A.1 along with the distribution of the resulting classes within the dataset. The mapping serves to reach reasonable classification and clustering performances, render the computations feasible, and make the results comparable to previous studies that also used this mapping [...].

Thus, the resulting data consisted of tuples $D = \{(s_i, p_i)|i = 1, ..., c\}$ with MFCC time series s_i, phone labels $p_i \in P$, and the set of phones P after folding ($|P| = 39$). The ready-made train-test split from TIMIT was used giving $D_{\text{TIMIT-train}}$ and $D_{\text{TIMIT-test}}$ of respective lengths $n_{\text{TIMIT-train}}$ and $n_{\text{TIMIT-test}}$.

2.3 ESN

The following ESN setup was used for all three experiments. Its hyperparameters are summarized in Table 2.1. The ESN consisted of $N = 100$ neurons with a connection density of $r = 10\%$. The entries of W^{in} and b were randomly sampled from a standard normal distribution and rescaled by factors of $k_{W^{in}} = 1.1$ and $k_b = 0.6$, respectively. W was obtained by random sampling from a standard normal distribution and rescaling the result to a spectral radius of $\rho = 2.57$. The spectral radius of an internal weight matrix is its largest absolute eigenvalue. The larger ρ, the farther W transforms the internal state during the state update along its first eigenvector, which tends to lead to a more chaotic behavior. ρ was adapted by rescaling the old (initial) internal weight matrix W_{old} to $W_{new} = \frac{\rho_{new}}{\rho(W_{old})} W_{old}$ where W_{new} has the desired spectral radius ρ_{new} instead of the previous ρ_{old}. [Effects on ESP]

Hyperparameter	**Value**
Number of neurons (N)	100
Connection density (r)	10%
Scaling factor for W^{in} ($k_{W^{in}}$)	1.1
Scaling factor for b (k_b)	0.6
Spectral radius (ρ)	2.57

Table 2.1: Final ESN hyperparameters.

The above hyperparameters were picked by hand based on their effects on the accuracy in Experiment 1, previous research, and resource constraints. All parameters were initially set to the values used in the demonstration experiments of (Jaeger, 2014) (Section 4.1, p. 161). N was kept as a largest possible value still feasible under the available computational resources. Larger sizes would likely improve performance after adapting the other hyperparameters but may increase the risk for overfitting (Lukoševičius, 2012). The remaining hyperparameters, r, $k_{W^{in}}$, k_b, and ρ, were adjusted by hand with the objective to maximize the development accuracy of phoneme classification in Experiment 1. Moreover, automated hyperparameters optimization was attempted but eventually not used due to its large computational cost and slow convergence. Its method and results are in the Appendix.

The resulting ESN was driven independently on each input signal s_i $(i = 1, ..., c)$ producing the reservoir state sequence x_i $(i = 1, ..., c)$. Concretely, each run started from the same state $x(0)$ sampled once from a standard normal distribution not to introduce meaningless between-sample differences while providing the network with an initial excitation. Indeed, using this normally distributed starting state led to a greater classification accuracy (on D_{dev} using the final hyperparameters) in Experiment 1 than when using the null vector as the starting state **[Can/Should I just state this without reporting the experiment? And is it tolerated to make such design decisions using the test set given the risk of overfitting?]**. The following states $x_i(t)$ $(t = 1, ..., L)$ were computed via update Equation 1.1 and collected column-wise in $N \times L$ matrix $X_i = [x_i(1)|...|x_i(L)]$. Concluding, an ESN response collection matrix X_i was computed for each training sample.

2.4 Experiment 1: Phoneme Classification

Objective Experiment 1 aimed to classify the pre-processed utterances using the corresponding ESN responses. The experiment's purpose is to (a) optimize the ESN hyperparameters for the subsequent clustering experiments and (b) evaluate and improve methods for conceptor-based time series classification. Concretely, the developed classifier takes as input the ESN's response to an utterance and outputs the assigned phoneme label. The assigned label is taken to equally apply to the ESN's response to the utterance and the utterance.

Data To optimize hyperparameters and take small design decisions, the pre-processed original dataset $D_{\text{TIMIT-train}}$ was initially divided into a preliminary training set, $D_{\text{pre-train}}$, and a development set, D_{dev}. Here, the split was 80/20, respectively, stratifying over phonemic classes. Once hyperparameters and methods were set, the classifier was retrained on the whole of $D_{\text{TIMIT-train}}$ and evaluated on $D_{\text{TIMIT-test}}$.

2.4.1 Training

Training amounted to computing one *positive conceptor* C_p^+ and one *negative conceptor* C_p^- per class $p \in P$. Each class' positive conceptor captures the linear state subspace that ESN states (responses to pre-processed phoneme utterances) of that class tend to occupy. For each class p, it was computed as follows. Let η_p be the number of training instances of p. The state collection matrices corresponding to signals of p were concatenated column-wise into a class-level collection matrix $X_p = [X_1|X_2|...|X_{\eta_p}]$ from which C_p^+ was computed with an initial aperture of $\alpha = 1$ by steps 2 and 3 of the procedure for conceptor computation. This was repeated for each class to obtain the set of preliminary positive conceptors $C_{pre}^+ = \{C_p^+ | p \in P\}$.

Aperture adaptation After computing the conceptors in C_{pre}^+ with the initial aperture of $\alpha = 1$, their apertures were optimized. First, one new aperture α_{opt} was chosen for all positive conceptors. The objective was to maximize their sensitivity to differences in the underlying ESN dynamics which would likely improve their capacity to classify the data to come. This objective is quantified as the maximization of the ∇-*criterion*. Concretely, the ∇-*criterion* is defined in function of conceptor C and a candidate aperture adaptation factor γ [¶]. It then returns the gradient of the Frobenius norm of the aperture-adapted conceptor with respect to the logarithm of γ:

$$\nabla(C, \gamma) = \frac{d}{d\ \log(\gamma)} ||\varphi(C, \gamma)||^2 \tag{2.1}$$

Indeed, this corresponds to the magnitude at which the size $||C||^2$ of a conceptor C changes (sensitivity) with respect to its log aperture (scaling of the underlying ESN states, see p. 49 of Jaeger (2014)). The optimal aperture, γ_p, was then approximated for each positive conceptor C_p^+ by sweeping through 200 candidate values $\gamma_{candidate}$ in the interval $[0.001, 500)$ on a logarithmic scale; logarithmic, for the optimal value was expected on the lower end of the interval. γ_p was set to the $\gamma_{candidate}$ that maximized a numerical approximation of $\nabla(C_p^+, \gamma_{candidate})$. This derivative was approximated using a change in γ of $\Delta\gamma = 10^{-4}$ **[Does the method for approximating the derivative need further explanation?]**. $|P| = 39$ values γ_p $(p \in P)$ resulted. Finally, the apertures of all positive conceptors were adapted using the mean $\gamma_{opt} = \frac{1}{|P|} \sum_{p \in P} \gamma_p \approx 133.98$. Let C_{opt}^+ be the resulting set of aperture-optimized positive conceptors.

[¶] In this case, γ equals the resulting aperture α_{new}, since $\gamma = \frac{\alpha_{new}}{\alpha}$ and the current aperture $\alpha = 1$.

The reason for adapting all conceptors to the same aperture instead of adapting each conceptor with their γ_p is to avoid classification bias. The larger a conceptor's trace (the sum of its singular values), the larger its *expected* Evidence, the similarity measure later used for classifying new instances [**Should I add the proof to appendix?**]. Thus, if there are differences between the traces of the conceptors, the classifier will be biased toward the classes whose conceptors have greater traces. A similar bias occurs for the expected conceptor similarity later used for clustering. Since a conceptor's trace strongly covaries with its aperture, the latter is used to control the trace [**Do I need a citation, proof, or explanation for this?**]. Thus, equating the apertures between classes helped as first step to approach their traces and debias the classifier.

However, although the conceptors were adapted to the same aperture, some variation in their traces remained as can be seen at $x = 1$ in Figure 2.2. To remove it, the conceptor's traces were again adapted until reaching a shared target value $tr_{target} = \bar{tr}$, the mean trace among the conceptors:

$$\bar{tr} = \frac{1}{|C^+|} \sum_{C \in C^+_{opt}} \mathrm{tr}(C) \approx 57.23 \qquad (2.2)$$

To adapt each conceptor's trace to this target value with an error tolerance of ϵ (here, $\epsilon = 0.01$), Algorithm 2.1 was developed. I shall call $\psi(C, tr_{target}, \epsilon)$ the function computed by the algorithm. The iterative procedure adapts a conceptor's aperture by a factor of the current trace error ratio until reaching the target trace. This works because, again, trace and aperture covary.

As shown in Figure 2.2 at $x > 1$, applying $\psi(\cdot, \bar{tr}, 0.01)$ to each of the conceptors in C^+_{opt} normalizes their traces. Thus, via the above two aperture adaptation procedures, the apertures of the conceptors in C^+_{pre} were optimized and their traces were normalized resulting in the final set of positive conceptors C^+.

Algorithm 2.1 Adapt the trace of a conceptor

```
Require:
  Conceptor $C$ whose trace is to be adapted to $tr_{target}$
  Target trace $tr_{target}$
  Error tolerance $\epsilon$

  while TRUE do
    if $|tr_{target} - tr(C)| < \epsilon$ then
      break
    end if
    $\gamma \leftarrow tr/\mathrm{tr}(C)$
    $C \leftarrow \varphi(C, \gamma)$
  end while
```

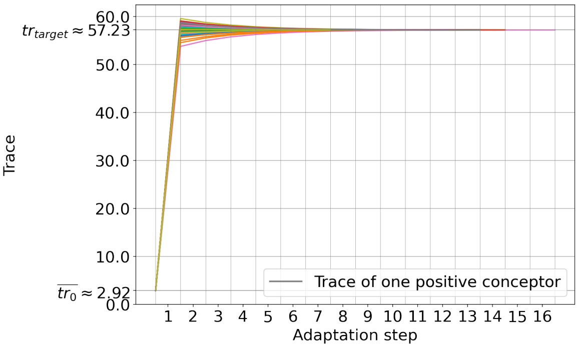

Figure 2.2: The traces of the positive conceptors in function of the adaption steps. The first adaptation step 1 is the aperture adaptation based on the ∇-criterion. The increase in aperture caused the traces to increase and diverge. The remaining steps ($x > 1$) are the results of normalizing the traces using Algorithm 2.1. Finally, these aperture adaptation steps resulted in an increase of the mean trace from an initial value of $\overline{tr_0}$ to approximately the target value tr_{target}.

Negative conceptors From C^+, the set of negative conceptors $C^- = \{C^-_p | p \in P\}$ was computed. Each class' negative conceptor models the linear state subspace that is complementary to the space occupied the states of all other classes. Or, equivalently, it models the subspace that states "from none of the other classes" are expected to occupy. These semantics are reflected in their formal definition:

$$C_p^- = \neg \bigvee \{C_q^+ | q \in P, q \neq p\}, \tag{2.3}$$

where $\bigvee S$ is the disjunction of the $|S|$ conceptors in set S computed by the associative disjunction of its elements:

$$\bigvee S = ((C_1 \vee C_2) \vee C_3) \vee ... \, C_{|S|} \tag{2.4}$$

Adaptating the apertures of the resulting C_p^-, using the methods with which the positive conceptors were adapted, did not improve the development accuracy.

2.4.2 Testing

The combined Evidence E was used to classify unforeseen speech samples via the corresponding ESN responses. The combined Evidence $E(x,p)$ that some ESN state x corresponds to class p is a measure of similarity between that state and the positive and negative conceptors of class p. Concretely, it is the combination of a positive Evidence $E^+(x,p)$, computed using the positive conceptor C_p^+, and a negative Evidence $E^-(x,p)$, computed using the negative conceptor C_p^-:

$$\begin{aligned} E(x,p) &= E^+(x,p) + E^-(x,p), \\ E^+(x,p) &= x'C_p^+x \\ E^-(x,p) &= x'C_p^-x \end{aligned} \tag{2.5}$$

$E(x,p)$ is large when x is close to the linear subspace modeled by conceptor C_p^+, but far from the linear subspace which the other conceptors model (see definition of C_p^-).

To classify a state x, the positive Evidence is maximized over all classes:

$$\text{Class}'(x) = \arg\max_{p \in P} E(x,p) \tag{2.6}$$

[Can I use self-explanatory symbols like `Class'` without explicitly stating what they are.] To classify a point cloud X (a column-wise state collection matrix), the mean Evidence for all states (columns of the collection matrix) is taken:

$$\text{Class}(X) = \arg\max_{p \in P} \frac{1}{L} \sum_{i=1}^{L} E(X[:,i],p), \tag{2.7}$$

where L is the width of X.

Finally, given a speech sample s_{new}, Class(X_{new}) returns the classification estimate, where $X_{new} = [r(s_{new})(1)|...|r(s_{new})(L)]$ is the column-wise concatenation of the states collected when driving the ESN with s_{new} (excluding the starting state).

2.4.3 Results

This testing procedure resulted in accuracies of 56.16% on the training set and 56.04% on the test set. For additional reference, the confusion matrix in Figure A.1 of Appendix A shows the classification rates across the classes. In an attempt to improve the accuracy, the experiment was repeated in slight adaptation to the type of data (Appendix A) which led to a training accuracy of 63.64% but test accuracy of 49.13%.

Conclusion In Experiment 1, an ESN setup (see Table 2.1) was found that could effectively represent time series as reflected in the accuracy achieved during conceptor-based classification. The same ESN setup and similar conceptor-based mechanisms were reused in Experiments 2 and 3 as I turn to unsupervised methods for identifying distinct symbols within ESN dynamics.

2.5 Experiment 2: Below-phoneme clustering

Objective Now blinded of any a-priori symbols – the classes $|P|$ previously available during training – an attempt to determine a set of maximally distinct symbolic classes was made tabula rasa. Concretely, in Experiment 2, three conditions for grouping earlier computed ESN state clouds into K hard (non-overlapping) clusters and one baseline condition were employed.

Dataset For computational constraints, the experiment was run on data subsamples $D_l \subset D_{\text{TIMIT-train}}$ but was repeated 10 times ($l = 1, ..., 10$). Each D_l encompassed utterances of an *independently* sampled set of 7 phonemes P_l to reduce selection bias. The cardinality of $|P_l| = 7$ ought roughly align the difficulty of the current task with the "simple" condition of a previous study, (Lerato and Niesler, 2012), that similarly performed hard clustering on TIMIT's phoneme recordings. However, since (Lerato and Niesler, 2012) restricted the data to vowels, which I did not for consistency with Experiment 3, their task was likely more difficult. For each phoneme in P_l, 15 phoneme utterances were sampled, to meet the centroid-based clustering assumption of equally large clusters, for a total of $n' = 105$ utterances per subset. Concretely, each utterance was sampled from 48 random speakers with an equal ratio across genders and dialect regions (newly sampled for each fold l). This stratification aims to evenly represent the population and enable the evaluation of any found clusters on and their generalization to new data. Moreover, only utterances from the phonetically compact sentences, that use each phonemes in a few phonetic contexts (e.g., /aa/ only before /f/), were considered to limit phonetic variability.

Clustering Scheme On each D_l, generalized centroid-based clustering was performed in four conditions. I will proceed by describing the algorithm, that is parameterized by three condition-dependent components, and then elaborate on the conditions. Besides the desired number of clusters K, the generalized centroid-based clustering algorithm is parametrized by three components:

1. A set of points $D = \{p_1, p_2, ..., p_n\}$ to be clustered.

2. A centroid computation function centroid(Cl) that provides the centroid of a given cluster Cl (a set of points).

3. A dissimilarity function $d(p_i, \mu_j)$ that provides the dissimilarity of a point p_i and centroid μ_j. During the assignment step, points are assigned to the cluster with the most similar centroid. This function is assumed to be non-negative and be monotonic increasing with dissimilarity. It may be asymmetric.

The algorithm minimizes the following loss function L:

$$L = \sum_{k=1}^{K} \sum_{p_i \in Cl_k} d(p_i, \text{centroid}(Cl_k)), \tag{2.8}$$

where Cl_k is the set of points in the k^{th} cluster.

L is minimized by as follows. Centroids are initialized in a K-means++ fashion. With this method, the initial centroids are sampled from a distribution that aims to spread them evenly across point space. Compared to the random centroid-initialization of classical K-means, this method tends to converge faster and more consistently used on points in Euclidian space (Arthur and Vassilvitskii, 2007). Then, two steps are iteratively repeated. In the *Assignment step*, each data point is assigned to the cluster with the nearest centroid according to some distance function. In the *Centroid update step*, the centroids are recalculated based on the newly formed clusters. This process terminates once all centroids converged in their position or a maximum number of iterations is reached.

Empty clusters were actively prevented. It is theoretically possible for empty clusters to arise after the *Assignment step* in each condition. An explanation will be given in the discussion. In practice, this case only occurred in the conceptor-based conditions (*ESN-Evidence* and *ESN-Mixed*). Regardless, since no decent centroid can be found for empty clusters, this case was handled in an additional *Reassignment step* after the *Assignment step.* Concretely, when an empty cluster arises, the most "misfit" point – the point with the largest dissimilarity to its current cluster's centroid – is reassigned to the empty cluster.

Conditions The experiment was performed in four conditions that would determine the parameters of the generalized centroid-based clustering algorithm:

- *MFCC-Euclidian* was a baseline where the MFCC time series were clustered using classical K-means(++).

- *ESN-Euclidian, ESN-Evidence*, and *ESN-Hybrid* clustered ESN state clouds, the responses to the MFCC time series.

First, in the *MFCC-Euclidian* condition, classical K-means was used to cluster the MFCC time series directly.

1. The set of points $D_{MFCC,l}$ contains the unlabeled MFCC time series D_l:

$$D_{MFCC,l} = D_l = \{s_i | 0 < i \leq n'\} \tag{2.9}$$

Algorithm 2.2 Generalized centroid-based hard clustering algorithm

```
Require:
  Number of clusters $K$
  [1] Set of points $D = \{p_1, p_2, ..., p_n\}$
  [2] Centroid computation function centroid$(Cl)$
  [3] Dissimilarity function $d(p_i, p_j)$

  Initialize $K$ cluster centroids via the K-means++
  procedure: $\mu = \{\mu_1, \mu_2, ..., \mu_K\}$

  while TRUE do
    Reset all clusters: $Cl = \{\emptyset, \emptyset, ..., \emptyset\}$

    Assignment step:
    for $p_i$ in $D$ do
      $k \leftarrow \arg\min_{1 \leq j \leq K} d(p_i, \mu_j)$
      Assign $p_i$ to cluster $k$: $Cl_k \leftarrow Cl_k \cup p_i$
    end for

    Reassignment step for empty clusters:
    while any $Cl_j$ is empty do
      Find point $p_{max}$ in $D$ that is most dissimilar
      to the centroid of its current cluster
      Move $p_{max}$ from its old cluster to $Cl_j$‖
    end while

    Centroid update step:
    for $j = 1$ to $K$ do
      $\mu_j \leftarrow$ centroid$(Cl_j)$
    end for

    if cluster assignments did not change then
      break because converged
    end if
  end while

  return Set of clusters $Cl =$
  $Cl_1, Cl_2, ..., Cl_K$
```

2. Given a cluster Cl, its centroid is the mean of its members:
$$\text{centroid}_{Eucl}(Cl) = \frac{1}{|Cl|} \sum_{p_i \in Cl} p_i \qquad (2.10)$$

3. The distance from point p_i to cluster Cl is the Euclidian distance between p_i and Cl's centroid:
$$d_{Eucl}(p_i, Cl) = ||p_i - \text{centroid}_{Eucl}(Cl)|| \qquad (2.11)$$

Second, in the *ESN-Euclidian* condition, the classical K-means clustered the ESN's responses to the MFCC time series.

1. The set of points $D_{ESN,l}$ contains the earlier collected, unlabeled, ESN responses to the MFCC time series from D_l:
$$D_{ESN,l} = \{X_i | s_i \in D_l\}, \qquad (2.12)$$
where X_i is the state collection matrix of the response to MFCC time series s_i.

2. Given a cluster Cl, its centroid is, again, the mean of its members centroid$_{Eucl}(Cl)$.

3. The distance from point p_i to cluster Cl is, again, the Euclidian distance between p_i and Cl's centroid $d_{Eucl}(p_i, Cl)$.

Third, in the *ESN-Evidence* condition, conceptors were used as centroids to represent the clusters. In essence, Experiment 1's classifier training procedure provides cluster centroids and its testing procedure acts to reassign points to clusters.

1. The set of points is, again, the ESN responses $D_{ESN,l}$.

2. Given a cluster Cl_j, its centroid is the tuple of the positive conceptor $C_j^{+'}$ and negative conceptor $C_j^{-'}$ computed from its and the other clusters' members:
$$\text{centroid}_{Evid}(Cl_j) = (C_j^{+'}, C_j^{-'}) \qquad (2.13)$$
Here, the clusters were construed as a set of classes. The same steps as used to train a phoneme-classifier in Experiment 1 were applied to train a "cluster-classifier". $C_j^{+'}$ and $C_j^{-'}$ are, then, the conceptors representing class i with training instances Cl_j.

3. The distance from point p to cluster Cl_j is the reciprocal of the combined Evidence $E(p, j)$ that p corresponds to cluster class j given the previously trained conceptors in $\text{centroid}_{Evid}(Cl_j)$:

$$d_{Evid}(p, \mu_j) = 1/\mathrm{E}(p, j) \tag{2.14}$$

   E indicates similarity on the range $(0, \infty)$ [**Can the combined Evidence be 0 in theory?**]; so, d_{Evid} ranges in $(0, 1)$ with larger values indicating a higher degree of dissimilarity between point p and the centroid μ_j of cluster Cl_j.

Fourth, in the *ESN-Hybrid* condition, the *ESN-Euclidian* and *ESN-Evidence* conditions were combined.

1. The set of points is, again, the ESN responses $D_{ESN,l}$.
2. Given a cluster Cl_j, its centroid is two-fold; It includes the mean of the states of the *ESN-Euclidian* condition and the positive and negative conceptors of the *ESN-Evidence* condition:

$$\begin{aligned} &\text{centroid}_{Hybrid}(Cl_j) \\ &\quad = (\text{centroid}_{Eucl}(Cl_j), \text{centroid}_{Evid}(Cl_j)) \end{aligned} \tag{2.15}$$

3. The distance from point p to cluster Cl_j is the mean of the distances used in the *ESN-Euclidian* and the *ESN-Evidence* conditions:

$$\begin{aligned} &d_{Hybrid}(p, Cl_j) \\ &\quad = (d_{Eucl}(p, Cl_j) + d_{Evid}(p, Cl_j))/2 \end{aligned} \tag{2.16}$$

   $d_{Eucl}(p, Cl_j)$ and $d_{Evid}(p, Cl_j)$ are computed using their respective elements in $\text{centroid}_{Hybrid}(p, Cl_j)$.

With these components as parameters, the clustering algorithm was run for 10 trials in each condition.

Evaluation The resulting clusters were evaluated using one intrinsic measure, the *mean intra-cluster dissimilarity* (MICD), and two extrinsic measures, the *normalized mutual information* (NMI) and *cluster classification accuracy* (CCA).

First, the MICD is the mean dissimilarity between the clusters' centroids and member points over all clusters. It is a measure of cluster cohesion, i.e., of how "tight" the clusters are, which translates to the crispness of the resulting symbols, i.e., how similar the symbol is to its designated referents. Moreover, it resembles the optimization objectives of the original and adapted K-means algorithm, and a downward trend of the MICD over the iterations is expected. Thus, it was applied to inform about cluster cohesion and the convergence behavior of the algorithm. d_{Evid} not being a metric, it also does not fulfill the assumptions of many alternative intrinsic cluster quality measures like the within-cluster sum of squares or the silhouette coefficient. Importantly, the MICD should not be mistaken as a means to compare between conditions, since they used different dissimilarity functions and data types. Given a clustering $Cl = \{Cl_1, Cl_2, ..., Cl_K\}$, where Cl_k is the set of points assigned to cluster k, the MICD is calculated as follows. For any cluster Cl_k, let the intra-cluster dissimilarity (ICD) be the mean dissimilarity between its member points p_i and its centroid $\text{centroid}(Cl_k)$:

$$ICD(Cl_k) = \frac{1}{|Cl_k|} \sum_{p_i \in Cl_k} d(p_i, \text{centroid}(Cl_k)) \tag{2.17}$$

where $d(p_i, \text{centroid}(Cl_k))$ is the dissimilarity function and depends on the condition. The MICD is then the mean of the ICD values for all clusters:

$$MICD = \frac{1}{K} \sum_{k=1}^{K} ICD(Cl_k) \tag{2.18}$$

where K is the number of clusters.

Second, the NMI is an extrinsic measure of the similarity between a clustering $Cl = \{Cl_1, Cl_2, ..., Cl_K\}$, where Cl_k is the set of points assigned to cluster k, and the ground-truth phonemic groups $G = \{G_1, G_2, ..., G_{|P_l|}\}$, where G_p is the set of points with label p in the dataset D_l. Its values range from 0 for completely dissimilar clusterings and 1 for identical clusterings. It was used to compare the empirically found clusters with the phonemic classes of TIMIT. To compute the NMI, the mutual (shared) information I between Cl and G is normalized by the mean entropy (uncertainty)

H within each clustering:

$$
\begin{aligned}
&I(G, Cl) \\
&= \sum_{G_p \in G} \sum_{Cl_k \in Cl} P(G_p \cap Cl_k) \log \frac{P(G_p \cap Cl_k)}{P(G_p)P(Cl_k)} \\
&H(G') \\
&= - \sum_{G'_p \in G'} P(G'_p) \log P(G'_p), \text{ for some clustering } G' \\
&NMI(G, Cl) = \frac{I(G, Cl)}{\frac{1}{2}[H(G) + H(Cl)]}
\end{aligned}
\tag{2.19}
$$

Third, I evaluated how accurately the conceptors derived from the clusters could classify new phonemes. This extrinsic measure, that I will refer to as *cluster classification accuracy* (CCA), ought to measure how accurately the acquired symbols this system could be used to represent certain entities, phoneme utterances in this case. Therefore, it was assumed that each cluster corresponded to one of the phonemic classes P. A bijective mapping of clusters to classes was made using Kuhn-Munkres algorithm to maximize the sum of intersections (Plummer and Lovász (1986) mentioned in Song, Liu, Huang, Wang, and Tan (2013)). Then, Experiment 1 was essentially replicated; a conceptor-based classifier was trained on the clusters to classify the respective matched class. The classifier was tested on the phonetically diverse utterances of phonemes P_l that remained from the eight selected speakers.

Run Algorithm 2.2 was repeatedly run on each condition for 20 *trials* to reduce the effects of the random cluster initialization. Thus, 20 replicates × 20 trials × 4 conditions = 1600 trials were performed in total. To limit the scope of this study, only a number of $K = 7$ clusters, consistent with the number of phonemic classes $|P_l|$, were searched for.

Results Table 2.2 compares the mean scores of the extrinsic performance measures (columns) of the clustering results across conditions (rows). Two additional rows were added, 'Random clusters' and 'Dataset classes', with the scores to expect of a random clustering and a clustering that perfectly matched the true class labels, respectively. The scores in all conditions were significantly above random [**Any way to support the significance claim?**]. Between the conditions, *MFCC-Euclidian* outperformed the others in terms of NMI by very little. No significant differences in NMI could be found between the ESN-based conditions. The *ESN-Mixed* condition and the *MFFC-Euclidian* outperformed the others in their accuracy.

Condition	**NMI**	**CCA**
MFCC-Euclidian	0.4934	0.5195
ESN-Euclidian	0.4774	0.5073
ESN-Evidences	0.4717	0.5040
ESN-Mixed	0.4757	0.5167
Random clusters	0.0871	0.2438
Dataset classes	1.0000	0.6971

Table 2.2: Mean scores of final clusterings between conditions.

Figures 2.3 and 2.4 depict the normalized MICDs and NMIs over the iteration. Horizontal lines provide comparison to the scores of randomly initialized clusters and the "ground truth" scores of the dataset's classes. The MICDs were rescaled to a unit range for all condition, to visualize all of their values, initially of different ranges, in one plot. Moreover, the plotted values are the means across trials, excluding any already terminated trials. Since trials terminated at different iterations, the scores at the higher iterations are the means of *fewer* values which explains the increase in variance. For the MICDs, a downward trend expontially decaying over the iterations can be observed for all conditions. Conversely, the NMIs across conditions increased over the first iteration, after which almost only very slight improvements occurred. About $\frac{2}{3}$ of change in the MICD and NMIs occured within the first three iterations.

2.6 Experiment 3: Above-phoneme

The second clustering experiment aimed to structure the conceptors C^+ in a hierarchy through an adaptation of HAC. Whereas in Experiment 2, symbols were identified among ESN states grouped by phoneme utterance (more concrete subsymbols), Experiment 3 aims to identify symbols among ESN states grouped by phoneme (more abstract subsymbols). Reiterating, symbols represent entities that

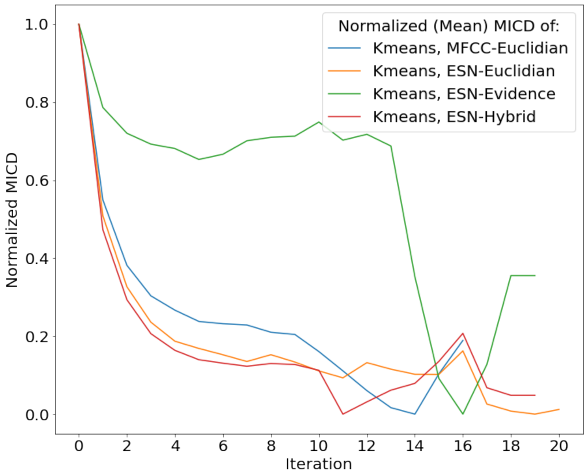

Figure 2.3: The MICDs of the clusterings over the iterations averaged across the trials.

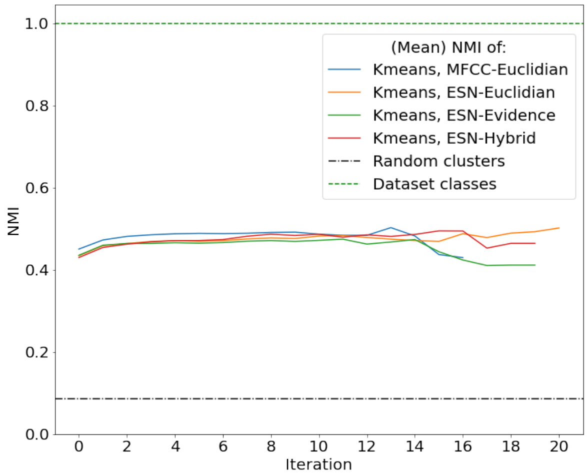

Figure 2.4: The NMIs of the clusterings over the iterations averaged across the trials. For reference, the ideal NMI of clusters corresponding to the dataset's classes (*Dataset*) and the baseline NMI of randomly initialized clusters (*Random*) are shown as horizontal lines.

themselves can be groups symbols. Indeed, many domains, like Phonetics, are conceptually structured as hierarchies or taxonomies of symbols of increasing levels of abstraction. This experiment is an attempt to distill such a symbolic abstraction hierarchy from a set of conceptors (one per phoneme) proxying for patterns among the corresponding ESN states.

Method The following three components were used to adapt traditional HAC to the clustering of conceptors:

1. The set of points to be clustered consisted of one conceptor embedding per phonemic class of the corresponding samples in the training set. Therefore, the set of positive conceptors from Experiment 1 was reused:

$$D_{HAC} = C^{+} = \{C_p^{+} | p \in P\} \tag{2.20}$$

2. The dissimilarity measure between points p_i and p_j was the negative logarithm of the conceptor similarity of p_i and p_j:

$$d(p_i, p_j) = 1 - \text{Sim}(p_i, p_j) \tag{2.21}$$

   This function fulfils several desirable properties for a dissimilarity measure: It is non-negative, symmetric, and equals 0 for identical inputs.

3. The linkage function $d_{link}(Cl_i, Cl_j)$ between clusters Cl_i and Cl_j is the mean pairwise dissimilarity of the points in the clusters. This is the defining function of *average* linkage algorithms:

$$\begin{aligned} &d_{link}(Cl_i, Cl_j) \\ &= \frac{1}{|Cl_i||Cl_j|} \sum_{p_x \in Cl_i, p_y \in Cl_j} d_{HAC}(p_x, p_y) \end{aligned} \tag{2.22}$$

From these components, the HAC adaptation was performed as detailed in Algorithm 2.3.

Results The resulting hierarchy is depicted as a dendrogram in Figure 2.5. Each leaf on the left represents a phonemic group. Moving toward the right, phoneme clusters emerge. The dissimilarities between merged child clusters is the abscissa of their link.

Algorithm 2.3 Generalized HAC

```
[1] Set of points $D = \{p_1, p_2, ..., p_n\}$
[2] Dissimilarity function $d(p_i, p_j)$
[3] Linkage function $d_{link}(Cl_i, Cl_j)$

Initialize a cluster for each point: $Cl \leftarrow D$

while number of clusters > 1 do
  Find the two most similar clusters:
     $Cl_i, Cl_j \leftarrow \underset{Cl_i \neq Cl_j}{\arg\min}\, d_{link}(Cl_i, Cl_j)$
  Update clusters in $Cl$:
     Remove $Cl_i$ and $Cl_j$
     Add $Cl_{merged} = Cl_i \cup Cl_j$
end while
```

Group		**Phonemes of Group**
Conso-nants	Fricatives	th f sh z s **dh v hh**
	Affricatives	jh ch
	Plosives	p b d t g k **dx**
	Nasals	m n ng
Vowels		ow aa oy ah uh er aw ay ey eh ae iy ih uw
Mixed		**l r w y**
Silence		h#

Table 2.3: Linguistic groups found within the phoneme clusters that resulted from HAC.

Several overlaps can be identified between the HAC phoneme clustering results and phonetic groups depending on the choice of phonetic model. I will compare the clusters with the classes provided by Pfeifer and Balik (2011) and shown in Figure 1.3, for their model is based on manner of production that seems the variable most correlated to the HAC clusters. Table 2.3 depicts these overlaps with the phonetic groups in the left column and their associated phonemes in the right column. Each group also corresponds to an HAC cluster with the exception of some **bold** phonemes that were moved between sibling clusters. The two primary clusters encompass consonants and vowels with a dissimilarity of about 0.015. Within the upper, consonants, cluster five subgroups can be identified. The red subcluster corresponds to fricatives and affricates (enclosed), both produced with air friction. The green subcluster encompasses plosives (top) and nasals (bottom). However, instead of the manner of production, the place of production may be considered as the group-determining variable. Thus, several other overlaps of the subclusters of consonants with phonetic groups may be found; the *bilabial* stops /p/ and /b/, the *alveolar* stops /t/ and /d/, the *velar* stops /k/ and /g/, and the *alveolar* fricatives /s/ and /z/ were each assigned to exclusive subclusters.

The lower cluster predominantly consists of vowel sounds. No significant correspondence between the sub-clusters of vowels and their pronunciation position or other articulatory features is apparent.

The phonemes in the Mixed group are considered separately, for their articulatory features resemble both vowels and consonants. The group contains the liquids /l/ and /r/ and semivowels /w/ and /y/. These instances all needed to be moved, since no cluster exclusively corresponded to the mixed group. Lastly, the silence /h#/ is considered separately like by Pfeifer and Balik (2011).

In summary, the table displays how the organization of phonemes derived from HAC coincides with traditional phonetic categories. The large resemblance suggests that the HAC algorithm produced relevant and mostly coherent clusters.

3 Discussion

Exp 1: Phoneme Classification Experiment 1 demonstrated phoneme classification using conceptors. [Comparison to phoneme classification benchmark...] [Comparison to Jaeger's experiments...]

Aperture normalization The methods of conceptor based time series classification used in Experiment 1 were largely inspired by (Jaeger, 2014) with the exception of the aperture adaptation. The first deviation here was to adapt explicitly only the apertures of the positive conceptors C^{+}...... **[normalizing/optimizing the aperture of the negative conceptors does not help]** It was made to

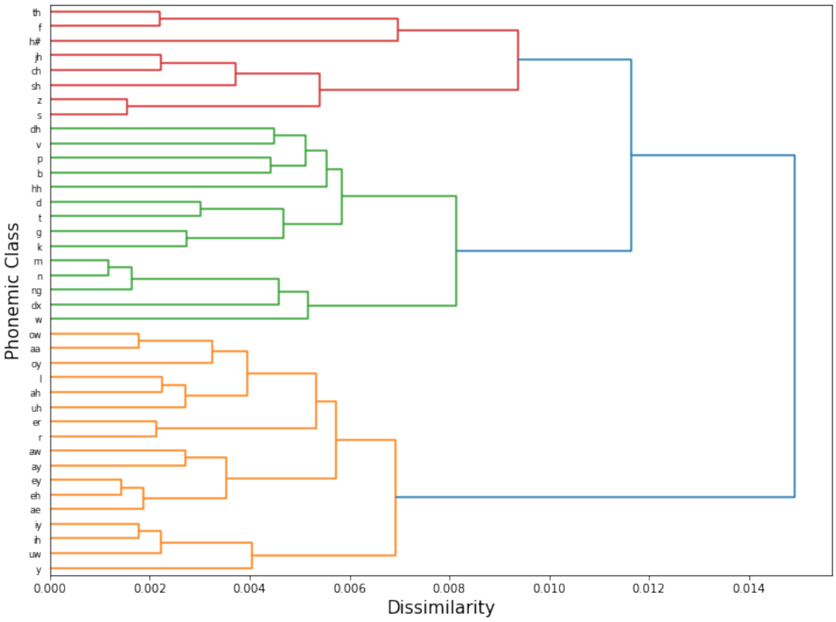

Figure 2.5: Dendrogram of the hierarchy that resulted from the adapted average linkage agglomerative clustering algorithm.

avoid to give conceptors with larger singular values an advantage during classification. To understand this effect, consider the example of a 2 × 2 conceptor matrix A. **[...]** To avoid such classification errors, the apertures were iteratively adapted until approaching a target singular value sum using the following procedure. Indeed this normalization further improved classification.

It must be remarked that, by that process, the final apertures will deviate from the one originally chosen aperture via the ∇-criterion. Figure 2.2 depicts this normalization process for the classification experiment.

Dataset Without normalization in time, performance decreases. How to handle different utterance speeds? Add leakiness to ESN to integrate time scales? Non-stationary signals

Z-Condition Worse due to overfitting: Exponential reason (AI2). The size of z and the associated computational costs are larger than for the former method, and it tends to lead to better *training* classification accuracy, but much worse test accuracy indicating overfitting.

Exp 2: Advantages of clustering method Conceptors do not [...]. As constant-sized matrices, they do not scale with the number of time steps or modeled states. This is a leap for analyzing time series since previous algorithms typically scale poorly with longer samples; for example, computing the distance between time series in dynamic time warping (DTW) scales quadratically with sample length dynamic. Notwithstanding, the quality of conceptors depends on the number of states used for their computation [...]. [Comparing responses of different lengths would require workarounds like Dynamic Time Warping (DTW).]

Third, ESNs deal with time-extended inputs causing several difficulties for preexisting clustering algorithms. For example, the euclidian distance can only be used to compare samples of different durations (for clustering algorithms that require distance functions) and have time-.

[Time complexity of clustering algorithms: especially Experiment 2]

[Mention esp might not be satisfied]

[Effects of spectral radius on ESP]

It should be stressed that, in the current work, *ESN activities* are classified and clustered using conceptors; Not speech signals, but an ESN's responses to these signals are used as input to classification and clustering algorithms.

Applications BPTT: To evaluate how well a classifier can distinguish between a set of classes, one may resort to performance metrics or investigate the degree of certainty in its predictions. However, such an evaluation relies only on the outputs and disregards the internal states that led to the predictions. For complex models or whole pipelines, it may be of interest to find groups or clusters present in the RNN's activations or within a subset thereof, e.g., corresponding to only one part of a pipeline. This study ought to be an exploratory step toward answering the following questions: How clustered are the activities within a NN? When, during training, do class differences in network activities arise?

Nonetheless, a phonetic interpretation about the speech signals will be made which, however, requires an assumption. Commonly, the echo state property is used to ensure a functional relationship between driver signal (phoneme recording) and RNN response Yildiz et al. (2012). However, for the echo state to be reached requires left-infinite or at least considerably long input sequences, for the starting state to be washed out. In our case, inputs will be downsampled to a length of 10 rendering the claim of the ESP impossible based on current theory. The analyzed states will most likely still correspond to the network's initial transient period. Thus, functionality between input and response cannot be guaranteed but only assumed. Hence, we shall nonetheless use the results of clustering and classification of ESN activities from the transient period for an interpretation about the input signals.

The current algorithm is not guaranteed to converge since it is an adaptation of K-Means that uses a non-Bregman divergence function (more on this in the comment under Experiment 2).

Adv of conceptor-based clustering to classical: Moreover, they compress one or more network states into a mathematical object (a matrix). These Another advantage, This hurdle can m and one advantage applying conceptors to time series is, as will be further argued, that they capture their information in a static mathematical compressing the temporal dimension and

Assumptions of generalized clustering: Banerjee, Merugu, Dhillon, Ghosh, and Lafferty (2005) bregman divergence, zero dist function when coinciding points and centroids.

To summarize, the three main motivations of the following clustering methods are to provide additional.

1. Explainability: Clustering RNN dynamics could help explain what groups of activity are present in the network. For example, both algorithms could be helpful tools for interpreting BPTT-trained RNNs. For example, Experiment 2 could show how classes become more distinct or numerous within the RNN (form activity clusters) while training.

2. Data analysis: Clustering time series with traditional clustering algorithms can be challenging. For instance, due to varying input lengths, methods for embedding time-series data are rather limited (e.g., latent-space encodings). This paper shows that the clusters present in RNN dynamics can be tied back to clusters within the inputs used to produce these dynamics.

3. Conceptor Classification: To the end of successful clustering, a phoneme classification experiment (Experiment 1) will be performed. In this experiment, the current methods for conceptor-based classification are applied and extended.

   Conceptors as state Embeddings The temporal and spatial compression realized by conceptors, allows them to be used as embeddings of ESN responses. Unifying responses in conceptor-space, independently of their length and maintaining only their most important information, allows state sequences to be related and compared. [Examples of the use of conceptors as embeddings]

Classically, these algorithms are used on data in Euclidian space and their components are well-defined for that space (mean, Euclidian distance, manhattan distance, etc.). However, the activities within the ESN are subjected to non-linear transformations like *tanh* at each time step as is the case for most neural networks. I hypothesize that Euclidian distance functions may not be well-suited to capturing the distances between network states.

Moving to conceptor space will be an attempt to improve the clustering of ESN activities through a more adequate means for cluster representation, comparison, and assignment.

Attempted explanation of swallowing problem

Finally, one may object that, for unsupervised symbol acquisition, the use of hard clustering is very artificial and something akin to Hebbian learning should be preferred, something that resembles how the brain acquires symbols. However, often in computing, and especially reservoir computing for material constraints, learning may not be possible. Yet, one may want to identify the symbols prevalent in the "artificial mind".

... We attempt to solve this mismatch by mapping the network states to be clustered to conceptors since conceptors offer several tools for comparing and analyzing network states. By moving to conceptor space, the following adaptations were made.

1. plus becomes or: Firstly, the operations of Euclidian vector space were changed.

2. division become aperture adaptation

3. Distance becomes the one proposed by Jaeger (2014) OR...

Problems. Distance metric is rather unclear, lack of semantics. Average linkage: Linkage algorithms: Pros and Cons slide of video

Categories may be due to coarticulation / correlations

Importantly, this is Not a Bregman divergence, because cosine similarity does not fulfill the triangle inequality. Therefore, this algorithm is not guaranteed to converge.

4 Conclusions

Conceptors are well-suited to apply clustering algorithms to time-series and ESN dynamics.

References

David Arthur and Sergei Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, Joydeep Ghosh, and John Lafferty. Clustering with bregman divergences. *Journal of machine learning research*, 6(10), 2005.

Margaret A Boden. *The creative mind: Myths and mechanisms*. Psychology Press, 2004.

Paul Bricman, Dr Herbert Jaeger, and Dr Jacolien van Rij-Tange. Nested state clouds: Distilling knowledge graphs from contextual embeddings. Bachelor's thesis, University of Groningen, 8 2022.

Ilana Bromberg, Qian Qian, Jun Hou, Jinyu Li, Chengyuan Ma, Brett Matthews, Antonio Moreno-Daniel, Jeremy Morris, Sabato Marco Siniscalchi, Yu Tsao, et al. Detection-based asr in the automatic speech attribute transcription project. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

William G Chase and Herbert A Simon. Perception in chess. *Cognitive psychology*, 4(1):55–81, 1973.

Satchit Chatterji. Cut the carp! using context to disambiguate similar signals using conceptors. Bachelor's thesis, University of Groningen, 8 2022.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Herbert Jaeger. Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*, 2014.

Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

Peter Karsmakers, Kristiaan Pelckmans, Johan AK Suykens, and Hugo Van hamme. Fixed-size kernel logistic regression for phoneme classification. In *INTERSPEECH*, pages 78–81, 2007.

K-F Lee and H-W Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

Lerato Lerato and Thomas Niesler. Investigating parameters for unsupervised clustering of speech segments using timit. In *Twenty-Third Annual*

Symposium of the Pattern Recognition Association of South Africa, page 83, 2012.

Carla Lopes and Fernando Perdigao. Phoneme recognition on the timit database. In Ivo Ipsic, editor, *Speech Technologies*, chapter 14. IntechOpen, Rijeka, 2011. doi: 10.5772/17600. URL `https://doi.org/10.5772/17600`.

Mantas Lukoševičius. A practical guide to applying echo state networks. *Neural Networks: Tricks of the Trade: Second Edition*, pages 659–686, 2012.

Jessica Maye and LouAnn Gerken. Learning phonemes without minimal pairs. In *Proceedings of the 24th annual Boston university conference on language development*, volume 2, pages 522–533, 2000.

Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. pages 18–24, 01 2015. doi: 10.25080/Majora-7b98e3ed-003.

George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63 (2):81, 1956.

Till Mossakowski, Razvan Diaconescu, and Martin Glauer. Towards fuzzy neural conceptors. *IfCoLog Journal of Logics and their Applications*, 6(4):725–744, 2019. URL `https://collegepublications.co.uk/ifcolog/?00033`.

Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. In *ACM Turing award lectures*. 1975.

Vaclav Pfeifer and Miroslav Balik. Comparison of current frame-based phoneme classifiers. *Advances in Electrical and Electronic Engineering*, 9, 12 2011. doi: 10.15598/aeee.v9i5.545.

Michael D Plummer and László Lovász. *Matching theory*, volume 121. Elsevier, 1986.

Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 18th Iberoamerican Congress, CIARP 2013, Havana, Cuba, November 20-23, 2013, Proceedings, Part I 18*, pages 117–124. Springer, 2013.

Jamie Vlegels. Multivariate time series classification using conceptors: Exploring methods using astronomical object data. Bachelor's thesis, University of Groningen, 2022.

Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

A Appendix

A.1 Computing Projector Matrix

The following algorithm for computing projector matrix $P_{R(C) \cap R(B)}$ is taken from Jaeger (2014). C and B are conceptors of size N. We begin by computing the basis matrix $B_{R(C) \cap R(B)}$ as follows:

1. Compute the SVDs $C = U\text{diag}(s_1, ..., s_l, s_{l+1}, ..., s_N)U'$ and $B = V\text{diag}(t_1, ..., t_m, t_{m+1}, ..., t_N)V'$, where l and m are the indices of the last singular values if C and B not below the arbitrary threshold $\epsilon_{impr} = 10^{-10} \approx 0$; This approximation is to account for computational imprecision [**Is this the right word?**], but in theory, $s_{l+1}, ..., s_N$ and $t_{m+1}, ..., t_N$ should all equal to 0.

2. Let $U_{>l}$ be the submatrix of U made from the last $N - l$ columns of U, and similarly, let $V_{>m}$ consist of the last $N - m$ columns of V.

3. Compute the SVD $U_{>l}(U_{>l})' + V_{>m}(V_{>m})' = W \Sigma W'$, where Σ =diag$(\sigma_1, ..., \sigma_k, \sigma_{k+1}, ..., \sigma_N)$. Again, k is the index of the last singular value of C not below ϵ_{impr}.

4. $B_{R(C) \cap R(B)} = W_{>k}$, the submatrix of W consisting of the last $N - k$ columns of W.

Finally, $P_{R(C) \cap R(B)} = B_{R(C) \cap R(B)} B'_{R(C) \cap R(B)}$.

A.2 Dataset

Table A.1 lists the phone classes and their frequencies within the processed TIMIT dataset.

A.3 Additional Results of Experiment 1

A.4 Additions of Experiment 2: Hyperparameter Optimization

Hyperparameters $k_b, k_{W^{in}}, r$ and ρ were also tuned automatically using bayesian optimization procedure of the Bayesian Optimization python package. The optimization objective was to maximize the testing accuracy of phoneme classification in Experiment 1. Concretely, after 10 initial exploration steps, 40 optimization steps were taken. At

	Phone	Folded	#Training	#Test
	iy		6953	2710
	ih	ix	13693	4654
	eh		3853	1440
	ae		3997	1407
	ah	ax-h ax	6291	2343
	uw	ux	2463	750
	uh		535	221
	aa	ao	6004	2289
	ey		2282	806
	ay		2390	852
	oy		684	263
	aw		729	216
	ow		2136	777
	l	el	6752	2699
	r		6539	2525
	y		1715	634
	w		3140	1239
	er	axr	5453	2183
	m	em	4027	1573
	n	en nx	8762	3112
	ng	eng	1368	419
	ch		822	259
	jh		1209	372
	dh		2826	1053
	b		2181	886
	d		3548	1245
	dx		2709	940
	g		2017	755
	p		2588	957
	t		4364	1535
	k		4874	1614
	z		3773	1273
	v		1994	710
	f		2216	912
	th		751	267
	s		7475	2639
	sh	zh	2389	870
	hh	hv	2111	725
	h# (silence)	dcl tcl kcl bcl pcl pau epi q gcl	39467	14021
Σ	39	22	177080	64145

Table A.1: Phone labels and their frequencies. Column 1: Phone classes. Column 2: Phones that were originally present in TIMIT but folded into a class with the left-adjacent phone. Columns 3 & 4: Number of speech samples of each class. Bottom row: Sums of classes or samples.

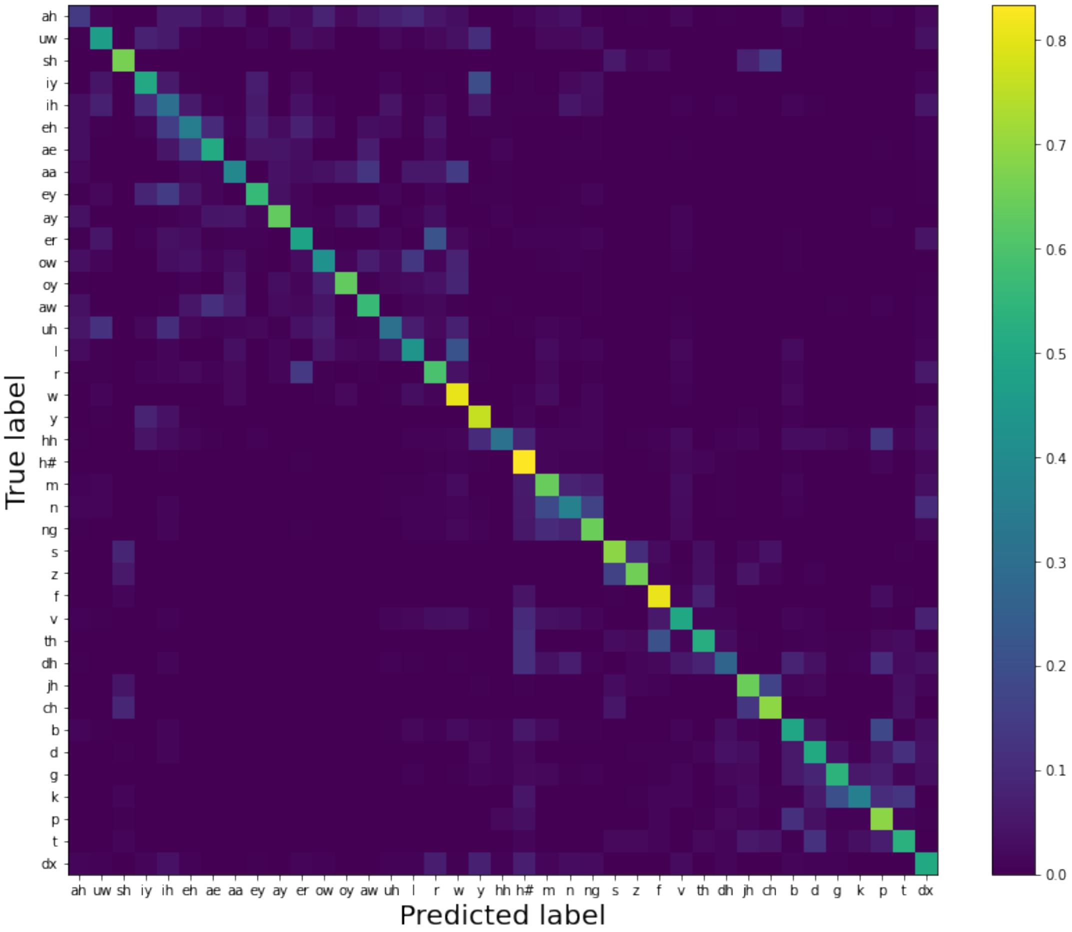

Figure A.1: Multi-class confusion matrix of the classification results. The colors represent the relative frequencies of the predictions (x-axis) made for each of the classes (y-axis). For every phoneme, their rate of correct classification (diagonal entries) was higher than the rate of misclassification to any of other classes (off-diagonal entries) suggesting an above-chance classification accuracy across all phonemic classes. Error rates seem elevated within groups of shared articulatory features like vowels (top left quadrant) and consonants (bottom right quadrant) [Elaborate if necessary].

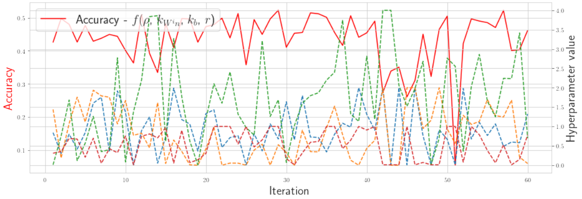

Figure A.2: Hyperparameter tuning with the bayesian optimizer and accuracy [Still no progress seems to occur. This experiment will need to be repeated.]

each optimization step, a set of hyperparameters is sampled from a promising region of the hyperparameter space trying to maximize an estimated surrogate $\bar{f}$ for the unknown objective function f [$f(\rho, k_{W^{in}}, k_b, r)$:=accuracy]. After training and testing the phoneme classifier with these hyperparameters on the training set (with a train-test split), the surrogate estimate is improved (for a detailed review of bayesian optimization, see Frazier). The hyperparameter space was restricted to:

- bias scaling parameter $b \in (0, 2)$
- input weight scaling parameter $k_{W^{in}} \in (0.01, 0.99)$
- spectral radius $r \in (0.01, 4)$
- internal weight density $\rho \in (0.01, 1)\%$

The progress of the bayesian optimizer is depicted in Figure A.2. Bayesian optimization was preferred over the more straightforward grid search, since, under consideration of the high computational complexity of training the classifier (about 30 minutes on my computer), the reduced number of training steps outweighed the overhead added by the bayesian optimizer.

A.5 Extension of Experiment 1: Inclusion of input states

Experiment 1 was repeated in slight adaptation of its methods to the stationary type of data. Jaeger (2014) demonstrated that conceptors may be used to classify signals produced by stationary and non-stationary processes. Stationary processes produce the same kind of signal (with the same probability distribution) over time; for example, white noise or sin waves are the results of stationary processes. Meanwhile, non-stationary processes change their properties over time leading to signals like speech whose probability distributions change over time. Meanwhile in the current classification method, the order within a sequence of states does not affect the resulting conceptor since it is lost when computing the correlation matrix. This works fine on signals from stationary sources where temporal order is of no relevance. However, for non-stationary sources, states can have different probabilities of occurring depending on their position in the sequence, information not accounted for in the current classification method. Jaeger (2014) approached this limitation by unrolling the ESN response $x(n)_n = 1, ..., L$ into a vector z reserving a dimension for each step in time. Moreover, the input signal s is appended to z for additional information. We have $z = [x(0); s(0); x(1); s(1); ...; x(L); s(L)]$. For z, the same classification procedure applies. New hyperparameters were picked by hand; the ESN size was reduced to $N' = 40$ neurons due to the larger computational complexity of this method, but the same density of $r' = 10\%$ was used. Scaling factors were changed to $k'_{W^{in}} = 1.5$ and $k'_b = 0.2$. A spectral radius of $\rho' = 1.5$ was used. A training accuracy of 63.64% and test accuracy of 49.13 were reached.

A.6 Additions of Experiment 2

Figures A.3 and A.4 relate the number of clusters to cluster quality as measured by the NMI and the SC scores, respectively.

Algorithm A.1 K-means++ improved initialization of centroids.

```
Require:
  Number of clusters $K$
  Set of points $D = \{p_1, p_2, ..., p_n\}$
  Dissimilarity function $d(p_i, p_j)$

  Choose first centroid $\mu_1$ uniformly at random from $D$

  for $k = 2$ to $K$ do
    For each point $p_i$, compute squared distance to nearest centroid**:
      $d_{min}(p_i) = \min_{0<j<k} d(p_i, \mu_j)^2$
    Choose $p_i \in D$ as the next centroid $\mu_k$ with
      probability $\frac{d_{min}(p_i)}{\sum_{j=1}^{n} d_{min}(p_j)}$
  end for

  return Set of centroids $\mu = \{\mu_1, \mu_2, ..., \mu_K\}$
```

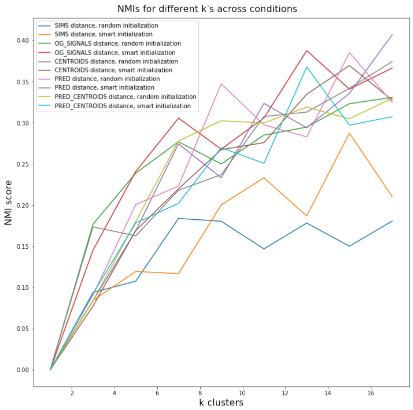

Figure A.3: NMI scores by values of k across conditions

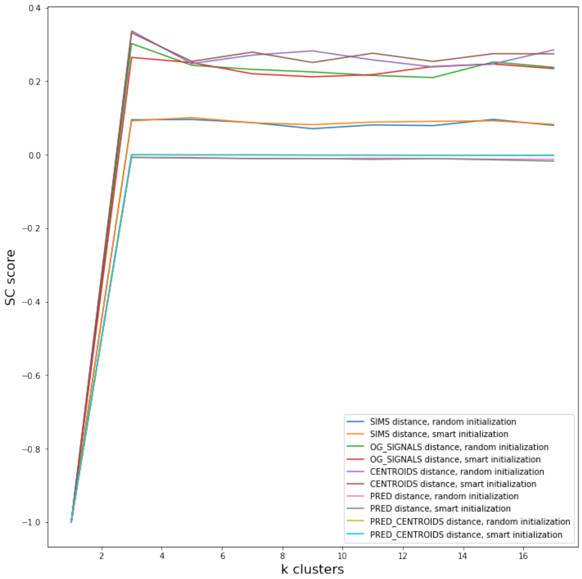

Figure A.4: SC scores by values of k across conditions

B Appendix

B.1 Mathematical notations

Notation	Meaning
A' or x'	Transposes of matrix A or vector x
I	$N \times N$ identity matrix, N to be inferred from context
$[x\|y]$	Matrix resulting from the column-wise concatenation of vectors x and y
$[x; y]$	Matrix resulting from the row-wise concatenation of vectors x and y
$A[:, y]$	Vector corresponding to the y'th column of matrix A
$A[x, :]$	Vector corresponding to the x'th row of matrix A
$\|S\|$	Cardinality of set S
$\|\|A\|\|$	Frobenius norm of matrix A
diag(A)	Vector containing the main diagonal of matrix A
$\|x\|$	Magnitude of vector x
$A^\dagger$	Pseudo-inverse of square matrix A
$R(A)$	Range of matrix A
$tr(A)$	Trace of square matrix A
P_S	$N \times N$ Projector matrix on the linear subspace S of $\mathbb{R}^N$, N to be inferred from context
$S \cap Z$	Intersection of linear spaces S and Z

Table B.1: List of some of the mathematical notations used throughout the paper.