Akanje in a Deep Boltzmann Machine

Searching for phonological categories in a deep unsupervised neural network trained on one specific case of vowel reduction in Russian

Anastasia Shchupak

Student number 13298321

BA Thesis Linguistics
Supervisor: Prof. P.P.G. (Paul) Boersma
University of Amsterdam
June 2023

Abstract

In recent years, advancements in Artificial Intelligence have enabled the development of increasingly sophisticated models capable of modelling different aspects of cognition. Phonology, as a purely cognitive process, can be studied using these new advancements. In this thesis, neural network modelling is employed to explore the underlying principles of phonological categorisation in vowel reduction. Specifically, a particular case of vowel reduction in Russian is selected as the primary focus of investigation. The study reveals two distinct phonological categorisations for language comprehension and language production. Additionally, the analysis identifies the specific position of the reduced vowel under examination within the phonological system.

Table of Contents

1 Introduction

1.1 Boundaries of phonology

The nature of phonology and its relation to phonetics has been a topic of discussion in the field of linguistics for several decades. The exact boundary between these disciplines, and even the existence of such a boundary, remains a point of disagreement: different theoretical models provide conflicting viewpoints.

Ohala (1990), for instance, argued against phonetic and phonology being two independent disciplines, advocating for their close integration rather than mere interfacing. Similarly, Flemming (2001) developed a unified model of phonetics and phonology, contending that these disciplines have similar properties and were artificially separated. Flemming suggests that the uncertainty surrounding the "hypothesized dividing line" between phonetics and phonology serves to emphasise the artificial nature of their separation. In his unified model, Flemming proposes the use of a weighted constraint system that incorporates both phonological and phonetic features.

Prince & Smolensky (1993, 2003) introduced a perspective that seemingly distinguishes phonology from phonetics by conceptualising phonology as a computational link connecting the lexicon and the phonetic form. Such a link in Prince & Smolensky's view serves to resolve the conflict between the demands of the lexical and phonetic interfaces through the generation of alternations. This proposal was made within the Optimality Theory (OT) framework (Prince & Smolensky 1993), depicted schematically in Figure 1. According to Prince & Smolensky (2003), in this model, markedness constraints support the phonetic interface and faithfulness constraints act as agents for the lexical interface.

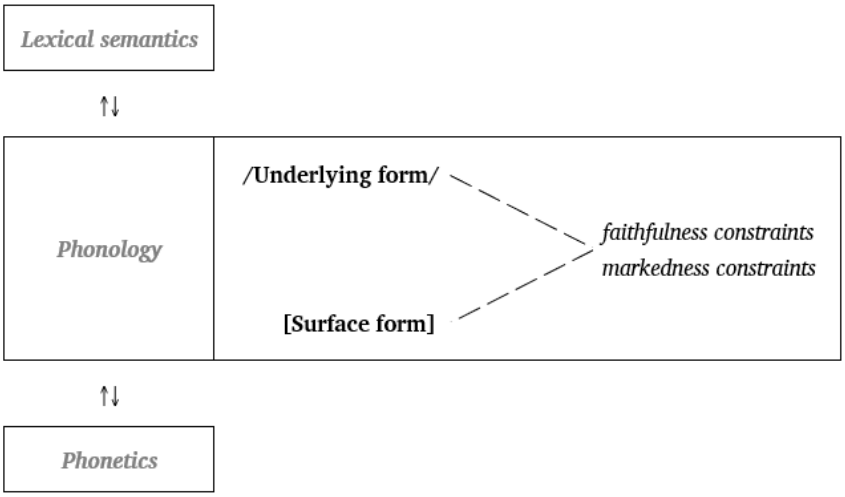


Figure 1. OT grammar model of Prince & Smolensky (1993, 2003)

While a computational perspective on phonology is an interesting and a very valuable viewpoint, the terminology used by Prince & Smolensky, specifically the term “interface”, can be somewhat confusing. The concept of an interface implies the existence of (at least) two distinct systems that communicate with each other. And addressing the issue of interface mismatch between the phonetic and lexical systems through the introduction of an intermediate phonology component would provide an elegant solution. This intermediate component could interface with both, phonetic and lexical, components. And inside, it could act as a translator, performing transformations that satisfy both interfaces. However, in their model, Prince & Smolensky do not present phonology as an independent system with separate interfaces to phonetics and lexical semantics. Instead, they include the phonetic and lexical interfaces within the phonological system itself, employing markedness and faithfulness constraints to resolve the tension between these interfaces. Consequently, phonetics extends beyond the confines of phonetic forms and penetrates the phonological grammar component, resulting in the coexistence of phonetic and phonological features in the surface phonological form. Thus, the boundary between phonetics and phonology in this model remains ambiguous.

Another grammar model within the framework of Optimality Theory, presented by Boersma (2011) and shown in Figure 2, introduces three distinct processing levels: semantic, phonological, and phonetic. This model, known as BiPhon-OT, effectively distinguishes between phonological and those processes that can affect phonological representations but are not phonological in nature. Essentially, BiPhon-OT entirely separates the domains of semantics, phonology, and phonetics, resulting in an increased number of processing levels while significantly reducing the complexity of each individual level.

Introducing a distinct semantic level of processing in BiPhon-OT was motivated by the challenges encountered in determining the correct underlying form (UF) during comprehension, as previously discussed by Smolensky (1996) and Hale & Reiss (1998). Smolensky (1996) proposed that constraints ranked for production would also be applicable for comprehension, suggesting that in the absence of markedness constraints (neutralised due to identical surface form (SF) in a comprehension process), the most faithful variant would be selected when evaluating all potential UF candidates. This solution offers a satisfactory explanation for the comprehension/production dilemma in a child language, which addresses the issue of why children exhibit more accurate comprehension earlier than production. However, it falls short when dealing with homophones, as highlighted by Hale & Reiss (1998). They criticised Smolensky’s approach, noting that in cases of surface phonological merger, the choice consistently favours the most faithful UF. They presented their own solution: retaining the entire list of possible UF candidates, which preserves valuable information during the phonological computation step. However, this means that the phonological

component of the grammar alone is incapable of resolving such ambiguity. Boersma (2011) suggests that this disambiguation is likely to occur at a higher level of processing. This concept was formalized in the BiPhon-OT grammatical model through the incorporation of a distinct level of semantic representations.

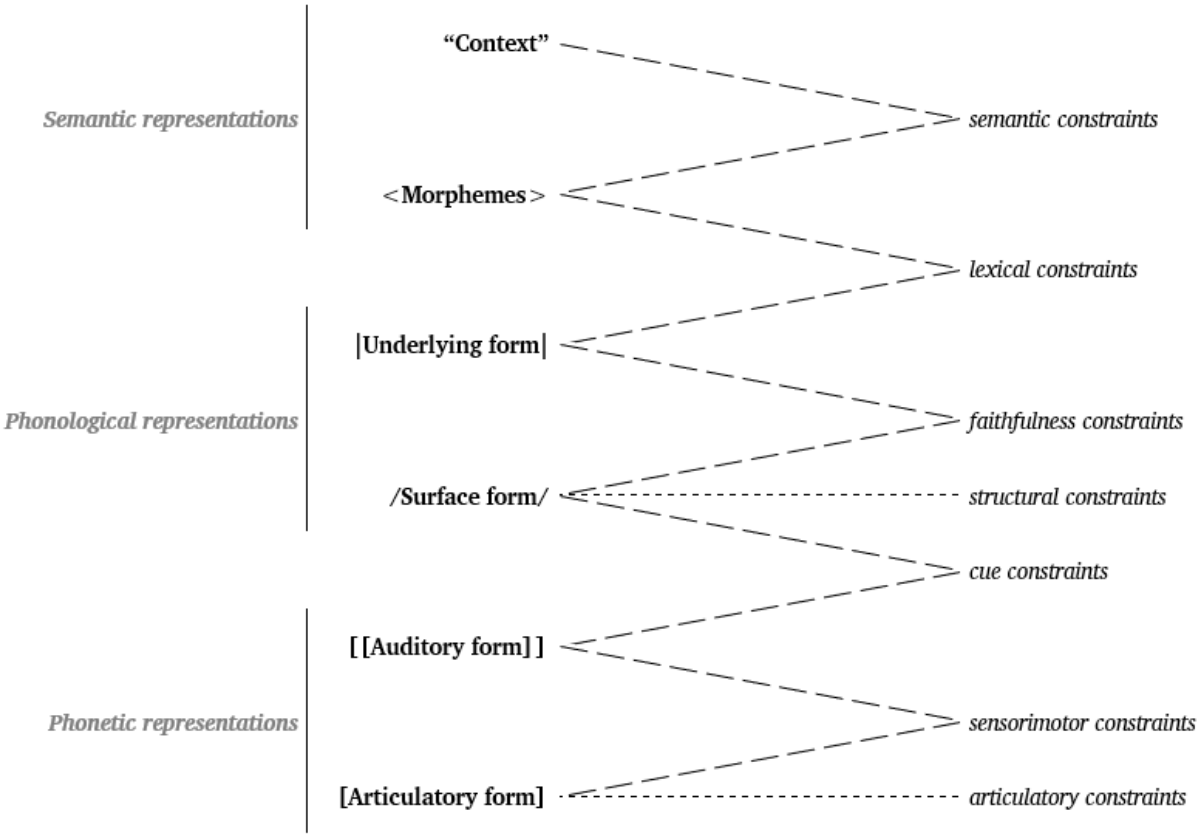


Figure 2. BiPhon-OT grammar model (Boersma 2011)

Additionally, in contrast to the Prince & Smolensky's (1993, 2003) approach, the BiPhon-OT grammar model maintains a distinct separation between phonology and phonetics, treating them as two independent systems. In BiPhon-OT, the phonological surface form is completely devoid of phonetic features and serves exclusively for phonological representations. This clear separation allows phonetics and phonology to exist as distinct systems, interconnected through the interface represented by cue constraints.

Two additional properties of the BiPhon-OT model, which are crucial for an effective grammar model, are bidirectionality and parallelism. Bidirectionality refers to using the same constraints, with the same rankings, in both the production and comprehension processes. Parallelism makes it possible for each constraint from each level to participate in a single

global ranking. This interconnectivity allows for feedback from later levels of processing to earlier ones, creating an interactive system (Boersma 2011).

Next to the question of separation of phonetics and phonology, there is another aspect of the relationship between these disciplines, namely the question whether phonology is phonetically grounded or not. Some researchers advocate for the phonetic motivation of phonology, arguing that phonological patterns are driven by phonetic factors (Ohala 1990; Browman & Goldstein 1992; Prince & Smolensky 1993, 2003). In contrast, others propose a substance-free perspective on phonology, positing that it is not directly tied to phonetic properties (Hale & Reiss 2000; Blaho 2008; Iosad 2013, 2017; Reiss 2017; Boersma, Chládková & Benders 2022; Reiss & Volenec 2022). There are also intermediate perspectives that propose a combination of phonetic grounding and abstract phonological principles (Hall & Teddiman 2014).

It is important to highlight that in case of being substance-free, in the absence of phonetic motivation, phonology becomes receptive to any form of phonetic variation, effectively using the interface between phonetics and phonology to facilitate the transmission of such variation to the phonological form. The mere possibility of accommodating any form of phonetic variation grants a significant advantage to the models developed within substance-free frameworks: their scalability. These models are capable to accommodate the continuous growth of linguistic data without requiring periodic modifications.

In summary, the distinguishing characteristics of phonological grammars include the separation of levels, the ability to process information in both top-down and bottom-up directions, and substance freedom. It is important to note that phonology is not directly observable in nature, but rather serves as an abstract framework aimed at explaining the cognitive processes that connect speech sounds and meanings in our minds. Therefore, when selecting a model, considerations should prioritise the ease of use, its ability to account for known phenomena, and scalability to accommodate expanding data. Taking these factors into account, the present thesis opts to employ a bidirectional system that effectively models both speech production and comprehension along with language acquisition (accounts for known phenomena), exhibits clear level separation (provides the ease of use), and demonstrates scalability by effortlessly incorporating additional data without the need for constant adaptation (is substance-free). The BiPhon-OT grammar model (Boersma 2011) fulfils all these criteria and is thus an appropriate choice.

1.2 Neural view on phonology

When considering the methods employed for studying phonology, it is important to remember that we are essentially modelling a cognitive process. In this regard, recent neurobiological

research can provide valuable insights. Popham et al. (2021) provide compelling evidence supporting the semantic alignment hypothesis. They discovered the coexistence of two interconnected types of cortical areas on both sides of the anterior border of the visual cortex. The inside-visual-cortex areas exhibit selectivity for specific visual categories, while the corresponding outside-visual-cortex areas specialise in processing semantic concepts of these visual categories in language. The presence of these specialised linguistic semantic areas within the cortex raises questions such as how information from these areas travels to the phonetic representation of corresponding semantic concepts in primary and secondary auditory cortex (Leonard & Chang 2014) and back. It is reasonable to hypothesize that this pathway may be where the abstract categories of phonology are formed and maintained.

In line with the advancements in neurobiology over the past few decades, the development of artificial intelligence has allowed new ways for studying the brain processes including those involved in phonology. Prince and Smolensky (1997) showed that optimisation principles in neural computation could be effectively applied to the theory of grammar. Reiss and Volenec (2022) take it a step further by postulating that phonology in the 20th century became neurologically oriented. Supporting a substance-free view on phonology, they argue that phonological symbols, being neither acoustic nor articulatory, are neurally encoded.

Consequently, one potential research approach involves using artificial neural networks to model phonological and phonetic phenomena. One example of such modelling is the simulation of the human brain's capacity for acquiring a language. By analysing the inner workings of such models, new hypotheses can be formed regarding how the connection between speech sound and meaning is organised within the brain. Boersma, Benders & Seinhorst (2020) propose such an artificial neural network model that accounts for both phonological and phonetic phenomena of a language, such as category creation and auditory dispersion. Another study by Boersma, Chládková & Benders (2022) introduces a deep neural network model along with the learning algorithm which enables the model to learn a simple toy language presented as a set of five $<$sound, meaning$>$ pairs. This model is referred to as BiPhon-NN and is demonstrated by the authors to serve as a neural-network counterpart of BiPhon-OT. Following training, the authors analyse the model's hidden levels and identify categories, emerging from both the phonetic and semantic inputs, which can be interpreted as phonological features.

The neural network type chosen by Boersma, Chládková & Benders (2022) for modelling is a deep Boltzmann machine (DBM) (Salakhutdinov & Hinton 2009). The deep architecture of DBMs, their capability to train on unlabelled data, and their capacity to generate new samples resembling the training data make them in general a valuable tool for studying cognitive processes. But even more specifically, as evidenced by Boersma, Chládková & Benders, DBMs are well-suited for the task of identifying potential phonological categories.

The goal of this thesis is to build on the work of Boersma, Chládková & Benders (2022) by using their model to learn a slightly more complex toy language that contains the phenomenon of vowel reduction. Specifically, the language under investigation is a small subset of Russian: twenty-two one- and two-syllabic words, representing a particular case of vowel reduction known as akanje. The focus of Section 2 will be on existing phonological analyses of akanje, concluding with a research question and a set of hypotheses. Section 3 will provide a comprehensive description of the methodology, including an overview of the model, a description of the training dataset, and details of the training algorithm. An analysis of the obtained results will be presented in Section 4. Section 5 and 6 will offer a discussion of the implications of the analysis and a conclusion, respectively.

In conclusion, it is important to address the notation used in this thesis, as different grammar models employ different notations for representing phonological forms. However, for the sake of consistency, one notation should be chosen. The OT grammar model (Prince & Smolensky 1993) uses forward slashes (*//*) for underlying forms and square brackets ([]) for surface forms. On the other hand, the BiPhon-OT grammar model (Boersma 2011) adopts pipes (||) for underlying forms and forward slashes (*//*) for surface forms, while reserving square brackets ([]) exclusively for phonetic representations.

Throughout the rest of this thesis, the BiPhon-OT notation will be used unless stated otherwise. This decision is based on the author's intention to differentiate between phonetic and phonological forms. However, when presenting accounts without such distinction, additional justification for the use of // or [] will be provided. Moreover, in accordance with the BiPhon-OT model, angle brackets (< >) will be employed for semantic forms.

2 *Akanje*, one specific case of vowel reduction in Russian

Phonemes |a| and |o|, when unstressed, can be realised differently across different East Slavic dialects. For example, speakers of some Russian dialects maintain a distinction between |a| and |o| when pronouncing |trava| 'grass' and |sova| 'owl' as [traˈva] and [soˈva] respectively. This distinction, however, is lost in some other Russian dialects, where both phonemes |a| and |o| are pronounced as a sound that is close to [a]. In such dialects, |trava| and |sova| might sound somewhat like [trɐˈva] and [sɐˈva], or perhaps [trʌˈva] and [sʌˈva]. This phenomenon of losing the distinction between phonetic realisations of the phonemes |a| and |o| in unstressed syllables in Russian is commonly referred to as *akanje* (Lunt 1979) or *akanye* (Enguehard 2019)[1].

[1] This phenomenon is also observed in Ukrainian and Belarussian dialects and referred to as *akannja* in Ukrainian and *akanne* in Belarussian (Lunt 1979).

In a broader context, akanje refers to a specific neutralisation pattern, involving the neutralisation of all non-high unstressed vowels (including |a|, |o|, and |e|) that result in a sound close to [a]. It is helpful to use two dimensions to categorise all possible kinds of this neutralisation: the presence or absence of palatalisation in the consonant preceding the neutralised vowel and the position of this vowel within the word (Iosad 2012: 522).

Regarding the first dimension (palatalisation of the context), neutralisations that occur after palatalised consonants are also referred to as *jakanje* (Kniazev & Shaulskiy 2007) or *yakanye* (Enguehard 2019). Some accounts classify these neutralisations apart from akanje, as seen in Enguehard's overview of Russian vowel reductions (Enguehard 2019: 114-116). Regarding the second dimension (position within the word), existing analyses of all types of Russian vowel reduction, including akanje, unanimously agree on the existence of two degrees of reduction: *moderate* reduction, occurring in the syllable immediately preceding the stress, and *extreme* reduction, occurring in other unstressed syllables. These terms were coined by Crosswhite (2000a, 2000b), while other terms such as *moderate* and *radical* reduction (Iosad, 2012) or *Degree I* and *Degree II* reduction (Barnes, 2007) can also be found in the literature.

The focus of this thesis is specifically on the moderate form of akanje in its narrow sense, excluding jakanje. Consequently, the analysis is limited to pretonic syllables and to the non-palatalized context. This means that the analysis essentially addresses the neutralisation of the phonemes |a| and |o|, because the third possible non-high phoneme in Russian, |e|, undergoes a reduction to [ɨ] in non-palatalized contexts, rather than to [a], as shown in Figure 3 (Iosad, 2012: 525-526).

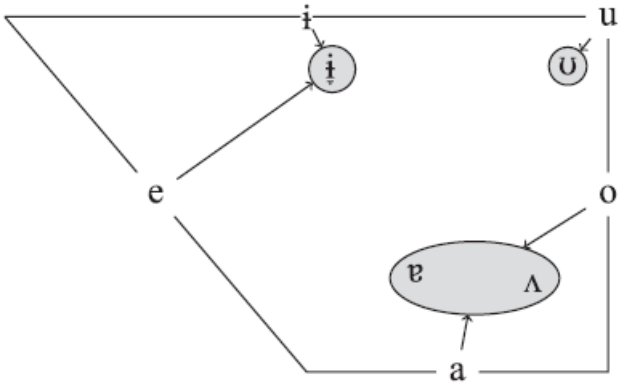


Figure 3. Moderate reduction, non-palatalised context (Iosad 2012, p. 526)

Despite the focus on the moderate degree of reduction, the following subsection will provide an overview of existing analyses of akanje, considering both moderate and extreme degrees. This is because some researchers view the boundary between phonetics and phonology, as mentioned in the previous section, precisely between these two degrees of reduction.

2.1 Existing phonological accounts of akanje

Phonological analyses of akanje vary from substance-full, accounting for all phonetic underpinnings, to substance-free, separating phonetic and phonological phenomena. This section aims to provide an overview of four existing analyses, starting with two substance-full approaches, and concluding with two substance-free approaches.

Presenting analyses that lack clear separation between phonology and phonetics, while using the notation that distinguishes phonological surface and phonetic forms (SF and PF, respectively), might be challenging. This is because in such analyses, SF gives place to both phonological and phonetic features. For example, despite the phoneme |ɐ| not being part of the Russian phonological inventory, it does appear in the SF of these substance-full analyses. It can be explained by the fact that SF in these analyses aims to reflect the actual sound using the International Phonetic Alphabet (IPA). Considering this motivation, I will use square brackets to denote SF when describing substance-full analyses below.

According to substance-full phonological accounts of akanje, the phonemes |a| and |o| are realised as [a] and [o], respectively, in SF of stressed syllables. However, in unstressed syllables, both phonemes undergo reduction and are realised as either [ɐ] or [ə]. The reduced form [ɐ] occurs in syllables immediately preceding the stress, indicating a moderate degree of reduction. The other reduced form [ə] appears in other unstressed positions, indicating an extreme degree of reduction. Examples illustrating these two degrees of reduction for |a| and |o| are provided in (2.1a) – (2.1d).

(2.1) *Two degrees of vowel reduction in some dialects of Russian: moderate* [ɐ] *and extreme* [ə]

a.	|s**o**va| [s**ɐ**ˈva]	‘owl’
b.	|s**a**mo| [s**ɐ**ˈmo]	‘itself’
c.	|g**o**l**o**va| [g**ə**ɬ**ɐ**ˈva]	‘head’
d.	|p**a**r**a**doks [p**ə**r**ɐ**ˈdoks]	‘paradox’

Crosswhite (2000a) proposed an analytical approach to explain these two reduction patterns, suggesting that they arise from two distinct types of reduction that serve different linguistic purposes: one to improve vowel contrast, and the other to increase articulatory ease. In a subsequent paper, Crosswhite (2000b) observed that reduced duration often leads to a decrease in vowel sonority and proposed using the sonority scale, together with foot form and processes of lengthening under stress, to account for the two distinct phonological categories of reduction. She argued that her analysis demonstrates the phonetic motivation behind these two phonological patterns (Crosswhite 2000b: 154). However, being a substance-full account, it does not identify any inherent problems associated with it.

Padgett (2004) employs the Dispersion Theory of Contrast (Flemming 2002) to account for both patterns of akanje. According to Flemming, phonology is influenced by both articulatory and auditory constraints. His dispersion theory of contrast suggests that phonology maximises the number and distinctiveness of contrasts while minimising articulatory effort. However, Padgett's analysis did not completely support the Dispersion Theory. It utilized quantitative phonetic data obtained from an experiment (Padgett & Tabain 2005), which examined more patterns of Russian vowel reduction in addition to akanje. The phonetic data showed inconsistent distances between the reduced vowels.

Enguehard (2019) presents a different explanation for akanje's phonology, which does not involve any phonetic substance. He suggests that the Russian vowel reduction can be explained as a quantitative difference between stressed and unstressed syllables in terms of phonological length, rather than a qualitative change in the sound of the vowel. This expands on the proposal of Crosswhite (2000b), who suggested that the sonority of vowels is influenced by the presence of a mora in stressed syllables and in syllables directly preceding stressed ones. Instead of moras, Enguehard uses the concept of "skeletal slots" to describe vowels, claiming that there are two of them in a stressed position and one in an unstressed position.

Barnes (2007) introduces a distinction between phonological and phonetic processes involved in akanje. According to Barnes, the neutralisation of |a| and |o| in unstressed syllables occurs in two distinct steps. The first step is purely phonological, involving the reduction of |a| and |o| to /a/, which is motivated by stress as an abstract structural factor. In the second step, which is purely phonetic, a raising process takes place: /a/ is raised to [ɐ] or [ə] in syllables undergoing extreme and moderate reduction, respectively. This raising process is driven by the articulatory challenge of producing a low vowel, which arises due to the reduced duration of the vowel.

Barnes' suggestion has been implemented within the framework of Optimality Theory in two ways. The first implementation was proposed by Iosad (2012), who shares Barnes' view of the phonological and phonetic aspects of akanje. Iosad employs the Parallel Structures Model of feature geometry (Morén 2003, in: Iosad 2012), offering a comprehensive and straightforward OT analysis of Barnes' phonological account of akanje.

The second implementation, carried out by the author (Shchupak 2022), employed the BiPhon-OT model (Boersma 2011). The analysis yielded successful results in both moderate and extreme contexts. Examples of the OT tableaus can be observed in Figures 4 and 5. Figure 4 illustrates a tableau for the perception of [s***X***v***Y***], where ***X*** and ***Y*** are sounds, the phonetic representation of which includes their first formant frequency and duration. ***X*** and ***Y*** can be denoted as follows: ***X*** = [700Hz, 70ms] and ***Y*** = [700Hz, 80ms]. This showcases how stress is

entering phonology from acoustic data. The list of candidates for this tableau includes various permutations of the potential phonemes /a/ and /o/, along with different stress patterns for each permutation. Figure 5 presents a tableau for the production of <head>|golova|, which displays the parallel working of four distinct levels in BiPhon-OT: <semantic form>, |underlying form|, /surface form/, and [phonetic form]. While not being exhaustive, this tableau serves to demonstrate the parallel functionality of the complete model.

[s***X***v***Y***] ***X*** = [700Hz, 70ms] ***Y*** = [700Hz, 80ms]	/*o/$_{NT}$	*/a/$_{T}$[470Hz]	*/o/$_{T}$[700Hz]	*/a/$_{NT}$[80ms]	*/a/$_{T}$[25ms]	*/a/$_{T}$[70ms]	*/o/$_{T}$[25ms]	*/o/$_{T}$[80ms]	*/a/$_{NT}$[470Hz]	*/a/$_{NT}$[25ms]	*/a/$_{NT}$[700Hz]	*/a/$_{NT}$[70ms]	*/o/$_{T}$[470Hz]	*/o/$_{T}$[70ms]	*/a/$_{T}$[700Hz]	*/a/$_{T}$[80ms]
/.ˈsa.va./[s***X***v***Y***]				*!		*					*				*	
☜/.sa.ˈva./[s***X***v***Y***]											*	*			*	*
/.ˈso.va./[s***X***v***Y***]			*!	*							*			*		
/.so.ˈva./[s***X***v***Y***]	*!														*	*
/.ˈsa.vo./[s***X***v***Y***]	*!					*									*	
/.sa.ˈvo./[s***X***v***Y***]			*!					*								
/.ˈso.vo./[s***X***v***Y***]	*!		*											*		
/.so.ˈvo./[s***X***v***Y***]	*!		*					*								

Figure 4. BiPhon-OT analysis of perception of [s***X***v***Y***] (***X*** = [700Hz, 70ms], ***Y*** = [700Hz, 80ms]) (performed in 2022)

<head>|golova| ***X*** = [470Hz, 25ms] ***Y*** = [470Hz, 70ms] ***Z*** = [470Hz, 80ms] ***U*** = [700Hz, 25ms] ***V*** = [700Hz, 70ms] ***W*** = [700Hz, 80ms]	* < > |M|	/*o/$_{NT}$	*[low, short]$_{ART}$	*/X/$_{2PT}$ [long]	*/a/$_{T}$[470Hz]	*/o/$_{T}$[700Hz]	*/a/$_{NT}$[80ms]	*/a/$_{T}$[25ms]	*/a/$_{T}$[70ms]	*/o/$_{T}$[25ms]	*/o/$_{T}$[80ms]	IDENT-IO (V_T)	*/a/$_{NT}$[470Hz]	*/a/$_{NT}$[25ms]	*/a/$_{NT}$[700Hz]	*/a/$_{NT}$[70ms]	*/o/$_{T}$[470Hz]	*/o/$_{T}$[70ms]	*/a/$_{T}$[700Hz]	*/a/$_{T}$[80ms]
<head>|golova| /.go.lo.ˈva./[g***X***l***Y***v***Z***]		* *!			*															*
<head>|golova| /.ga.la.ˈva./[g***U***l***V***v***W***]			*!												*		*			*	*
<head>|golova| /.ga.la.ˈvo./[g***X***l***V***v***Y***]												*!	*	*	*	*	*	*		
☞<head>|golova| /.ga.la.ˈva./[g***X***l***V***v***W***]													*	*	*	*			*	*
<head>|golova| /.ga.la.ˈva./[g***V***l***V***v***W***]				*!											*	*				

Figure 5. BiPhon-OT analysis of production of *<head>|golova|* (performed in 2022)

The tableaus in Figures 4 and 5 incorporate the following constraints, listed top down as represented in BiPhon-OT:

1. *Lexical constraint* *<>|X|, which prevents underlying forms that are not associated with any morpheme.
2. *Faithfulness constraint* IDENT-IO (V_T), which deprecates changes to the tonic vowel.
3. *Structural constraint* */o/$_{NT}$, which disallows unstressed /o/ in the surface form.
4. A series of *cue constraints*, listed in Table 1.
5. *Articulatory constraint* *[low, short]$_{ART}$, which militates against short vowels being low.

Table 1. Cue constraints in BiPhon-OT analysis of akanje (performed in 2022)

Super high-ranked	**High-ranked**		**Middle-ranked**		**Low-ranked**	
	F1	**Duration**	**F1**	**Duration**	**F1**	**Duration**
*/X/$_{2PT}$ [long]	*/a/$_{T}$[470Hz] */o/$_{T}$[700Hz]	*/a/$_{NT}$[80ms] */a/$_{T}$[25ms] */a/$_{T}$[70ms] */o/$_{T}$[25ms] */o/$_{T}$[80ms]	*/a/$_{NT}$[700Hz]	*/a/$_{NT}$[25ms]	*/a/$_{NT}$[470Hz] */a/$_{T}$[700Hz] */o/$_{T}$[470Hz]	*/a/$_{NT}$[70ms] */a/$_{T}$[80ms] */o/$_{T}$[70ms]

Here follow the examples of interpreting the cue constraints. The cue constraint */X/$_{2PT}$ [long] puts restrictions on the duration of the second pretonic syllable: it cannot be long, specifically not 70 or 80 milliseconds in the given context. The cue constraint */o/$_{T}$[700Hz] indicates that the occurrence of /o/ in a stressed syllable is prohibited when the vowel has a first formant frequency (F1) of 700Hz. Similarly, the cue constraint */a/$_{NT}$[25ms] specifies that /a/ in an unstressed syllable is not allowed when the vowel has a duration of 25 milliseconds.

In summary, this subsection presented an overview of four different accounts of akanje. Two of these accounts follow a substance-full approach and do not demonstrate a clear separation between phonology and phonetics. In contrast, the other two accounts adopt a substance-free perspective, clearly distinguishing between phonology and phonetics. Notably, one of the substance-free accounts was implemented twice using two different frameworks within Optimality Theory: the parallel structures model of feature geometry and the BiPhon-OT model.

2.2 Akanje in a deep Boltzmann machine

Section 1.2 discusses the potential advantage of using neural networks in phonological research, specifically highlighting effectiveness of DBM in detecting phonological categories. The objective of this thesis is to use a neural network language grammar model to explore underlying principles of phonological categorisation in akanje. The procedure will include building a DBM model capable of learning a language consisting of a set of Russian words that represent the phenomenon of akanje (specifically, its moderate part), training the model on this language, and subsequently analysing the hidden levels of the model in an attempt to identify potential phonological categories.

The analysis will address the following questions: will distinct categories for different sounds emerge? If so, will the neutralised realisation of |a| and |o| form its own category? If such a category is found, will it exhibit phonological similarities with |a|, |o|, or an equal degree of similarity to both[2]? Expectations vary depending on what influences the phonological representation more: phonetic or semantic representation. In the case of strong phonetic influence, the new category will be more similar to |a|, because the neutralised sound is phonetically closer to [a]. In the case of strong semantic influence, the similarity will differ for different words, depending on which phoneme is neutralised. For example, in the word <owl>|sova|[sɐˈva] (2.1a), the phonological category of [ɐ] when will be more similar to |o|. On the other hand, in the word <itself>|samo| [sɐˈmo] (2.1b), the phonological category of [ɐ] will be more similar to |a|.

3 Methodology

3.1 Deep Boltzmann machines

A deep Boltzmann machine (DBM) is a deep artificial neural network that serves as a generative model with multiple layers of hidden variables and is capable of training without supervision (Salakhutdinov & Hinton 2009; Salakhutdinov 2010; Salakhutdinov & Larochelle 2010). It consists of three or more levels of nodes with layers of bidirectional connections between nodes in different levels, while nodes within the same levels remain unconnected. Figure 6 illustrates a schematic representation of a DBM used by Boersma, Chládková & Benders (2022) in their research, consisting of two hidden levels and an input level.

[2] To measure phonological similarity between two utterances, the *cosine similarity* of the network's hidden levels will be computed after it has been exposed to the respective utterances, following the method described by Boersma, Chládková & Benders (2022: 28).

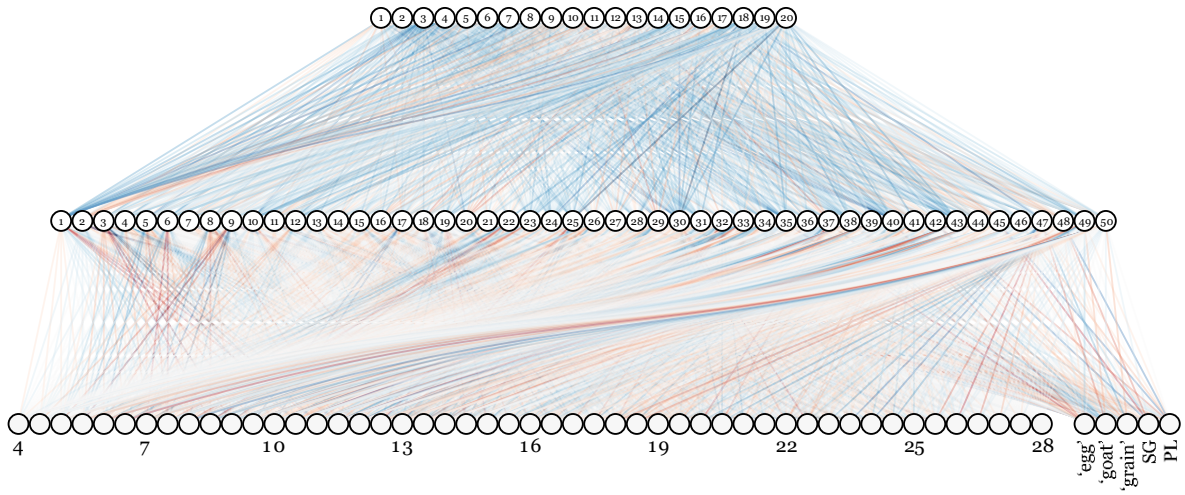


Figure 6. Boersma, Chládková & Benders' (2022) DBM

The input level of their DBM comprised 54 nodes, with 49 nodes dedicated to auditory representation and 5 nodes dedicated to semantic representation. The auditory nodes were designed to represent the part of the basilar membrane relevant for hearing F1 and F2, covering a spectral continuum ranging from 4.0 to 28.0 ERB. This choice was made because the phonetic representation of utterances in the researched toy language contained exclusively vowels. In Figure 6, the colours representing connections between nodes indicate that the network has already undergone training. The connections between levels are depicted in varying intensities of red and blue, where brighter red signifies a stronger positive weight of connection, while brighter blue signifies a stronger negative weight of connection.

This network can be trained using a generated sequence of utterances from the toy language, encoding both auditory and semantic information into the input level for processing. Following training, the model can be utilised by exposing it to a specific input and considering the input level to be an output level as well (if unclamped). In this manner, the model can display its "comprehension" or "production" capabilities, depending on the type of information used as an input – either auditory or semantic. Boersma, Chládková & Benders demonstrated that their model successfully learned the toy language. When only auditory information was provided, their model accurately restored the corresponding semantic information, and vice versa. Additionally, the analysis of hidden levels of the network revealed the emergence of categories, leading to the assumption that these categories might represent phonological categories.

3.2 Training dataset

3.2.1 Toy language with akanje

A toy language used in this thesis consists of a set of twenty-two Russian words that represent the phenomenon of akanje in its moderate form (as described in Section 2). It comprises eight pairs and two triplets of lexically equivalent words, as shown in Table 2. For both phonemes |o| and |a|, there are two pairs of words demonstrating akanje, two pairs of words without akanje, and one triplet in which akanje occurs in two out of the three words. This approach ensures that each pair or triplet of words always includes one word without akanje, serving as a reference.

Table 2. Toy language inventory

	Russian			***phonetic form***			***meaning (lexical)***
	feminine	*masculine*	*plural*	*feminine*	*masculine*	*plural*	
|o|	сова		совы	[sɐvˈa]		[sˈovɨ]	<*owl*>
	роза		розы	[rˈozə]		[rˈozɨ]	<*rose*>
		кот	коты		[kot]	[kɐtˈɨ]	<*cat*>
		бот	боты		[bot]	[bˈotɨ]	<*bot*>
	моя	мой	мои	[mɐjˈa]	[moj]	[mɐjˈɨ]	<*mine*>
|a|	жара	жар		[ʐɐˈra]	[ʐar]		<*heat*>
	жаба		жабы	[ʐˈabə]		[ʐˈabɨ]	<*toad*>
		таз	тазы		[taz][3]	[tɐzˈɨ]	<*basin*>
		бар	бары		[bar]	[bˈarɨ]	<*bar*>
	раба	раб	рабы	[rɐˈba]	[rab][4]	[rɐˈbɨ]	<*slave*>

Each word can be analysed as having two parts: a stem that represents the lexical meaning of the word and consists of three sounds (CVC), and possibly a suffix consisting of one vowel

[3] Final-obstruent devoicing is ignored for simplicity (in real Russian it would be [tas])

[4] Final-obstruent devoicing is ignored for simplicity (in real Russian it would be [rap])

(V), indicating gender or plurality. For example, the suffixes [a] and [ə] are associated with feminine forms, the suffixes [ɨ] and [ï] indicate plural forms, and the masculine forms are characterised by the absence of a suffix. Interestingly, in the case of suffixes, [ə] is a reduced form of [a], and [ï] is a reduced form of [ɨ], both representing the extreme form of vowel reduction. However, it is mentioned here only for the sake of completeness and will not be the focus of the further analysis. Furthermore, throughout the rest of this thesis, the absence of a suffix is interpreted as a null sound [∅], rather than the absence of one.

The vowel inventory of the created language is presented in Table 3. Vowels can appear in the first or second syllable and can be either stressed or unstressed. Although the moderate form of akanje exclusively occurs in the first syllables, the phonetic qualities of the vowels in the second syllables are also accurately preserved. This preservation is important because, as demonstrated by Chrabaszcz et al. (2014), vowel quality is the most influential acoustic cue for stress perception in Russian. Other cues, such as pitch, intensity, and duration, were also investigated by Chrabaszcz et al. and found to have much less influence than vowel quality. This is good news for modelling the auditory input for the network, as it means that including additional representations for pitch, intensity, and duration is not necessary. However, it also implies that the acoustic quality of all vowels, rather than only the vowels under investigation, should be explicitly preserved to account for stress position in the word.

Table 3. Vowel inventory

	stressed syllable	*unstressed syllable*
first syllable	[a] [o]	[ɐ]
second syllable	[a] [ɨ]	[ə] [ï]

Once the properties of the toy language have been established, the next step in creating a training dataset for the DBM involves defining word representations that provide valuable input for the network to acquire this toy language. Following the approach taken by Boersma, Chládková & Benders (2022), the input for the network is divided into auditory and semantic components. The step-by-step construction of these two components is presented in Sections 3.2.2 and 3.2.3, with examples of the complete input representations showcased in Section 3.2.4. Throughout the rest of this thesis, the term toy language will be used to refer to the toy language constructed for this thesis. Whenever referring to the toy language from Boersma, Chládková & Benders (2022), it will be explicitly mentioned.

3.2.2 Auditory input representation

The auditory input encoding a word from the toy language should consist of data representing four sounds: two consonants and two vowels (CVCV), with the second vowel sometimes being a null sound. This section will discuss the process of designing separate representations for vowels, consonants, and null sounds. By doing so, it will eventually become possible to concatenate the four representations into a single auditory input. The discussion begins with vowels.

The phonetic quality of a vowel is chosen to be represented by the first two formants, following the approach of Boersma, Chládková & Benders (2022). The values for the first and second formants are derived from an experiment conducted by Padget & Tabain (2005), which explored various patterns of Russian vowel reduction. The findings of this experiment are depicted in Figure 7. The study of Padget & Tabain categorises vowels into stressed, prestressed, and unstressed categories, which corresponds to stressed vowels, vowels undergoing moderate reduction, and vowels undergoing extreme reduction, respectively, as discussed in Section 2.

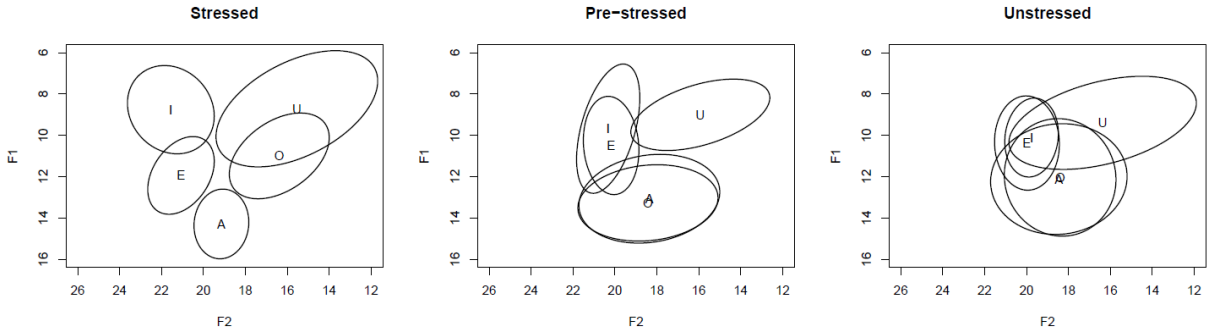


Figure 7. Vowel spaces (in ERB) of stressed as well as moderately reduced (pre-stressed) and extremely reduced (unstressed) vowels in Russian (Padget & Tabain 2005)

Considering the importance of accurately representing vowel qualities to account for stress position in words, the next objective is to determine parameters for normal distributions in F1-F2 space that result in random realisations of vowels fitting reasonably well within the ovals and circles depicted in Figure 7. Tokens for each vowel will be drawn from these distributions. Achieving a fit within circles of different diameters for different vowels requires selecting specific standard deviations for each vowel's distribution, while achieving a good fit within the ovals involves adjusting standard deviations for F1 and F2 separately. Table 4 provides a list of the six vowels possible in the toy language, along with their mean (M) and standard deviation (SD) values for F1 and F2. The corresponding auditory realisations of the

generated vowels, based on these distributions, are illustrated in Figure 8 (for the three vowels [a], [o], and [ɐ] appearing in the first syllable), Figure 9 (for the four vowels [a], [ɨ], [ə], and [ï] appearing in the second syllable), and Figure 10 (for all six vowels).

Table 4. Six possible vowels

	F1 (ERB)		*F2 (ERB)*	
phonetic form	*M*	*SD*	*M*	*SD*
[a]	14.0	0.7	19.0	0.5
[o]	11.0	0.7	16.5	1.0
[ɐ]	13.0	0.6	18.5	1.1
[ɨ]	9.0	0.8	21.5	0.8
[ə]	12.0	0.9	18.5	1.0
[ï]	10.0	0.7	20.0	0.4

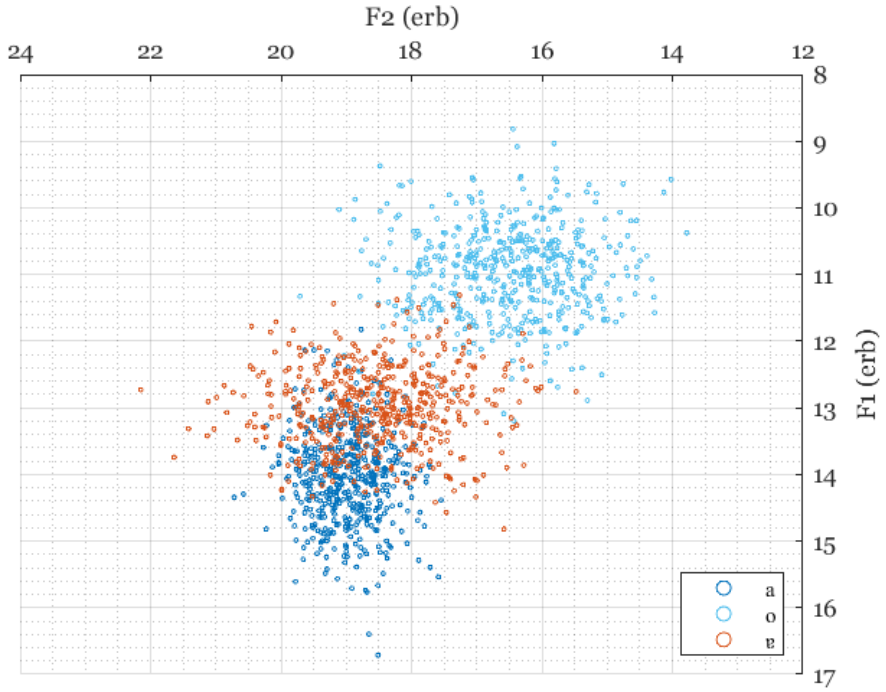


Figure 8. Auditory realisations of 1500 randomly generated vowels for the first syllable

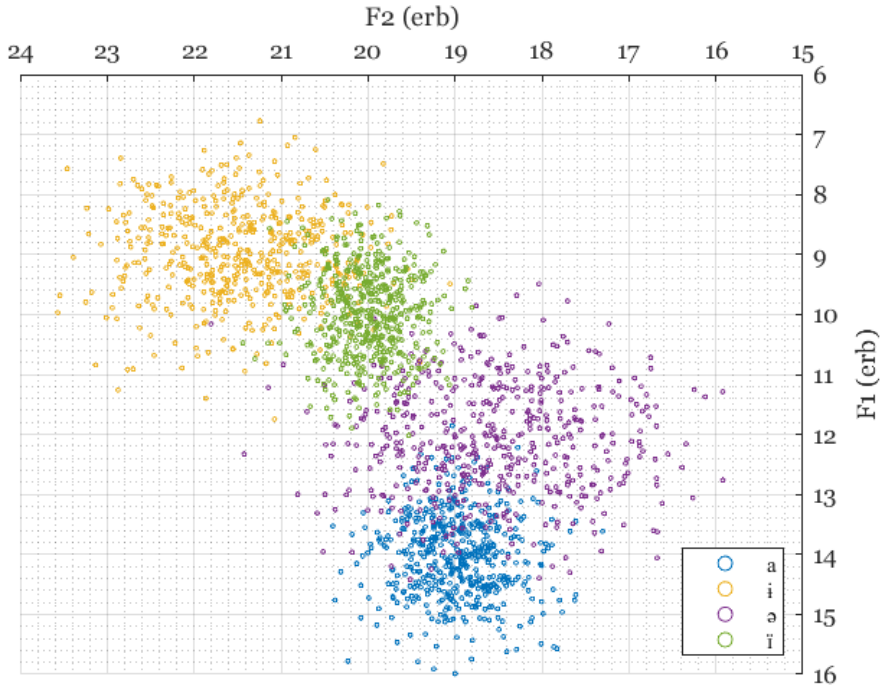


Figure 9. Auditory realisations of 2000 randomly generated vowels for the second syllable

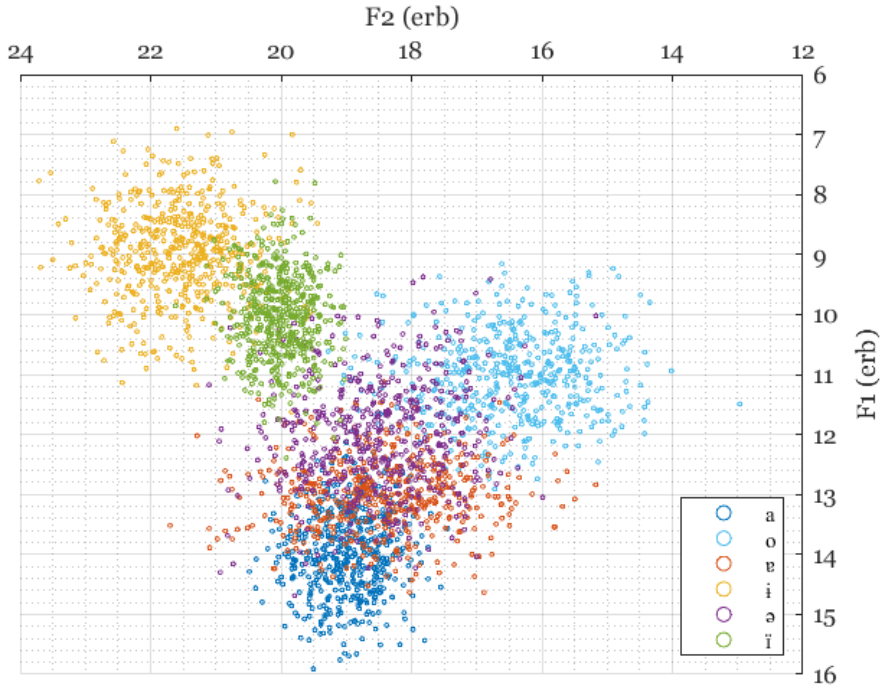


Figure 10. Auditory realisations of 3000 randomly generated vowels for both syllables

Following Boersma, Chládková & Benders (2022), the auditory nodes for vowels will represent the part of the basilar membrane relevant for hearing F1 and F2, covering a spectral continuum ranging from 4.0 to 28.0 ERB. Figure 11 illustrates the input representations for all six possible vowels using the following colour code: the red gradient indicates positive activity, with darker shades of red representing greater activity or stronger activation, while the blue gradient indicates negative activity, with darker shades of blue representing larger absolute values of negative activity or stronger inhibition. The activation level of a node for a vowel varies from -1.0 (most inhibited) to 4.0 (most activated).

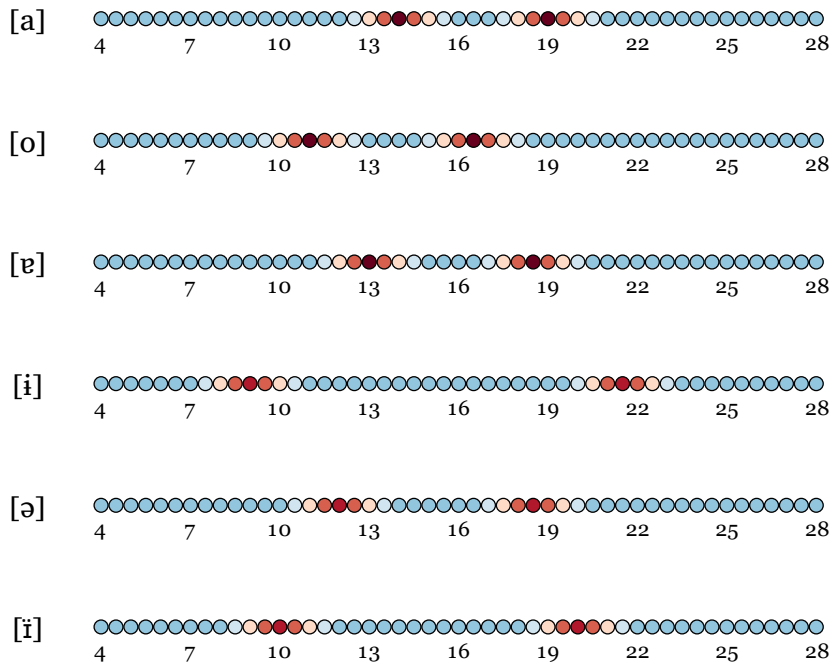


Figure 11. Six possible vowel input representations

As seen in Figure 11, the nodes associated with the not-excited regions of the basilar membrane are inhibited. This is done to ensure that the lack of activation in these nodes genuinely reflects the absence of sound at the corresponding frequencies, rather than a potentially present sound that goes unnoticed by the listener. Furthermore, it is important to consider phonetic variation, as different speakers might pronounce the same vowels with slight variation (see Figure 7). To address this, when preparing the training dataset for the model, random samples of both vowels are drawn from the vowel distributions described in Table 4 for each word. Figure 12 illustrates examples of vowels generated using this approach.

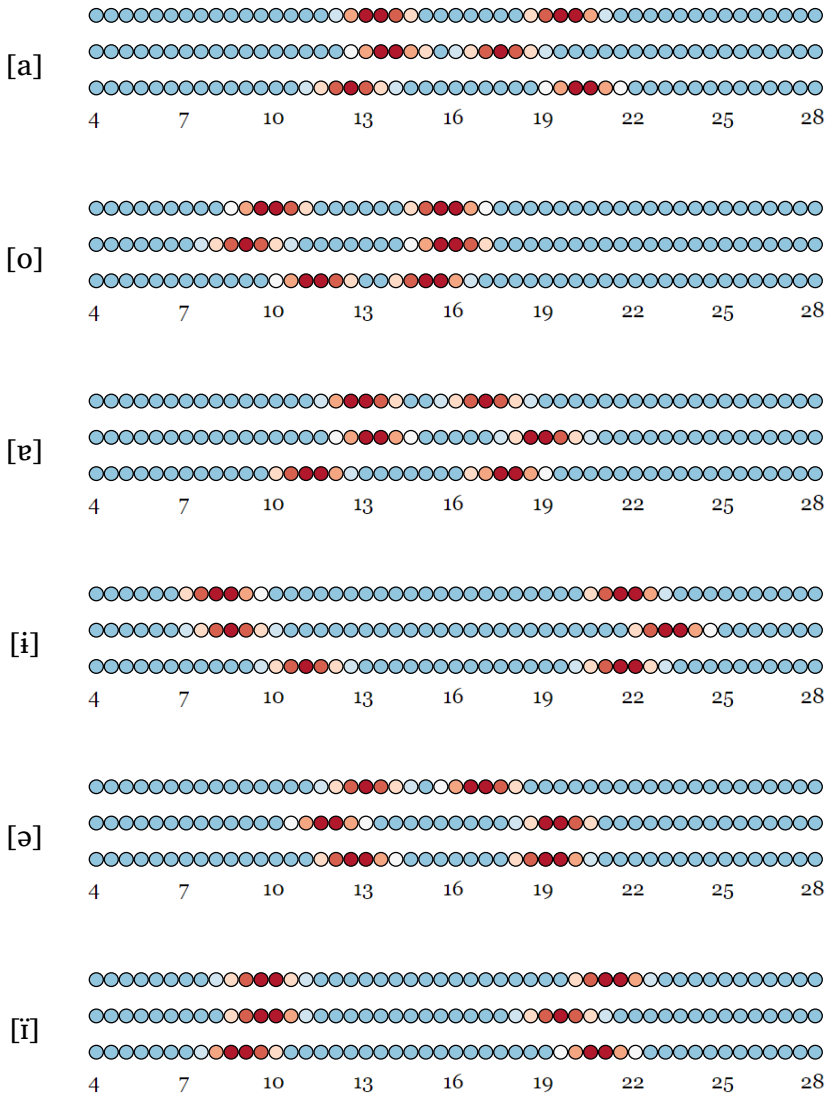


Figure 12. Randomly sampled vowel input representations

Moving on to modelling auditory input for consonants, the decision was made to represent their phonemic forms instead of modelling their specific phonetic qualities. There are two reasons for this choice. First, it is challenging to find unified phonetic qualities that apply to different types of consonants, such as plosives, fricatives, and nasals, which are all present in the toy language. Second, and most importantly, the primary focus of this thesis is on vowels, and including consonant information into a word representation is only important for differentiation between word meanings. For this purpose, it seems sufficient to represent consonants phonemically in the auditory input. The inventory of all possible consonants along with their input representations is shown in Figure 13, using the same colour code as the one used for the vowel input representation, with one difference: there is no gradient present here because phonemic representations are discrete.

The activation level of a node for a consonant can be either -1.0 (for inhibited nodes) or 5.5 (for activated nodes). The reason for choosing a higher activation level for consonants compared to the maximum activation level for vowels (which is 4.0) is that only one node is activated for a consonant, whereas several nodes are activated for each formant frequency of a vowel. As a result, setting the maximum activation level of a consonant and a vowel at the same value resulted in poor consonant learning performance of the model. Through experimental analysis, it was determined that a value of 5.5 for consonant activation yielded the most successful training results.

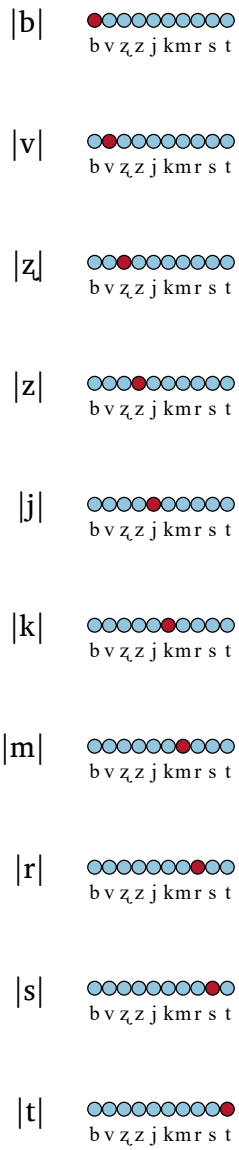


Figure 13. The ten possible consonant input representations

Another representation that needs to be addressed is the representation of a null sound. The auditory nodes for the null sound are chosen to be inhibited, following the same motivation as the inhibition of nodes representing the not-excited parts of the basilar membrane for vowels, as described earlier. Moreover, the inhibition level of the nodes representing the null sound is slightly higher than the typical inhibition of inactive parts of the basilar membrane, specifically set as -2.0 instead of -1.0. This value was chosen for input normalisation and confirmed through experimental analysis by comparing the learning performance of models trained on sets with different inhibition values. Finally, Figure 14 illustrates examples of full word representations, combining representations for two consonants and two vowels.

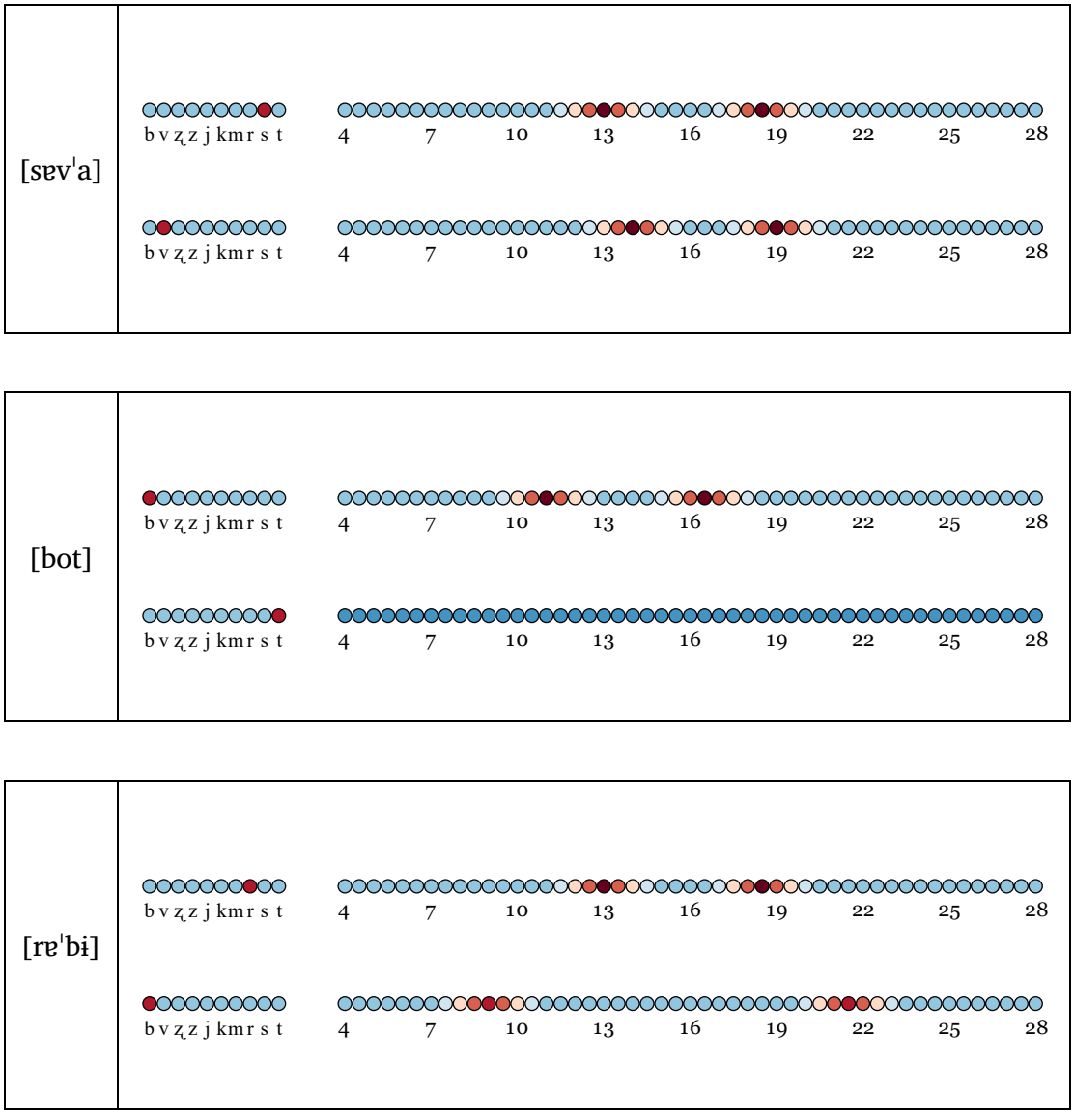


Figure 14. Examples of whole-word auditory input representations

3.2.3 Semantic input representation

The semantic input encoding a word from the toy language should include data that represents the lexical meaning of the word and its morphosyntactic meaning, such as gender and number. The words from the toy language always have one of ten possible lexical meanings and one of three possible morphosyntactic meanings. An overview of all the possible meanings is presented in Table 5.

Table 5. The possible meanings in the toy language

lexical meanings	*morphosyntactic meanings*
<owl>	<F>
<rose>	<M>
<cat>	<PL>
<bot>	
<mine>	
<heat>	
<toad>	
<basin>	
<bar>	
<slave>	

Figure 15 illustrates the inventory of all possible composite meanings, along with their corresponding input representations, using the same colour code as the auditory input representation. Similar to the input representation of consonants, there is no gradient present in semantic representations as they are also discrete. The activation level of a semantic node can be either –2.0 (for inhibited nodes) or +3.5 (for activated nodes), aligning with the choice of Boersma, Chládková & Benders (2022). Furthermore, following their approach, during the evaluation of the model for the production of partial meaning (such as only the lexical, or only the morphosyntactic part of the full meaning), the activation levels will be –1.0 for inhibited nodes and +4.5 for activated nodes.

The difference in activation strengths of an active node, which is +3.5 when two nodes are activated and +4.5 when one node is activated, is chosen for input normalisation. Furthermore, the variation in inhibition between these two cases can be conceptualised as different strengths of lateral inhibition on the input level, which also contributes to input normalisation. When only one semantic node is active, the inactive nodes are supressed to a lesser extent compared to when two semantic nodes are active. In the former case, the activation level of inhibited nodes is –1, whereas in the latter case, it is –2.

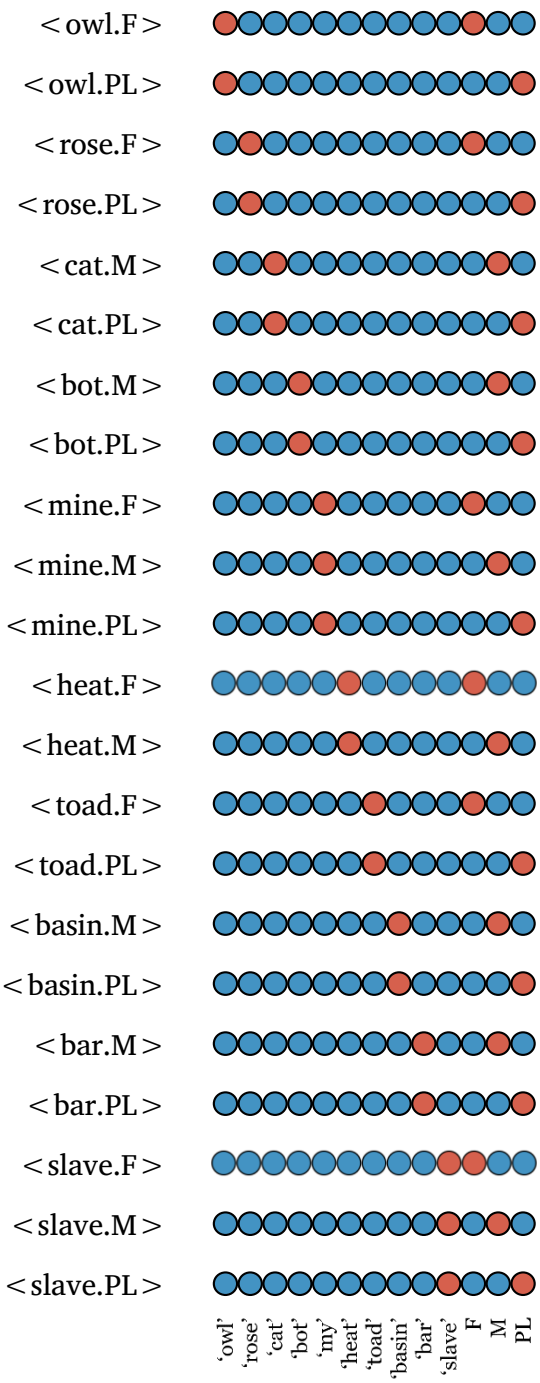


Figure 15. The twenty-two possible semantic input representations

3.2.4 Full input representation

Finally, it is possible to combine the auditory and semantic components of the input to observe examples of the complete input used for training the model. Examples of such complete input are shown in Figure 16.

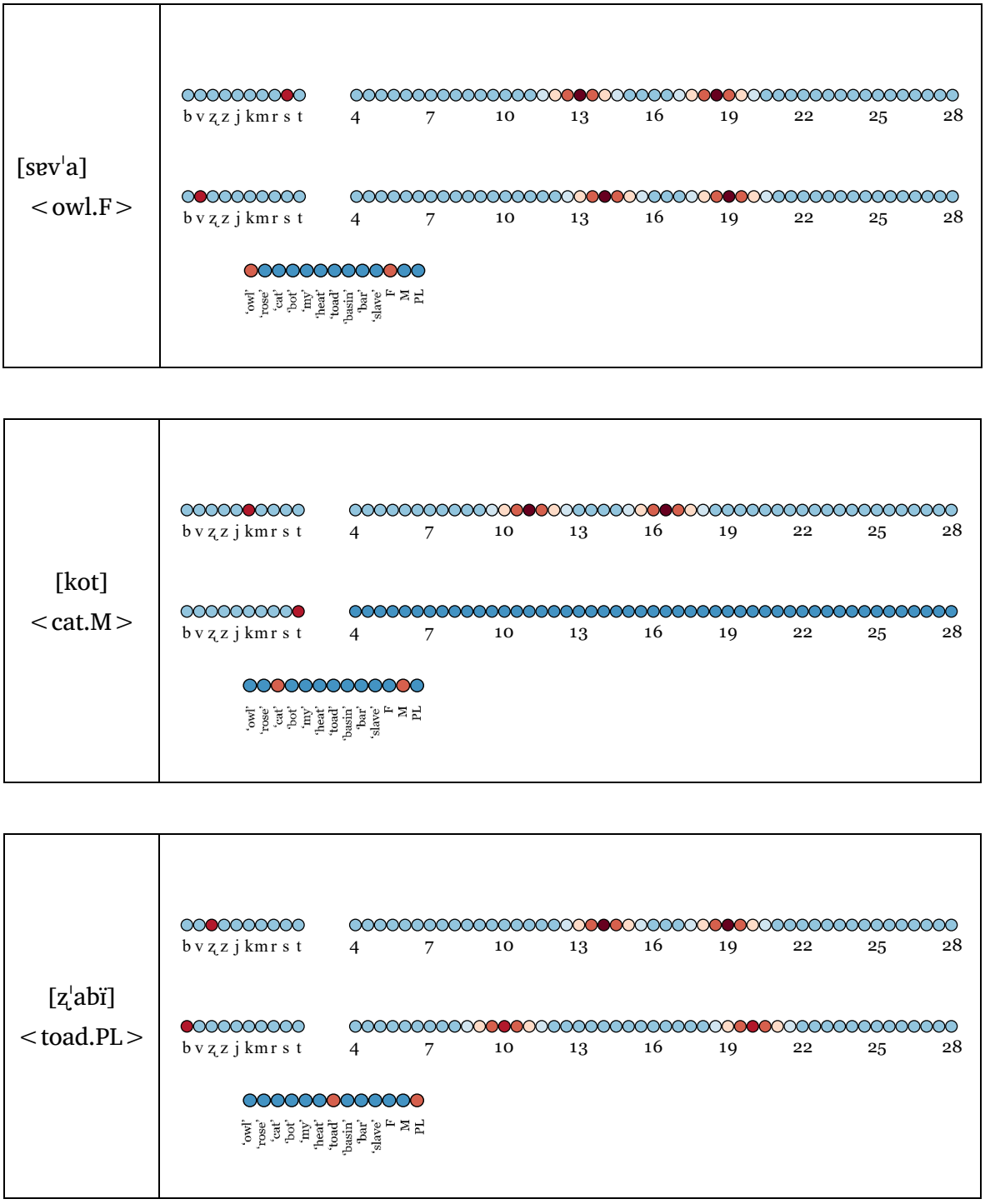


Figure 16. Examples of the complete input representations

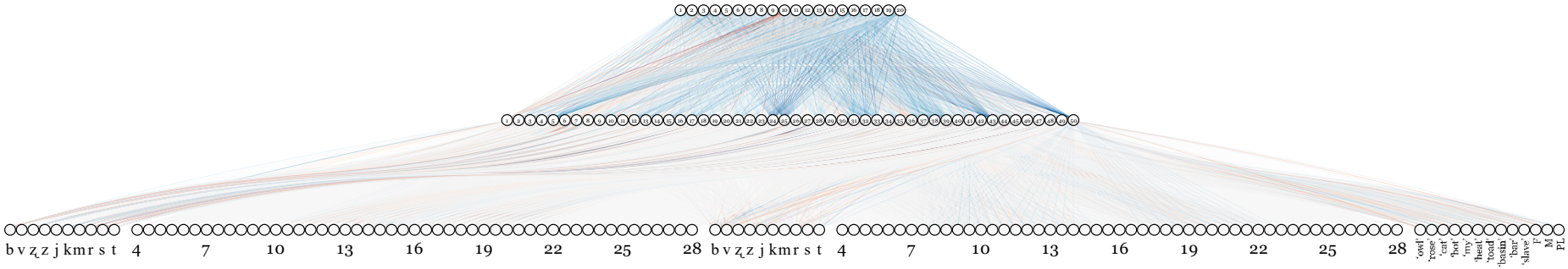


Figure 17. The neural network, trained on 13200 instances

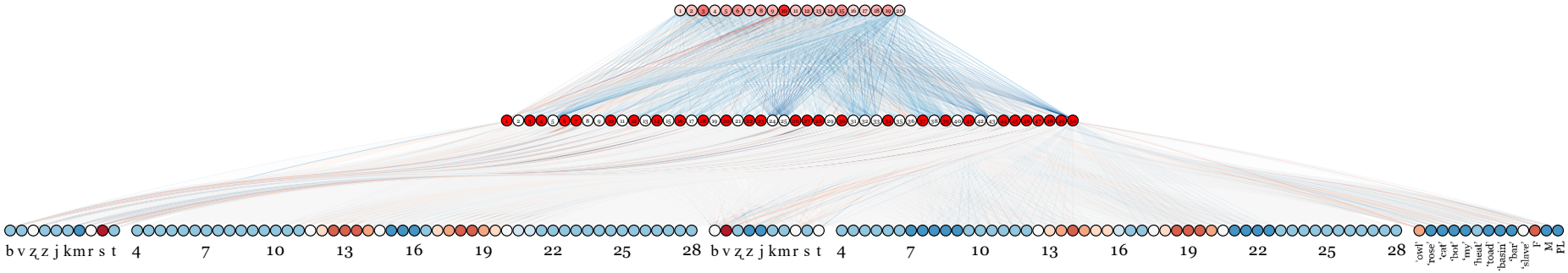


Figure 18. The neural network, trained on 13200 instances and utilised for comprehension of the word [sʁvˈa]

3.3 Network architecture

An example of a complete neural network, trained on 13200 instances is presented in Figure 17. The input level of this network consists of auditory and semantic components, as described in Section 3.2, comprising a total of 131 nodes (49 nodes for each of the two vowels, 10 nodes for each of the two consonants, and 13 nodes for meaning). The first and second hidden levels of the network consist of 50 and 20 nodes respectively, mirroring the configuration of the network in Boersma, Chládková & Benders (2022). Various alternative node configurations for the hidden levels were tested but did not result in noticeable improvements in the network's learning performance. Figure 18 illustrates the same network employed for comprehension of one of the words from the toy language, which sounds like [sɐvˈa] and has the meaning <owl.F>.

3.4 Training algorithm

The procedure for training the network was originally introduced by Boersma (2019) and is based on an algorithm for training deep Boltzmann machines previously proposed by Salakhutdinov & Hinton (2009). This training procedure was also employed by Boersma, Chládková & Benders (2022). The training process consists of four phases: initial settling, Hebbian learning, dreaming, and anti-Hebbian learning. These phases are repeated for every instance in the training dataset. In this section, a brief description of all four phases will be provided. For a deeper understanding of the concepts and details, please refer to the original papers (Boersma 2019; Boersma, Chládková & Benders 2022). The enumeration of formulas is maintained in accordance with these papers for convenience.

The computation process involves several key terms: the activities of the nodes, the biases of the nodes, and the weights of the connections between the nodes. Let us assume that K represents the number of nodes in the input level (131 in our case), while L and M represent the numbers of nodes in the second and third hidden levels, respectively (50 and 20 in our case). Then the activities of the input level nodes will be denoted as $x_k(k = 1..K)$, the activities of the first hidden level nodes as $y_l(l = 1..L)$, and the activities of the second hidden level nodes as $z_m(m = 1..M)$. Furthermore, the biases of the input level nodes will be denoted as $a_k(k = 1..K)$, the biases of the first hidden level nodes as $b_l(l = 1..L)$, and the biases of the second hidden level nodes as $c_m(m = 1..M)$. Each node k in the input level is connected to each node l in the first hidden level, and the weight of this connection is represented by u_{kl}. Similarly, each node l in the first hidden level is connected to each node m in the second hidden level, and the weight of this connection is represented by v_{lm}.

While the activities of the nodes in all levels play a crucial role in training and will be the primary focus of analysis when using the trained network, it is important to note that they

are not stored as part of the network's long-term memory. On the other hand, the biases, and the weights of the connections between the nodes *are* retained in the network's long-term memory. These biases and weights are the parameters that undergo changes in every training iteration, and they ultimately define the characteristics and behaviour of the trained network.

3.4.1 Initial settling phase

During the initial settling phase, the instance being used for the current training iteration is applied to the network's input level. Subsequently, the input is propagated to the first hidden level and then to the second hidden level, while the input level is kept clamped. This propagation is governed by the following formulas:

(3.1) $y_l \leftarrow \sigma(b_l + \sum_{k=1}^{K} x_k u_{kl} + \sum_{m=1}^{M} v_{lm} z_m)$

where $\sigma(x)$ is the standard logistic function:

(3.2) $\sigma(x) = \frac{1}{1+e^{-x}}$

(3.3) $z_m \leftarrow \sigma(c_m + \sum_{l=1}^{L} y_l v_{lm})$

Steps (3.1) and (3.3) are repeated until the network reaches a near-equilibrium state, meaning that the activities of the nodes in all levels almost stop changing between the steps. This process is referred to as mean-field approximation (Boersma 2019). The number of steps was set to 10 in both papers (Boersma 2019; Boersma, Chládková & Benders 2022) and will be determined for our network in Section 4.1.

3.4.2 Hebbian learning phase

During the Hebbian learning phase, actual learning takes place as connections between active nodes are strengthened, and the biases of the active nodes are increased. This process is described by the following formulas:

(3.4) $a_k \leftarrow a_k + \eta x_k$

(3.5) $b_l \leftarrow b_l + \eta y_l$

(3.6) $c_m \leftarrow c_m + \eta z_m$

(3.7) $u_{kl} \leftarrow u_{kl} + \eta x_k y_l$

(3.8) $v_{lm} \leftarrow v_{lm} + \eta y_l z_m$

where η is a learning rate of 0.001

3.4.3 Dreaming phase

In the dreaming phase, all the short-term memory information currently present in the network (which is defined by the activities of the nodes at the end of the initial settling phase) circulates between the levels. This circulation occurs using the weights of the connections and the biases of the nodes, which are established during the Hebbian learning phase. The input level is not clamped and participates in this process:

(9) $x_k \leftarrow a_k + \sum_{l=1}^{L} u_{kl} y_l$

(10) $z_m \sim \mathcal{B}(\sigma(c_m + \sum_{l=1}^{L} y_l v_{lm}))$

(11) $y_l \sim \mathcal{B}(\sigma(c_m + \sum_{k=1}^{K} x_k u_{kl} + \sum_{m=1}^{M} v_{lm} z_m))$

where $\mathcal{B}(x)$ represents the Bernoulli distribution, which introduces a degree of randomness into the network's state. Steps (9) – (11) are repeated until the network reaches a near-equilibrium state, meaning that the activities of the input nodes and the probability distributions of the possible states of the hidden nodes (as defined by (10) and (11) before applying $\mathcal{B}(x)$) are almost not changing between the steps. The number of steps was set to 10 by Boersma, Chládková & Benders (2022). For the network used in this thesis, this number will be determined in Section 4.1.

3.4.4 Anti-Hebbian learning phase

During the anti-Hebbian phase, the network is unlearning from the state it reached during the dreaming phase. The connections between active nodes are weakened, and the biases of the active nodes are decreased:

(3.12) $a_k \leftarrow a_k - \eta x_k$

(3.13) $b_l \leftarrow b_l - \eta y_l$

(3.14) $c_m \leftarrow c_m - \eta z_m$

(3.15) $u_{kl} \leftarrow u_{kl} - \eta x_k y_l$

(3.16) $v_{lm} \leftarrow v_{lm} - \eta y_l z_m$

3.5 Programming and visualising tools

The model used in this study was implemented in MATLAB (The MathWorks Inc. 2023). The learning algorithm for the DBM was adapted from the source code used in the original study by Boersma, Chládková & Benders (2022) in Praat (Boersma & Weenink 2023). The implementation of a MATLAB class for the DBM functionality can be found in Appendix 3. Additionally, the implementation of a MATLAB class used for analysing the network and creating all the tables shown in Section 4 can be found in Appendix 4.

To visualise the training and subsequent usage of the network, a user interface was created using MATLAB App Designer. This interface provides convenient control over the training process and allows the user to conduct visual experiments with the trained network, such as observing its behaviour when dealing with partial input. The source code for this app, including all the computation scripts, is available at the UvA Phonetic Sciences archive (https://www.fon.hum.uva.nl/archive). The visualisations presented in Sections 3 and 4, as well as all the visualisations in the app, were created using Graphviz, an open-source graph visualisation software (Ellson et al. 2023).

4 Analysis

To ensure the accurate implementation of the algorithm, the first step was to replicate the results of the original study (Boersma, Chládková & Benders 2022). This involved modelling the learning of their toy language, as well as computing and visualising the results. It was verified that the same results were obtained as reported in the original study.

Subsequently, a model was created consisting of a DBM as described in Section 3.3, a training algorithm as described in Section 3.4, and a training dataset generator, capable of generating datasets as described in Section 3.2. For the analysis, the network was trained on a dataset consisted of 2000 samples for each word, resulting in a total dataset size of 44000 samples. This process was repeated 100 times consecutively to simulate 100 different learners of the toy language. For each virtual learner, the necessary measurements were conducted and then averaged across all the learners.

The analysis involves applying specific inputs to the trained network, resulting in different patterns of node activation at three different levels of the network. These node activation patterns correspond to the activities of the input level nodes $x_k(k = 1..K)$, the activities of the first hidden level nodes $y_l(l = 1..L)$, and the activities of the second hidden level nodes $z_m(m = 1..M)$, as defined in Section 3.2. These patterns can be viewed as vectors in a multidimensional space, with the number of dimensions matching the number of nodes at each respective level. The term “similarity” between the levels of the network is then used to

refer to the *cosine similarity* between the vectors representing the node activation patterns for those levels.

Cosine similarity is a measure of similarity between two vectors in a multidimensional space, determined by the cosine of the angle between them. It is computed by taking the inner product of the two vectors and dividing it by the product of their Euclidean norm. The resulting value ranges between 0 (indicating that the vectors are orthogonal, and therefore not similar) and 1 (indicating maximal similarity). Each analysis section below will specify the exact levels of the network being compared.

Section 4.1 provides an overview of the verification process for some of the model's training parameters. Following that, in Section 4.2, the modelling of comprehension and production of the words in the toy language is outlined. This is followed by Section 4.3, where the proficiency of the trained network in the toy language is measured. Sections 4.4, 4.5, and 4.6 will present an analysis of the hidden levels of the network after processing different partial inputs to examine the potential emergence of categories, referred to as phonological categories. Furthermore, please note that in the analysis tables, angle brackets (< >) used to denote a semantic form are omitted to save space.

4.1 Verifying training parameters

There are two phases in the training process where a sequence of steps is repeated until the network reaches a near-equilibrium state: the initial settling phase and the dreaming phase. In the model by Boersma, Chládková & Benders (2022), the number of repetitions in both cases was set to 10. However, considering the larger number of nodes in the input level of the DBM in this thesis, as well as a slightly more complex toy language, both parameters were tested to determine if 10 iterations were truly sufficient for the network states to converge. It was found that, in both cases, a near-equilibrium state is, in fact, reached much earlier, specifically after just one iteration in the initial settling phase, and after two iterations in the dreaming phase. Nevertheless, to stay on the safe side, the number 10 was preserved for both phases.

4.2 Modelling comprehension and production of words in the toy language

Once the network is trained, it is possible to model comprehension and production of words in the toy language. Boersma, Chládková & Benders (2022) have provided a detailed procedure for this evaluation, which involves applying partial input to the network. Specifically, auditory input is used for comprehension evaluation, while semantic input is

used for production evaluation. This partial input is initially spread to the first hidden level using the step (3.1). Subsequently, a sequence of 10 echoes is performed without clamping the input level. Each echo consists of steps (3.3), (3.9), (3.1). Boersma, Chládková & Benders (2022) suggest using 10 echoes as it has been found to be sufficient for the network in their study to reach a near-equilibrium state. As a result of this echoing process, the input level transforms into output level, revealing how the network interpreted the provided partial input. For example, when the network receives auditory input, the semantic part of the output reflects how the network *comprehended* the given sound. Conversely, if semantic input is provided, the auditory part of the output demonstrates how the network *produced* the given meaning.

In this thesis, the same echoing method is employed for modelling comprehension and production, with two adjustments. First, the number of echoes is increased from 10 to 30. This adjustment is necessary since the network in this thesis does not always reach a near-equilibrium state after 10 echoes and may require up to 25 echoes. To determine if the network has reached a near-equilibrium state, cosine similarities are computed between two consecutive echoes at each of the three levels of the network. If these cosine similarities for all three levels approach 1.000000 (accurate to six decimal places), then the network is considered to have reached a near-equilibrium state.

The second adjustment involves clamping the semantic part of the input during echoing in production. This decision is motivated by the observed better production performance of the network when the semantic input is clamped[5]. The question arises: How can this improved performance be explained? In the context of comprehension, Boersma, Chládková & Benders (2022, p.28) suggested that the unclamped version may reflect a more realistic scenario compared to the clamped version. In comprehension, if the auditory output does not match the input, it can be interpreted as what the network "thinks" it has heard. This behaviour of the network also supports the existence of the perceptual magnet effect (Kuhl 1991).

However, I suggest that production may inherently differ from comprehension in terms of their potential to deviate from the input. In comprehension, the listener hears the sound once before entering the process of deriving a semantic representation for that sound. During this process, the internal representation of the heard sound might wander off from the original one. In contrast, in production, the meaning is not heard from the outside world but is created

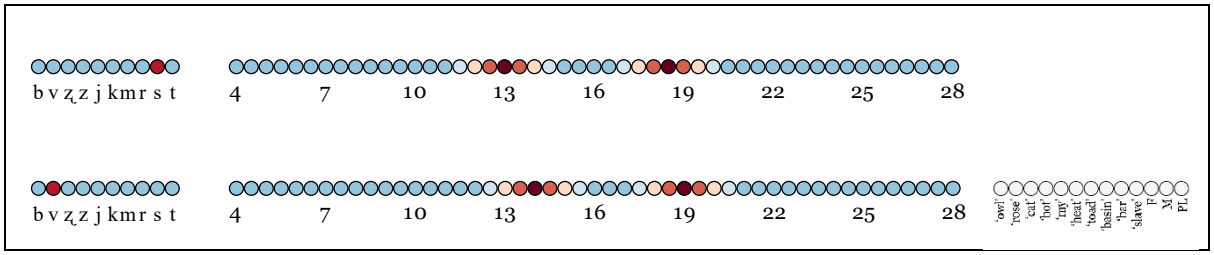
[5] Tables A1.3 and A1.4 in Appendix 1 present the measurements of proficiency in production, as further explained in Section 4.3. Table A1.3 corresponds to the measurements with the semantic nodes unclamped, while Table A1.4 corresponds to the measurements with the semantic nodes clamped.

in the speaker's brain. Therefore, the brain might be capable of maintaining the intended meaning during the process of finding an auditory representation to articulate it. Considering this interpretation and the better performance of the network in production with the semantic input nodes being clamped, this method will be further employed for the analysis of production.

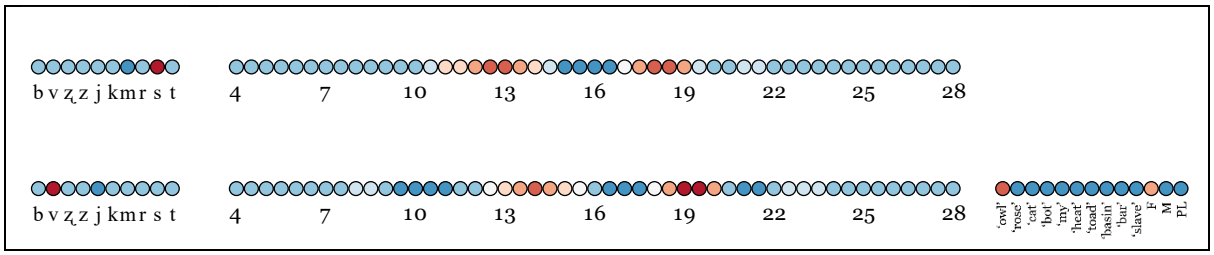
4.3 Proficiency of the network in the toy language

To simulate comprehension, the trained network is provided with input where only the auditory nodes are activated, while the semantic nodes' activation is set to 0. An example of such input for the word <owl.F> [sɐvˈa] is demonstrated in Figure 19(a).

(a) Input level for [sɐvˈa] used for measuring comprehension performance



(b) Comprehension output level for [sɐvˈa]



(c) Standard representation for <owl.F> [sɐvˈa]

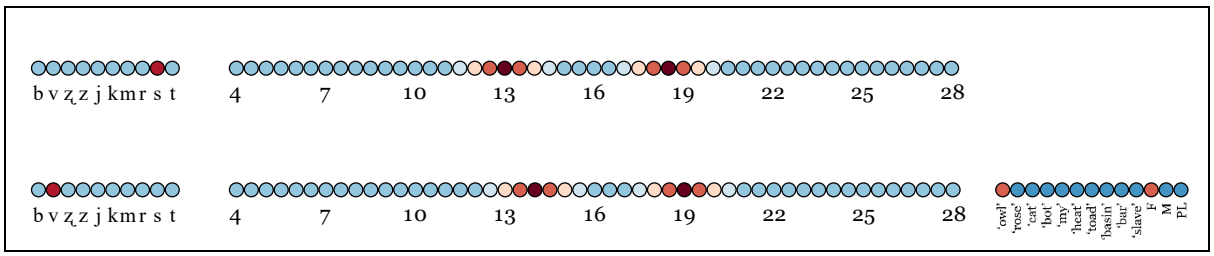


Figure 19. Example of input, output, and standard representations for [sɐvˈa] for measuring the network's **comprehension** performance

Subsequently, this input is propagated through the network in a series of 30 echoes following the procedure described in Section 4.2, while keeping the whole input level unclamped. As a result, the input level becomes the output level, representing how the network comprehended the given auditory input. An example of such an output level for the word <owl.F> [sɛvˈa], comprehended by the network trained on 44000 samples, is depicted in Figure 19(b).

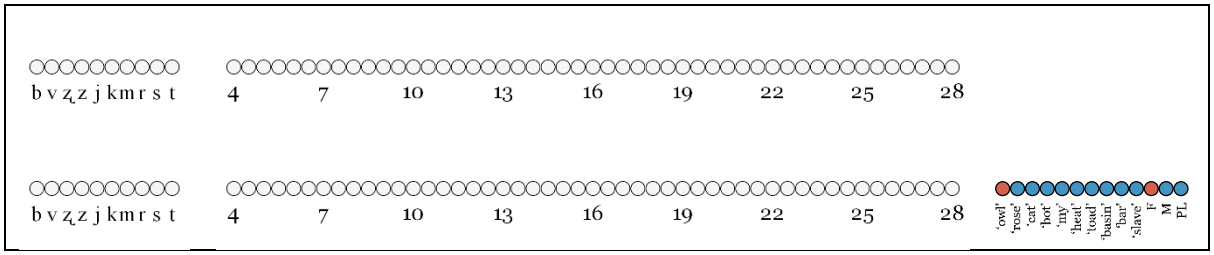
Afterwards, the entire comprehension output, as shown in Figure 19(b), is compared with the entire standard input representation of the intended word, as illustrated in Figure 19(c) for the word <owl.F> [sɛvˈa]. This comparison involves computing the cosine similarity between them. Table 6 displays the cosine similarities in percents between the outputs for all possible auditory inputs (in rows) and the entire input representations of all possible words in the toy language (in columns), averaged across 100 toy language learners. The table utilises a gradient of green to encode higher similarities with darker green and lower similarities with lighter green.

In a similar manner to simulating comprehension, when simulating production, the trained network is provided with partial input. However, this time only the semantic nodes are activated, and the auditory nodes' activation is set to 0. Figure 20(a) demonstrates an example of such input for the word <owl.F> [sɛvˈa]. After propagating this input through the network in a series of 30 echoes, the input level becomes the output level, representing how the network produced the given meaning. Figure 20(b) shows an example of such an output level for the word <owl.F> [sɛvˈa], produced by the network trained on 44000 samples.

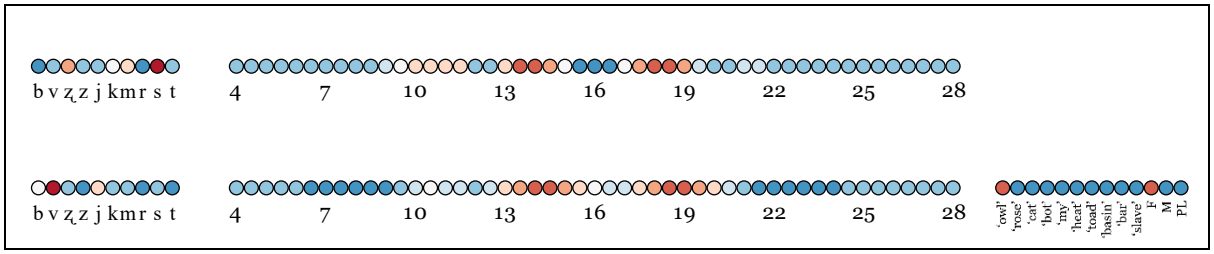
Similar to measuring comprehension proficiency, the cosine similarity is then computed between the entire production output, as shown in Figure 20(b), and the entire standard input representation of the word, as shown in Figure 20(c) for the word <owl.F> [sɛvˈa]. Table 7 presents the cosine similarities in percents between the outputs of all possible semantic inputs (in rows) and the entire standard input representations of all possible words in the toy language (in columns), averaged across 100 toy language learners. The table utilises a gradient of red to encode higher similarities with darker red and lower similarities with lighter red.

Tables 6 and 7 visually demonstrate that the network has successfully learned how to comprehend and produce words in the toy language. The diagonals of these tables are highlighted with darker colours, indicating higher similarities. These diagonal similarities in Tables 6 and 7 can also be referred to as the accuracies of word comprehension (Table 6) and the accuracies of word production (Table 7).

(a) Input level for <owl.F> used for measuring production performance



(b) Production output level for <owl.F>



(c) Standard representation for <owl.F> [sɐvˈa]

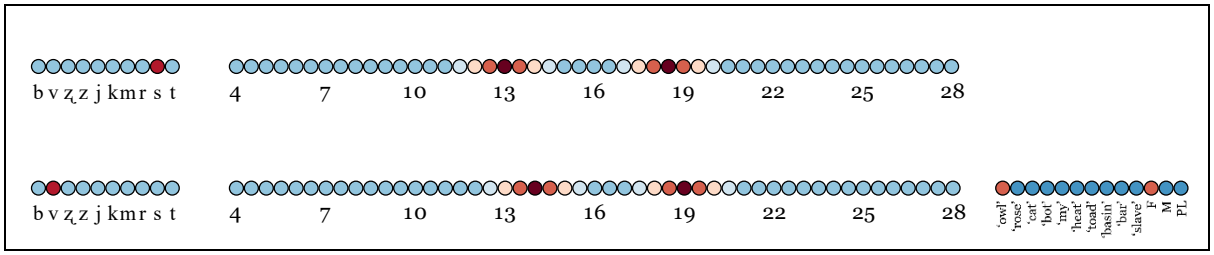


Figure 20. Example of input, output, and standard representations for <owl> [sɐvˈa] for measuring the network's **production** performance

To facilitate visual analysis, the standard deviations of the similarities have been omitted from Tables 6 and 7. Nevertheless, it is worth noting that the average standard deviation of word comprehension accuracies over 100 learners is 3.68% and the average standard deviation of words production accuracies over 100 learners is 7.59%. These standard deviations provide insight into the variability of comprehension and production performance across the learners. For the specific standard deviations of each similarity across 100 learners, please refer to Appendix 1, where Tables A1.1 and A1.4 represent the same tables as Table 6 and Table 7 but with the standard deviations included.

Table 6. Comprehension proficiency: cosine similarities between the entire comprehension outputs for all possible standard auditory inputs (in rows) and the entire standard input representations of all possible words (in columns); averaged over 100 toy language learners (in percents). Colour-coding: gradient of green, with darker green indicating higher similarities. The network is trained on 44000 samples.

	owl.F [sɛvˈa]	owl.PL [sˈovɨ]	rose.F [rˈozə]	rose.PL [rˈozɨ]	cat.M [kot]	cat.PL [kɐtˈɨ]	bot.M [bot]	bot.PL [bˈotɨ]	mine.F [mɐjˈa]	mine.M [moj]	mine.PL [mɐjˈɨ]	heat.F [ʐɐˈra]	heat.M [ʐˌar]	toad.F [ʐˌˈabə]	toad.PL [ʐˌˈabɨ]	basin.M [taz]	basin.PL [tɐzˈɨ]	bar.M [bar]	bar.PL [bˈarɨ]	slave.F [rɐˈba]	slave.M [rab]	slave.PL [rɐˈbɨ]
[sɛvˈa]	96	38	20	6	8	23	9	6	63	8	22	64	26	39	25	26	23	26	26	64	26	23
[sˈovɨ]	37	95	30	64	33	17	33	63	5	33	17	6	5	-3	31	4	16	5	31	5	4	17
[rˈozə]	20	32	95	65	33	-7	33	32	20	33	-7	20	3	29	-1	14	6	3	-2	33	15	6
[rˈozɨ]	3	62	63	95	32	14	32	62	3	33	15	3	2	-2	29	14	27	2	28	16	14	27
[kot]	7	30	31	30	98	36	83	42	7	73	7	7	48	4	3	48	7	48	3	7	48	7
[kɐtˈɨ]	24	17	-5	18	40	94	22	30	24	11	61	25	28	14	37	28	61	28	37	25	28	62
[bot]	6	32	32	32	83	17	98	60	6	73	6	6	47	3	3	47	5	57	14	6	47	6
[bˈotɨ]	4	61	30	61	47	29	64	93	5	36	16	5	8	-2	29	7	16	19	42	5	8	16
[mɐjˈa]	63	5	19	5	8	24	8	5	96	37	57	63	25	39	24	24	23	25	24	63	25	24
[moj]	7	30	31	30	72	7	73	31	36	98	36	7	48	3	3	48	7	48	3	7	48	7
[mɐjˈɨ]	24	18	-6	18	9	62	8	17	58	38	95	24	26	14	38	26	62	26	38	24	27	62
[ʐɐˈra]	63	6	19	6	9	24	9	6	63	9	24	96	56	52	39	27	24	38	39	63	27	24
[ʐˌar]	26	4	3	3	48	26	48	3	26	48	26	54	99	44	44	74	26	84	44	26	74	25
[ʐˌˈabə]	41	-2	28	-1	4	16	4	-1	41	4	16	54	46	96	66	34	15	35	33	54	46	29
[ʐˌˈabɨ]	24	29	0	30	5	36	5	30	24	4	35	38	46	65	95	34	36	34	63	38	45	49
[taz]	25	3	14	14	48	26	48	3	25	48	26	25	73	32	32	99	55	74	32	25	73	26
[tɐzˈɨ]	24	17	7	30	10	62	10	18	25	10	62	25	28	15	39	57	95	28	39	25	28	62
[bar]	25	4	4	4	48	25	58	15	25	48	24	37	84	34	33	74	25	99	61	25	74	25
[bˈarɨ]	27	30	-1	30	6	38	18	43	27	7	38	39	47	32	64	36	38	64	96	26	35	37
[rɐˈba]	63	4	32	17	7	24	7	4	63	7	24	63	25	52	38	25	24	25	25	96	55	57
[rab]	24	3	14	14	48	25	47	3	25	48	25	24	73	43	42	73	25	73	31	54	99	54
[rɐˈbɨ]	24	17	6	29	9	62	9	17	24	9	62	24	28	27	51	28	62	28	38	58	57	95

Table 7. Production proficiency: cosine similarities between the entire production outputs for all possible semantic inputs (in rows) and the entire standard input representations of all possible words (in columns); averaged over 100 toy language learners (in percents). Colour-coding: gradient of red, with darker red indicating higher similarities. The network is trained on 44000 samples; semantic nodes of the input are clamped.

	owl.F [sɛvˈa]	owl.PL [sˈoviˈ]	rose.F [rˈozə]	rose.PL [rˈoziˈ]	cat.M [kot]	cat.PL [kɛtˈi]	bot.M [bot]	bot.PL [bˈotiˈ]	mine.F [mɐjˈa]	mine.M [moj]	mine.PL [mɐjˈi]	heat.F [ʐɐˈra]	heat.M [ʐar]	toad.F [ʐˈabə]	toad.PL [ʐˈabiˈ]	basin.M [taz]	basin.PL [tɐzˈi]	bar.M [bar]	bar.PL [bˈariˈ]	slave.F [rɛˈba]	slave.M [rab]	slave.PL [rɛˈbi]
owl.F	76	47	37	12	18	8	18	12	41	18	8	41	19	38	13	19	8	19	13	41	19	8
owl.PL	45	78	13	43	21	34	21	43	10	21	34	10	19	11	41	20	34	19	42	10	19	33
rose.F	27	18	81	52	24	3	24	19	28	24	2	28	16	37	8	25	13	16	10	39	25	13
rose.PL	3	47	52	81	23	28	24	49	4	23	28	5	16	9	38	23	38	16	40	15	24	39
cat.M	14	19	20	20	89	44	72	30	14	63	15	15	60	15	15	59	14	59	15	14	59	15
cat.PL	11	37	11	38	49	75	29	48	10	19	40	11	21	12	39	21	41	21	38	12	22	42
bot.M	11	24	26	26	75	19	91	53	11	67	10	11	54	12	13	56	11	62	20	13	56	12
bot.PL	8	47	19	48	34	41	54	80	8	25	31	7	15	10	39	17	33	25	47	10	17	33
mine.F	40	9	35	9	18	9	18	9	75	49	44	39	19	37	11	19	9	19	10	40	20	10
mine.M	13	20	20	20	63	13	64	20	43	90	43	13	59	14	15	59	13	59	15	13	59	13
mine.PL	10	39	11	39	19	38	20	39	45	50	73	9	20	12	41	21	38	21	40	10	21	38
heat.F	43	9	35	12	20	13	20	9	43	19	12	76	49	44	21	20	13	30	20	44	21	13
heat.M	18	16	18	18	61	19	60	17	19	60	18	47	85	26	27	61	19	69	27	20	61	19
toad.F	32	11	38	10	16	6	16	11	33	17	7	42	32	80	52	24	6	21	17	41	32	15
toad.P	9	41	12	40	16	31	16	40	9	17	32	18	31	52	79	24	32	21	45	18	31	41
basin.M	20	13	22	23	56	21	57	13	20	56	21	20	65	22	22	91	50	65	22	20	65	20
basin.PL	16	30	15	41	14	45	15	30	16	14	45	15	27	17	44	57	79	27	44	15	27	45
bar.M	19	10	11	11	54	19	62	20	21	55	20	29	75	25	25	67	19	92	53	21	69	21
bar.PL	15	33	6	35	11	38	21	43	16	13	40	25	37	21	50	28	38	56	83	17	30	41
slave.F	44	4	41	16	15	14	15	4	44	15	14	44	23	50	25	22	13	23	13	78	53	48
slave.M	19	14	23	23	58	19	58	15	19	58	19	19	63	29	29	62	18	63	20	48	88	48
slave.PL	15	32	17	43	17	43	18	33	15	17	43	14	24	24	51	23	42	25	41	49	54	76

4.4 Phonological categories in comprehension and production

Now that it has been established that the network has successfully learned to comprehend and produce words in the toy language, the next step of the analysis becomes relevant. Specifically, it is now interesting to compare the hidden representations of all the words in the toy language. These hidden representations can be found in the hidden levels of the network following the comprehension and production of each word and are viewed as phonological representations of these words. Therefore, comparing the hidden representations of different words can potentially lead to the discovery of phonological categories.

In their study, Boersma, Chládková & Benders (2022) focused on investigating the first hidden level of their network for this comparison. They justified this choice by noting that the second level showed minimal variation across different utterances. Therefore, to determine the phonological similarities between two utterances in their toy language, they computed the cosine similarities between these utterances' corresponding first hidden levels.

In this thesis, the same method for comparing hidden representations is employed, with one difference: instead of comparing only the first hidden level of the network, the entire network's hidden state, including both hidden levels, will be compared. To implement this, two vectors representing the first and second hidden levels will be concatenated for each word to compute the cosine similarities between them. This approach allows for a more comprehensive analysis, as it includes any potential influence on categorisation, no matter how small, that may occur at the second hidden level. The fact that cosine similarity reflects a comparison of the individual vector dimensions ensures the validity of comparing two concatenated vectors in a single operation.

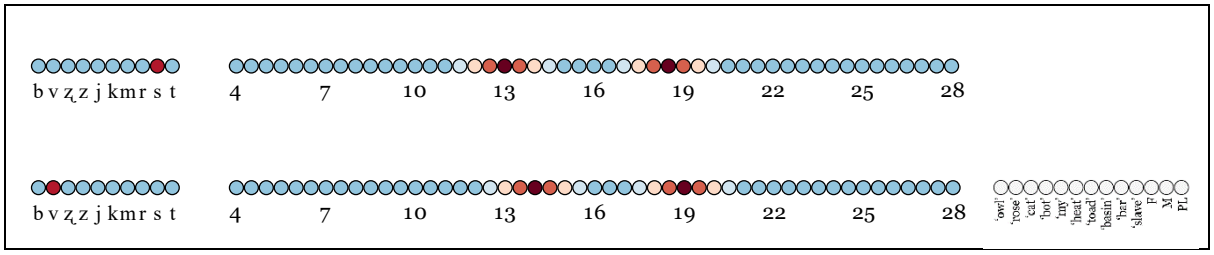
The decision to include the second hidden level in the analysis was driven by two main reasons. Firstly, in the network used in this thesis, the second hidden level exhibited a greater variation (ranging from 97% to 100% with standard deviations of 0% and 1% across 100 language learners) compared to the network used by Boersma, Chládková & Benders (2022) (ranging from 98% to 100% with standard deviation of 0% and 1% across 100 language learners). This broader range of similarities suggests a potentially stronger influence of the second hidden level on the categorisation process. Secondly, and more importantly, upon examining the similarity tables for the first hidden level alone and for both levels together, it became evident that including the second hidden level resulted in clearer categorisation, or rather, multiple categorisations. The fact that multiple categorisations emerged in the *hidden states* of the network is explained below.

Throughout the rest of this thesis, the term "hidden states of the network after applying input X" will be used to refer to the activation pattern of both hidden levels of the network following

the application of a specific input X. Furthermore, the terms “hidden representation of input X”, and “phonological representation of input X” will be used interchangeably to describe the hidden states of the network after applying input X. Consequently, the term “phonological similarity between inputs A and B” will denote the cosine similarity between the distinct hidden states of the network, one obtained after applying input A and the other after applying input B.

The first step in examining the hidden levels of the network is to measure the phonological similarities between different auditory inputs in comprehension and different semantic inputs in production. Figure 21 provides examples of the input levels given to the network for subsequent comprehension (21(a)) and production (21(b)). After such inputs are provided for each word in the toy language, the network’s hidden states are compared with each other separately for comprehension and production.

(a) Input level for [sɐvˈa] used for analysing the network’s hidden states in comprehension



(b) Input level for <owl.F> used for analysing the network’s hidden states in production

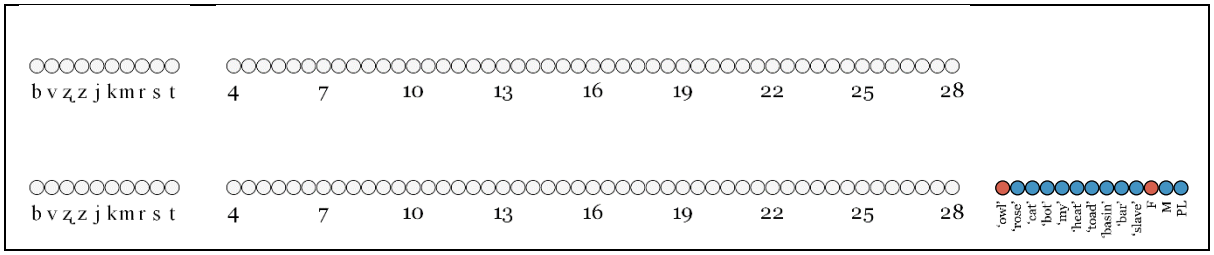


Figure 21. Example of input levels for <owl> [sɐvˈa] for analysing the network’s hidden states in comprehension(a) and production(b)

Table 8 displays the phonological similarities between standard auditory representations of all possible words in the toy language, averaged over 100 language learners. A gradient of green is applied to all cells containing similarities higher than 60%, with darker green representing stronger similarities. It can be observed that the similarities in Table 8 are stronger between words that share the same set of vowel sounds. For example, [sɐvˈa] shares the set of vowel sounds with [mɐjˈa], [ɻɐˈra], and [rɐˈba]. Similarly, [sˈovɪ̈] shares the set of

vowel sounds with [rˈozï] and [bˈotï], etc. Additionally, pairs that share one vowel and both consonants are marked high on the similarity scale, such as the pairs [rˈozə] – [rˈozï], [bot] – [bˈotï], [zˌˈabə] – [zˌˈabï], [bar] – [bˈarï], [rɐˈba] – [rɐˈbɨ].

In the same manner as Table 8, Table 9 displays the phonological similarities between all possible word meanings in the toy language, averaged across 100 language learners. A gradient of red is applied to all cells containing similarities higher than 60%, with darker red representing stronger similarities. Categorisation in this table appears different compared to comprehension categorisation seen in Table 8: the similarities are stronger here between words that share the same meaning, either in terms of its lexical or morphosyntactic aspect. For example, <owl.F> shares the lexical meaning with <owl.PL> and the morphosyntactic meaning with <rose.F>, <mine.F>, <heat.F>, <toad.F>, and <slave.F>. Similarly, <owl.PL> shares the lexical meaning with <owl.F> and the morphosyntactic meaning with <rose.PL>, <cat.PL>, <bot.PL>, <mine.PL>, <toad.PL>, <basin.PL>, <bat.PL>, and <slave.PL>, etc.

Table 8. Phonological similarities between all possible words in comprehension, averaged over 100 toy language learners (in percents). Colour-coding: gradient of green for values above 60, with darker green indicating higher similarities. The network is trained on 44000 samples.

	owl.F [sɛvˈa]	owl.PL [sˈovï]	rose.F [rˈozə]	rose.PL [rˈozï]	cat.M [kot]	cat.PL [kɐtˈi]	bot.M [bot]	bot.PL [bˈotï]	mine.F [mɐjˈa]	mine.M [moj]	mine.PL [mɐjˈi]	heat.F [zɐˈra]	heat.M [zˌar]	toad.F [zˌˈabə]	toad.PL [zˌˈabï]	basin.M [taz]	basin.PL [tɐzˈi]	bar.M [bar]	bar.PL [bˈarï]	slave.F [rɐˈba]	slave.M [rab]	slave.PL [rɐˈbi]
owl.F [sɛvˈa]	100	69	60	56	51	63	52	55	82	52	65	81	60	68	62	59	64	59	64	81	59	64
owl.PL [sˈovï]	69	100	69	81	65	57	66	81	55	65	58	56	49	51	62	49	57	49	63	55	49	57
rose.F [rˈozə]	60	69	100	85	65	50	65	68	59	65	50	59	48	64	51	54	55	48	50	65	54	56
rose.PL [rˈozï]	56	81	85	100	64	56	65	80	54	65	57	54	46	50	62	53	62	47	61	60	53	62
cat.M [kot]	51	65	65	64	100	65	90	72	51	86	51	51	69	49	49	70	52	69	49	51	70	52
cat.PL [kɐtˈi]	63	57	50	56	65	100	57	62	65	53	82	64	59	59	65	60	81	59	65	64	60	81
bot.M [bot]	52	66	65	65	90	57	100	80	51	86	52	51	69	49	49	69	52	75	54	51	69	52
bot.PL [bˈotï]	55	81	68	80	72	62	80	100	55	67	58	55	49	50	62	49	58	54	68	54	49	57
mine.F [mɐjˈa]	82	55	59	54	51	65	51	55	100	65	79	82	58	67	62	58	64	57	63	82	58	64
mine.M [moj]	52	65	65	65	86	53	86	67	65	100	65	52	70	49	48	70	52	70	49	51	70	52
mine.PL [mɐjˈi]	65	58	50	57	51	82	52	58	79	65	100	64	59	59	65	59	82	58	66	64	59	82
heat.F [zɐˈra]	81	56	59	54	51	64	51	55	82	52	64	100	72	73	68	59	65	64	68	82	58	64
heat.M [zˌar]	60	49	48	46	69	59	69	49	58	70	59	72	100	71	70	86	59	91	70	58	85	59
toad.F [zˌˈabə]	68	51	64	50	49	59	49	50	67	49	59	73	71	100	84	65	59	66	68	73	71	65
toad.PL [zˌˈabï]	62	62	51	62	49	65	49	62	62	48	65	68	70	84	100	64	65	65	80	68	70	71
basin.M [taz]	59	49	54	53	70	60	69	49	58	70	59	59	86	65	64	100	73	86	65	59	86	60
basin.PL [tɐzˈi]	64	57	55	62	52	81	52	58	64	52	82	65	59	59	65	73	100	59	66	65	60	82
bar.M [bar]	59	49	48	47	69	59	75	54	57	70	58	64	91	66	65	86	59	100	79	58	85	59
bar.PL [bˈarï]	64	63	50	61	49	65	54	68	63	49	66	68	70	68	80	65	66	79	100	62	64	65
slave.F [rɐˈba]	81	55	65	60	51	64	51	54	82	51	64	82	58	73	68	59	65	58	62	100	72	79
slave.M [rab]	59	49	54	53	70	60	69	49	58	70	59	58	85	71	70	86	60	85	64	72	100	74
slave.PL [rɐˈbi]	64	57	56	62	52	81	52	57	64	52	82	64	59	65	71	60	82	59	65	79	74	100

Table 9. Phonological similarities between all possible words in production, averaged over 100 toy language learners (in percents). Colour-coding: gradient of red for values above 60, with darker red indicating higher similarities. The network is trained on 44000 samples.

	owl.F [sɛvˈa]	**owl.PL** [sˈovĭ]	**rose.F** [rˈozə]	**rose.PL** [rˈozĭ]	**cat.M** [kot]	**cat.PL** [kɐtˈi]	**bot.M** [bot]	**bot.PL** [bˈotĭ]	**mine.F** [mɐjˈa]	**mine.M** [moj]	**mine.PL** [mɐjˈi]	**heat.F** [ʐɐˈra]	**heat.M** [ʐar]	**toad.F** [ʐˌˈabə]	**toad.PL** [ʐˌˈabĭ]	**basin.M** [taz]	**basin.PL** [tɐzˈi]	**bar.M** [bar]	**bar.PL** [bˈarĭ]	**slave.F** [rɐˈba]	**slave.M** [rab]	**slave.PL** [rɐˈbi]
owl.F [sɛvˈa]	100	79	71	60	58	60	57	60	74	59	60	73	58	72	60	59	61	58	59	74	59	61
owl.PL [sˈovĭ]	79	100	61	72	59	72	58	72	60	60	73	59	57	61	72	59	71	58	71	60	58	71
rose.F [rˈozə]	71	61	100	82	60	61	60	63	71	60	61	70	58	71	60	60	62	57	60	73	61	63
rose.PL [rˈozĭ]	60	72	82	100	59	72	59	73	59	58	71	60	57	59	70	59	71	57	71	62	59	72
cat.M [kot]	58	59	60	59	100	74	83	62	58	81	59	59	80	58	58	79	58	78	57	58	79	59
cat.PL [kɐtˈi]	60	72	61	72	74	100	60	75	61	59	75	60	58	60	70	59	74	58	71	61	59	74
bot.M [bot]	57	58	60	59	83	60	100	75	57	81	58	58	78	58	58	78	57	79	58	57	79	59
bot.PL [bˈotĭ]	60	72	63	73	62	75	75	100	61	60	74	59	57	61	72	58	72	59	72	60	58	72
mine.F [mɐjˈa]	74	60	71	59	58	61	57	61	100	73	78	73	58	72	60	59	61	58	60	75	59	62
mine.M [moj]	59	60	60	58	81	59	81	60	73	100	75	58	79	59	59	80	58	79	58	58	80	60
mine.PL [mɐjˈi]	60	73	61	71	59	75	58	74	78	75	100	58	57	61	72	59	73	58	72	60	58	74
heat.F [ʐɐˈra]	73	59	70	60	59	60	58	59	73	58	58	100	75	72	60	59	60	60	60	74	60	61
heat.M [ʐar]	58	57	58	57	80	58	78	57	58	79	57	75	100	61	59	80	58	81	59	58	80	59
toad.F [ʐˌˈabə]	72	61	71	59	58	60	58	61	72	59	61	72	61	100	82	60	61	58	60	73	61	63
toad.PL [ʐˌˈabĭ]	60	72	60	70	58	70	58	72	60	59	72	60	59	82	100	59	72	58	71	62	61	73
basin.M [taz]	59	59	60	59	79	59	78	58	59	80	59	59	80	60	59	100	75	80	59	59	80	60
basin.PL [tɐzˈi]	61	71	62	71	58	74	57	72	61	58	73	60	58	61	72	75	100	59	72	62	59	75
bar.M [bar]	58	58	57	57	78	58	79	59	58	79	58	60	81	58	58	80	59	100	76	60	81	61
bar.PL [bˈarĭ]	59	71	60	71	57	71	58	72	60	58	72	60	59	60	71	59	72	76	100	60	59	73
slave.F [rɐˈba]	74	60	73	62	58	61	57	60	75	58	60	74	58	73	62	59	62	60	60	100	74	80
slave.M [rab]	59	58	61	59	79	59	79	58	59	80	58	60	80	61	61	80	59	81	59	74	100	75
slave.PL [rɐˈbi]	61	71	63	72	59	74	59	72	62	60	74	61	59	63	73	60	75	61	73	80	75	100

An attentive reader might notice that the set of darker green cells in Table 8 could be a subset of the set of darker red cells in Table 9. This can be confirmed by examining Tables 10 and 11, which represent the excerpts from Tables 8 and 9, specifically displaying phonological similarities between words with the same morphosyntactic meaning <F>. In production, all such words fall into one group, exhibiting phonological similarity ranging from 70% to 75%, as shown in Table 11. Consequently, the similarities between all pairs are marked with very close shades of red in this table. On the other hand, Table 10 reveals clear categorisation within the same subgroup of words: words with the phonetic forms [sɐvˈa], [mɐjˈa], [ʐɐˈra], and [rɐˈba] are highly similar to each other (with similarities of 81%-82%), while words with the phonetic forms [rˈozə] and [ʐˈabə] form their own categories, being less similar to any other word.

Table 10. Phonological similarities between the six words sharing morphosyntactic meaning <F> **in comprehension**, averaged over 100 toy language learners (in percents). Colour-coding: gradient of green for values above 60, with darker green indicating higher similarities. The network is trained on 44000 samples.

	owl.F [sɐvˈa]	rose.F [rˈozə]	mine.F [mɐjˈa]	heat.F [ʐɐˈra]	toad.F [ʐˈabə]	slave.F [rɐˈba]
owl.F [sɐvˈa]	100	60	82	81	68	81
rose.F [rˈozə]	60	100	59	59	64	65
mine.F [mɐjˈa]	82	59	100	82	67	82
heat.F [ʐɐˈra]	81	59	82	100	73	82
toad.F [ʐˈabə]	68	64	67	73	100	73
slave.F [rɐˈba]	81	65	82	82	73	100

Table 11. Phonological similarities between the six words sharing morphosyntactic meaning <F> **in production**, averaged over 100 toy language learners (in percents). Colour-coding: gradient of red for values above 60, with darker red indicating higher similarities. The network is trained on 44000 samples.

	owl.F [sɐvˈa]	rose.F [rˈozə]	mine.F [mɐjˈa]	heat.F [ʐɐˈra]	toad.F [ʐˈabə]	slave.F [rɐˈba]
owl.F [sɐvˈa]	100	71	74	73	72	74
rose.F [rˈozə]	71	100	71	70	71	73
mine.F [mɐjˈa]	74	71	100	73	72	75
heat.F [ʐɐˈra]	73	70	73	100	72	74
toad.F [ʐˈabə]	72	71	72	72	100	73
slave.F [rɐˈba]	74	73	75	74	73	100

An even clearer example that visually illustrates the emergence of different phonological categorisations in comprehension and production can be seen in Tables 12 and 13. These tables are excerpts from Tables 8 and 9, respectively, and specifically represent phonological similarities between words with the same morphosyntactic meaning <M>. Table 13 demonstrates how all the words in this selection fall into one group. In Table 12, it can be observed that one production category from Table 13 is split into exactly two comprehension categories, based on the quality of the vowel. The first comprehension category includes the words [kot], [bot], and [moj], while the second comprehension category consists of the words [ʐar], [taz], [bar], and [rab].

Table 12. Phonological similarities between the seven words sharing morphosyntactic meaning <M> **in comprehension**, averaged over 100 toy language learners (in percents). Colour-coding: gradient of green for values above 60, with darker green indicating higher similarities. The network is trained on 44000 samples.

	cat.M [kot]	bot.M [bot]	mine.M [moj]	heat.M [ʐar]	basin.M [taz]	bar.M [bar]	slave.M [rab]
cat.M **[kot]**	100	90	86	69	70	69	70
bot.M **[bot]**	90	100	86	69	69	75	69
mine.M **[moj]**	86	86	100	70	70	70	70
heat.M **[ʐar]**	69	69	70	100	86	91	85
basin.M **[taz]**	70	69	70	86	100	86	86
bar.M **[bar]**	69	75	70	91	86	100	85
slave.M **[rab]**	70	69	70	85	86	85	100

Table 13. Phonological similarities between the seven words sharing morphosyntactic meaning <M> **in production**, averaged over 100 toy language learners (in percents). Colour-coding: gradient of red for values above 60, with darker red indicating higher similarities. The network is trained on 44000 samples.

	cat.M [kot]	bot.M [bot]	mine.M [moj]	heat.M [ʐar]	basin.M [taz]	bar.M [bar]	slave.M [rab]
cat.M [kot]	100	83	81	80	79	78	79
bot.M [bot]	83	100	81	78	78	79	79
mine.M [moj]	81	81	100	79	80	79	80
heat.M [ʐar]	80	78	79	100	80	81	80
basin.M [taz]	79	78	80	80	100	80	80
bar.M [bar]	78	79	79	81	80	100	81
slave.M [rab]	79	79	80	80	80	81	100

Another way to observe the difference in phonological categorisations between comprehension and production is by examining the phonological representations of words in both processes, placing them side by side for each word. Table 14 provides an example of such an analysis for two words: <owl.F>[sɐvˈa] and <owl.PL>[sˈovï]. For both words, the colour coding suggests some similar categories in comprehension and production. However, zooming into these categories, it is possible to see that sometimes the production similarities are higher than the comprehension similarities, and vice versa. For example, the comprehension similarities between [sɐvˈa] and three other words with the same vowel qualities ([mɐjˈa], [ʐɐˈra], and [rɐˈba]) are higher (82%, 81%, and 81%, respectively) than the corresponding production similarities (74%, 73%, and 74%, respectively). At the same time, the production similarity between <owl.F> and <owl.PL> is higher (79%) than the corresponding comprehension similarity (69%).

The discovery of two different phonological categorisations of entire words in comprehension and production indicates that each word in the toy language has two distinct phonological representations: one in comprehension and one in production. This raises the question of how these two phonological representations compare to each other. To address this question, Table 15 provides such a comparison for two types of virtual language learners: those who learned the language using a training dataset of 44000 samples (row 1) and those who trained on 88000 samples (row 2). Each column in Table 15 represents the similarity between phonological representations in comprehension and production for one word. The inclusion of more experienced learners in the analysis aims to demonstrate that the observed categorisation in the first row of Table 15 becomes even more pronounced as learners are exposed to more training data.

Table 15 illustrates that the highest similarities between two phonological representations of a word are observed for <cat.M>[kot], <bot.M>[bot], <mine.M>[moj], <heat.M>[ʐar], <basin.M>[taz], <bar.M>[bar], and <slave.M>[rab]. These words share the morphosyntactic meaning <M>, which is the only meaning among all lexical and morphosyntactic meanings which does not have variation in its phonetic realisations, always being realised as [∅]. Moreover, with more training, similarity between the two phonological representations for these words remains high compared to all the other words, where it decreases.

Table 14. Phonological similarities between <owl.F> [sɐvˈa], <owl.PL> [sˈovïˈ] and all possible words in the toy language, averaged over 100 language learners (in percents). Colour-coding: gradient of green for values above 60 in comprehension, gradient of red for values above 60 in production, with darker red and darker green indicating higher similarities. The network is trained on 44000 samples.

(a) Phonological similarities between [sɐvˈa] and all possible words in comprehension and between <owl.F> and all possible words in production

	owl.F [sɐvˈa]	owl.PL [sˈovïˈ]	rose.F [rˈozə]	rose.PL [rˈozïˈ]	cat.M [kot]	cat.PL [kɐtˈˌi]	bot.M [bot]	bot.PL [bˈotïˈ]	mine.F [mɐjˈa]	mine.M [moj]	mine.PL [mɐjˈˌi]	heat.F [ʐɐˈra]	heat.M [ʐˌar]	toad.F [ʐˌˈabə]	toad.PL [ʐˌˈabïˈ]	basin.M [taz]	basin.PL [tɐzˈˌi]	bar.M [bar]	bar.PL [bˈarïˈ]	slave.F [rɐˈba]	slave.M [rab]	slave.PL [rɐˈbˌi]
[sɐvˈa]	100	69	60	56	51	63	52	55	82	52	65	81	60	68	62	59	64	59	64	81	59	64
owl.F	100	79	71	60	58	60	57	60	74	59	60	73	58	72	60	59	61	58	59	74	59	61

(a) Phonological similarities between [sˈovïˈ] and all possible words in comprehension and between <owl.PL> and all possible words in production

	owl.F [sɐvˈa]	owl.PL [sˈovïˈ]	rose.F [rˈozə]	rose.PL [rˈozïˈ]	cat.M [kot]	cat.PL [kɐtˈˌi]	bot.M [bot]	bot.PL [bˈotïˈ]	mine.F [mɐjˈa]	mine.M [moj]	mine.PL [mɐjˈˌi]	heat.F [ʐɐˈra]	heat.M [ʐˌar]	toad.F [ʐˌˈabə]	toad.PL [ʐˌˈabïˈ]	basin.M [taz]	basin.PL [tɐzˈˌi]	bar.M [bar]	bar.PL [bˈarïˈ]	slave.F [rɐˈba]	slave.M [rab]	slave.PL [rɐˈbˌi]
[sˈovïˈ]	69	100	69	81	65	57	66	81	55	65	58	56	49	51	62	49	57	49	63	55	49	57
owl.PL	79	100	61	72	59	72	58	72	60	60	73	59	57	61	72	59	71	58	71	60	58	71

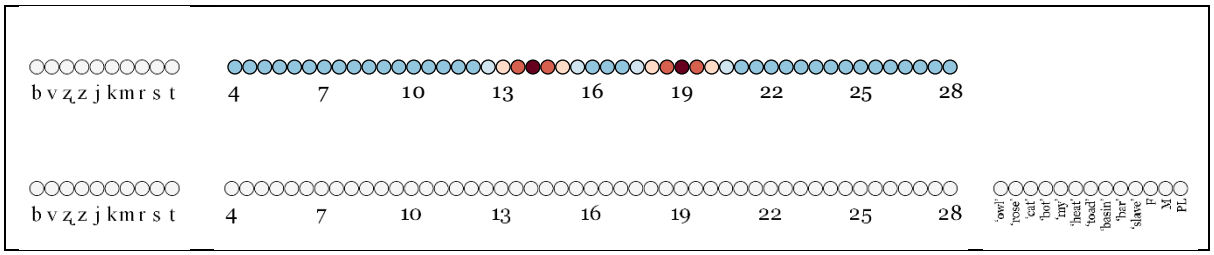
Table 15. Similarities between phonological representations of the same words in comprehension and production; means and standard deviations averaged over 100 language learners (in percents). Colour-coding: gradient of blue, with darker blue indicating higher similarities The network is trained on 44000 samples (1st row) and on 88000 samples (2nd row).

	owl.F [sɐvˈa]	owl.PL [sˈovïˈ]	rose.F [rˈozə]	rose.PL [rˈozïˈ]	cat.M [kot]	cat.PL [kɐtˈˌi]	bot.M [bot]	bot.PL [bˈotïˈ]	mine.F [mɐjˈa]	mine.M [moj]	mine.PL [mɐjˈˌi]	heat.F [ʐɐˈra]	heat.M [ʐˌar]	toad.F [ʐˌˈabə]	toad.PL [ʐˌˈabïˈ]	basin.M [taz]	basin.PL [tɐzˈˌi]	bar.M [bar]	bar.PL [bˈarïˈ]	slave.F [rɐˈba]	slave.M [rab]	slave.PL [rɐˈbˌi]
network trained on 44000 samples	83	83	87	86	91	82	93	83	83	92	83	83	89	86	84	93	85	93	86	85	91	85
network trained on 88000 samples	79	80	82	79	90	76	91	77	77	91	78	77	86	80	78	91	79	89	79	79	88	79

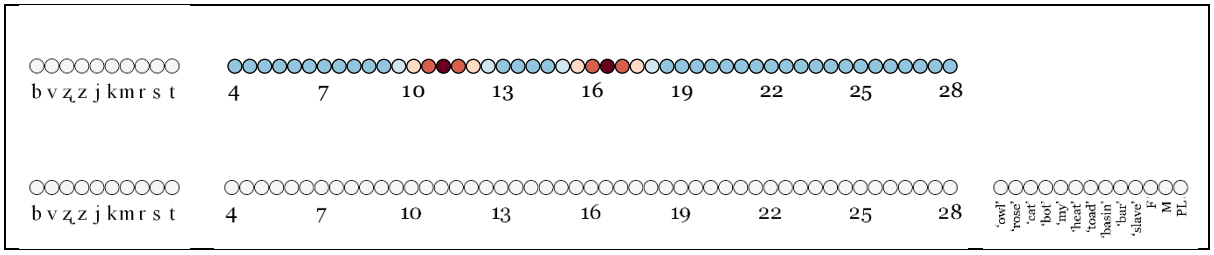
4.5 Phonological category of [ɐ] in comprehension

Now that the question regarding the phonological categories for entire words in comprehension and production has been addressed, it is time to specifically examine the sound [ɐ] and to determine if it forms a distinct category in the hidden representation of the network. From the perspective of comprehension, this can be achieved by comparing the hidden states of the network after processing inputs consisting of only one vowel, namely [a], [o], or [ɐ], in the first syllable, as illustrated in Figure 22. To indicate a partial auditory input, symbol " – " is used to represent input components with activities set to 0.

(a) Input level for [– a – –]



(b) Input level for [– o – –]



(c) Input level for [– ɐ – –]

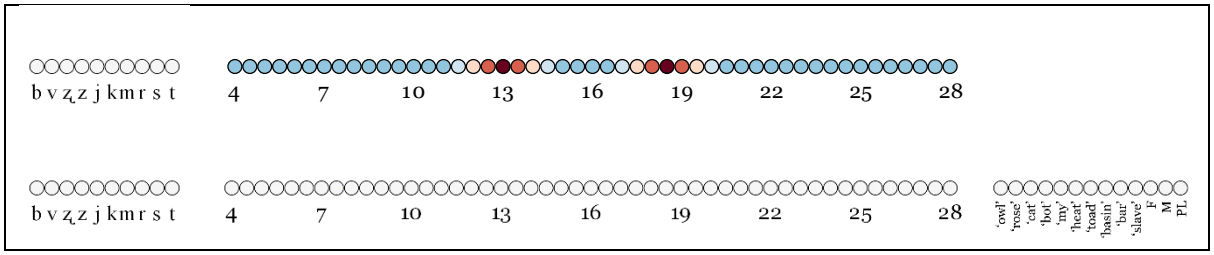


Figure 22. Input levels for [– a – –], [– o – –], and [– ɐ – –]

Table 16 presents the phonological similarities between the inputs [– a – –], [– o – –], and [– ɐ – –]. It can be observed that the phonological representations of [– ɐ – –] and [– a – –] are much more similar than those of [– ɐ – –] and [– o – –]. However, an 83% similarity between [– ɐ – –] and [– a – –] does not necessarily imply that they belong to

the same phonological category, especially considering the relatively high similarity of 62% between [−ɐ− −] and [−o− −]. In other words, the difference between [ɐ] and [a] is only approximately twice as small as the difference between [ɐ] and [o], given that the concept of difference is the inverse of similarity. Considering this, it is possible that, from a comprehension perspective, [ɐ] indeed forms its own category. Moreover, this category is phonologically more similar to the category formed by [a] then the one formed by [o].

Table 16. Phonological similarities between [−a− −], [−o− −], and [−ɐ− −] **in comprehension**, averaged over 100 toy language learners (in percents); means and standard deviations averaged over 100 toy language learners (in percents). The network is trained on 44000 samples.

	[-a--]	**[-o--]**	**[-ɐ--]**
[-a--]	100 (0)	62 (9)	83 (8)
[-o--]	62 (9)	100 (0)	62 (10)
[-ɐ--]	83 (8)	62 (10)	100 (0)

4.6 Phonological category of [ɐ] in production

Section 4.4 provides evidence for the emergence of a distinct phonological category for [ɐ] in comprehension. However, the categorisation of [ɐ] in production might differ from that in comprehension, since different phonological categories emerge in comprehension and production, as shown in Section 4.3. How can we approach determining a possible category for [ɐ] in production? This question is seemingly less straightforward than a similar question in comprehension because the input for production consists of different meanings rather than different sounds. Finding phonological similarities between different meanings cannot provide much insight into sound categorisations, especially when each meaning has several phonetic realisations in the language.

However, upon further consideration, answering this question is easier than it initially seemed. The key to answering it is to remember that the hidden states of the network, after producing the same meanings, are always the same because the steps (3.3), (3.9), and (3.1), which are repeated 30 times in a production process, are deterministic. Therefore, each meaning in the toy language will have its own unique phonological representation in production. On the one hand, when we examine the lexical and morphosyntactic components of the meanings separately, we can observe that almost all of them can be phonetically realised in two different ways, except for the morphosyntactic meaning <M> (which can

only be realised as [∅]). But on the other hand, in production, none of them can have two different phonological representations.

Let us take the lexical pair <owl.F>[sɐvˈa] – <owl.PL>[sˈovï] as an example. The lexical meaning <owl> can be phonetically realised in two ways: [sɐv] and [sov]. However, in production, there is only one possible phonological representation of <owl> because there is only one possible hidden state of the network after the meaning <owl> has been produced. Therefore, the question of whether there is a separate phonological category for [ɐ] in production can be answered directly, without analysing any hidden representations: no, there is not.

However, another question that arises is: is the phonological representation of <owl> in production closer to the phonological representation of [sɐv] or that of [sov] in comprehension? Answering this question will help determine which of the two possible phonetic realisations of <owl>, [sɐv] or [sov], is phonologically more similar to the lexical meaning <owl>.

I will attempt to answer this question for six lexical pairs (or triplets) that represent the phenomenon of moderate akanje in the toy language. Figure 23 illustrates an example of three input levels for one lexical pair (in this case, the pair <owl.F>[sɐvˈa] – <owl.PL>[sˈovï]). Once again, the symbol " – " is used to represent input components with activities set to 0. After applying these three inputs to the network and registering the hidden representations of each input, the cosine similarities are computed between the hidden representation of <owl. – > and both hidden representations of [sov –] and [sɐv –]. The results of these comparisons for all six lexical pairs with akanje are presented in Table 17.

Table 17. Phonological similarities between the lexical meanings and their corresponding phonetic realisations; means and standard deviations averaged over 100 language learners (in percents). The network is trained on 44000 samples.

	[sov-]	**[sɐv-]**		**[ʐar-]**	**[ʐɐr-]**
<**owl**>	82 (8)	80 (8)	<**heat**>	83 (8)	83 (9)
	[kot-]	**[kɐt-]**		**[taz-]**	**[tɐz-]**
<**cat**>	87 (5)	84 (9)	<**basin**>	88 (6)	87 (7)
	[moj-]	**[mɐj-]**		**[rab-]**	**[rɐb-]**
<**my**>	88 (5)	83 (7)	<**slave**>	84 (8)	83 (8)

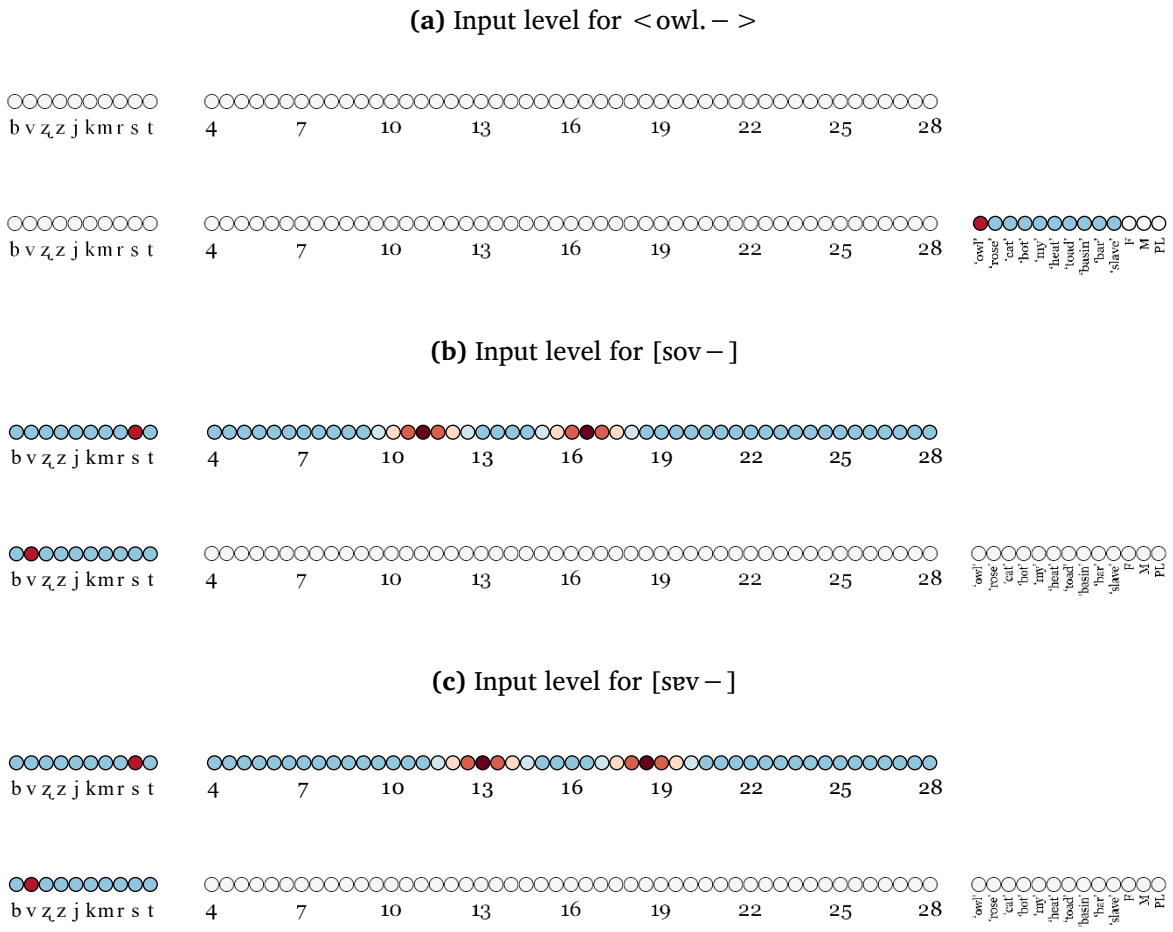


Figure 23. Input levels for (a) the lexical meaning <owl>,
(b) its phonetic realisation [sov –], and
(c) its phonetic realisation [sɐv –]

Table 17 reveals that for each lexical pair (or triplet), the difference in phonological similarities between the lexical meaning and its two phonetic realisations is minimal. Nevertheless, there is a slight trend of higher phonological similarity between the lexical meanings <owl>, <cat>, and <my> and the sound [o], as opposed to the sound [ɐ]. Furthermore, there is an even subtler trend of higher phonological similarity between the lexical meanings <heat>, <basin>, and <slave> and the sound [a], compared to the sound [ɐ]. However, these findings should be interpreted with caution, considering the relatively small difference between the mean values and the relatively high standard deviations across 100 language learners.

5 Discussion

5.1 Answering the research questions

In this thesis, I aimed to answer the following two questions: first, whether distinct categories for different sounds would emerge in the hidden levels of the trained network; and second, if such categories were observed, whether the category for the sound [ɐ], resulting from moderate akanje, would exhibit greater similarity to the category for [a] or to the category for [o].

However, these questions now appear somewhat incomplete due to certain underlying assumptions that have proven to be inaccurate. The initial premise was to conceptualise the hidden levels of the neural network in my model as the phonological level in an abstract grammar model. By the phonological level in an abstract grammar model, I mean a realm of hidden representations within the human brain that connect sounds and meanings in a language. With this conceptualisation, both questions were based on the assumption that the phonological level is a monolith, and that processing in both directions, from sound to meaning and from meaning to sound, is symmetrical. This assumption implied that a single phonological representation was expected for both processes. However, the analysis conducted in Section 4.4 revealed a discrepancy: the phonological categorisation that emerges in the comprehension process differs from the one observed in the production process.

Therefore, the answer to the first research question is yes, distinct categories for different sounds do emerge in the hidden levels of the trained network. However, this is not the end of the story, as distinct categories for different meanings also emerge in the hidden levels of the trained network, and the two sets of categories do not coincide.

The second research question regarding the phonological category for [ɐ] is similarly incomplete, much like the first one. It primarily focuses on the comprehension process, not because the production process was unimportant, but rather because a difference between the two was not anticipated. This becomes evident when considering my initial expectations. I hypothesised that [ɐ] would be more phonologically similar to [a] in case of the “strong phonetic influence” and to [o] in case of “strong semantic influence” on phonological categorisation. However, it is now apparent that the phonological representations of sounds in comprehension are strongly influenced by phonetics, while the phonological forms of meanings in production are strongly influenced by semantics. Thus, there is no need to choose between these two influences, as they both coexist in phonology but manifest in two distinct phonological representations.

Therefore, the answer to the second question is that the phonological category for the sound [ɐ] does indeed emerge in comprehension, and it exhibits greater similarity to the phonological category for [a] than to the category for [o]. However, there is no distinct category for [ɐ] in production.

5.2 What do different phonological categorisations mean?

The discovery of two distinct phonological categorisations in production and comprehension aligns with the BiPhon-OT grammar model (Boersma 2011), as depicted in Figure 2. In BiPhon-OT, phonological representations consist of two distinct forms: the underlying form and the surface form. By interpreting the results obtained in the Section 4.4, it becomes possible to conceptualise the phonological representations in production as the underlying forms in BiPhon-OT, while the phonological representations in comprehension can be seen as the surface forms in BiPhon-OT.

The underlying form in BiPhon-OT is connected to semantic representations through a set of lexical constraints, which can account for the observed semantic influence on the phonological representations in production. Simultaneously, the surface form is connected to phonetic representations through a set of cue constraints, which in turn explains the influence of phonetics on the phonological representations observed in comprehension.

The question then arises, what in my analysis could represent the faithfulness constraints that connect the underlying and surface forms? The answer to this question lies in Table 15, which illustrates similarities between the phonological representations in comprehension and production for the same words. Additionally, Table 17 demonstrates phonological similarities between the lexical meanings and their corresponding phonetic realisations for the lexical pairs and triplets with moderate akanje.

It would be interesting to use all the findings and their conceptualisations, aligning them with the BiPhon-OT grammar model, and attempt to write down the underlying and surface forms for some of the words in the toy language. Expressing the surface forms does not pose a significant challenge since they are motivated by phonetics. Therefore, it is logical to reuse the IPA symbols that were employed throughout this thesis to approximate the acoustic qualities of sounds.

I will solely focus on the stems of the words for this exercise, as only the stems in the toy language exhibit the phenomenon of moderate akanje and were the primary focus of the analysis. For example, the surface form for the stem of [sɐvˈa] could be written as /sɐv/, while

the surface form for the stem of [sˈovï] could be written as /sov/. These two different notations reflect the distinct category of /ɐ/ in phonological categorisation for comprehension. Similarly, for the lexical triplet [rɐˈba]—[rab]—[rɐˈbɨ], two possible surface forms are /rɐb/ and /rab/. On the other hand, for the lexical pair [bot]— [bˈotï], there is only one possible surface form: /bot/.

Contrary to the relatively straightforward task of writing down the surface forms, expressing the underlying forms presents a slightly greater challenge. Traditionally, in phonology, the IPA is employed to represent both surface and underlying forms. However, as observed in Section 4.4, the categories that emerge in production, and thus belong to the underlying form, have little to do with sounds, and are instead more motivated by semantics. Moreover, Section 4.6 provides evidence that a single semantic form corresponds to only one phonological representation in production, and therefore can have only one underlying form. For example, there exists one and only one underlying form for the lexical meaning <owl>, which exhibits nearly equal phonological similarity with both surface forms /sɐv/ and /sov/, with a slight preference for /sov/ as indicated in Table 17.

Choosing how to denote the underlying form in this case may appear arbitrary; however, I will follow the phonological tradition by using the IPA for this purpose. Furthermore, when deciding between |ɐ| and |o|, as well as between |ɐ| and |a|, I will select |o| and |a|, respectively. This choice is motivated by the slightly more faithful connection between the underlying forms and the phonemes /o/ and /a/, compared to the phoneme /ɐ/, as demonstrated in Table 17. For example, the underlying form for the lexical meaning <owl> will be represented as |sov|, while for the lexical meaning <slave>, the underlying form will be denoted as |rab|, etc.

Taking these notations into consideration, it becomes easier to interpret the results presented in Table 15 and address the question of why underlying and surface forms are the most similar for the words that share the morphosyntactic meaning <M>. The reason behind this lies in the fact that the surface forms of these words are the most faithful to their corresponding underlying form as demonstrated by the notations |kot|/kot/, |bot|/bot/, |moj|/moj/, |ʐar|/ʐar/, |taz|/taz/, |bar|/bar/, and |rab|/rab/. On the other hand, all the other words in the toy language consist of two syllables and undergo either a moderate (in the first syllable) or extreme (in the second syllable) form of vowel reduction. As a result, their surface forms are less faithful to their underlying forms. This is precisely what we can observe in Table 15, especially in its second row, which compares the phonological representations of more experienced learners.

6 Conclusion

In this thesis, I have developed a neural network model capable of successfully acquiring a toy language composed of a set of Russian words representing a specific case of vowel reduction in Russian known as akanje. Through this model, I have investigated the underlying principles of phonological categorisation in akanje. The trained network has demonstrated its effectiveness in language comprehension and production, resulting in the emergence of phonological categories at the hidden levels of the network. Remarkably, it has been observed that the set of phonological categories that emerges in language comprehension differs from the set of phonological categories that emerges in language production. These findings align with the grammar model of Bidirectional Phonology and Phonetics (BiPhon-OT), highlighting the distinction between the two phonological representations used in this model: Underlying and Surface forms.

References

Barnes, Jonathan. 2007. Phonetics and phonology in Russian unstressed vowel reduction: A study in hyperarticulation. Boston University, ms.

Blaho, Sylvia. 2008. *The syntax of phonology: A radically substance-free approach*. Tromsø: University of Tromsø doctoral dissertation.

Boersma, Paul. 2011. A programme for bidirectional phonology and phonetics and their acquisition and evolution. In Anton Benz & Jason Mattausch (eds.), *Linguistik Aktuell/Linguistics Today*, vol. 180, 33–72. Amsterdam: John Benjamins Publishing Company. https://doi.org/10.1075/la.180.02boe.

Boersma, Paul. 2019. Simulated distributional learning in deep Boltzmann machines leads to the emergence of discrete categories. In *Proceedings of the 19th International Congress of Phonetic Sciences*, 1520–1524.

Boersma, Paul, Titia Benders & Klaas Seinhorst. 2020. Neural network models for phonology and phonetics. *Journal of Language Modelling* 8(1). 103-177. https://doi.org/10.15398/jlm.v8i1.224.

Boersma, Paul, Kateřina Chládková & Titia Benders. 2022. Phonological features emerge substance-freely from the phonetics and the morphology. *Canadian Journal of Linguistics/Revue canadienne de linguistique* 67(4). 611–669. https://doi.org/10.1017/cnj.2022.39.

Boersma, Paul & David Weenink. 2023. Praat: doing phonetics by computer [Computer program]. Version 6.3.09. http://www.praat.org/. (20 April, 2023).

Browman, Catherine P. & Louis Goldstein. 1992. Articulatory phonology: An overview. *Phonetica* 49(3–4). 155–180. https://doi.org/10.1159/000261913.

Chrabaszcz, Anna, Matthew Winn, Candise Y. Lin & William J. Idsardi. 2014. Acoustic cues to perception of word stress by English, Mandarin, and Russian speakers. *Journal of Speech, Language, and Hearing Research* 57(4). 1468–1479. https://doi.org/10.1044/2014_JSLHR-L-13-0279.

Crosswhite, Katherine. 2000a. The analysis of extreme vowel reduction. *UCLA Working Papers in Linguistics* 4(Papers in Phonology 4). 1–12.

Crosswhite, Katherine. 2000b. Vowel reduction in Russian: A unified account of standard, dialectal, and “dissimilative” patterns. *University of Rochester Working Papers in the Language Sciences* Spring 2000 (1). 107–172.

Ellson, John, Emden Gansner, Yifan Hu & Stephen North. 2023. Graphviz: open-source graph visualisation software [Computer program]. Version 8.0.5. https://graphviz.org/. (3 May, 2023).

Enguehard, Guillaume. 2019. A thought on the form and the substance of Russian vowel reduction. In Denisa Lenertová, Roland Meyer, Radek Šimík & Luka Szucsich (eds.), *Advances in formal Slavic linguistics 2016*, 109–125. Berlin: Language Science Press. http://dx.doi.org/10.5281/zenodo.2545515.

Flemming, Edward. 2001. Scalar and categorical phenomena in a unified model of phonetics and phonology. *Phonology* 18(1). 7–44. https://doi.org/10.1017/S0952675701004006.

Flemming, Edward. 2002. *Auditory representations in phonology* (Outstanding Dissertations in Linguistics). New York: Routledge.

Hale, Mark & Charles Reiss. 1998. Formal and empirical arguments concerning phonological acquisition. *Linguistic Inquiry* 29(4). 656–683. https://doi.org/10.1162/002438998553914.

Hale, Mark & Charles Reiss. 2000. "Substance abuse" and "dysfunctionalism": Current trends in phonology. *Linguistic Inquiry* 31(1). 157–169. https://doi.org/10.1162/002438900554334.

Hall, Daniel Currie & Laura Teddiman. 2014. On substance in phonology. In *Proceedings of the 2014 annual conference of the Canadian Linguistic Association.*, 1–14.

Iosad, Pavel. 2012. Vowel reduction in Russian: No phonetics in phonology. *Journal of Linguistics* 48(3). 521–571. https://doi.org/10.1017/S0022226712000102.

Iosad, Pavel. 2013. *Representation and variation in substance-free phonology: A case study in Celtic*. Tromsø: University of Tromsø doctoral dissertation.

Iosad, Pavel. 2017. *A Substance-free framework for phonology: an analysis of the Breton dialect of Bothoa* (Edinburgh Studies in Theoretical Linguistics). Edinburgh: Edinburgh University Press.

Kniazev, Sergey & Evgeny Shaulskiy. 2007. The development of akanje in Russian: New data. In *Proceedings of ICPhS XVI (16th International Congress of Phonetic Sciences)*, 1425–1428.

Kuhl, Patricia. 1991. Human adults and human infants show a "perceptual magnet effect" for the prototypes of speech categories, monkeys do not. *Perception & psychophysics* 50(2). 93–107.

Leonard, Matthew K. & Edward F. Chang. 2014. Dynamic speech representations in the human temporal lobe. *Trends in Cognitive Sciences* 18(9). 472–479. https://doi.org/10.1016/j.tics.2014.05.001.

Lunt, Horace G. 1979. On akanje and linguistic theory. *Harvard Ukrainian Studies* 3/4(Part 2). 595–608.

Morén, Bruce. 2003. The parallel structures model of feature geometry. *Working Papers of the Cornell Phonetics Laboratory* 15. 194–270.

Ohala, John J. 1990. There is no interface between phonology and phonetics: a personal view. *Journal of Phonetics* 18(2). 153–171. https://doi.org/10.1016/S0095-4470(19)30399-7.

Padgett, Jaye. 2004. Russian vowel reduction and Dispersion Theory. *Phonological studies* 7. 81–96.

Padgett, Jaye & Marija Tabain. 2005. Adaptive Dispersion Theory and phonological vowel reduction in Russian. *Phonetica* 62(1). 14–54. https://doi.org/10.1159/000087223.

Popham, Sara F., Alexander G. Huth, Natalia Y. Bilenko, Fatma Deniz, James S. Gao, Anwar O. Nunez-Elizalde & Jack L. Gallant. 2021. Visual and linguistic semantic representations are aligned at the border of human visual cortex. *Nature Neuroscience* 24(11). 1628–1636. https://doi.org/10.1038/s41593-021-00921-6.

Prince, Alan & Paul Smolensky. 1993. *Optimality Theory: Constraint interaction in generative grammar*. Technical report 2. Rutgers University Center for Cognitive Science. (Published in 2002 as technical documentation by Rutgers University.) https://doi.org/10.7282/T34M92MV

Prince, Alan & Paul Smolensky. 1997. Optimality: From neural networks to universal grammar. *Science* 275(5306). 1604–1610. https://doi.org/10.1126/science.275.5306.1604.

Prince, Alan & Paul Smolensky. 2003. Optimality Theory in phonology. In William Frawley (ed.), *International encyclopedia of linguistics* (3), 212–222. 2nd edn. Oxford University Press. https://doi.org/10.1093/acref/9780195139778.001.0001.

Reiss, Charles. 2017. Substance Free phonology. In S. J. Hannahs & Anna R. K. Bosch (eds.), *The Routledge Handbook of Phonological Theory*, 425–452. 1st edn. Routledge.

Reiss, Charles & Veno Volenec. 2022. Conquer primal fear: Phonological features are innate and substance-free. *Canadian Journal of Linguistics/Revue canadienne de linguistique* 67(4). 581–610.

Salakhutdinov, Ruslan & Geoffrey Hinton. 2009. Deep Boltzmann machines. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Clearwater Beach, Florida, USA.

Salakhutdinov, Ruslan. 2010. Learning deep Boltzmann machines using adaptive MCMC. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 943–950.

Salakhutdinov, Ruslan & Hugo Larochelle. 2010. Efficient learning of deep Boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (JMLR Workshop and Conference Proceedings), vol. 9, 693–700. PMLR.

Shchupak, Anastasia. 2022. *Akanje and Optimality Theory: employing the bidirectional grammar model BiPhon to analyse one case of vowel reduction in Russian*. Amsterdam: University of Amsterdam paper for the course Linguistic Theories in the BA Linguistics. https://www.fon.hum.uva.nl/archive/2022/other/2022-lingthy-AnastasiaShchupak.pdf.

Smolensky, Paul. 1996. On the comprehension/production dilemma in child language. *Linguistic Inquiry* 27(4). 720–731.

The MathWorks Inc. 2023. MATLAB [Computer program]. Version 9.14 (R2023a). Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com. (4 May, 2023).

Appendix 1: Proficiency tables

Table A1.1. Comprehension proficiency: cosine similarities between the entire comprehension outputs for all possible standard auditory inputs (in rows) and the entire standard input representations of all possible words (in columns); means and standard deviations averaged over 100 toy language learners (in percents) [6]. The network is trained on **44000 samples**.

	owl.F	**owl.PL**	**rose.F**	**rose.PL**	**cat.M**	**cat.PL**	**bot.M**	**bot.PL**	**mine.F**	**mine.M**	**mine.PL**	**heat.F**	**heat.M**	**toad.F**	**toad.PL**	**basin.M**	**basin.PL**	**bar.M**	**bar.PL**	**slave.F**	**slave.M**	**slave.PL**
	[sɐvˈa]	**[sˈovĭ]**	**[rˈozə]**	**[rˈozĭ]**	**[kot]**	**[kɐtˈɨ]**	**[bot]**	**[bˈotĭ]**	**[mɐjˈa]**	**[moj]**	**[mɐjˈɨ]**	**[ʐɐˈra]**	**[ʐ̩ar]**	**[ʐ̩ˈabə]**	**[ʐ̩ˈabĭ]**	**[taz]**	**[tɐzˈɨ]**	**[bar]**	**[bˈarĭ]**	**[rɐˈba]**	**[rab]**	**[rɐˈbɨ]**
[sɐvˈa]	96 (4)	38 (5)	20 (4)	6 (5)	8 (6)	23 (4)	9 (6)	6 (5)	63 (4)	8 (5)	22 (4)	64 (6)	26 (6)	39 (5)	25 (5)	26 (6)	23 (4)	26 (6)	26 (6)	64 (3)	26 (7)	23 (4)
[sˈovĭ]	37 (6)	95 (6)	30 (5)	64 (5)	33 (7)	17 (7)	33 (6)	63 (6)	5 (5)	33 (7)	17 (7)	6 (5)	5 (9)	-3 (4)	31 (4)	4 (8)	16 (7)	5 (8)	31 (5)	5 (5)	4 (8)	17 (6)
[rˈozə]	20 (5)	32 (4)	95 (2)	65 (3)	33 (4)	-7 (4)	33 (4)	32 (4)	20 (5)	33 (4)	-7 (4)	20 (5)	3 (4)	29 (3)	-1 (5)	14 (4)	6 (4)	3 (4)	-2 (4)	33 (5)	15 (4)	6 (4)
[rˈozĭ]	3 (5)	62 (5)	63 (4)	95 (6)	32 (8)	14 (6)	32 (9)	62 (5)	3 (4)	33 (9)	15 (5)	3 (5)	2 (10)	-2 (5)	29 (5)	14 (10)	27 (5)	2 (9)	28 (5)	16 (5)	14 (8)	27 (6)
[kot]	7 (2)	30 (2)	31 (2)	30 (2)	98 (1)	36 (2)	83 (1)	42 (2)	7 (3)	73 (2)	7 (2)	7 (3)	48 (2)	4 (2)	3 (2)	48 (2)	7 (2)	48 (2)	3 (3)	7 (3)	48 (2)	7 (3)
[kɐtˈɨ]	24 (4)	17 (6)	-5 (4)	18 (5)	40 (9)	94 (7)	22 (10)	30 (6)	24 (4)	11 (10)	61 (6)	25 (4)	28 (9)	14 (5)	37 (6)	28 (10)	61 (6)	28 (10)	37 (5)	25 (4)	28 (9)	62 (6)
[bot]	6 (3)	32 (1)	32 (2)	32 (2)	83 (1)	17 (3)	98 (1)	60 (1)	6 (3)	73 (1)	6 (3)	6 (3)	47 (2)	3 (2)	3 (2)	47 (3)	5 (3)	57 (2)	14 (2)	6 (3)	47 (2)	6 (3)
[bˈotĭ]	4 (6)	61 (7)	30 (4)	61 (6)	47 (10)	29 (7)	64 (10)	93 (9)	5 (5)	36 (11)	16 (6)	5 (5)	8 (13)	-2 (5)	29 (5)	7 (13)	16 (6)	19 (12)	42 (7)	5 (6)	8 (13)	16 (6)
[mɐjˈa]	63 (3)	5 (5)	19 (4)	5 (5)	8 (5)	24 (3)	8 (6)	5 (5)	96 (3)	37 (5)	57 (3)	63 (3)	25 (5)	39 (5)	24 (5)	24 (5)	23 (4)	25 (5)	24 (5)	63 (3)	25 (5)	24 (3)
[moj]	7 (3)	30 (2)	31 (2)	30 (2)	72 (2)	7 (3)	73 (1)	31 (2)	36 (3)	98 (1)	36 (3)	7 (3)	48 (2)	3 (2)	3 (3)	48 (3)	7 (3)	48 (3)	3 (3)	7 (3)	48 (2)	7 (3)
[mɐjˈɨ]	24 (4)	18 (5)	-6 (4)	18 (5)	9 (6)	62 (3)	8 (6)	17 (5)	58 (3)	38 (6)	95 (4)	24 (3)	26 (6)	14 (4)	38 (5)	26 (6)	62 (4)	26 (6)	38 (5)	24 (4)	27 (6)	62 (4)
[ʐɐˈra]	63 (4)	6 (6)	19 (5)	6 (6)	9 (8)	24 (3)	9 (8)	6 (6)	63 (5)	9 (7)	24 (3)	96 (5)	56 (5)	52 (6)	39 (5)	27 (6)	24 (4)	38 (6)	39 (5)	63 (4)	27 (6)	24 (3)
[ʐ̩ar]	26 (3)	4 (2)	3 (2)	3 (2)	48 (2)	26 (3)	48 (2)	3 (2)	26 (3)	48 (2)	26 (3)	54 (2)	99 (1)	44 (2)	44 (2)	74 (1)	26 (3)	84 (1)	44 (2)	26 (3)	74 (2)	25 (3)
[ʐ̩ˈabə]	41 (4)	-2 (4)	28 (3)	-1 (4)	4 (4)	16 (4)	4 (4)	-1 (4)	41 (4)	4 (4)	16 (4)	54 (4)	46 (3)	96 (1)	66 (3)	34 (3)	15 (4)	35 (3)	33 (4)	54 (4)	46 (3)	29 (4)
[ʐ̩ˈabĭ]	24 (5)	29 (5)	0 (4)	30 (4)	5 (10)	36 (6)	5 (9)	30 (6)	24 (5)	4 (10)	35 (7)	38 (5)	46 (8)	65 (5)	95 (7)	34 (9)	36 (6)	34 (8)	63 (6)	38 (5)	45 (8)	49 (7)
[taz]	25 (2)	3 (2)	14 (2)	14 (2)	48 (2)	26 (3)	48 (2)	3 (2)	25 (2)	48 (2)	26 (2)	25 (2)	73 (1)	32 (1)	32 (1)	99 (1)	55 (2)	74 (1)	32 (1)	25 (2)	73 (1)	26 (2)
[tɐzˈɨ]	24 (4)	17 (5)	7 (4)	30 (5)	10 (7)	62 (4)	10 (7)	18 (4)	25 (4)	10 (7)	62 (5)	25 (4)	28 (7)	15 (5)	39 (5)	57 (6)	95 (5)	28 (7)	39 (5)	25 (4)	28 (7)	62 (5)
[bar]	25 (2)	4 (2)	4 (2)	4 (2)	48 (2)	25 (3)	58 (2)	15 (2)	25 (3)	48 (2)	24 (3)	37 (3)	84 (1)	34 (2)	33 (2)	74 (2)	25 (3)	99 (1)	61 (1)	25 (3)	74 (1)	25 (3)
[bˈarĭ]	27 (4)	30 (5)	-1 (4)	30 (5)	6 (10)	38 (6)	18 (10)	43 (5)	27 (5)	7 (10)	38 (6)	39 (5)	47 (8)	32 (4)	64 (6)	36 (9)	38 (7)	64 (8)	96 (6)	26 (4)	35 (8)	37 (6)
[rɐˈba]	63 (3)	4 (5)	32 (4)	17 (5)	7 (5)	24 (3)	7 (6)	4 (5)	63 (3)	7 (6)	24 (4)	63 (3)	25 (5)	52 (5)	38 (5)	25 (5)	24 (3)	25 (5)	25 (4)	96 (3)	55 (4)	57 (4)
[rab]	24 (3)	3 (2)	14 (2)	14 (2)	48 (2)	25 (3)	47 (2)	3 (2)	25 (3)	48 (2)	25 (2)	24 (3)	73 (1)	43 (2)	42 (1)	73 (1)	25 (2)	73 (1)	31 (1)	54 (3)	99 (1)	54 (2)
[rɐˈbɨ]	24 (4)	17 (5)	6 (4)	29 (5)	9 (8)	62 (5)	9 (8)	17 (5)	24 (4)	9 (8)	62 (5)	24 (4)	28 (7)	27 (4)	51 (6)	28 (7)	62 (5)	28 (8)	38 (6)	58 (4)	57 (7)	95 (6)

[6] The average comprehension accuracy over all comprehended words (per word shown in the diagonal of the table): M = 96.27%, STD = 3.68%.

Table A1.2. Comprehension proficiency: cosine similarities between the entire comprehension outputs for all possible standard auditory inputs (in rows) and the entire standard input representations of all possible words (in columns); means and standard deviations averaged over 100 toy language learners (in percents)[7]. The network is trained on **88000 samples**.

	owl.F	**owl.PL**	**rose.F**	**rose.PL**	**cat.M**	**cat.PL**	**bot.M**	**bot.PL**	**mine.F**	**mine.M**	**mine.PL**	**heat.F**	**heat.M**	**toad.F**	**toad.PL**	**basin.M**	**basin.PL**	**bar.M**	**bar.PL**	**slave.F**	**slave.M**	**slave.PL**
	[sɐvˈa]	**[sˈovɨ]**	**[rˈozə]**	**[rˈozɨ]**	**[kot]**	**[kɐtˈɨ]**	**[bot]**	**[bˈotɨ]**	**[mɐjˈa]**	**[moj]**	**[mɐjˈɨ]**	**[ʐɐˈra]**	**[ʐˌar]**	**[ʐˌˈabə]**	**[ʐˌˈabɨ]**	**[taz]**	**[tɐzˈɨ]**	**[bar]**	**[bˈarɨ]**	**[rɐˈba]**	**[rab]**	**[rɐˈbɨ]**
[sɐvˈa]	92 (11)	37 (5)	17 (8)	5 (6)	12 (12)	23 (5)	13 (13)	6 (7)	61 (8)	13 (13)	23 (5)	60 (10)	29 (11)	38 (7)	25 (6)	30 (12)	24 (4)	29 (12)	25 (6)	61 (8)	30 (12)	24 (4)
[sˈovɨ]	38 (5)	94 (9)	30 (4)	62 (7)	36 (10)	17 (6)	36 (10)	62 (7)	6 (5)	36 (10)	17 (6)	6 (5)	8 (12)	-2 (5)	30 (5)	8 (12)	17 (6)	8 (12)	30 (5)	6 (5)	8 (12)	16 (6)
[rˈozə]	18 (6)	32 (5)	93 (8)	64 (6)	35 (9)	-7 (5)	35 (9)	32 (5)	18 (6)	35 (8)	-6 (5)	18 (5)	5 (10)	28 (5)	-1 (6)	17 (8)	6 (5)	6 (10)	-1 (6)	30 (6)	16 (8)	6 (5)
[rˈozɨ]	5 (6)	60 (8)	63 (8)	92 (11)	37 (11)	14 (7)	38 (11)	60 (7)	5 (7)	38 (11)	14 (7)	4 (7)	9 (13)	-1 (7)	28 (6)	20 (12)	27 (8)	9 (13)	28 (6)	17 (7)	19 (12)	26 (7)
[kot]	7 (2)	31 (2)	32 (2)	31 (2)	98 (2)	36 (3)	83 (2)	43 (2)	7 (3)	73 (2)	7 (3)	7 (3)	48 (2)	4 (3)	3 (3)	47 (3)	7 (4)	47 (3)	3 (3)	7 (3)	48 (2)	8 (3)
[kɐtˈɨ]	28 (7)	13 (7)	-2 (6)	13 (7)	48 (12)	88 (12)	31 (14)	26 (7)	28 (7)	20 (14)	56 (10)	28 (7)	36 (13)	17 (7)	32 (8)	37 (13)	57 (10)	37 (13)	33 (8)	28 (7)	37 (13)	57 (9)
[bot]	7 (2)	32 (2)	32 (2)	32 (2)	83 (1)	18 (2)	98 (2)	61 (2)	7 (3)	73 (2)	6 (3)	7 (3)	47 (3)	3 (2)	3 (2)	47 (2)	6 (3)	57 (3)	14 (3)	7 (2)	47 (2)	6 (2)
[bˈotɨ]	7 (6)	58 (9)	30 (5)	58 (9)	53 (14)	27 (8)	69 (12)	89 (11)	6 (6)	41 (15)	14 (7)	6 (6)	14 (16)	-1 (5)	27 (8)	13 (15)	14 (8)	24 (15)	38 (9)	7 (6)	14 (16)	14 (8)
[mɐjˈa]	58 (9)	7 (7)	16 (7)	7 (7)	16 (15)	24 (5)	16 (15)	6 (7)	90 (12)	45 (13)	56 (5)	58 (10)	33 (14)	35 (9)	25 (6)	33 (14)	24 (5)	32 (14)	25 (7)	58 (10)	33 (14)	24 (5)
[moj]	7 (4)	31 (3)	32 (2)	31 (3)	72 (2)	7 (2)	73 (2)	31 (3)	36 (4)	98 (3)	36 (3)	7 (4)	48 (4)	3 (3)	3 (3)	48 (3)	7 (3)	47 (4)	3 (5)	7 (3)	48 (2)	7 (2)
[mɐjˈɨ]	27 (5)	15 (7)	-3 (6)	15 (8)	18 (14)	58 (9)	18 (14)	15 (8)	59 (6)	46 (12)	89 (12)	28 (6)	35 (14)	16 (7)	34 (9)	35 (13)	57 (10)	35 (13)	34 (9)	28 (6)	35 (14)	58 (9)
[ʐɐˈra]	60 (10)	6 (7)	16 (7)	6 (6)	14 (14)	25 (5)	15 (14)	7 (6)	60 (9)	15 (14)	25 (5)	92 (11)	61 (11)	49 (10)	39 (7)	33 (14)	25 (5)	44 (12)	39 (7)	60 (9)	33 (13)	25 (5)
[ʐˌar]	25 (2)	4 (2)	3 (2)	3 (2)	48 (2)	25 (2)	48 (2)	4 (2)	25 (2)	48 (2)	25 (2)	54 (2)	99 (1)	44 (1)	44 (1)	74 (1)	25 (2)	84 (1)	44 (1)	25 (2)	74 (1)	25 (2)
[ʐˌˈabə]	39 (6)	-1 (5)	28 (5)	-1 (5)	6 (10)	14 (5)	7 (10)	-1 (5)	39 (7)	6 (10)	14 (5)	52 (6)	47 (7)	94 (7)	65 (5)	36 (8)	14 (5)	36 (8)	33 (4)	51 (6)	47 (8)	26 (4)
[ʐˌˈabɨ]	27 (6)	27 (5)	2 (7)	28 (6)	11 (13)	34 (8)	11 (13)	28 (5)	27 (6)	11 (13)	34 (7)	40 (6)	51 (10)	66 (7)	92 (9)	40 (11)	34 (8)	40 (11)	60 (7)	40 (6)	51 (10)	46 (8)
[taz]	25 (2)	3 (2)	15 (2)	15 (2)	48 (2)	25 (2)	48 (2)	3 (2)	25 (2)	48 (2)	25 (2)	25 (2)	73 (1)	32 (2)	32 (1)	99 (1)	54 (2)	73 (1)	32 (1)	25 (2)	73 (1)	25 (2)
[tɐzˈɨ]	28 (6)	13 (8)	10 (6)	26 (7)	19 (15)	57 (9)	19 (15)	13 (7)	28 (6)	19 (15)	57 (9)	29 (7)	38 (14)	18 (6)	34 (9)	66 (12)	89 (11)	38 (13)	34 (9)	29 (6)	38 (13)	57 (9)
[bar]	25 (3)	4 (2)	4 (2)	4 (2)	48 (2)	24 (3)	58 (2)	15 (2)	25 (2)	48 (2)	24 (2)	36 (2)	84 (1)	33 (1)	33 (1)	73 (1)	24 (2)	99 (1)	62 (2)	25 (3)	74 (1)	24 (3)
[bˈarɨ]	28 (6)	28 (7)	2 (6)	28 (6)	14 (14)	34 (8)	25 (14)	40 (7)	28 (6)	14 (15)	35 (8)	41 (7)	53 (12)	34 (5)	60 (8)	42 (12)	35 (8)	70 (11)	91 (10)	28 (6)	42 (12)	34 (8)
[rɐˈba]	60 (9)	6 (6)	28 (8)	18 (6)	14 (14)	25 (4)	14 (14)	6 (6)	60 (9)	13 (13)	24 (5)	60 (8)	31 (13)	48 (11)	37 (7)	31 (12)	24 (4)	31 (12)	26 (6)	92 (11)	59 (10)	57 (4)
[rab]	25 (2)	3 (2)	14 (2)	14 (2)	48 (2)	25 (3)	47 (2)	3 (2)	25 (3)	48 (3)	25 (3)	25 (3)	73 (1)	43 (1)	43 (1)	73 (2)	25 (2)	73 (1)	32 (2)	54 (3)	99 (2)	54 (2)
[rɐˈbɨ]	28 (6)	14 (7)	9 (6)	26 (8)	17 (12)	58 (9)	16 (13)	14 (8)	28 (6)	16 (12)	58 (9)	29 (6)	35 (12)	30 (7)	47 (9)	35 (12)	58 (8)	35 (12)	35 (8)	61 (7)	63 (10)	90 (10)

[7] The average comprehension accuracy over all comprehended words (per word shown in the diagonal of the table): M = 93.5%, STD = 7.59%.

Table A1.3. Production proficiency: cosine similarities between the entire production outputs for all possible semantic inputs (in rows) and the entire standard input representations of all possible words (in columns); means and standard deviations averaged over 100 toy language learners (in percents) [8]. The network is trained on **44000 samples**; semantic nodes of the input are **unclamped**.

	owl.F [sɐvˈa]	**owl.PL** [sˈoviˈ]	**rose.F** [rˈozə]	**rose.PL** [rˈoziˈ]	**cat.M** [kot]	**cat.PL** [kɐtˈɨ]	**bot.M** [bot]	**bot.PL** [bˈotiˈ]	**mine.F** [mɐjˈa]	**mine.M** [moj]	**mine.PL** [mɐjˈɨ]	**heat.F** [ʐɐˈra]	**heat.M** [ʐˌar]	**toad.F** [ʐˌˈabə]	**toad.PL** [ʐˌˈabiˈ]	**basin.M** [taz]	**basin.PL** [tɐzˈɨ]	**bar.M** [bar]	**bar.PL** [bˈariˈ]	**slave.F** [rɐˈba]	**slave.M** [rab]	**slave.PL** [rɐˈbɨ]
owl.F	74 (8)	44 (10)	37 (10)	11 (10)	18 (8)	8 (11)	17 (9)	11 (10)	40 (8)	17 (9)	8 (10)	40 (9)	18 (9)	39 (9)	14 (10)	19 (9)	8 (10)	18 (9)	12 (11)	41 (9)	20 (9)	9 (10)
owl.PL	44 (9)	75 (10)	13 (9)	42 (11)	22 (10)	33 (9)	22 (10)	42 (10)	10 (9)	22 (10)	33 (9)	9 (9)	20 (11)	12 (9)	41 (10)	22 (11)	34 (9)	20 (11)	40 (11)	11 (10)	21 (11)	34 (9)
rose.F	26 (11)	17 (11)	74 (12)	47 (11)	26 (9)	5 (9)	24 (9)	19 (10)	28 (9)	25 (10)	3 (10)	30 (10)	19 (12)	38 (11)	10 (12)	21 (11)	8 (12)	19 (12)	14 (12)	34 (11)	22 (10)	9 (10)
rose.PL	3 (10)	45 (11)	47 (11)	73 (12)	25 (10)	29 (10)	23 (10)	46 (10)	5 (9)	24 (11)	28 (9)	8 (11)	19 (10)	11 (11)	37 (11)	20 (11)	32 (13)	19 (11)	42 (11)	12 (10)	22 (9)	34 (11)
cat.M	14 (6)	18 (8)	20 (8)	20 (7)	86 (7)	42 (7)	68 (8)	26 (8)	14 (8)	62 (7)	15 (8)	15 (9)	59 (8)	16 (8)	15 (9)	58 (7)	14 (8)	59 (7)	15 (9)	15 (7)	60 (7)	16 (8)
cat.PL	12 (10)	37 (12)	13 (11)	38 (11)	47 (9)	69 (10)	27 (10)	43 (12)	11 (10)	20 (9)	39 (12)	11 (10)	23 (12)	14 (11)	39 (11)	21 (12)	37 (11)	22 (12)	38 (10)	12 (10)	23 (11)	39 (11)
bot.M	11 (8)	22 (8)	24 (8)	24 (8)	68 (9)	15 (10)	85 (7)	46 (8)	14 (8)	65 (8)	12 (9)	13 (10)	55 (8)	15 (9)	16 (9)	57 (8)	14 (9)	57 (8)	15 (9)	16 (10)	58 (8)	15 (9)
bot.PL	11 (11)	44 (12)	17 (9)	44 (11)	30 (12)	35 (12)	50 (11)	70 (12)	10 (11)	27 (11)	31 (10)	9 (12)	18 (14)	12 (10)	39 (10)	22 (14)	34 (10)	21 (14)	37 (13)	13 (11)	21 (13)	33 (11)
mine.F	38 (11)	10 (10)	36 (10)	10 (10)	18 (10)	8 (10)	19 (11)	9 (10)	70 (13)	49 (9)	42 (11)	36 (12)	19 (9)	39 (10)	13 (11)	19 (10)	7 (11)	19 (11)	10 (12)	38 (12)	21 (10)	9 (10)
mine.M	13 (7)	20 (7)	20 (6)	20 (6)	63 (6)	13 (6)	63 (6)	20 (7)	43 (7)	89 (6)	43 (6)	14 (7)	58 (6)	14 (6)	15 (6)	58 (7)	14 (7)	59 (6)	15 (7)	14 (7)	59 (6)	14 (7)
mine.PL	9 (11)	38 (10)	12 (10)	39 (10)	20 (11)	36 (11)	21 (10)	39 (10)	42 (11)	50 (9)	69 (13)	8 (10)	20 (10)	13 (9)	40 (10)	21 (10)	35 (12)	21 (10)	39 (11)	9 (11)	21 (10)	36 (12)
heat.F	40 (13)	10 (11)	39 (11)	17 (10)	25 (14)	14 (11)	24 (14)	12 (12)	39 (11)	21 (13)	10 (10)	68 (11)	46 (11)	38 (11)	16 (11)	20 (13)	11 (11)	26 (13)	14 (11)	43 (14)	23 (14)	14 (11)
heat.M	17 (8)	17 (8)	20 (7)	21 (7)	62 (8)	20 (10)	62 (8)	20 (9)	18 (8)	60 (8)	18 (8)	43 (8)	80 (8)	22 (9)	23 (9)	59 (8)	19 (9)	64 (9)	22 (10)	21 (9)	60 (8)	20 (9)
toad.F	30 (12)	13 (12)	38 (10)	12 (10)	19 (11)	7 (11)	20 (11)	14 (13)	34 (12)	23 (12)	11 (12)	32 (14)	27 (13)	69 (14)	43 (14)	27 (12)	10 (12)	19 (14)	12 (17)	32 (14)	26 (13)	9 (12)
toad.P	10 (11)	40 (11)	12 (10)	37 (11)	18 (13)	29 (13)	19 (14)	39 (12)	13 (12)	22 (12)	34 (13)	12 (13)	27 (12)	45 (11)	70 (13)	27 (12)	33 (12)	20 (13)	40 (17)	11 (13)	27 (13)	33 (14)
basin.M	21 (7)	13 (7)	20 (8)	21 (8)	55 (6)	19 (8)	57 (7)	14 (8)	20 (7)	56 (7)	20 (8)	19 (7)	63 (8)	22 (7)	22 (7)	88 (8)	47 (7)	63 (8)	20 (9)	19 (7)	63 (7)	19 (8)
basin.PL	17 (10)	31 (10)	14 (10)	40 (11)	15 (10)	42 (10)	18 (10)	32 (10)	16 (9)	17 (10)	43 (10)	15 (10)	27 (10)	18 (9)	44 (10)	56 (8)	74 (10)	27 (9)	41 (11)	15 (10)	27 (10)	42 (11)
bar.M	19 (8)	9 (8)	12 (7)	12 (8)	54 (7)	20 (9)	57 (8)	15 (9)	22 (8)	55 (8)	21 (7)	24 (9)	71 (9)	24 (8)	24 (8)	65 (8)	18 (9)	88 (7)	49 (8)	23 (9)	69 (7)	23 (9)
bar.PL	15 (12)	33 (11)	9 (11)	35 (11)	14 (11)	36 (12)	19 (10)	36 (11)	16 (11)	16 (11)	39 (10)	19 (12)	34 (12)	20 (10)	46 (11)	27 (11)	34 (11)	54 (10)	76 (11)	16 (14)	32 (12)	39 (12)
slave.F	44 (11)	6 (11)	38 (12)	14 (10)	17 (13)	14 (10)	19 (13)	6 (10)	44 (11)	18 (12)	16 (10)	42 (13)	24 (13)	44 (12)	21 (11)	21 (12)	11 (10)	24 (13)	12 (10)	73 (12)	51 (11)	45 (10)
slave.M	20 (9)	15 (9)	21 (9)	21 (8)	58 (8)	18 (10)	60 (8)	17 (9)	20 (8)	59 (8)	19 (8)	19 (11)	61 (9)	23 (9)	24 (10)	57 (9)	14 (9)	61 (9)	19 (11)	44 (8)	83 (8)	44 (8)
slave.PL	17 (13)	32 (11)	14 (10)	40 (10)	20 (13)	40 (11)	22 (11)	33 (11)	16 (11)	21 (13)	41 (10)	16 (12)	27 (12)	19 (13)	45 (12)	24 (12)	38 (13)	28 (11)	39 (12)	45 (12)	53 (11)	70 (12)

[8] The average production accuracy over all produced words (per word shown in the diagonal of the table): M = 76.05%, STD = 10.18%.

Table A1.4. Production proficiency: cosine similarities between the entire production outputs for all possible semantic inputs (in rows) and the entire standard input representations of all possible words (in columns); means and standard deviations averaged over 100 toy language learners (in percents) [9]. The network is trained on **44000 samples**; semantic nodes of the input are **clamped**.

	owl.F [sɐvˈa]	**owl.PL** [sˈovĭ]	**rose.F** [rˈozə]	**rose.PL** [rˈozĭ]	**cat.M** [kot]	**cat.PL** [kɐtˈɨ]	**bot.M** [bot]	**bot.PL** [bˈotĭ]	**mine.F** [mɐjˈa]	**mine.M** [moj]	**mine.PL** [mɐjˈɨ]	**heat.F** [ʐɐˈra]	**heat.M** [ʐˌar]	**toad.F** [ʐˌˈabə]	**toad.PL** [ʐˌˈabĭ]	**basin.M** [taz]	**basin.PL** [tɐzˈɨ]	**bar.M** [bar]	**bar.PL** [bˈarĭ]	**slave.F** [rɐˈba]	**slave.M** [rab]	**slave.PL** [rɐˈbɨ]
owl.F	76 (7)	47 (9)	37 (9)	12 (10)	18 (7)	8 (10)	18 (8)	12 (10)	41 (8)	18 (7)	8 (10)	41 (8)	19 (7)	38 (9)	13 (10)	19 (7)	8 (9)	19 (7)	13 (9)	41 (8)	19 (7)	8 (10)
owl.PL	45 (9)	78 (8)	13 (8)	43 (9)	21 (7)	34 (9)	21 (7)	43 (9)	10 (9)	21 (7)	34 (8)	10 (9)	19 (7)	11 (8)	41 (9)	20 (7)	34 (8)	19 (8)	42 (9)	10 (9)	19 (8)	33 (8)
rose.F	27 (8)	18 (8)	81 (8)	52 (8)	24 (7)	3 (8)	24 (6)	19 (8)	28 (8)	24 (7)	2 (7)	28 (9)	16 (8)	37 (8)	8 (9)	25 (8)	13 (9)	16 (8)	10 (9)	39 (9)	25 (7)	13 (7)
rose.PL	3 (7)	47 (9)	52 (7)	81 (8)	23 (6)	28 (9)	24 (6)	49 (7)	4 (7)	23 (7)	28 (8)	5 (9)	16 (8)	9 (8)	38 (8)	23 (7)	38 (9)	16 (7)	40 (8)	15 (8)	24 (6)	39 (8)
cat.M	14 (6)	19 (6)	20 (6)	20 (6)	89 (6)	44 (6)	72 (6)	30 (7)	14 (6)	63 (6)	15 (6)	15 (6)	60 (6)	15 (7)	15 (7)	59 (6)	14 (7)	59 (6)	15 (7)	14 (6)	59 (6)	15 (6)
cat.PL	11 (9)	37 (10)	11 (9)	38 (10)	49 (7)	75 (9)	29 (7)	48 (10)	10 (9)	19 (7)	40 (9)	11 (8)	21 (8)	12 (8)	39 (9)	21 (8)	41 (9)	21 (8)	38 (9)	12 (8)	22 (7)	42 (9)
bot.M	11 (6)	24 (6)	26 (5)	26 (6)	75 (5)	19 (7)	91 (4)	53 (5)	11 (6)	67 (5)	10 (6)	11 (6)	54 (5)	12 (6)	13 (6)	56 (6)	11 (6)	62 (6)	20 (6)	13 (6)	56 (5)	12 (6)
bot.PL	8 (8)	47 (9)	19 (7)	48 (9)	34 (7)	41 (8)	54 (7)	80 (9)	8 (8)	25 (7)	31 (8)	7 (8)	15 (8)	10 (8)	39 (8)	17 (9)	33 (8)	25 (8)	47 (9)	10 (8)	17 (8)	33 (9)
mine.F	40 (9)	9 (10)	35 (9)	9 (10)	18 (8)	9 (9)	18 (8)	9 (10)	75 (9)	49 (7)	44 (9)	39 (10)	19 (8)	37 (8)	11 (9)	19 (8)	9 (10)	19 (8)	10 (10)	40 (9)	20 (8)	10 (9)
mine.M	13 (6)	20 (6)	20 (6)	20 (6)	63 (5)	13 (6)	64 (5)	20 (6)	43 (6)	90 (5)	43 (6)	13 (6)	59 (6)	14 (6)	15 (6)	59 (6)	13 (6)	59 (6)	15 (7)	13 (6)	59 (6)	13 (6)
mine.PL	10 (9)	39 (8)	11 (8)	39 (8)	19 (7)	38 (9)	20 (7)	39 (9)	45 (9)	50 (7)	73 (9)	9 (8)	20 (6)	12 (7)	41 (8)	21 (7)	38 (9)	21 (6)	40 (8)	10 (8)	21 (7)	38 (9)
heat.F	43 (10)	9 (9)	35 (9)	12 (9)	20 (9)	13 (9)	20 (9)	9 (9)	43 (9)	19 (9)	12 (9)	76 (9)	49 (8)	44 (10)	21 (11)	20 (9)	13 (9)	30 (9)	20 (10)	44 (9)	21 (9)	13 (9)
heat.M	18 (7)	16 (6)	18 (7)	18 (7)	61 (7)	19 (7)	60 (6)	17 (7)	19 (7)	60 (6)	18 (7)	47 (7)	85 (7)	26 (8)	27 (7)	61 (6)	19 (7)	69 (7)	27 (8)	20 (6)	61 (6)	19 (7)
toad.F	32 (9)	11 (10)	38 (8)	10 (8)	16 (8)	6 (8)	16 (8)	11 (9)	33 (9)	17 (8)	7 (8)	42 (10)	32 (9)	80 (9)	52 (10)	24 (8)	6 (8)	21 (9)	17 (10)	41 (10)	32 (9)	15 (9)
toad.P	9 (9)	41 (9)	12 (9)	40 (8)	16 (10)	31 (9)	16 (10)	40 (9)	9 (9)	17 (9)	32 (9)	18 (10)	31 (9)	52 (9)	79 (10)	24 (8)	32 (9)	21 (9)	45 (11)	18 (10)	31 (9)	41 (10)
basin.M	20 (6)	13 (6)	22 (6)	23 (6)	56 (5)	21 (6)	57 (6)	13 (6)	20 (6)	56 (6)	21 (6)	20 (6)	65 (6)	22 (7)	22 (7)	91 (6)	50 (6)	65 (6)	22 (7)	20 (6)	65 (6)	20 (6)
basin.PL	16 (8)	30 (9)	15 (7)	41 (9)	14 (7)	45 (8)	15 (8)	30 (9)	16 (8)	14 (7)	45 (8)	15 (8)	27 (7)	17 (7)	44 (8)	57 (6)	79 (8)	27 (7)	44 (8)	15 (8)	27 (7)	45 (8)
bar.M	19 (6)	10 (5)	11 (5)	11 (5)	54 (5)	19 (7)	62 (5)	20 (6)	21 (6)	55 (5)	20 (5)	29 (5)	75 (5)	25 (6)	25 (6)	67 (6)	19 (6)	92 (5)	53 (6)	21 (6)	69 (5)	21 (6)
bar.PL	15 (10)	33 (8)	6 (8)	35 (8)	11 (8)	38 (9)	21 (7)	43 (8)	16 (10)	13 (8)	40 (8)	25 (10)	37 (8)	21 (8)	50 (9)	28 (8)	38 (9)	56 (7)	83 (8)	17 (11)	30 (7)	41 (9)
slave.F	44 (8)	4 (9)	41 (9)	16 (9)	15 (7)	14 (9)	15 (7)	4 (9)	44 (9)	15 (7)	14 (9)	44 (9)	23 (8)	50 (8)	25 (9)	22 (8)	13 (10)	23 (8)	13 (9)	78 (8)	53 (7)	48 (9)
slave.M	19 (7)	14 (7)	23 (7)	23 (7)	58 (6)	19 (8)	58 (6)	15 (7)	19 (7)	58 (6)	19 (6)	19 (7)	63 (6)	29 (7)	29 (7)	62 (6)	18 (6)	63 (6)	20 (7)	48 (7)	88 (6)	48 (6)
slave.PL	15 (9)	32 (9)	17 (8)	43 (9)	17 (9)	43 (9)	18 (9)	33 (9)	15 (9)	17 (9)	43 (8)	14 (9)	24 (7)	24 (9)	51 (10)	23 (8)	42 (10)	25 (8)	41 (10)	49 (9)	54 (7)	76 (9)

[9] The average production accuracy over all produced words (per word shown in the diagonal of the table): M = 81.64%, STD = 7.59%.

Table A1.5. Production proficiency: cosine similarities between the entire production outputs for all possible semantic inputs (in rows) and the entire standard input representations of all possible words (in columns); means and standard deviations averaged over 100 toy language learners (in percents) [10]. The network is trained on **88000 samples**; semantic nodes of the input are **clamped**.

	owl.F	**owl.PL**	**rose.F**	**rose.PL**	**cat.M**	**cat.PL**	**bot.M**	**bot.PL**	**mine.F**	**mine.M**	**mine.PL**	**heat.F**	**heat.M**	**toad.F**	**toad.PL**	**basin.M**	**basin.PL**	**bar.M**	**bar.PL**	**slave.F**	**slave.M**	**slave.PL**
	[sɐvˈa]	**[sˈovɨ]**	**[rˈozə]**	**[rˈozɨ]**	**[kot]**	**[kɐtˈɨ]**	**[bot]**	**[bˈotɨ]**	**[mɐjˈa]**	**[moj]**	**[mɐjˈɨ]**	**[ʐɐˈra]**	**[ʐˌar]**	**[ʐˌabə]**	**[ʐˌabɨ]**	**[taz]**	**[tɐzˈɨ]**	**[bar]**	**[bˈarɨ]**	**[rɐˈba]**	**[rab]**	**[rɐˈbɨ]**
owl.F	71 (9)	46 (10)	34 (9)	11 (10)	18 (8)	11 (9)	18 (8)	11 (10)	36 (10)	18 (8)	11 (8)	36 (10)	17 (8)	34 (9)	11 (10)	17 (8)	11 (9)	17 (8)	10 (10)	37 (9)	18 (7)	11 (9)
owl.PL	46 (8)	73 (10)	14 (10)	39 (10)	19 (7)	33 (11)	19 (7)	39 (10)	12 (8)	19 (7)	32 (10)	11 (8)	17 (7)	12 (8)	37 (9)	17 (7)	32 (11)	17 (7)	37 (9)	12 (9)	18 (7)	33 (11)
rose.F	25 (9)	16 (10)	74 (9)	48 (10)	23 (7)	7 (10)	23 (7)	17 (9)	27 (9)	23 (8)	7 (8)	28 (11)	16 (9)	33 (9)	6 (10)	22 (9)	15 (10)	15 (9)	9 (10)	35 (10)	22 (7)	15 (9)
rose.PL	3 (9)	42 (10)	49 (9)	73 (10)	24 (7)	30 (10)	23 (8)	43 (10)	4 (10)	23 (7)	29 (9)	6 (10)	16 (9)	8 (9)	32 (11)	22 (9)	37 (11)	16 (7)	36 (9)	13 (9)	23 (8)	38 (10)
cat.M	13 (7)	18 (6)	19 (7)	18 (6)	87 (6)	43 (7)	70 (6)	28 (7)	13 (7)	61 (6)	14 (7)	13 (7)	58 (7)	15 (7)	14 (8)	58 (7)	14 (7)	58 (6)	14 (7)	14 (8)	59 (7)	15 (8)
cat.PL	11 (9)	35 (11)	13 (9)	35 (11)	48 (7)	69 (11)	28 (8)	44 (11)	9 (10)	18 (8)	35 (11)	10 (10)	19 (8)	12 (9)	34 (10)	18 (8)	36 (11)	17 (8)	32 (10)	11 (9)	19 (8)	36 (11)
bot.M	11 (7)	22 (6)	24 (6)	24 (6)	72 (6)	19 (7)	89 (5)	50 (6)	12 (8)	65 (6)	12 (8)	11 (7)	54 (6)	12 (6)	12 (6)	56 (6)	13 (8)	62 (7)	19 (8)	13 (7)	56 (5)	13 (7)
bot.PL	10 (11)	41 (12)	20 (8)	42 (10)	31 (8)	39 (11)	50 (7)	71 (10)	9 (10)	23 (8)	31 (11)	8 (11)	13 (9)	10 (8)	33 (9)	16 (9)	32 (12)	22 (10)	39 (11)	11 (10)	16 (8)	32 (11)
mine.F	35 (10)	10 (10)	34 (8)	10 (10)	18 (7)	12 (9)	18 (7)	10 (10)	69 (11)	48 (7)	46 (9)	34 (11)	16 (8)	33 (9)	8 (9)	16 (8)	11 (9)	16 (8)	8 (10)	35 (11)	17 (7)	12 (9)
mine.M	12 (6)	20 (6)	20 (5)	20 (6)	63 (5)	13 (6)	63 (5)	20 (6)	43 (6)	89 (5)	43 (6)	13 (6)	57 (4)	14 (5)	14 (5)	57 (5)	13 (6)	57 (5)	14 (5)	13 (6)	57 (4)	13 (6)
mine.PL	11 (9)	37 (10)	13 (8)	37 (10)	19 (7)	34 (10)	19 (7)	37 (10)	45 (9)	50 (7)	69 (10)	10 (9)	17 (7)	11 (7)	35 (10)	17 (7)	34 (10)	17 (7)	35 (9)	11 (9)	18 (6)	35 (10)
heat.F	37 (11)	8 (11)	33 (10)	11 (11)	19 (8)	16 (10)	18 (8)	8 (10)	38 (11)	19 (8)	15 (10)	69 (11)	45 (8)	40 (11)	18 (11)	18 (8)	15 (10)	26 (9)	17 (12)	38 (9)	18 (8)	16 (10)
heat.M	14 (8)	18 (7)	19 (7)	20 (7)	63 (6)	16 (8)	62 (6)	20 (7)	16 (8)	62 (7)	15 (8)	43 (8)	82 (7)	23 (8)	23 (8)	57 (7)	14 (8)	66 (7)	23 (8)	15 (8)	57 (7)	15 (8)
toad.F	29 (11)	14 (12)	37 (10)	10 (12)	19 (11)	9 (10)	17 (10)	12 (13)	29 (12)	19 (9)	9 (10)	34 (13)	25 (11)	70 (11)	43 (12)	20 (10)	8 (10)	15 (10)	9 (12)	35 (12)	26 (10)	15 (11)
toad.P	9 (11)	40 (11)	15 (10)	37 (10)	19 (10)	29 (11)	18 (9)	39 (11)	10 (10)	19 (9)	30 (11)	15 (11)	26 (10)	48 (10)	71 (11)	21 (9)	29 (11)	17 (9)	38 (12)	16 (10)	27 (9)	36 (11)
basin.M	18 (7)	13 (6)	23 (6)	23 (6)	57 (6)	18 (7)	57 (6)	13 (6)	18 (7)	57 (6)	18 (7)	17 (8)	62 (7)	19 (7)	19 (8)	88 (6)	47 (7)	62 (7)	19 (8)	17 (8)	62 (7)	18 (8)
basin.PL	16 (10)	29 (9)	18 (7)	40 (8)	15 (7)	41 (9)	15 (7)	29 (9)	16 (10)	15 (7)	41 (10)	15 (10)	23 (8)	17 (9)	39 (11)	53 (7)	74 (9)	23 (8)	38 (11)	15 (10)	23 (7)	41 (10)
bar.M	16 (8)	12 (7)	15 (6)	14 (6)	55 (6)	15 (9)	63 (6)	21 (7)	18 (8)	57 (6)	17 (8)	24 (8)	70 (6)	20 (7)	20 (7)	63 (7)	15 (9)	87 (6)	48 (7)	18 (8)	64 (7)	17 (8)
bar.PL	15 (10)	32 (10)	11 (9)	33 (9)	13 (7)	34 (11)	23 (7)	41 (10)	16 (9)	15 (8)	36 (10)	22 (10)	33 (9)	19 (10)	42 (12)	24 (9)	34 (11)	52 (8)	75 (11)	16 (10)	27 (9)	37 (10)
slave.F	39 (10)	5 (10)	39 (9)	16 (10)	16 (7)	16 (8)	16 (7)	5 (10)	38 (10)	16 (7)	17 (9)	38 (10)	20 (8)	44 (11)	21 (11)	18 (8)	15 (9)	20 (8)	9 (11)	72 (10)	49 (8)	50 (8)
slave.M	16 (9)	16 (7)	24 (7)	24 (7)	59 (7)	16 (9)	60 (7)	17 (8)	16 (9)	59 (6)	15 (9)	15 (10)	59 (9)	25 (9)	25 (10)	57 (9)	13 (9)	60 (8)	17 (9)	43 (9)	84 (9)	43 (9)
slave.PL	14 (10)	31 (9)	18 (10)	41 (10)	17 (7)	39 (11)	17 (7)	31 (10)	14 (10)	17 (7)	40 (10)	13 (9)	21 (8)	23 (9)	46 (10)	19 (7)	37 (10)	22 (8)	36 (10)	46 (9)	50 (7)	72 (10)

[10] The average production accuracy over all produced words (per word shown in the diagonal of the table): M = 76.27%, STD = 8.95%.

Appendix 2: Source code of class dbm (MATLAB)

```
classdef dbm < handle
    properties (Constant)
        n_nodes_deep_levels = [config.getInstance.dbm_h1_nodes config.getInstance.dbm_h2_nodes]
        learning_rate = config.getInstance.dbm_learning_rate

        % equilibriums
        n_mean_field_echoes = config.getInstance.dbm_n_mean_field_echoes
        n_gibbs_echoes = config.getInstance.dbm_n_gibbs_echoes
        n_use_echoes = config.getInstance.dbm_use_echoes
    end

    properties (SetAccess = immutable)
        lang
        n_visible_nodes
        n_levels
    end

    properties (SetAccess = public)
        activities
        biases
        weights
        n_training_iterations
    end

    methods
        function obj = dbm(lang)
            obj.n_levels = size(dbm.n_nodes_deep_levels, 2) + 1;
            obj.lang = lang;
            obj.n_visible_nodes = lang.n_input_nodes;
            obj.activities = cell(obj.n_levels, 1);
            obj.biases = cell(obj.n_levels, 1);
            obj.weights = cell(obj.n_levels-1, 1);
            obj.n_training_iterations = 0;
```

```
    obj.reset;

    disp("DBM parameters:")
    disp("Number of hidden nodes in level h1: " + dbm.n_nodes_deep_levels(1))
    disp("Number of hidden nodes in level h2: " + dbm.n_nodes_deep_levels(2))
    disp("Learning rate: " + dbm.learning_rate)
    disp("Initialise weights with random small numbers: " ...
        + config.getInstance.dbm_initial_non_zero_weights)
    disp("n_mean_field_echoes: " + dbm.n_mean_field_echoes)
    disp("n_gibbs_echoes: " + dbm.n_gibbs_echoes)
    disp("use_echoes: " + dbm.n_use_echoes)
end

function load_trained_dbm(obj, n_instances)
    % 1st layer activities and biases
    obj.activities{1} = zeros(1, obj.n_visible_nodes, 'double');
    obj.biases{1} = zeros(1, obj.n_visible_nodes, 'double');

    % 1st layer weights
    dbm_filename = fullfile(config.getInstance.path_dbms, obj.lang.name ...
        + "_dbm_layer1_" + n_instances + "." + config.getInstance.dbm_format);
    obj.weights{1} = readmatrix(dbm_filename);

    for i = 2:obj.n_levels
        % i-th layer activities and biases
        obj.activities{i} = zeros(1, dbm.n_nodes_deep_levels(i-1), 'double');
        obj.biases{i} = zeros(1, dbm.n_nodes_deep_levels(i-1), 'double');

        % i-th layer weights
        if i < obj.n_levels
            dbm_filename = fullfile(config.getInstance.path_dbms, obj.lang.name ...
                + "_dbm_layer" + i + "_" + n_instances + "." ...
                + config.getInstance.dbm_format);
            obj.weights{i} = readmatrix(dbm_filename);
        end
    end
```

```
    % update training iterations counter
    obj.n_training_iterations = n_instances;
end

function reset(obj)
    obj.activities{1} = zeros(1, obj.n_visible_nodes, 'double');
    obj.biases{1} = zeros(1, obj.n_visible_nodes, 'double');

    if config.getInstance.dbm_initial_non_zero_weights
        obj.weights{1} = rand(obj.n_visible_nodes, dbm.n_nodes_deep_levels(1)) / 1000;
    else
        obj.weights{1} = zeros(obj.n_visible_nodes, dbm.n_nodes_deep_levels(1), 'double');
    end

    for i = 2:obj.n_levels
        obj.activities{i} = zeros(1, dbm.n_nodes_deep_levels(i-1), 'double');
        obj.biases{i} = zeros(1, dbm.n_nodes_deep_levels(i-1), 'double');
        if i < obj.n_levels
            obj.weights{i} = zeros( ...
                dbm.n_nodes_deep_levels(i-1), dbm.n_nodes_deep_levels(i), 'double');
        end
    end

    obj.n_training_iterations = 0;
end

function learn(obj, train_batch_data)
    for batch = 1:size(train_batch_data, 3)
        for step = 1:size(train_batch_data, 1)
            % Populate input level and spread it up
            input_data = train_batch_data(step, :, batch);
            obj.spread_up_input(input_data, 0)

            % Hebbian learning phase
            obj.hebbian_learning(dbm.learning_rate)
```

```
            % Dreaming phase
            obj.resonate(1)

            % Anti-Hebbian learning phase
            obj.hebbian_learning(-dbm.learning_rate)
        end
    end

    % clean up activities
    obj.clean_up_activities

    % update training iterations
    obj.n_training_iterations = obj.n_training_iterations ...
        + size(train_batch_data, 3) * size(train_batch_data, 1);

    % save trained dbm to disk
    obj.save
    comment("TRAINED!!!!!!", "else")

    % clean big data from memory
    clear train_batch_data
end

function comprehend(obj, input_vector)
    if config.getInstance.dbm_compute_similarities
        comment("comprehending", 'header')
    end
    obj.use(input_vector, 1:obj.lang.n_input_nodes)
end

function produce(obj, input_vector)
    if config.getInstance.dbm_compute_similarities
        comment("producing", 'header')
    end
    obj.use(input_vector, 1:obj.lang.n_sound_nodes)
```

```
end

function similarity = measure_similarity_between_dbm_states(obj, a1, a2)
    similarity = zeros(1, 3);
    for level = 1:obj.n_levels
        if norm(a1{level}) ~= 0 && norm(a2{level}) ~= 0
            similarity(level) =  dbm.cosine_similarity(a1{level}, a2{level});
        end
    end
end

function table_similarity = measure_phonological_similarity(obj, input_utterances, unclamped)
    % input: is a vector for every utterance
    % output: create 3 tables with activities {2}, {3},
    % and a combination of them (after spreading)

    table_similarity{1} = zeros(size(input_utterances, 1), size(input_utterances, 1));
    table_similarity{2} = zeros(size(input_utterances, 1), size(input_utterances, 1));
    table_similarity{3} = zeros(size(input_utterances, 1), size(input_utterances, 1));

    second_levels_per_utterance = zeros(size(input_utterances, 1), ...
        obj.n_nodes_deep_levels(1));
    third_levels_per_utterance = zeros(size(input_utterances, 1), ...
        obj.n_nodes_deep_levels(2));
    both_levels_per_utterance = zeros(size(input_utterances, 1), ...
        obj.n_nodes_deep_levels(1) + obj.n_nodes_deep_levels(2));

    % spread and remember the hidden levels
    for i=1:size(input_utterances, 1)
        input_vector = input_utterances(i, :);
        obj.use(input_vector, unclamped{i,1})
        second_levels_per_utterance(i,:) = obj.activities{2};
        third_levels_per_utterance(i,:) = obj.activities{3};
        both_levels_per_utterance(i,:) = [obj.activities{2} obj.activities{3}];
    end
```

```
    % compare activities{2} vectors between each other for similarity
    for i=1:size(input_utterances, 1)
        for j=1:size(input_utterances, 1)
            v1 = second_levels_per_utterance(i,:);
            v2 = second_levels_per_utterance(j,:);
            table_similarity{1}(i, j) = dbm.cosine_similarity(v1, v2);

            v1 = third_levels_per_utterance(i,:);
            v2 = third_levels_per_utterance(j,:);
            table_similarity{2}(i, j) = dbm.cosine_similarity(v1, v2);

            v1 = both_levels_per_utterance(i,:);
            v2 = both_levels_per_utterance(j,:);
            table_similarity{3}(i, j) = dbm.cosine_similarity(v1, v2);
        end
    end
end

function io_similarity = measure_io_similarity(obj, ...
        input_utterances, full_utterances, unclamped)
    % create similarity table with tested output in rows and all
    % the inputs in columns
    io_similarity = zeros( ...
        size(input_utterances, 1), size(input_utterances, 1));
    for i=1:size(input_utterances, 1)
        input_vector = input_utterances(i, :);
        obj.use(input_vector, unclamped{i,1})
        for j=1:size(input_utterances, 1)
            io_similarity(i, j) = dbm.cosine_similarity( ...
                obj.activities{1}, full_utterances(j, :));
        end
    end
end

function draw_trained(obj)
    if obj.n_visible_nodes == lang1.n_input_nodes
```

```
            draw.dbm(obj.activities, obj.weights, draw.lang1_trained_dbm);
        elseif obj.n_visible_nodes == lang2.n_input_nodes
            draw.dbm(obj.activities, obj.weights, draw.lang2_trained_dbm);
        else
            error("Attempting to draw DBM for unknown language with " ...
                + obj.n_visible_nodes + " input nodes")
        end
    end

    function draw(obj)
        if obj.n_visible_nodes == lang1.n_input_nodes
            draw.dbm(obj.activities, obj.weights, draw.lang1_dbm);
        elseif obj.n_visible_nodes == lang2.n_input_nodes
            draw.dbm(obj.activities, obj.weights, draw.lang2_dbm);
        else
            error("Attempting to draw DBM for unknown language with " ...
                + obj.n_visible_nodes + " input nodes")
        end
    end
end


methods (Access = private)
    function spread_up_input(obj, input_data, stochastic)
        if config.getInstance.dbm_compute_similarities
            comment("learning spread up similarity", 'header')
        end

        % populate input level
        obj.activities{1} = input_data;

        % zero deep levels
        for level = 2:obj.n_levels
            obj.activities{level} = zeros(1, size(obj.activities{level}, 2), "double");
        end
```

```
    % spread up
    for e = 1:dbm.n_mean_field_echoes
        if config.getInstance.dbm_compute_similarities
            prev_activities = obj.activities;
        end

        % do the spreading
        for level = 2:obj.n_levels
            obj.spread_to_level(level, stochastic);
        end

        % measure similarity to know about the equilibrium state
        if config.getInstance.dbm_compute_similarities
            similarity = obj.measure_similarity_between_dbm_states( ...
                prev_activities, obj.activities);
            comment("learning spread up similarity, step " + e + ": " ...
                + sprintf('%f %f %f', similarity(:)), "else")
        end
    end
    if config.getInstance.dbm_compute_similarities
        disp(obj.activities)
    end
end

function hebbian_learning(obj, learning_rate)
    for level = 1:obj.n_levels
        obj.biases{level} = obj.biases{level} + ...
            learning_rate * obj.activities{level};
    end

    for level = 1:obj.n_levels-1
        obj.weights{level} = obj.weights{level} + ...
            learning_rate * obj.activities{level}.' * obj.activities{level+1};
    end
end
```

```
function resonate(obj, stochastic)
    if config.getInstance.dbm_compute_similarities
        comment("resonating", 'header')
    end
    for e = 1:dbm.n_gibbs_echoes
        if config.getInstance.dbm_compute_similarities
            prev_activities = obj.activities;
        end

        % do the spreading
        obj.spread_to_level1(1:obj.lang.n_input_nodes);
        obj.spread_to_level(obj.n_levels, stochastic);
        for level = 2:obj.n_levels-1
            obj.spread_to_level(level, stochastic);
        end

        % measure similarity to know about the equilibrium state
        if config.getInstance.dbm_compute_similarities
            similarity = obj.measure_similarity_between_dbm_states( ...
                prev_activities, obj.activities);
            comment("learning resonate similarity, step " + e + ": " ...
                + sprintf('%f %f %f', similarity(:)), "else")
        end
    end
end

function spread_to_level (obj, level, stochastic)
    if level == 1
        error("call spread_to_level1 to spread to level 1")
    elseif level == obj.n_levels
        obj.activities{level} = ...
            obj.activities{level-1} * obj.weights{level-1} + obj.biases{level};
    else
        obj.activities{level} = ...
            obj.activities{level+1} * obj.weights{level}.' + obj.biases{level} ...
            + obj.activities{level-1} * obj.weights{level-1};
```

```
    end

    obj.activities{level} = dbm.sigmoid(obj.activities{level});
    if stochastic
        obj.activities{level} = binornd(1, obj.activities{level});
    end
end

function spread_to_level1 (obj, unclamped)
    activities_1 = obj.activities{2} * obj.weights{1}.' + obj.biases{1};
    obj.activities{1}(unclamped) = activities_1(unclamped);
end

function clean_up_activities(obj)
    obj.activities{1} = zeros(1, obj.n_visible_nodes, 'double');
    for i = 2:obj.n_levels
        obj.activities{i} = zeros(1, dbm.n_nodes_deep_levels(i-1), 'double');
    end
end

function use(obj, input_vector, unclamped)
    % populate input level
    obj.activities{1} = input_vector;

    % zero deep levels
    for level = 2:obj.n_levels
        obj.activities{level} = zeros(1, size(obj.activities{level}, 2), "double");
    end

    % spread up
    for e = 1:dbm.n_use_echoes
        if config.getInstance.dbm_compute_similarities
            prev_activities = obj.activities;
        end

        % spread to hidden levels
```

```
                for level = 2:obj.n_levels
                    obj.spread_to_level(level, 0);
                end
                % spread to visible level
                obj.spread_to_level1(unclamped);

                % measure similarity to know about the equilibrium state
                if config.getInstance.dbm_compute_similarities
                    similarity = obj.measure_similarity_between_dbm_states( ...
                        prev_activities, obj.activities);
                    comment("using spread up similarity, step " + e + ": " ...
                        + sprintf('%f %f %f', similarity(:)), "else")
                end
            end
        end

        function save(obj)
            for i = 1:obj.n_levels-1
                filename = fullfile(config.getInstance.path_dbms, obj.lang.name ...
                    + "_dbm_layer" + i + "_" + obj.n_training_iterations ...
                    + "." + config.getInstance.dbm_format);
                writematrix(obj.weights{i}, filename)
            end
        end
    end

    methods (Static)
        function s = sigmoid(z)
            s = 1.0 ./ (1.0 + exp(-z));
        end

        function similarity = cosine_similarity (v1, v2)
            similarity = 1 - pdist2(v1, v2, 'cosine');
        end
    end
end
```

Appendix 3: Source code of class measure (MATLAB)

```
classdef measure

    properties(Constant)
        lex = [
            utterance("sova",  lang2_sound.SOVA,  lang2_meaning.SOVA)
            utterance("roza",  lang2_sound.ROZA,  lang2_meaning.ROZA)
            utterance("kot",   lang2_sound.KOT,   lang2_meaning.KOT)
            utterance("bot",   lang2_sound.BOT,   lang2_meaning.BOT)
            utterance("moj",   lang2_sound.MOJ,   lang2_meaning.MOJ)
            utterance("zhar",  lang2_sound.ZHAR,  lang2_meaning.ZHAR)
            utterance("zhaba", lang2_sound.ZHABA, lang2_meaning.ZHABA)
            utterance("taz",   lang2_sound.TAZ,   lang2_meaning.TAZ)
            utterance("bar",   lang2_sound.BAR,   lang2_meaning.BAR)
            utterance("rab",   lang2_sound.RAB,   lang2_meaning.RAB)
        ]
        gen = [
            utterance("roza",  lang2_sound.ROZA,  lang2_meaning.ROZA)
            utterance("kot",   lang2_sound.KOT,   lang2_meaning.KOT)
            utterance("boty",   lang2_sound.BOTY,   lang2_meaning.BOTY)
        ]
    end

    methods (Static)
        function similarities(lang, samples_per_utterance, n_learners)
            % 1) generates sample_per_utterance samples for language lang
            % 2) trains DBM on this set n_learners times
            % 3) after every training, uses it for every sound and meaning and measures similarity
            %    between
            %    - 1st hidden levels
            %    - 2nd hidden levels
            %    - both levels


            dataset = data_set(lang, samples_per_utterance);
```

```
% PROFICIENCY
io_comprehension_similarity = zeros(lang.n_utterances, lang.n_utterances, n_learners);
io_production_similarity_unclamped = zeros(lang.n_utterances, lang.n_utterances, n_learners);
io_production_similarity_clamped = zeros(lang.n_utterances, lang.n_utterances, n_learners);

% PHONOLOGY WORDS
similarity_sounds{1} = zeros(lang.n_utterances, lang.n_utterances, n_learners);
similarity_sounds{2} = zeros(lang.n_utterances, lang.n_utterances, n_learners);
similarity_sounds{3} = zeros(lang.n_utterances, lang.n_utterances, n_learners);

similarity_meanings_clamped{1} = zeros(lang.n_utterances, lang.n_utterances, n_learners);
similarity_meanings_clamped{2} = zeros(lang.n_utterances, lang.n_utterances, n_learners);
similarity_meanings_clamped{3} = zeros(lang.n_utterances, lang.n_utterances, n_learners);

similarity_meanings_unclamped{1} = zeros(lang.n_utterances, lang.n_utterances, n_learners);
similarity_meanings_unclamped{2} = zeros(lang.n_utterances, lang.n_utterances, n_learners);
similarity_meanings_unclamped{3} = zeros(lang.n_utterances, lang.n_utterances, n_learners);

similarity_faithfullness{1} = zeros(lang.n_utterances*2, lang.n_utterances*2, n_learners);
similarity_faithfullness{2} = zeros(lang.n_utterances*2, lang.n_utterances*2, n_learners);
similarity_faithfullness{3} = zeros(lang.n_utterances*2, lang.n_utterances*2, n_learners);

% PHONOLOGY SOUNDS-PHONEMES
similarity_first_vowels{1} = zeros(lang2_sound.n_vowels, lang2_sound.n_vowels, n_learners);
similarity_first_vowels{2} = zeros(lang2_sound.n_vowels, lang2_sound.n_vowels, n_learners);
similarity_first_vowels{3} = zeros(lang2_sound.n_vowels, lang2_sound.n_vowels, n_learners);

similarity_second_vowels{1} = zeros(lang2_sound.n_vowels, lang2_sound.n_vowels, n_learners);
similarity_second_vowels{2} = zeros(lang2_sound.n_vowels, lang2_sound.n_vowels, n_learners);
similarity_second_vowels{3} = zeros(lang2_sound.n_vowels, lang2_sound.n_vowels, n_learners);

similarity_three_sounds{1} = zeros(3 * size(measure.lex, 1), 3 * size(measure.lex, 1), ...
    n_learners);
similarity_three_sounds{2} = zeros(3 * size(measure.lex, 1), 3 * size(measure.lex, 1), ...
    n_learners);
```

```
similarity_three_sounds{3} = zeros(3 * size(measure.lex, 1), 3 * size(measure.lex, 1), ...
    n_learners);

similarity_lex = cell(size(measure.lex, 1), 3);
for j=1:size(measure.lex, 1)
    similarity_lex{j, 1} = zeros(4, 4, n_learners);
    similarity_lex{j, 2} = zeros(4, 4, n_learners);
    similarity_lex{j, 3} = zeros(4, 4, n_learners);
end

similarity_gen = cell(size(measure.gen, 1), 3);
for j=1:size(measure.gen, 1)
    similarity_gen{j, 1} = zeros(6, 6, n_learners);
    similarity_gen{j, 2} = zeros(6, 6, n_learners);
    similarity_gen{j, 3} = zeros(6, 6, n_learners);
end

for i=1:n_learners
    comment(lang.name + ": generating new data for learner " ...
        + i + " of " + n_learners, 'header');
    data = dataset.generate_observations;

    comment("Training DBM", 'else');
    dbm_instance = dbm(lang);
    dbm_instance.learn(data)

    full_vectors = measure.create_word_input(lang);
    sound_vectors = measure.create_sound_input(lang);
    meaning_vectors = measure.create_meaning_input(lang);
    unclamped_all = measure.unclamped(lang.n_input_nodes, lang.n_utterances);
    unclamped_sound = measure.unclamped(lang.n_sound_nodes, lang.n_utterances);

    % PROFICIENCY
    comment("Measure input-output comprehension similarity for learner " ...
        + i + " of " + n_learners, 'else')
    io_comprehension_similarity(:,:,i) = dbm_instance.measure_io_similarity( ...
```

```
    sound_vectors, full_vectors, unclamped_all);

comment("Measure input-output production similarity (meaning unclamped) for learner " ...
    + i + " of " + n_learners, 'else')
io_production_similarity_unclamped(:,:,i) = dbm_instance.measure_io_similarity( ...
    meaning_vectors, full_vectors, unclamped_all);

comment("Measure input-output production similarity (meaning clamped) for learner " ...
    + i + " of " + n_learners, 'else')
io_production_similarity_clamped(:,:,i) = dbm_instance.measure_io_similarity( ...
    meaning_vectors, full_vectors, unclamped_sound);

% PHONOLOGY
comment("Measure similarity comprehension for learner " ...
    + i + " of " + n_learners, 'else')
similarity = dbm_instance.measure_phonological_similarity(sound_vectors, unclamped_all);
similarity_sounds{1}(:,:,i) = similarity{1};
similarity_sounds{2}(:,:,i) = similarity{2};
similarity_sounds{3}(:,:,i) = similarity{3};

comment("Measure similarity production (meaning clamped) for learner " ...
    + i + " of " + n_learners, 'else')
similarity = dbm_instance.measure_phonological_similarity(meaning_vectors, ...
    unclamped_sound);
similarity_meanings_clamped{1}(:,:,i) = similarity{1};
similarity_meanings_clamped{2}(:,:,i) = similarity{2};
similarity_meanings_clamped{3}(:,:,i) = similarity{3};

comment("Measure similarity production (meaning UNclamped) for learner " ...
    + i + " of " + n_learners, 'else')
similarity = dbm_instance.measure_phonological_similarity(meaning_vectors, ...
    unclamped_all);
similarity_meanings_unclamped{1}(:,:,i) = similarity{1};
similarity_meanings_unclamped{2}(:,:,i) = similarity{2};
similarity_meanings_unclamped{3}(:,:,i) = similarity{3};
```

```
comment("Measure similarity comprehension-production (faithfullness) for learner " ...
    + i + " of " + n_learners, 'else')
similarity = dbm_instance.measure_phonological_similarity( ...
    [sound_vectors; meaning_vectors], ...
    [unclamped_all; unclamped_sound]);
similarity_faithfullness{1}(:,:,i) = similarity{1};
similarity_faithfullness{2}(:,:,i) = similarity{2};
similarity_faithfullness{3}(:,:,i) = similarity{3};

% PHONOLOGY SOUNDS-PHONEMES
if strcmp(lang.name, "lang2")
    comment("Measure similarity first vowel for learner " ...
        + i + " of " + n_learners, 'else')
    input_vectors = measure.create_first_vowel_input;
    similarity = dbm_instance.measure_phonological_similarity( ...
        input_vectors, measure.unclamped(lang.n_input_nodes, lang2_sound.n_vowels));
    similarity_first_vowels{1}(:,:,i) = similarity{1};
    similarity_first_vowels{2}(:,:,i) = similarity{2};
    similarity_first_vowels{3}(:,:,i) = similarity{3};

    comment("Measure similarity second vowel for learner " ...
        + i + " of " + n_learners, 'else')
    input_vectors = measure.create_second_vowel_input;
    similarity = dbm_instance.measure_phonological_similarity( ...
        input_vectors, measure.unclamped(lang.n_input_nodes, lang2_sound.n_vowels));
    similarity_second_vowels{1}(:,:,i) = similarity{1};
    similarity_second_vowels{2}(:,:,i) = similarity{2};
    similarity_second_vowels{3}(:,:,i) = similarity{3};

    comment("Measure similarity three first sounds for learner " ...
        + i + " of " + n_learners, 'else')
    input_vectors = measure.create_three_sounds_input;
    similarity = dbm_instance.measure_phonological_similarity( ...
        input_vectors, measure.unclamped(lang.n_input_nodes, 3 * size(measure.lex, 1)));
    similarity_three_sounds{1}(:,:,i) = similarity{1};
    similarity_three_sounds{2}(:,:,i) = similarity{2};
```

```
            similarity_three_sounds{3}(:,:,i) = similarity{3};

            comment("Measure lexical mapping for learner " ...
                + i + " of " + n_learners, 'else');
            for j=1:size(measure.lex, 1)
                input_vectors = measure.create_lex_input(j);
                similarity = dbm_instance.measure_phonological_similarity( ...
                    input_vectors, measure.unclamped_lex);
                similarity_lex{j, 1}(:,:,i) = similarity{1};
                similarity_lex{j, 2}(:,:,i) = similarity{2};
                similarity_lex{j, 3}(:,:,i) = similarity{3};
            end

            comment("Measure morphosyntactic mapping for learner " ...
                + i + " of " + n_learners, 'else');
            for j=1:size(measure.gen, 1)
                input_vectors = measure.create_gen_input(j);
                similarity = dbm_instance.measure_phonological_similarity( ...
                    input_vectors, measure.unclamped_gen);
                similarity_gen{j, 1}(:,:,i) = similarity{1};
                similarity_gen{j, 2}(:,:,i) = similarity{2};
                similarity_gen{j, 3}(:,:,i) = similarity{3};
            end
        end
    end

    % average over all learners
    % PROFICIENCY
    table_io_comprehension_similarity = measure.average_input_output( ...
        io_comprehension_similarity);
    table_io_production_similarity_clamped = measure.average_input_output( ...
        io_production_similarity_clamped);
    table_io_production_similarity_unclamped = measure.average_input_output( ...
        io_production_similarity_unclamped);

    % PHONOLOGY WORDS
```

```
table_similarity_sounds = measure.average(similarity_sounds);
table_similarity_meanings_clamped = measure.average(similarity_meanings_clamped);
table_similarity_meanings_unclamped = measure.average(similarity_meanings_unclamped);
table_similarity_faithfullness = measure.average(similarity_faithfullness);

% PHONOLOGY SOUNDS-PHONEMES
if strcmp(lang.name, "lang2")
    table_similarity_first_vowels = measure.average(similarity_first_vowels);
    table_similarity_second_vowels = measure.average(similarity_second_vowels);
    table_similarity_three_sounds = measure.average(similarity_three_sounds);

    table_similarity_lex = cell(size(measure.lex, 1), 1); %cell array of 10 cell arrays
    for j=1:size(measure.lex, 1)
        s{1} = similarity_lex{j, 1};
        s{2} = similarity_lex{j, 2};
        s{3} = similarity_lex{j, 3};
        table_similarity_lex{j} = measure.average(s); % cell array of 6 elements
    end
    table_similarity_gen = cell(size(measure.gen, 1), 1); %cell array of 3 cell arrays
    for j=1:size(measure.gen, 1)
        s{1} = similarity_gen{j, 1};
        s{2} = similarity_gen{j, 2};
        s{3} = similarity_gen{j, 3};
        table_similarity_gen{j} = measure.average(s); % cell array of 6 elements
    end
end

% create tables with column and row names
utterance_names_comp = strings(1, lang.n_utterances);
for i=1:lang.n_utterances
    utterance_names_comp(1, i) = char("[" + lang.utterances(i).name + "]");
end
utterance_names_comp = cellstr(utterance_names_comp);

utterance_names_prod = strings(1, lang.n_utterances);
for i=1:lang.n_utterances
```

```
    utterance_names_prod(1, i) = char("<" + lang.utterances(i).name+ ">");
end
utterance_names_prod = cellstr(utterance_names_prod);

% PROFICIENCY
table_io_comprehension_similarity = measure.create_tables_input_output( ...
    table_io_comprehension_similarity, utterance_names_comp);
table_io_production_similarity_clamped = measure.create_tables_input_output( ...
    table_io_production_similarity_clamped, utterance_names_prod);
table_io_production_similarity_unclamped = measure.create_tables_input_output( ...
    table_io_production_similarity_unclamped, utterance_names_prod);

% PHONOLOGY WORDS
table_similarity_sounds = measure.create_tables( ...
    table_similarity_sounds, utterance_names_comp);
table_similarity_meanings_clamped = measure.create_tables( ...
    table_similarity_meanings_clamped, utterance_names_prod);
table_similarity_meanings_unclamped = measure.create_tables( ...
    table_similarity_meanings_unclamped, utterance_names_prod);
table_similarity_faithfullness = measure.create_tables( ...
    table_similarity_faithfullness, [utterance_names_comp utterance_names_prod]);

% PHONOLOGY SOUNDS-PHONEMES
if strcmp(lang.name, "lang2")
    % first and second vowels
    vowel_names = strings(1, lang2_sound.n_vowels);
    for i=1:lang2_sound.n_vowels
        vowel_names(1, i) = char(lang2_sound.vowels(i));
    end
    vowel_names = cellstr(vowel_names);
    table_similarity_first_vowels = measure.create_tables( ...
        table_similarity_first_vowels, vowel_names);
    table_similarity_second_vowels = measure.create_tables( ...
        table_similarity_second_vowels, vowel_names);

    % three sounds
```

```
    three_sounds_names = strings(1, 3 * size(measure.lex, 1));
    j = 1;
    for i=1:size(measure.lex, 1)
        c1 = lang2_sound.cons(measure.lex(i).sound.c1_index);
        c2 = lang2_sound.cons(measure.lex(i).sound.c2_index);

        three_sounds_names(1, j) = char("[" + c1 + lang2_sound.vowels(1) + c2 + "]");
        j = j + 1;

        three_sounds_names(1, j) = char("[" + c1 + lang2_sound.vowels(2) + c2 + "]");
        j = j + 1;

        three_sounds_names(1, j) = char("[" + c1 + lang2_sound.vowels(3) + c2 + "]");
        j = j + 1;
    end
    three_sounds_names = cellstr(three_sounds_names);

    table_similarity_three_sounds = measure.create_tables( ...
        table_similarity_three_sounds, three_sounds_names);

    % lexical
    table_similarity_lex = measure.create_lex_table(table_similarity_lex);

    % morphosyntactic
    table_similarity_gen = measure.create_gen_table(table_similarity_gen);
end

% write similarity matrices to files
% PROFICIENCY
measure.write_to_file_input_output(table_io_comprehension_similarity, ...
    "io_comprehension", lang, samples_per_utterance, n_learners)
measure.write_to_file_input_output(table_io_production_similarity_clamped, ...
    "io_production_clamped", lang, samples_per_utterance, n_learners)
measure.write_to_file_input_output(table_io_production_similarity_unclamped, ...
    "io_production_unclamped", lang, samples_per_utterance, n_learners)
```

```
        % PHONOLOGY WORDS
        measure.write_to_file(table_similarity_sounds, ...
            "comprehension", lang, samples_per_utterance, n_learners)
        measure.write_to_file(table_similarity_meanings_clamped, ...
            "production_meaning_clamped", lang, samples_per_utterance, n_learners)
        measure.write_to_file(table_similarity_meanings_unclamped, ...
            "production_meaning_unclamped", lang, samples_per_utterance, n_learners)
        measure.write_to_file(table_similarity_faithfullness, ...
            "faithfullness", lang, samples_per_utterance, n_learners)

        % PHONOLOGY SOUNDS-PHONEMES
        if strcmp(lang.name, "lang2")
            measure.write_to_file(table_similarity_first_vowels, ...
                "first_vowels", lang, samples_per_utterance, n_learners)
            measure.write_to_file(table_similarity_second_vowels, ...
                "second_vowels", lang, samples_per_utterance, n_learners)
            measure.write_to_file(table_similarity_three_sounds, ...
                "three_sounds", lang, samples_per_utterance, n_learners)
            measure.write_to_file(table_similarity_lex, ...
                "lexical_mapping", lang, samples_per_utterance, n_learners)
            measure.write_to_file(table_similarity_gen, ...
                "gender_number_mapping", lang, samples_per_utterance, n_learners)
        end
    end
end


methods (Static, Access=private)
    function input_vectors = create_word_input(lang)
        input_vectors = zeros(lang.n_utterances, lang.n_input_nodes);
        for j=1:lang.n_utterances
            input_vectors(j, :) = [lang.utterances(j).sound.create_input_vector ...
                lang.utterances(j).meaning.create_input_vector];
        end
    end
```

```
function input_vectors = create_sound_input(lang)
    input_vectors = zeros(lang.n_utterances, lang.n_input_nodes);
    for j=1:lang.n_utterances
        input_vectors(j, :) = [lang.utterances(j).sound.create_input_vector ...
            zeros(1, lang.utterances(j).meaning.n_semantic_nodes)];
    end
end

function input_vectors = create_meaning_input(lang)
    input_vectors = zeros(lang.n_utterances, lang.n_input_nodes);
    for j=1:lang.n_utterances
        input_vectors(j, :) = [zeros(1, lang.utterances(j).sound.n_auditory_nodes) ...
                lang.utterances(j).meaning.create_input_vector];
    end
end

function input_vectors = create_first_vowel_input()
    input_vectors = zeros(lang2_sound.n_vowels, lang2.n_input_nodes);

    c1_vector = lang2_sound.create_zero_consonant_input_vector;
    c2_vector = lang2_sound.create_zero_consonant_input_vector;
    v2_vector = vowel.create_zero_input_vector;

    for j=1:lang2_sound.n_vowels
        v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(j), lang2_sound.f2_erbs(j));
        input_vectors(j, :) = [c1_vector v1_vector c2_vector v2_vector ...
            zeros(1, lang2_meaning.n_semantic_nodes)];
    end
end

function input_vectors = create_second_vowel_input()
    input_vectors = zeros(lang2_sound.n_vowels, lang2.n_input_nodes);

    c1_vector = lang2_sound.create_zero_consonant_input_vector;
    c2_vector = lang2_sound.create_zero_consonant_input_vector;
    v1_vector = vowel.create_zero_input_vector;
```

```
    for j=1:lang2_sound.n_vowels
        v2_vector = vowel.create_input_vector(lang2_sound.f1_erbs(j), lang2_sound.f2_erbs(j));
        input_vectors(j, :) = [c1_vector v1_vector c2_vector v2_vector ...
            zeros(1, lang2_meaning.n_semantic_nodes)];
    end
end

function input_vectors = create_three_sounds_input()
    n_comparisons = 3 * size(measure.lex, 1);
    input_vectors = zeros(n_comparisons, lang2.n_input_nodes);

    i = 1;
    for j=1:size(measure.lex, 1)
        c1_vector = lang2_sound.create_consonant_input_vector(measure.lex(j).sound.c1_index);
        c2_vector = lang2_sound.create_consonant_input_vector(measure.lex(j).sound.c2_index);
        v2_vector = vowel.create_zero_input_vector;

        % [a]
        v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(1), lang2_sound.f2_erbs(1));
        input_vectors(i, :) = [c1_vector v1_vector c2_vector v2_vector ...
            zeros(1, lang2_meaning.n_semantic_nodes)];
        i = i + 1;

        % [o]
        v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(2), lang2_sound.f2_erbs(2));
        input_vectors(i, :) = [c1_vector v1_vector c2_vector v2_vector ...
            zeros(1, lang2_meaning.n_semantic_nodes)];
        i = i + 1;

        % [ɐ]
        v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(3), lang2_sound.f2_erbs(3));
        input_vectors(i, :) = [c1_vector v1_vector c2_vector v2_vector ...
            zeros(1, lang2_meaning.n_semantic_nodes)];
        i = i + 1;
    end
```

```
end

function input_vectors = create_lex_input(i)
    input_vectors = zeros(4, lang2.n_input_nodes);

    % sounds
    c1_vector = lang2_sound.create_consonant_input_vector(measure.lex(i).sound.c1_index);
    c2_vector = lang2_sound.create_consonant_input_vector(measure.lex(i).sound.c2_index);
    v2_vector = vowel.create_zero_input_vector;
    meaning_vector = measure.lex(i).meaning.create_lex_input_vector;

    % input vector with lexical meaning
    input_vectors(1, :) = [ ...
        lang2_sound.create_zero_consonant_input_vector ...
        vowel.create_zero_input_vector ...
        lang2_sound.create_zero_consonant_input_vector ...
        vowel.create_zero_input_vector ...
        meaning_vector];

    % [a]
    v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(1), lang2_sound.f2_erbs(1));
    input_vectors(2, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];

    % [o]
    v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(2), lang2_sound.f2_erbs(2));
    input_vectors(3, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];

    % [ɐ]
    v1_vector = vowel.create_input_vector(lang2_sound.f1_erbs(3), lang2_sound.f2_erbs(3));
    input_vectors(4, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];
end

function input_vectors = create_gen_input(i)
```

```
    input_vectors = zeros(4, lang2.n_input_nodes);
    % sounds
    c1_vector = lang2_sound.create_zero_consonant_input_vector;
    v1_vector = vowel.create_zero_input_vector;
    c2_vector = lang2_sound.create_zero_consonant_input_vector;
    v2_vector = vowel.create_zero_input_vector;
    meaning_vector = measure.gen(i).meaning.create_gen_input_vector;

    % input vector with morphosyntactic meaning
    input_vectors(1, :) = [c1_vector v1_vector c2_vector v2_vector meaning_vector];

    % [a]
    v2_vector = vowel.create_input_vector(lang2_sound.f1_erbs(1), lang2_sound.f2_erbs(1));
    input_vectors(2, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];

    % [ɨ]
    v2_vector = vowel.create_input_vector(lang2_sound.f1_erbs(4), lang2_sound.f2_erbs(4));
    input_vectors(3, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];

    % [ə]
    v2_vector = vowel.create_input_vector(lang2_sound.f1_erbs(5), lang2_sound.f2_erbs(5));
    input_vectors(4, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];

    % [ɪ]
    v2_vector = vowel.create_input_vector(lang2_sound.f1_erbs(6), lang2_sound.f2_erbs(6));
    input_vectors(5, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];

    % [NONE]
    v2_vector = vowel.create_input_vector(0, 0);
    input_vectors(6, :) = [c1_vector v1_vector c2_vector v2_vector ...
        zeros(1, lang2_meaning.n_semantic_nodes)];
end
```

```
function table_similarity = average(similarity)
    m{1} = round(mean(similarity{1}, 3)*100);
    m{2} = round(mean(similarity{2}, 3)*100);
    m{3} = round(mean(similarity{3}, 3)*100);

    sd{1} = round(std(similarity{1}, 0, 3)*100);
    sd{2} = round(std(similarity{2}, 0, 3)*100);
    sd{3} = round(std(similarity{3}, 0, 3)*100);

    table_similarity{1} = string(m{1}) + " (" + string(sd{1}) + ")";
    table_similarity{2} = string(m{2}) + " (" + string(sd{2}) + ")";
    table_similarity{3} = string(m{3}) + " (" + string(sd{3}) + ")";

    table_similarity{4} = string(m{1});
    table_similarity{5} = string(m{2});
    table_similarity{6} = string(m{3});
end

function table_similarity = average_input_output(similarity)
    m = round(mean(similarity, 3)*100);
    sd = round(std(similarity, 0, 3)*100);
    table_similarity{1} = string(m) + " (" + string(sd) + ")";
    table_similarity{2} = string(m);
end

function result_table_similarity = create_tables(table_similarity, headers)
    result_table_similarity = cell(6, 1);
    for i=1:6
        result_table_similarity{i} = array2table(table_similarity{i});
        result_table_similarity{i}.Properties.RowNames = headers;
        result_table_similarity{i}.Properties.VariableNames = headers;
        result_table_similarity{i}.Properties.DimensionNames{1} = ' ';
    end
end
```

```
function result_table_similarity = create_tables_input_output(table_similarity, headers)
    result_table_similarity = cell(2, 1);
    for i=1:2
        result_table_similarity{i} = array2table(table_similarity{i});
        result_table_similarity{i}.Properties.RowNames = headers;
        result_table_similarity{i}.Properties.VariableNames = headers;
        result_table_similarity{i}.Properties.DimensionNames{1} = ' ';
    end
end

function result_table_similarity_lex = create_lex_table(table_similarity_lex)
    % table_similarity_lex: cell array of 10 cell arrays of 6 elements
    % for output, we want to return one cell array of 6 elements
    result_table_similarity_lex = cell(6, 1);
    for i=1:6
        lex_matrix = strings(size(measure.lex, 1), 3);
        row_names = strings(size(measure.lex, 1), 1);

        for j=1:size(measure.lex, 1)
            lex_matrix(j, :) = table_similarity_lex{j}{i}(1, 2:4);
            row_names(j) = ...
                "<" + measure.lex(j).meaning.lex(measure.lex(j).meaning.lex_index) + ">";
        end

        result_table_similarity_lex{i} = array2table(lex_matrix);
        result_table_similarity_lex{i}.Properties.RowNames = row_names;
        result_table_similarity_lex{i}.Properties.VariableNames = ["a", "o", "e"];
        result_table_similarity_lex{i}.Properties.DimensionNames{1} = ' ';
    end
end

function result_table_similarity_gen = create_gen_table(table_similarity_gen)
    % table_similarity_lex: cell array of 10 cell arrays of 6 elements
    % for output, we want to return one cell array of 6 elements
    result_table_similarity_gen = cell(6, 1);
    for i=1:6
```

```
        gen_matrix = strings(size(measure.gen, 1), 5);
        row_names = strings(size(measure.gen, 1), 1);

        for j=1:size(measure.gen, 1)
            gen_matrix(j, :) = table_similarity_gen{j}{i}(1, 2:6);
            row_names(j) = measure.gen(j).meaning.gen(measure.gen(j).meaning.gen_index);
        end
        result_table_similarity_gen{i} = array2table(gen_matrix);
        result_table_similarity_gen{i}.Properties.RowNames = row_names;
        result_table_similarity_gen{i}.Properties.VariableNames = ["a", "ɨ", "ə", "ɪ̈" "NONE"];
        result_table_similarity_gen{i}.Properties.DimensionNames{1} = ' ';
    end
end

function write_to_file(table_similarity, name, lang, n_samples, n_learners)
    f_common = fullfile(config.getInstance.path_measures, lang.name ...
            + "_" + name + "_similarities_" ...
            + n_samples + "samples_" ...
            + n_learners + "learners_h");
    filename{1} = strcat(f_common, "_with_std_1.csv");
    filename{2} = strcat(f_common, "_with_std_2.csv");
    filename{3} = strcat(f_common, "_with_std_12.csv");
    filename{4} = strcat(f_common, "_without_std_1.csv");
    filename{5} = strcat(f_common, "_without_std_2.csv");
    filename{6} = strcat(f_common, "_without_std_12.csv");

    writetable(table_similarity{1}, filename{1}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
    writetable(table_similarity{2}, filename{2}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
    writetable(table_similarity{3}, filename{3}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
    writetable(table_similarity{4}, filename{4}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
    writetable(table_similarity{5}, filename{5}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
    writetable(table_similarity{6}, filename{6}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
end

function write_to_file_input_output(table_similarity, name, lang, n_samples, n_learners)
    f_common = fullfile(config.getInstance.path_measures, lang.name ...
```

```
            + "_" + name + "_similarities_" ...
            + n_samples + "samples_" ...
            + n_learners + "learners_h");
    filename{1} = strcat(f_common, "_with_std.csv");
    filename{2} = strcat(f_common, "_without_std.csv");

    writetable(table_similarity{1}, filename{1}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
    writetable(table_similarity{2}, filename{2}, 'WriteRowNames', true, 'Encoding', 'UTF-8')
end

function unclamped_indeces = unclamped(n_nodes, n_comparisons)
    unclamped_indeces = repmat(1:n_nodes, n_comparisons, 1);
    unclamped_indeces = mat2cell( ...
        unclamped_indeces, ones(1, size(unclamped_indeces, 1)), size(unclamped_indeces, 2));
end

function unclamped_indeces = unclamped_lex()
    % only for lang2
    unclamped_indeces = {
        1:lang2.n_sound_nodes;
        1:lang2.n_input_nodes;
        1:lang2.n_input_nodes;
        1:lang2.n_input_nodes;
    };
end

function unclamped_indeces = unclamped_gen()
    % only for lang2
    unclamped_indeces = {
        1:lang2.n_sound_nodes;
        1:lang2.n_input_nodes;
        1:lang2.n_input_nodes;
        1:lang2.n_input_nodes;
        1:lang2.n_input_nodes;
        1:lang2.n_input_nodes;
    };
```

```
        end
    end
end
```