University of Amsterdam

# The emergence of consonant categories in a virtual brain

Modeling the learning of voiceless plosives using a deep Restricted Boltzmann Machine

BA Thesis Linguistics
Supervisor: Paul Boersma
21-6-2021

Sandra Meijer
11289267

# Abstract

In the past years, neural networks have been used for many applications, one of which is getting insight into how human beings learn and process language. In this thesis, a deep Restricted Boltzmann machine is created and an existing framework is adapted in order to process consonants and replicate a perception experiment with simplified sound inputs. In two language conditions, the model learns to comprehend and produce nine syllables consisting of a consonant and a vowel. The model is tested with different sound inputs to determine the locations of the categories and boundaries of the different consonants. Additionally, the perceptual magnet effect is observed and one of the hidden layers of the model is analyzed to investigate the existence of phonological categories after learning. The results suggest that phonological categories emerged for vowels and consonants, but further research is needed to support these findings.

# Table of Contents

# 1. Introduction

Children's ability to learn language has been a topic in linguistic research for years. An important finding is the ability of infants up to 10 months to discriminate between non-native speech sounds, whereas older children have more difficulty discriminating these sounds (Werker & Tees, 1984). This finding suggests a categorization of speech sounds based on the phoneme inventory of the child's native language, supposedly through distributional learning (Maye, Werker & Gerken, 2002). This distributional learning can be modeled using artificial neural networks (Boersma, 2019; Boersma, Benders & Seinhorst, 2020; Boersma, Chládková & Benders, 2021).

The model proposed by Boersma (2019) is bidirectional in the sense that it can model both production and comprehension. Sound and meaning are represented at the same level, but information flows in two directions, making both processing from meaning to sound (top-down) and from sound to meaning (bottom-up) possible. The neural network used in the model is a deep Restricted Boltzmann machine, in which both the input and the output of the model can be represented on the same level. This property makes the model especially suitable for the modeling of bidirectional processes. In Figure 1, both sound and meaning are represented at the bottom layer of the model. When modeling production, the meaning part of the visible layer is the input, and the sound part is the output. For comprehension, it is the other way around.

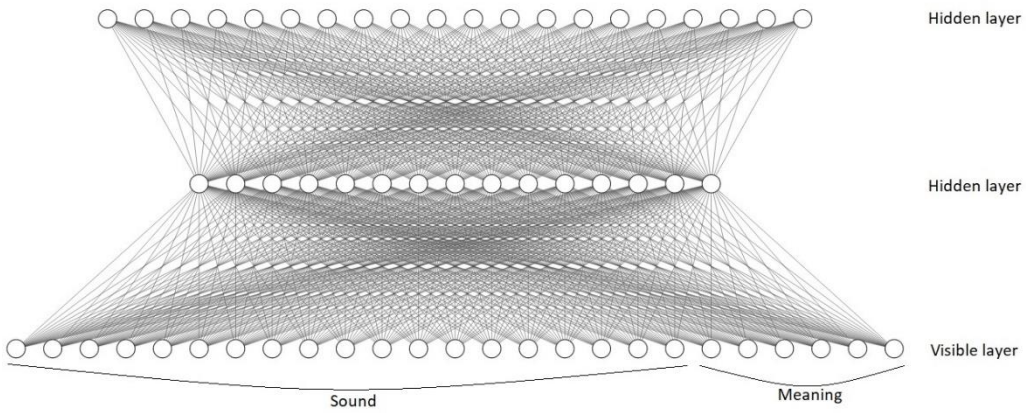


***Figure 1 –*** *The structure of a deep Restricted Boltzmann machine for bidirectional processing*

In both the original model proposed by Boersma (2019) and the adaptation by Beemdelust (2020), a deep Restricted Boltzmann machine is used to show the perceptual magnet effect on vowels. However, the perceptual magnet effect has not only been observed for vowels (Kuhl, 1991) but also consonants (Davis & Kuhl, 1994; Iverson & Kuhl, 1996). The simplified artificial language that is used only contains vowels or combinations of vowels. Yet, there are no known languages without consonants, and in order to model language learning as accurately as possible, the model should be able to process consonants too. Therefore, this thesis aims to expand the phoneme inventory of the

proposed artificial language to contain consonants and to adapt the model to process consonant-vowel pairs.

The addition of consonants to the model not only increases the validity of the model but also creates an opportunity to reproduce perception experiments on consonants. An example would be the experiment by Liberman, Delattre, and Cooper (1952), in which they investigated the speech perception of synthetic speech. In their experiment, participants had to listen to synthetic consonant–vowels pairs and indicate which voiceless stop they heard; *p, t,* or *k*. The stimuli were created using a machine called *pattern playback*. This device transforms drawings of spectrograms into synthetic speech. Because of the technique used to create the synthetic speech stimuli, the original spectrograms had to be simplified and only the most important cues were included. For the vowel, the first and second formant were included and the cues for the different plosives were bursts at different frequencies.

Low-frequency bursts are generally associated with *p*, high-frequency bursts with *t* and the domain of middle frequencies is mainly perceived as k, but the perception of the consonant depends on the following vowel. The results of the study were quite complex, which Kiefte and Kluender (2005) found was the result of harmonic distortion. They were able to show that the claim that the perception of stops is context-dependent remains valid. Nevertheless, research by Liberman and colleagues shows that the main auditory cues for the place of articulation are the second formant transitions.

To distinguish voiced and voiceless stops, voicing is the most important cue. If it is necessary to distinguish plosives from nasal consonants, steady-state resonance is the main cue (Liberman, Delattre, Gerstman & Cooper, 1956). In this research, voicing and steady-state resonance are left out of the input because the place of articulation is the only varying property of the consonants used in the artificial language.

For both diphthongs and consonant-vowel pairs, the aspect of time has to be implemented into the model. Although it has been successfully implemented in several types of neural networks, only Beemdelust (2020) and Yang (2020) have tried to implement time in a Restricted Boltzmann machine. Beemdelust added a second representation of the basilar membrane to the input layer to process two vowels at the same time. Yang modeled tone by using three representations of the fundamental frequency at different time points. This network structure, with several representations of the basilar membrane, will serve as a basis for the implementation of consonants, although changes will have to be made to the shape of the input. In contrast to vowels, consonants do not have a steady state with separate formants. Yet, plosives have a distinctive burst followed by a formant transition to the vowel. The burst can be modeled as a concentration of noise spread over several nodes on one basilar membrane, while the vowel is represented as two formants on the other basilar membrane.

This research aims to model a language with CV syllables. The phoneme inventory comprises three voiceless consonants and three vowels. The choice of voiceless consonants is mainly because of the lack of voicing. In Section 2, the network structure of the restricted Boltzmann machine and its application in language modeling are explained. Section 3 describes the creation of vowels and consonants as input for the model. In Section 4, the network is tested in two language environments. The comparison between those two environments is described in Section 5. The final two sections, Sections 6 and 7, contain the discussion and conclusion of the research respectively.

# 2. Network Structure

## 2.1 The Boltzmann Machine

A Boltzmann Machine is a type of stochastic artificial neural network that can learn a probability distribution of its input. The network consists of a visible layer and a hidden layer with nodes that are both connected to the nodes in the same layer and nodes in the other layer. If any more layers are added, the network is called a deep Boltzmann Machine. In the current research, a network with two hidden layers is used. Moreover, the connections between the nodes at the same level are removed, creating a so-called deep Restricted Boltzmann Machine. The nodes correspond to the neurons that fire in the brain, which also have weighted connections to other neurons. The weights of the connections can be seen as the strength of the connections, and the biases can be seen as the threshold values for which the second neuron is activated.

The model consists of a visible layer and two hidden layers. Both the sound and its meaning are represented on the visible layer. The sound input is composed of activations on two separate basilar membranes. The membranes are formed by a series of 30 nodes that represent the continuum of the membrane, ranging from 5 to 30 ERB. ERB stands for Equivalent Rectangular Bandwidth and is a measure used for human hearing. In Figure 2, the rightmost part of the visual layer represents the meaning of the sound. In this research, the words are the IPA transcriptions of the consonant and the vowel displayed on the basilar membranes.

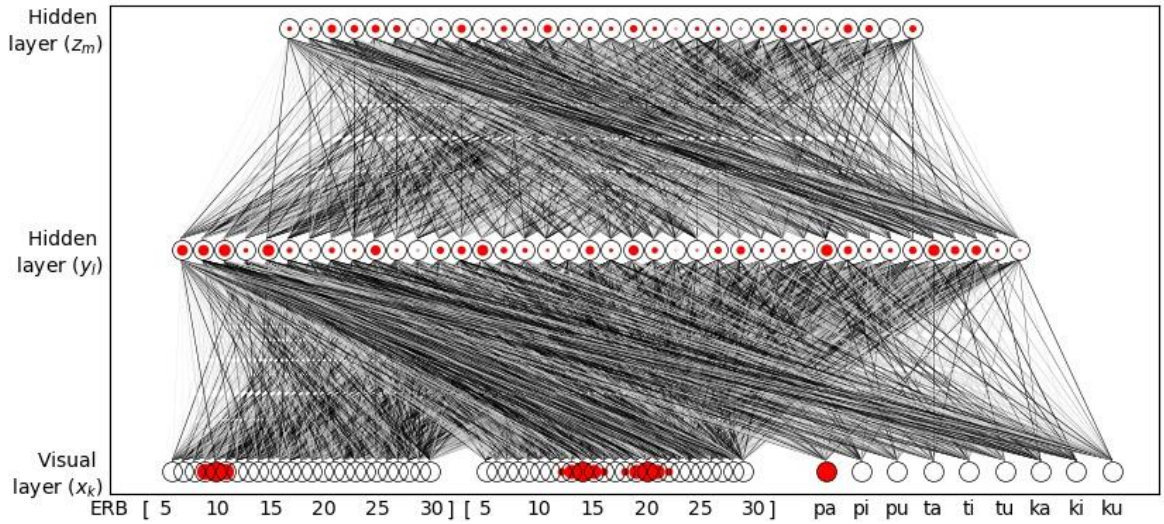


***Figure 2*** *– Network structure for processing CV clusters*

The two hidden layers have the function of information processing. The number of nodes in these layers is set to the arbitrary number of 40 for the middle layer and 30 for the top layer.

## 2.2 Training the network

Although the most common learning algorithm for Restricted Boltzmann Machines is contrastive divergence, the current model uses a Hebbian learning algorithm. The learning algorithm is based on the idea that the connection between two neurons becomes stronger the more the two are activated at the same time. The idea that neurons that fire together, wire together is called Hebb's rule, hence the name Hebbian learning.

To fully train the network, the network has to work through four phases. First, the network has to settle the input. The actual learning starts in the Hebbian learning phase. Third, the network input is no longer given and the network is able to create its own inputs. The last phase is the anti-Hebbian learning phase, in which the network unlearns part of the connections it has made. Since the model is based on Boersma (2019) and later Beemdelust (2020), the mathematical operations presented in the following sections are the same.

### 2.2.1 Initial settling phase

The initial settling phase takes place before the network starts learning. In this phase, the activities of the hidden nodes are initialized. The network is given an input of both a sound and its meaning at the visible layer and the activities of the input nodes are "clamped" during the entire phase. This means the activities of the input nodes are kept constant at the value the network was provided with. The activities of the nodes in the other two layers are initially set to zero but the activations of the layers will change based on the activity of the first layer. The activity of a node in the middle layer ($y_l$) is calculated by taking the sum of all weights of its connections multiplied by the corresponding

activations, both for the bottom ($x_k$) and the top layer ($z_m$). On top of that, a sigmoid function is used to enable the network to learn more complicated relationships between the nodes and eventually reproduce the input. The sigmoid function used in (1) is defined in (2).

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

(2) $\sigma(x) := 1/(1 + \mathrm{e}^{-x})$

The activities at the top level ($z_m$) can be computed in a similar way to the middle layer, yet, the activations are only influenced by the activations of and the connections to the second layer ($y_l$), resulting in formula (3).

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

While the input at the bottom layer ($x_k$) remains clamped, (1) through (3) are repeated 10 times, bringing the network to a steady state before moving on to the Hebbian Learning phase.

### 2.2.2 Hebbian learning phase

The model starts learning during the Hebbian learning phase. In this phase, the weights of the connections and the biases change based on the activation of the nodes in the network with a learning rate $\eta$ of 0.001. Formulas (4) to (6) describe the update of the biases for every layer, increasing the activity of a node in the future. Formulas (7) and (8) are the changes of the connections between two layers, strengthening connections between nodes that are active.

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

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

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

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

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

### 2.2.3 Dreaming phase

During the dreaming phase, the bottom layer is no longer clamped and the activation of the nodes can also be influenced by the middle layer. This way, the network can create its own patterns, similar to dreaming. The activation of the bottom layer ($x_k$) changes as described in (9).

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

After the activation of the bottom layer has changed, the top and middle layers change too. Because of backpropagation, the values for the top layer will change before the middle layer. Formulas (10) and (11) are the Bernoulli deviates of (3) and (1) respectively, forcing the activation of a node to be either 0 or 1. Thus, a node is either fully activated, or is not activated at all. The formulas also have

a randomization factor. For instance, when the value of the sigmoid function inside the Bernoulli deviate is 0.2, the output has a 0.2 probability of being 1.

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

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

Just as in the initial settling phase, sequences (9) to (11) are repeated 10 times to reach a near-equilibrium of the network.

### 2.2.4 Anti-Hebbian learning phase

The last phase, the anti-Hebbian learning phase, is similar to the Hebbian learning phase, yet its purpose is to weaken the connections between active nodes to prevent them from growing too large. The network is unlearning some of the connections it learned during the dreaming phase. (12) to (16) are the same as (4) to (8), yet weakening the connections instead of strengthening them.

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

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

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

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

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

The four steps together form one learning step and thousands of those steps are needed for the network to be thoroughly trained and represent the human brain.

## 3. Model Input: vowels and consonants

The input layer of the model contains two representations of the basilar membrane, both represented by 30 nodes. The leftmost node of each slab corresponds to the lowest basilar frequency and the rightmost one to the highest basilar frequency, dividing the continuum from 5 to 30 ERB in 29 equal steps. Therefore, the distance between each node is ≈ 0.86 ERB.

As mentioned before, a vowel is represented by its first and second formant, leaving out all other auditory clues to keep the model as simple as possible. The mean frequencies for the formants are given in Table 1. The formant values for the input ($\mu_{F1}$ and $\mu_{F2}$) are sampled from a Gaussian distribution with a mean from Table 1 and a standard deviation of $\sigma$ = 1.0 ERB. Subsequently, the activation of the nodes is determined using the formula described in (17), where $w$ = 1.5 ERB is the half-width of the Gaussian peak on the basilar membrane. Here, $ERB_k$ denotes the ERB value that is represented at node $k$.

(17) $x_k = 10\left(e^{-\frac{1}{2}\left(\frac{ERB_k - \mu_{F1}}{w}\right)^2} + e^{-\frac{1}{2}\left(\frac{ERB_k - \mu_{F2}}{w}\right)^2}\right) - 0.5$

***Table 1*** *– Mean frequency of the vowels in ERB units*

| **Vowels** | **i** | **u** | **a** |
|---|---|---|---|
| Mean frequency of F1 | 7 | 7 | 13 |
| Mean frequency of F2 | 25 | 13 | 19 |

The implementation of consonants is less straightforward since there are many auditory cues to recognize consonants in speech, for example by the voiced onset time (VOT) or the stop-burst pattern of plosives. Since the burst frequencies of plosives are different when the places of articulation vary, they can easily be implemented in the proposed model without the addition of nodes for specific cues such as silence or VOT. For a language in which all consonants are plosives, a plosive can be modeled as a concentration of noise spread over several nodes, with a different mean for each of the plosives.

The mean frequencies used in the language environment with a fixed mean for each plosive are based on the findings of Halle, Hughes, and Radley (1957) for American English. They found the frequencies of the plosive bursts are between 11 and 18 ERB for [p], and between 18 and 27 ERB for [k]. For [t], the burst has a high frequency, above 27 ERB, which is accompanied by a concentration of energy around 11 ERB. The lower frequency burst found for [t] will not be part of the input of the model because they were not included in the experiments we are trying to reproduce. In the experiment by Hall et al. (1957, participants listened to plosives in isolation, without transitions to a vowel. Therefore, the frequencies for the bursts can be used as a basis for the condition with a fixed burst mean.

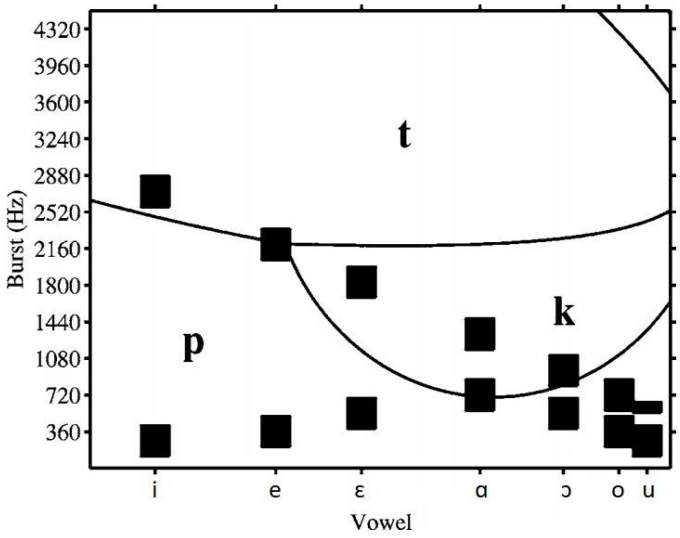


***Figure 3*** *– Estimated model response regions for /p/, /t/ and /k/ based on a quadratic logistic model (Kiefte & Kluender, 2005).*

The burst means that depend on the following vowel are based on the experiment by Liberman, Delattre, and Cooper (1952) and the additional research by Kiefte and Kluender (2005). Figure 3 shows the estimated regions for the perception of each plosive, depending on the burst frequency (in Hz) and the subsequent vowel.

From this figure, we can deduce that /ki/ and /ke/ are syllables that are not perceived. Since /e/ is not included in the artificial language, /ke/ does not pose a problem, while /ki/ does. To reproduce the results from Liberman, Delattre and Cooper (1952) as closely as possible, the best adaptation to the model is to leave out the syllable completely, creating a gap in the language. Yet, we can also choose to place /k/ at the boundary between /p/ and /t/, assuming there is a small region of /k/ in between the other two plosives.

For the other kV combinations, the mean burst frequency is determined by taking the middle of the plosive category at the location of each of the vowels. For /t/, the top of the graph in Figure 3 is not the upper boundary of the plosive category, but the mean bursts will be below 30 ERB to ensure the burst is visible on the basilar membrane representations of the neural network. Both the fixed mean and the varying mean frequencies per plosive can be found in Table 2.

***Table 2*** *– Mean frequencies for the consonants in ERB units*

| **Consonants** | **p** | **t** | **k** |
|---|---|---|---|
| Mean frequency of burst | 14 | 29 | 23 |
| Mean frequency before /a/ | 9 | 27 | 19 |
| Mean frequency before /i/ | 18 | 29 | 23 |
| Mean frequency before /u/ | 14 | 25 | 21 |

The formula for the activation of the nodes that represent the consonant part of the input is given in (18). Here, the half-width of the Gaussian peak is 1.0 ERB, creating a smaller range of activated nodes. Yet, the activation of the nodes is higher than for vowels, indicating the higher concentration of noise in plosives. The mean consonant values are sampled with a standard deviation of $\sigma$ = 1.5 ERB.

(18) $x_k = 12\, e^{-\frac{1}{2}\left(\frac{ERB_k - \mu_{burst}}{w}\right)^2} - 0.5$

Whether the consonant is represented on the first or second auditory slab depends on the type of language that is simulated. In a language with only CV syllables, the first slab always contains the consonant and the second slab the vowel, while it is the other way around for VC languages. In a language that contains both CV and VC syllables, the order depends on the type of syllable that is created. The ranges of both slabs of the basilar membrane are the same in order to make analysis

easier and to ensure switching around the order of the consonant and the vowel can be done without any complications.

# 4. Simulate language learning

During the learning phase, the four learning steps described in paragraph 2.2 are completed thousands of times. For every input, the activations in the hidden layers are settled while the visual layer remains clamped, a Hebbian learning step is taken, the visual layer is 'unclamped' and the network is left dreaming, creating its own activation pattern. In the last step, part of the newly learned connections are unlearned. After the iteration of these learning steps, the ability of the network to comprehend and produce sounds is tested.

To test the comprehension of the network, it is given an input that only comprises activations on the two basilar membrane parts of the network. The meaning nodes are not activated. The activations from the bottom layer ($x_k$) spread up to the middle and top layers ($y_l$ and $z_m$), resonate between the layers and spread back to the bottom layer. For this last step, the network will also show activation at the meaning nodes and if it has learned to classify the sound correctly, the meaning node corresponding to the sound is activated. To test the production of the network, only one of the input nodes is active: a meaning node. After the activation has spread to the top layer and back, the network shows both the meaning and the sound it has created.

## 4.1 Syllables with a constant mean burst

The least complicated language environment to create is one in which there are three voiceless plosives and three vowels, with only one possible syllable structure. In this language, the only possible structure is CV. The mean bursts of the plosives can be found in Table 2, where only the value 'mean ERB of burst' is used, so the vowel that follows does not influence the burst frequency. This is not consistent with the findings that the burst frequency plays a role in the perception of a plosive. A stop with a certain burst frequency can be perceived as different plosives, depending on the following vowel (Liberman, Delattre & Cooper, 1952).

The use of a fixed mean for each plosive creates the possibility to show the categorization of the three distinct consonants without the potential complications of a more spread distribution of the plosives. By keeping the mean burst frequency the same for all combinations with one plosive, the categorization of the plosives will be easier to achieve.

After the network has been trained with 10,000 pieces of data, each plosive and vowel have been activated approximately 3,333 times in total. Each meaning has been activated around two-thirds less because there are three combinations for each plosive, and three for each vowel, resulting in around 1,111 activations per meaning node.

### 4.1.1 The comprehension of the model

The testing phase of the network starts with the testing of the comprehension. The network is given a sound input without the activation of one of the meaning nodes. After resonance, the output layer shows both the echo of the input sound and the meaning the network thinks the input corresponded to. Figure 4 shows the state of the neural network after it is given a sound input and the activations have spread up and down through the network. In this case, a random sound was created, meant to correspond to the sound pattern of /pa/. From Figure 4, it follows that the network has correctly identified the sound.

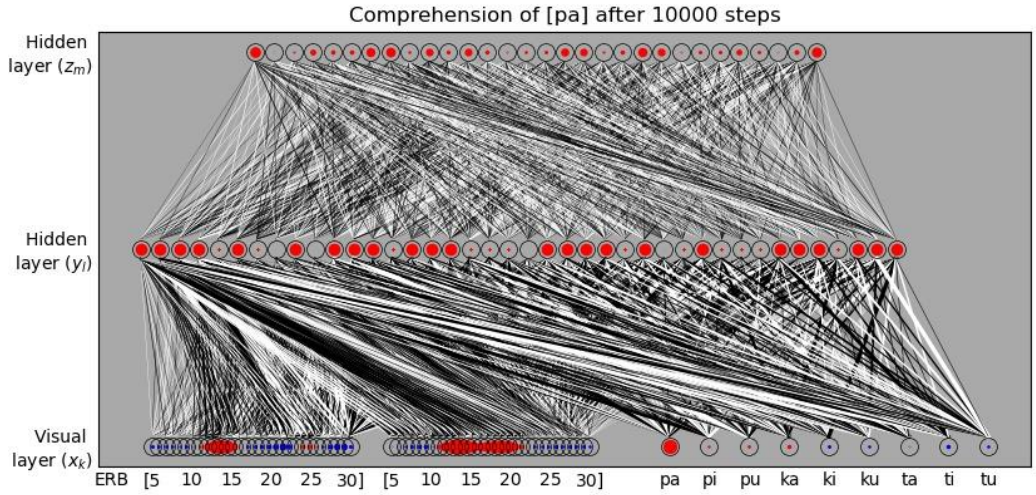


***Figure 4*** *– Neural network after 10,000 steps in a test from sound to meaning for /pa/*

It is clearly visible that /pa/ is given the highest activation, indicating that the model has correctly identified the input. Since the activations flow up and down in the network 10 times, the sound displayed on the visual layer does not correspond to the input sound; it is an echo of the original input. The model also activated all the other meanings that begin with /p/, and the ones that end in /a/. This indicates that the model might recognize the different parts of the syllable, although the activations of the other meaning nodes are significantly less than the activation of /pa/.

### 4.1.2 The production of the model

The comprehension of the network is determined by the activations of the nine meaning nodes. The production of the model is more complicated to analyze, because the performance is determined by the activations of all 60 auditory nodes, instead of only 9 meaning nodes. Not only is the analysis more complicated, but it is also harder for the network to learn to produce sounds for the same reason. The state of the network during testing from meaning to sound is given in Figure 5, where the network is given the input /ku/. After the activation has spread through the network, the location of

the burst lies around 23 ERB, and the vowel formants around 9 and 13 ERB, exactly as predicted for /k/ and /u/.

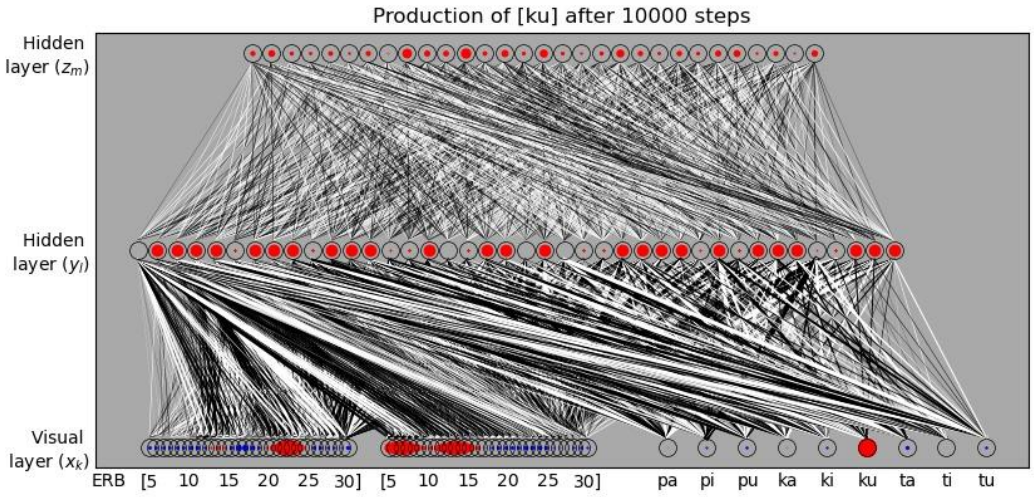


***Figure 5*** *– Neural network after 10000 steps in a test from meaning to sound for /ku/*

### 4.1.3 Observing the Perceptual Magnet Effect

A trained network is not only able to comprehend and produce sounds, but it can also demonstrate the perceptual magnet effect. The sound inputs for each syllable are located around certain mean frequencies so that the nodes at the mean or directly next to it will be most often activated when that sound is produced and given to the network. Because of this, the network creates sounds that are close to the mean bursts of the plosives when they are given only a meaning as input.

As the name of the effect suggests, the perceptual magnet effect occurs in perception, or in this case, the comprehension of the network. With the current network structure, we can see what the network thinks the incoming sound means. Moreover, we can see what auditory input (sound) the network thinks it has heard. In testing the comprehension of the network, the focus lies on the active meaning nodes, while the perceptual magnet effect can be observed on the two auditory slabs. The auditory slabs in Figure 4 show the burst and formants at the exact locations of their means, while the auditory input is chosen at random with a standard deviation of 1.5 ERB for the burst and 1.0 ERB for the vowel formants. However, the network thinks the burst and formants were located at their mean frequencies.

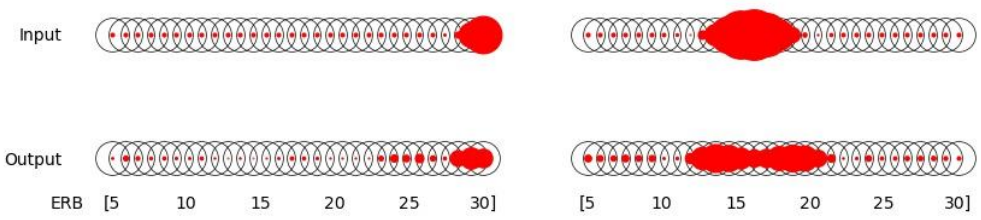


***Figure 6*** *– Sound input and output for /ta/ after 10,000 steps*

In Figure 6, the sound input and the echo of the model are shown for /ta/. The burst and both formants have frequencies a few ERB away from the mean, but after resonance in the network, the echo of the sound shows activations at the mean position and not at the input positions. From Table 2, it follows that the average burst mean for /t/ lies at 29 ERB, and the formants of /a/ are located around 13 and 19 ERB. The input consonant in Figure 6 has a burst of 30 ERB, while the burst of the echo lies around 29 ERB. Also, the vowel formants, both located around 16 ERB in the input, shifted towards the mean formant values for /a/, creating two distinct formants again. So, when the network is tested on a sound input slightly away from the mean, the network not only outputs the correct meaning, but the echo shows the vowel formants and the consonant burst around the predetermined means, perceiving the sound as located at the mean frequency for its category.

### 4.1.4 Category boundaries

The concept of categorical perception is similar to the perceptual magnet effect, although they are not entirely the same (Iverson & Kuhl, 2000). Both phenomena have categorical aspects, and Lacerda (1995) even proposed that the perceptual magnet effect emerges as a side-effect of a classification problem. Since the network is trained to categorize speech, it comes as no surprise that both phenomena can be observed in the current model.

The perceptual magnet effect is visible on the auditory slabs, whereas categorical perception can be observed on the meaning slab, by comparing the activations of the nodes for sound inputs with varying frequencies. The activation of a meaning node is not only high when the mean frequency is put into the network, but also when the frequency of the input is slightly away from the mean. The transition from one consonant to the other is rather abrupt. Where for one active node, the network perceives /pa/, for the following active node, the meaning node for /ka/ is much more active. The boundaries between two consonants are sharp, leaving little room for ambiguous sounds.

In Figures 7 to 9, the activations of the meaning nodes are given for several different burst frequencies. The network is tested with 150 different inputs, with 50 different bursts before three vowels. In testing, the network is given 50 input bursts with increasing frequencies, located on the interval from 5 to 30 ERB, in steps of 0.5 ERB. The following vowel has constant formant frequencies, to ensure that the difference in activation of the meaning nodes can be attributed solely to the frequency of the burst. The activations in the figures are an average of 50 listeners. Because the burst frequency is not influenced by the following vowel, the graphs should show similar categories.

***Table 3*** *– Mean frequencies of the bursts for the three consonants*

| **Consonants** | **p** | **t** | **k** |
|---|---|---|---|
| Mean frequency of burst | 14 | 29 | 23 |

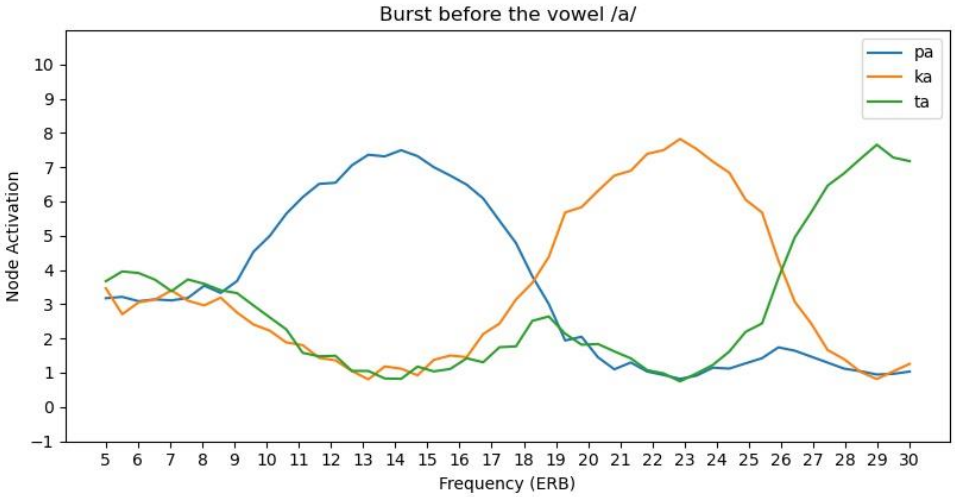


***Figure 7*** *– Average node activation for /pa/, /ka/, and /ta/*

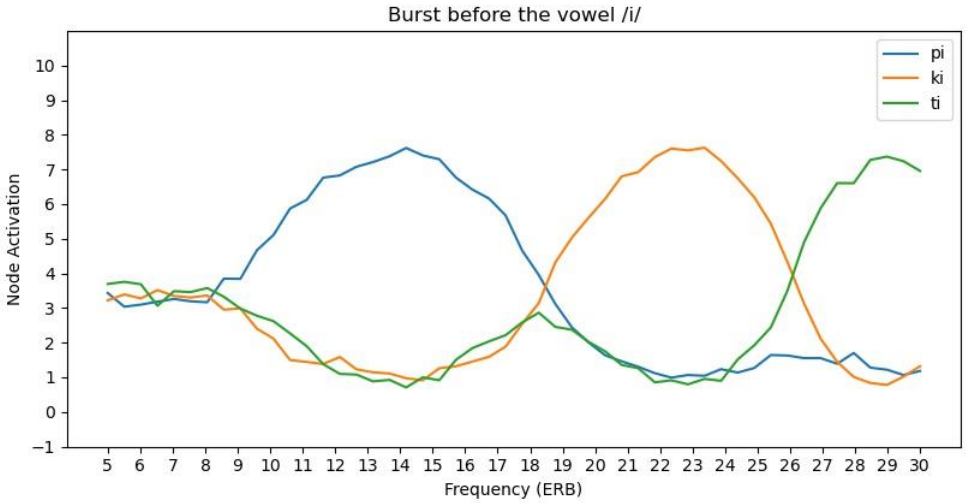


***Figure 8*** *– Average node activation for /pi/, /ki/, and /ti/*

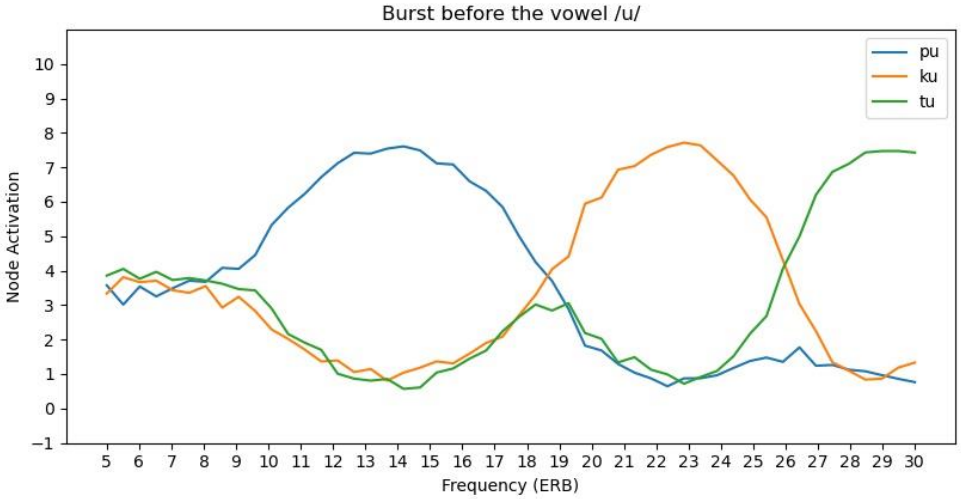


***Figure 9*** *– Average node activation for /pu/, /ku/, and /tu/*

The x-axis corresponds to the frequency of the input burst, and the y-axis shows the activation of the meaning node. In Figure 8, for example, the network is tested on Ci syllables. When the input burst has a frequency of ≈ 14.5 ERB, the activation of the node /pi/ is more than 7, while the activations for /ki/ and /ti/ are close to 1. During training, the meaning node that corresponds to the sound receives an activation of 10, while the other meaning nodes have zero activation. Therefore, an activation of 7 indicates that the network is quite certain it has heard the syllable /pi/. Near the category boundaries, the activation slowly decreases, until the middle of the adjacent category. The boundaries between the consonants are located exactly in the middle between the burst means of the two consonants involved. The boundary between /p/ and /k/ lies around 18.5 ERB and the boundary between /k/ and /t/ lies around 26 ERB.

The activity patterns in the figures correspond to the locations of the bursts in Table 3, and have the shape of a Bell curve, corresponding to the distribution of the input bursts. The boundaries between the different consonants are not very sharp, but very clear. For the lower frequencies, below 9 ERB, all consonants have a similar activation because there is no consonant with such a low burst. This activation is relatively high for all three consonants because the vowel part of the syllable also plays a role in the activation of the meaning node. Additionally, around the border between /p/ and /k/, the network also perceives /t/ more often, resulting in a bump in the graph. At the boundary between /k/ and /t/, the bump for /p/ is much lower, if present at all.

The categories are not all equally wide. The category for /p/ stretches out further than /k/, and the category for /t/ is not completely included in the figures because the category expands beyond the range of the basilar membrane that is used in the network. This is the result of the chosen range for the basilar membrane, 5 to 30 ERB, and the high mean frequency for /t/ at 29 ERB.

## 4.2 Syllables with vowel-affected mean bursts

After demonstrating the network behaves as expected with a fixed burst mean value for each plosive, a new network is trained with different burst means depending on the following vowel. The syllable /ki/ is included in the model language, despite its absence in the results of earlier perception experiments (Kiefte & Kluender, 2005; Liberman, Delattre & Cooper, 1952). Removing /ki/ from the language results in a dissimilar distribution for /k/, /p/, and /t/, which is unwanted for the comparison of the fixed-mean and variable-mean language types.

After 10,000 training steps, each vowel and plosive is represented in the input approximately 3,333 times and each meaning node approximately 1,111 times. Because there are three different variations for each plosive, the occurrence of each variation is around 1,111. This is three times less than in the language with a fixed mean burst, but it is still enough for the network to learn to produce most of the sounds.

### 4.2.1 The comprehension of the model

The network is tested from a sound input to a meaning output to test how well it performs when it is asked to find the correct meaning for an unlabeled sound input. The testing procedure is equal to the one used in section 4.1.

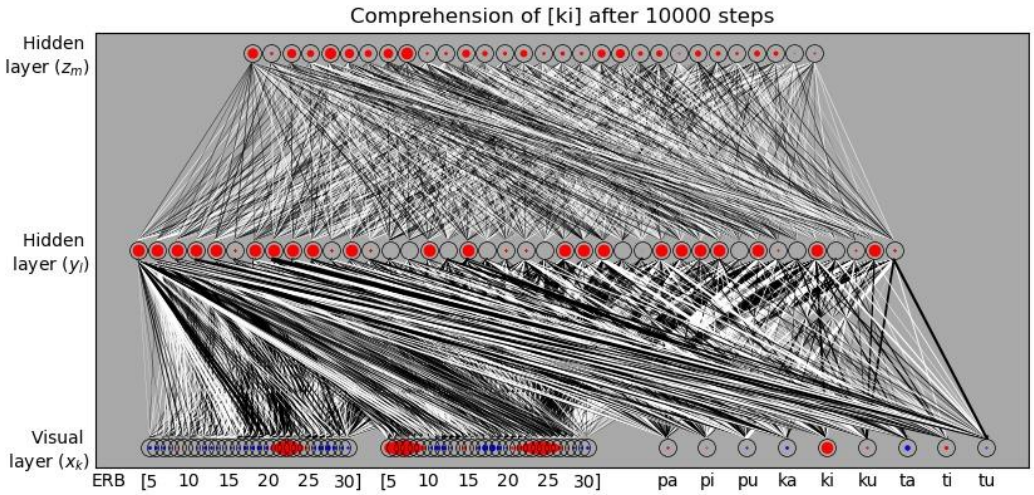


***Figure 10*** *– Network after 10,000 steps when tested from sound to meaning for /ki/*

In Figure 10, the sound input is correctly identified as /ki/, while the other meaning nodes are either slightly active or bear negative activations. All syllables ending in /i/ are active, while the other syllables starting with /k/ are not all active, only /ku/ has a small activation. This is most likely the result of the shared F1 of /i/ and /u/. Moreover, the kV syllables do not share the same mean burst frequency, while the Ci syllables share their formants values. Yet, the activations of all 'wrong' syllables are neglectable.

The mean frequencies for the bursts are far enough apart for separate categories to form, but with a standard deviation of 1.5 ERB in the sampling of the sound inputs, the categories may overlap. In Figure 11, the network encountered a sound input that is supposed to be/pi/, but the network does not know how to categorize the sound. The echo of the network places the burst between the expected locations for /p/ before /i/ and /u/. Because the input of the network is not visible in the figure, it is not possible to determine what input the network encountered in this specific case, but it is most likely further away from the mean burst for /pi/ around 14 ERB, situated closer to the 18 ERB mean burst of /pu/, causing confusion about the meaning of the sound.

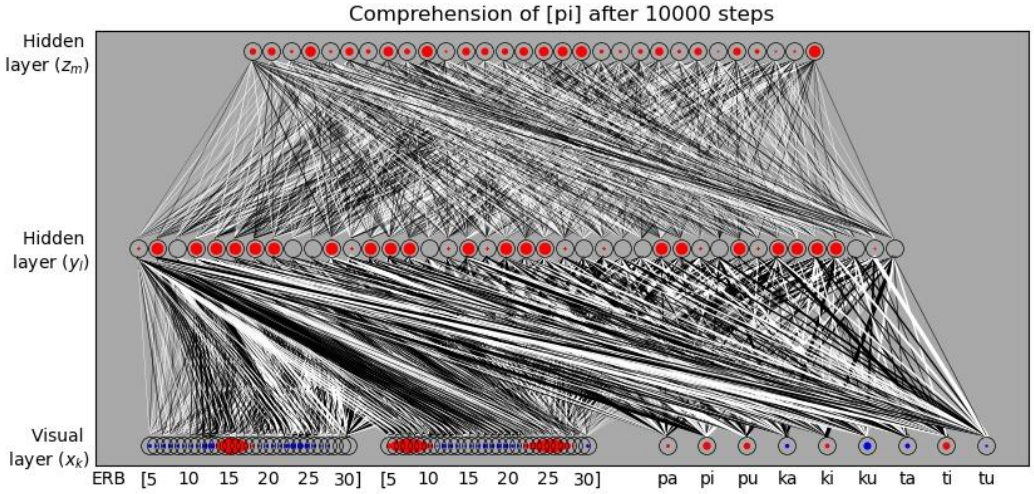


***Figure 11*** *– Network after 10,000 steps when tested from sound to meaning for /pi/*

### 4.2.2 The production of the model

Testing the same network with only a meaning input is more demanding for the model than testing with a sound input, but when tested on the input *pu*, the network produces a sound output that corresponds to the meaning, as can be seen in Figure 12. For /tu/, the burst should lie around 18 ERB, and the vowel formants around 7 and 13 ERB, which is indeed what is visible in Figure 12. However, the model is not equally accurate for other syllables, for example /pa/ in Figure 13.

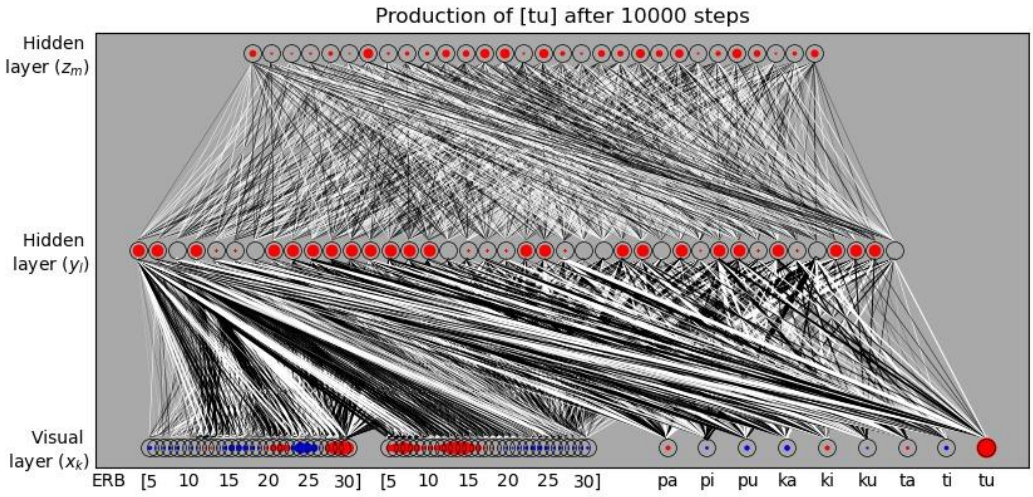


***Figure 12*** *– Network after 10,000 steps, tested on the meaning /tu/*

In Figure 13, the network is given /pa/ as input, and after resonating in the hidden layers, a sound output is produced. The network produces a burst around 9 ERB, which corresponds to /p/ before /a/, but there is also activation 25 and 30 ERB, where the bursts for /t/ would be. The vowel formants are located around 13 and 19 ERB, which is expected for /a/, but there are two additional formants around

6 and 23 ERB. The network creates an entirely new vowel with four formants, resulting in a meaning echo of /pa/ and /ta/, /tu/, and /ti/.

The activation on the basilar membrane partially explains the echo of the meaning nodes, where /pa/, /ti/, and /tu/ have strong activations, but because the network is tested from meaning to sound, the echo on the meaning nodes is not what we are interested in. However, it might give a good insight into how the network processes its input. Apparently, the input /pa/ activates hidden nodes that are also activated for the higher frequency bursts of /t/, resulting in an output far away from the desired output.

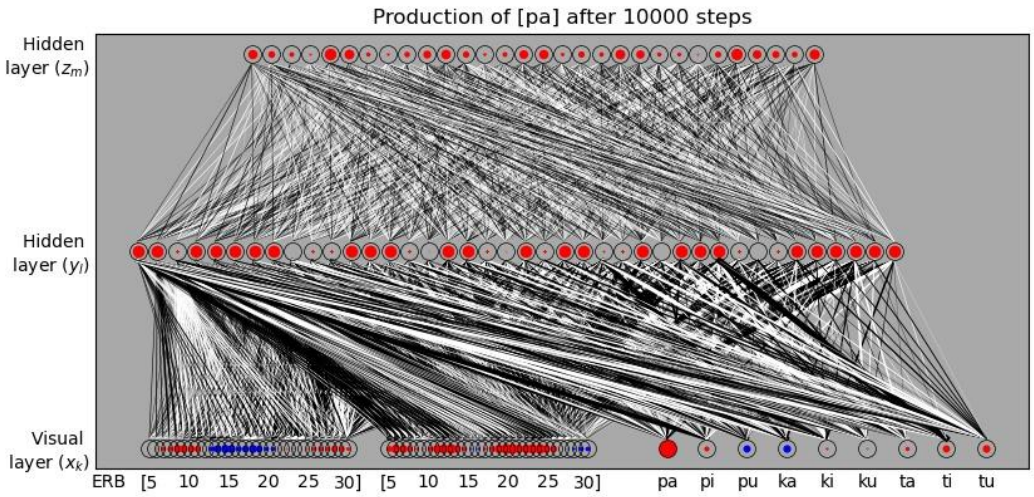


***Figure 13*** *– Network after 10,000 steps, tested on the meaning /pa/*

The network's inconsistent behavior in the production of sounds might be the result of the small number of training data for each plosive variant. However, after 30,000 and even 50,000 learning steps, the model does not perform significantly better. Figures 14 and 15 show the network for /pa/ after 30,000 and 50,000 learning steps respectively. After 30,000 steps, the burst for /p/ shows higher activation of the lower frequency nodes. However, on the vowel slab, an extra formant occurs. Now, the meaning node for /ku/ is also activated.

After 50,000 steps, the burst is more defined on the first slab. Yet, the formants on the second auditory slab are not corresponding to /a/. The first formant corresponds to the expected F2 of /a/, but the second formant corresponds to the F2 of /i/. Hence, all Ci nodes are activated as well.

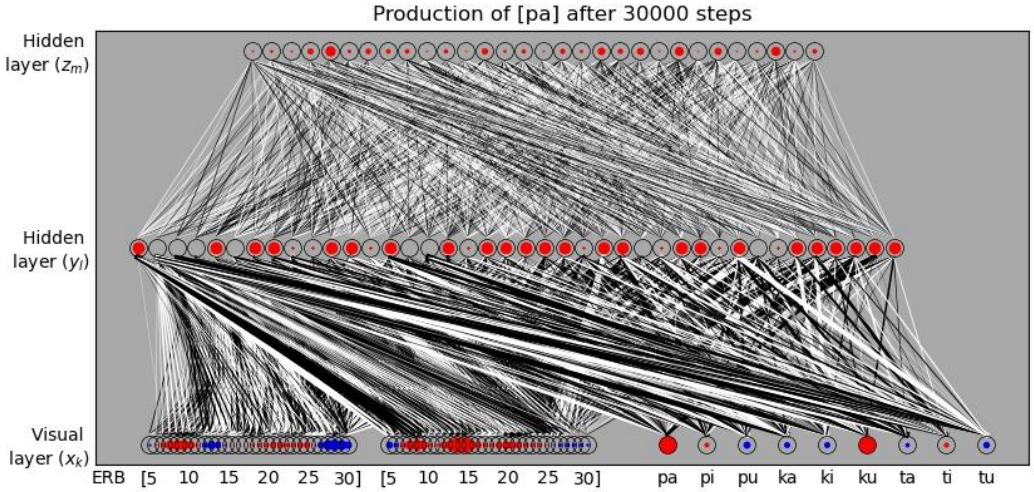


***Figure 14*** *– Network after 30,000 steps, tested on the meaning /pa/*

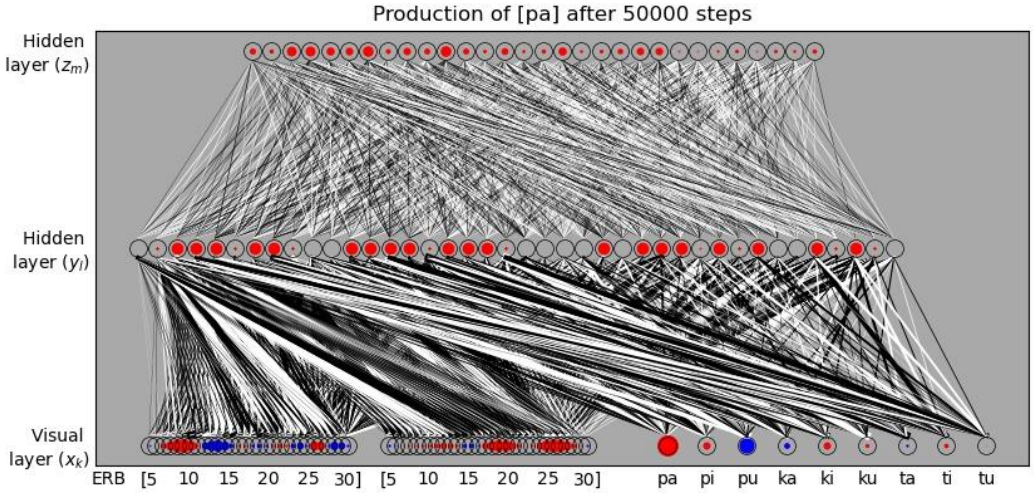


***Figure 15*** *– Network after 50,000 steps, tested on the meaning /pa/*

Because increasing the number of learning steps does not improve the performance of the model, we can conclude that the irregularities are not the result of an undertrained network. Hence, it remains to be investigated what causes the bad performance on this example. Also, because the output quality of the network does not improve when the number of learning steps increases, in all further steps, the network is considered trained after 10,000 learning steps for all further analyses.

### 4.2.3 Category boundaries

Figures 16 to 18 show the consonant categories that are formed when the network is tested with various bursts. The categories are no longer formed at the same frequencies for each vowel, resulting

in three distinct figures. Just as for the fixed-mean plosives, the three consonant categories are clearly visible for all vowel conditions. However, the locations and spread of the categories are variable.

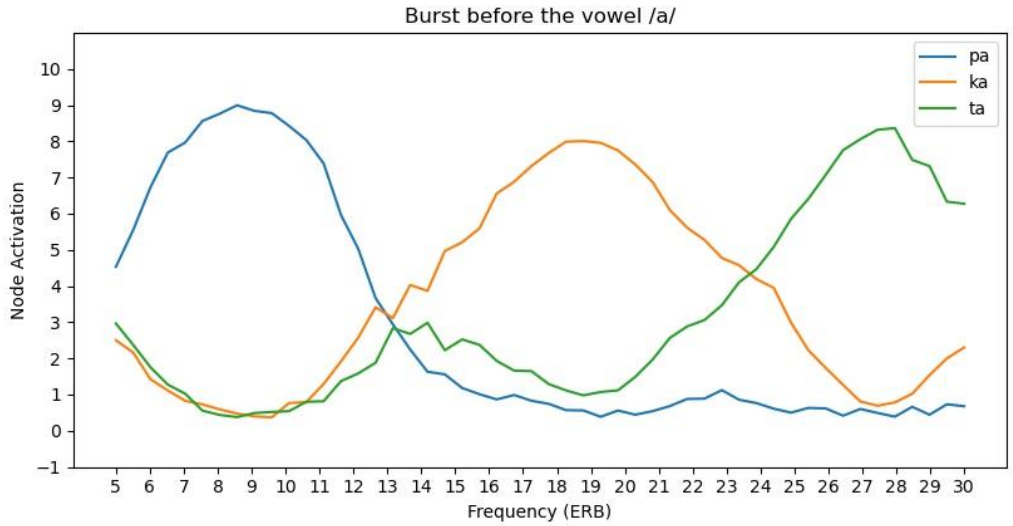


***Figure 16*** *– Average node activation for /pa/, /ka/, and /ta/*

In Figure 16, the categories are well-formed, with clear boundaries between the categories, and the mean bursts for the three plosives are evenly spaced on the basilar membrane. As a result, the model has no regions where it does not know what sound it has heard, except for the inevitable confusion at the category boundaries.

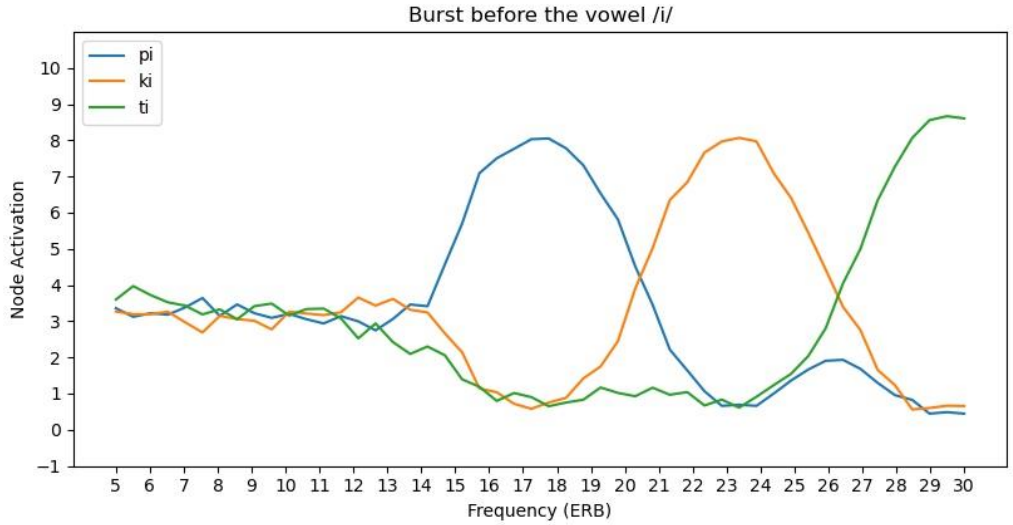


***Figure 17*** *– Average node activation for /pi/, /ki/, and /ti/*

The consonants before /i/ have mean bursts that are close together. The burst for /p/ has a wide spread, but the maximum activation is lower than for /a/. The same is true for /k/. The activation for /t/, however, is comparable to the situation in Figure 16.

Because the mean frequency for /p/ is much higher before /i/ than before /a/, the model does not know which one of the syllables to choose for the lower frequencies. Starting from 14 ERB, the

model correctly identifies /pi/, but for the frequencies below, the activations for all three consonants are comparable. For /u/ in Figure 18, the situation is similar to the one in Figure 17. For the low frequencies (<10 ERB), the model cannot correctly identify the input, and for frequencies above 30 ERB, this also seems to be the case.

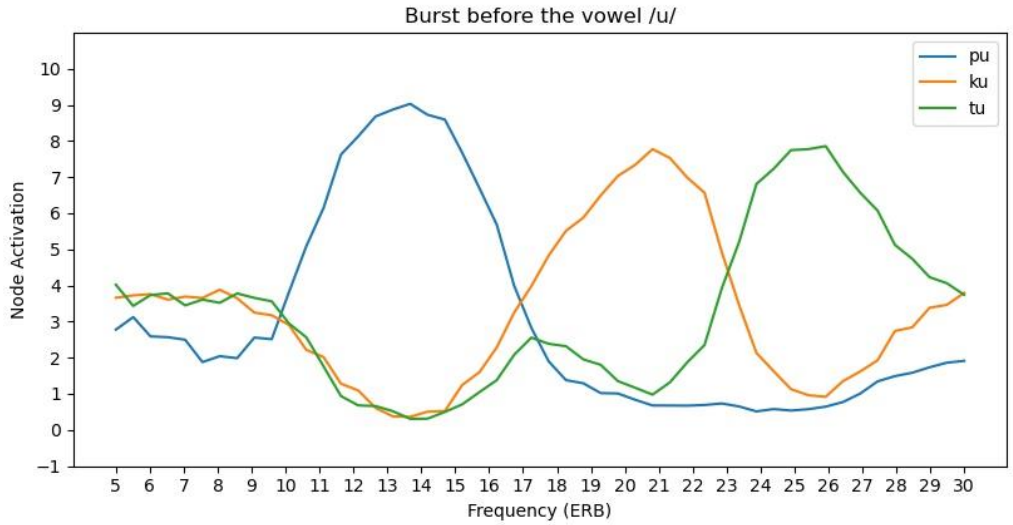


***Figure 18*** *– Average node activation for /pu/, /ku/, and /tu/*

When comparing the activations in the previous figures with the frequencies in Table 4, it is evident that the highest activation occurs at the mean frequency for each of the bursts.

***Table 4*** *– Mean frequencies for the consonants in ERB units*

| **Consonants** | **p** | **t** | **k** |
|---|---|---|---|
| Mean frequency before /a/ | 9 | 27 | 19 |
| Mean frequency before /i/ | 18 | 29 | *23* |
| Mean frequency before /u/ | 14 | 25 | 21 |

## 5. Cosine Similarities

The models described in sections 4.1 and 4.2 share their network structure, with two basilar membrane representations and nine meaning nodes. This makes it possible to compare the activation of the two models for the hidden layers. The middle layer of the network is connected to both the auditory nodes and the meaning nodes, forming the connection between the phonetic representation and the meaning. Thus, this layer can be seen as the phonological layer, following the model of speech production and perception displayed in Figure 19. Since the emergence of phonological categories is expected to be visible at this layer (Boersma, Chládková & Benders, 2021), we compare the activations

at this layer for the nine different syllables to determine the phonological similarity within each of the models.

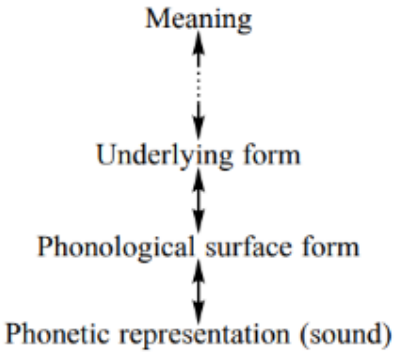


***Figure 19*** *– Bidirectional phonology and phonetics (Boersma, 2019)*

The similarity of the syllables is measured using cosine similarity, which is the inner product of two normalized vectors. The formula for normalization is given in (19), where $v_i$ is the vector to be normalized and n is the number of dimensions, in this case n = 9, because the language used in the model contains nine syllables.

(19) $V_i = \frac{v_i}{\sqrt{\sum_{j=1}^{n} v_j^2}}$

The activations of the middle layer are all between 0 and 1 because of the sigmoid function. Therefore, the cosine similarity returns a value between 0 and 1, 0 being completely different and 1 being completely identical. We compare the similarity of the middle layer for sound inputs, which is almost identical to the similarity for meaning inputs, which can be found in Appendix IV. It is expected that syllables ending in /a/ and /u/ are more similar than syllables ending in /a/ and /i/, because /a/ and /u/ share a formant around 13 ERB, while /a/ and /i/ do not share a formant. Likewise, /u/ and /i/ share their F1 at 7 ERB and will have higher similarity than /a/ and /i/.

Tables 5 and 6 show the cosine similarity in percentages for the fixed-mean and variable-mean burst conditions respectively. In Table 5, it is visible that syllables sharing their vowel are more similar than syllables sharing a consonant. For example, /pa/ is more similar to /ta/ and /ka/ than to /pu/ and /pi/. This is presumably the result of the amount of activation on the auditory slabs. For the consonant, there is one small range of nodes active, while a vowel consists of two formants, which have a bigger range of activation than the burst.

***Table 5*** *– Fixed burst cosine similarity for comprehension*

| | **pa** | **pu** | **pi** | **ta** | **tu** | **ti** | **ka** | **ku** | **ki** |
|---|---|---|---|---|---|---|---|---|---|
| **pa** | 100 | 61 | 54 | 68 | 56 | 51 | 67 | 54 | 49 |
| **pu** | 61 | 100 | 60 | 57 | 69 | 55 | 55 | 68 | 53 |
| **pi** | 54 | 60 | 100 | 49 | 54 | 64 | 49 | 54 | 65 |
| **ta** | 68 | 57 | 49 | 100 | 62 | 55 | 66 | 57 | 50 |
| **tu** | 56 | 69 | 54 | 62 | 100 | 61 | 55 | 68 | 54 |
| **ti** | 51 | 55 | 64 | 55 | 61 | 100 | 52 | 55 | 63 |
| **ka** | 67 | 55 | 49 | 66 | 55 | 52 | 100 | 62 | 55 |
| **ku** | 54 | 68 | 54 | 57 | 68 | 55 | 62 | 100 | 59 |
| **ki** | 49 | 53 | 65 | 50 | 54 | 63 | 55 | 59 | 100 |

For the variable burst condition, the similarities are comparable to the fixed mean condition, although the similarities are all lower. Syllables that share a vowel are still more similar than syllables that share a consonant, but the decrease is similar for all syllables, not only for consonant-sharing syllables. Because none of the syllables share a burst frequency anymore, the similarity between the consonant-sharing syllables was expected to be much lower than in the fixed-mean condition. Yet, the similarity seems to be unaltered.

***Table 6*** *– Variable burst cosine similarity for comprehension*

| | **pa** | **pu** | **pi** | **ta** | **tu** | **ti** | **ka** | **ku** | **ki** |
|---|---|---|---|---|---|---|---|---|---|
| **pa** | 100 | 55 | 53 | 62 | 59 | 51 | 62 | 57 | 52 |
| **pu** | 55 | 100 | 55 | 56 | 66 | 57 | 57 | 66 | 55 |
| **pi** | 53 | 55 | 100 | 49 | 56 | 64 | 50 | 55 | 62 |
| **ta** | 62 | 56 | 49 | 100 | 57 | 52 | 68 | 56 | 52 |
| **tu** | 59 | 66 | 56 | 57 | 100 | 54 | 55 | 67 | 55 |
| **ti** | 51 | 57 | 64 | 52 | 54 | 100 | 48 | 57 | 67 |
| **ka** | 62 | 57 | 50 | 68 | 55 | 48 | 100 | 58 | 51 |
| **ku** | 57 | 66 | 55 | 56 | 67 | 57 | 58 | 100 | 56 |
| **ki** | 52 | 55 | 62 | 52 | 55 | 67 | 51 | 56 | 100 |

# 6. Discussion

The Restricted Boltzmann machine is capable of processing consonants-vowel pairs and can learn to comprehend and produce those syllables in different language environments. The language with fixed-mean bursts demonstrated that the model can process consonants when modeled as a single burst. The language with variable-mean bursts shares features with the natural languages, where the frequency of the burst can vary based on the following vowel.

In both conditions, the perceptual magnet effect and categorical perception can be observed. The category boundaries created in the variable-burst condition can be compared to the results of perception experiments with simplified artificial speech prompts (Liberman, Delattre & Cooper, 1952; Kiefte & Kluender, 2005). The input of the model closely resembles the prompts in the experiments and the results are similar: the category boundaries between the plosives are found at similar frequencies. Yet, the burst frequencies in this research were based on the categories found by Kiefte & Kluender (2005), and one could argue whether it is justified to conclude that the model has similar perception to human beings.

In the fixed-mean language environment, the mean bursts for the plosives are not equivalent to the mean of all variants of that plosive in the variable-mean environment. The fixed-mean burst frequencies were determined based on Halle, Hughes, and Radley (1957) before the variable-mean condition was created, and the variable means were established. The frequency of the fixed-mean burst does not influence the results significantly. The location of the category boundaries would change, but the general pattern would stay intact.

The use of a single burst with a small width on a continuum that ranges from 5 to 30 ERB can pose a problem in the learning of the model. The auditory slabs contain 30 nodes each, and for the burst, only three or four nodes have a reasonable activation. For the vowel, there are two formants with a broader activation spread, more than double the number of nodes. The result is that the vowel has a more prominent place in determining which meaning node is active. The activation level of the burst nodes in the current model is presumably too low to outweigh the higher number of active nodes at the auditory slab for the vowel.

The cosine similarities in the two language conditions were comparable, although the similarities for the varying-mean bursts were all lower than for the fixed-mean condition. This suggests that the phonological categories that emerged in the fixed-mean condition are also present in the varying-mean condition. So, even though /p/ does not have the same burst frequency in /pa/ as in /pi/ in this condition, they still share an almost equal amount of features as in the fixed-mean condition, where the burst frequency is identical. Therefore, we can argue that the /p/ in /pa/, /pi/, and /pu/ are all instances of the same phoneme and not three different phonemes. However, this proposition is hard

to extend to /t/ and /k/, because the similarities for these two consonants are almost identical. For example, /ku/ is have an equal similarity to /ka/ and /ki/ as to /ta/ and /ti/, making it impossible to divide them into two distinct phonological categories.

When using a model to enlarge the understanding of a phenomenon or theory, most often concessions have to be made. This model is holistic, using two representations of the basilar membrane to include the aspect of time. Moreover, the model uses simplified language input, vowels consisting of two formants only, and plosives with the burst frequency as the key feature for categorization. This frequency distinguishes the places of articulation of the plosives, and because all consonants are voiceless plosives, taking the burst frequency as the main feature is justified. However, when other consonants were to be included in the model, more concessions would have to be made.

Voiceless plosives are not the only consonants that exist. Therefore, the model has to be expanded to include consonants with other features, which means the structure of the network has to change too. For example, if voiced plosives are added, there has to be a node or slab that accounts for the presence or absence of voicing. If other consonants are added, the presence or absence of a stop has to be taken into account. The downside of this technique is that the input layer is getting further and further away from the way sounds are received at the basilar membrane in the human ear.

Because of the complications that come with the addition of different kinds of consonants, the current research focused on the addition of voiceless plosives in the initial position, knowing that it is a severe limitation to include just one type of consonants at a specific place in the syllable. Further research can focus on the implementation of single consonants, VC syllables, consonant clusters, or different types of consonants in the currently used structure; for example by adding voiced plosives or nasals. Also, one could try to determine whether using this model is justified, or that a different network structure or learning algorithm has to be used in order to model language in a way that resembles the processing of natural language more closely.

Lastly, the behavior of the network around the category boundary between /p/ and /k/ can be a subject for further research. In both the fixed-mean condition and the variable-mean condition, the activation of the meaning node for /t/ increases, at least preceding /a/ and /u/. It would be interesting to test whether this behavior can also be found in human beings.

# 7. Conclusion

In this study, the main focus was to adapt Boersma's (2019) framework for the modeling of categorical perception to process not only vowels but also consonants. Three voiceless plosives were modeled as a burst and used as input for the model. After the goal of incorporating consonants was reached, the network was tested in two language conditions, one in which plosives had a fixed mean

burst frequency, and one with varying-mean bursts. In both conditions, the model was able to correctly identify incoming sounds and also create sounds from the lexicon. Also, the perceptual magnet effect and categorical perception were observed.

The category boundaries found for the plosives with varying mean frequencies were comparable to the ones found by Liberman et al. (1952) and later Kiefte and Kluender (2005). The similarities in the hidden layer were comparable between the two language conditions, suggesting that the categorization of all /p/, /t/, and /k/ consonants is justified, even though there is intracategorical variation for all these phonemes. Further research is needed to determine whether this is still the case when the network is not holistic but uses memory to process sound sequences. Also, it still has to be determined whether the model behaves the same for VC syllables and whether the results from this research can be reproduced in a human perception experiment.

# 8. References


Beemdelust, Angelica. 2020. The addition of time to a restricted Boltzmann machine using a holistic model, and the machine's ability to distinguish between different sequences of sounds. *Thesis BA Linguistics*, University of Amsterdam.

Boersma, Paul. 2019. Simulated distributional learning in deep Boltzmann machines leads to the emergence of discrete categories. *Proceedings of the 19th International Congress of Phonetic Sciences*, Melbourne, 5-9 August 2019. 1520–1524.

Boersma, Paul, Titia Benders & Klaas Seinhorst. 2020. Neural networks for phonology and phonetics. *Journal of Language Modelling* 8: 103–177.

Boersma, Paul, Kateřina Chládková & Titia Benders. 2021. Phonological features emerge substance-freely from the phonetics and the morphology. Submitted to a journal.

Davis, Katharine & Patricia Kuhl. 1994. Tests of the perceptual magnet effect for American English /k/and /g/. *Journal of the Acoustical Society of America* 95. 2976.

Halle, Morris, George W. Hughes, & Jean-Pierre A. Radley. 1957. Acoustic properties of stop consonants. *The Journal of the Acoustical Society of America 29*(1). 107-116.

Iverson, Paul & Patricia Kuhl. 1996. Influences of phonetic identification and category goodness on American listeners' perception of /r/ and/l/. *Journal of the Acoustical Society of America* 99. 1130-1140.

Iverson, Paul & Patricia Kuhl. 2000. Perceptual magnet and phoneme boundary effects in speech perception: Do they arise from a com-mon mechanism? *Perception and Psychophysics* 62(4), 874-886.

Kiefte, Michael & Keith Kluender. 2005. Pattern Playback revisited: Unvoiced stop consonant perception. *The Journal of the Acoustical Society of America*, *118*(4), 2599-2606.

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. 93–107.

Lacerda, Francisco. 1995. The perceptual-magnet effect: An emergent consequence of exemplar-based phonetic memory. In Kjell Elenius & Peter Branderud (Eds.), *Proceedings of the XIII$^{th}$ International Congress of Phonetic Sciences* (2). Stockholm: KTH and Stockholm University. 140-147.

Liberman, Alvin, Pierre Delattre & Franklin Cooper. 1952. The role of selected stimulus-variables in the perception of the unvoiced stop consonants. *The American Journal of Psychology*, 497-516.

Liberman, Alvin, Pierre Delattre, Louis Gerstman & Franklin Cooper. 1956. Tempo of frequency change as a cue for distinguishing classes of speech sounds. *Journal of Experimental Psychology*, *52*(2), 127.

Maye, Jessica, Janet Werker & LouAnn Gerken. 2002. Infant sensitivity to distributional information can affect phonetic discrimination. *Cognition*, *82*(3), B101-B111.

Werker, Janet & Richard Tees. 1984. Cross-Language Speech Perception: Evidence for Perceptual Reorganization During the First Year of Life. *Infant Behavior and Development 7*. 49-63.

Yang, Jingnan. 2020. Distributional Learning of Mandarin Lexical Tones in Bidirectional Deep Neural Network. [Unpublished paper], University of Amsterdam.

# Appendices

## Appendix I – Code for the Deep Restricted Boltzmann Machine

```
# dRBM visualization and model
import numpy as np
import variable_input as ig
import random
from scipy.special import expit


def bernoulli(arr):
    for x in range(len(arr)):
        arr[x] = int(random.random() < arr[x])
    return arr

class DRBM:
    # initialize all variables of the network
    def __init__(self, size_y, size_z, learning_rate, vc_dict):
        self.sizes = (size_y, size_z)
        self.vc_dict = vc_dict
        self.learning_rate = learning_rate

        # activations
        self.x_k = ig.create_random_pair(vc_dict)
        self.y_l = np.zeros((size_y, 1))
        self.z_m = np.zeros((size_z, 1))

        # biases
        self.bias_a = np.zeros((len(self.x_k), 1))
        self.bias_b = np.zeros((size_y, 1))
        self.bias_c = np.zeros((size_z, 1))

        # weights
        self.weights_u = np.random.random_sample((len(self.x_k), size_y)) - 0.5
        self.weights_v = np.random.random_sample((size_y, size_z)) - 0.5

    # Functions to update the weights and biases
    def update_biases(self, anti = False):
        '''This function updates all biases in the network during the Hebbian
              and anti-Hebbian learning phases.
              When "anti" is true, the network is in the anti-
              Hebbian learning phase.'''
        if anti == True:
            self.bias_a -= self.learning_rate * self.x_k
            self.bias_b -= self.learning_rate * self.y_l
            self.bias_c -= self.learning_rate * self.z_m
        else:
```

```
        self.bias_a += self.learning_rate * self.x_k
        self.bias_b += self.learning_rate * self.y_l
        self.bias_c += self.learning_rate * self.z_m

def update_u(self, anti = False):
    '''This function updates the weights between the bottom two layers.
    When "anti" is true, the network is in the anti-Hebbian learning phase.'''
    if anti == True:
        self.weights_u -= self.learning_rate * np.matmul(self.x_k, self.y_l.T)
    else:
        self.weights_u += self.learning_rate * np.matmul(self.x_k, self.y_l.T)

def update_v(self, anti = False):
    '''This function updates the weights between the top two layers.
    When "anti" is true, the network is in the anti-Hebbian learning phase.'''
    if anti == True:
        self.weights_v -= self.learning_rate * np.matmul(self.y_l, self.z_m.T)
    else:
        self.weights_v += self.learning_rate * np.matmul(self.y_l, self.z_m.T)

# Functions to change activation in the three layers
def update_y(self, dream = False):
    '''This function updates the activations of the middle layer.
    When "dream" is true, the network is in the dreaming phase.'''
    if dream == False:
        self.y_l = expit(self.bias_b + np.matmul(self.weights_u.T, self.x_k)
          + np.matmul(self.weights_v, self.z_m))
    else:
        self.y_l = bernoulli(expit(self.bias_b + np.matmul(self.weights_u.T,
          self.x_k) + np.matmul(self.weights_v, self.z_m)))

def update_z(self, dream = False):
    '''This function updates the activations of the top layer.
    When "dream" is true, the network is in the dreaming phase.'''
    if dream == False:
        self.z_m = expit(self.bias_c + np.matmul(self.weights_v.T, self.y_l))
    else:
        self.z_m = bernoulli(expit(self.bias_c +
                        np.matmul(self.weights_v.T, self.y_l)))

def update_x(self):
    '''This function updates the activations of the visible layer.'''
    self.x_k = self.bias_a + np.matmul(self.weights_u, self.y_l)

# Initial settling and dreaming phase
def initial_dreaming(self, dream = False):
     '''This function executes the initial settling or the dreaming phase.
    When "dream" is true, the network is in the dreaming phase,
```

```
   otherwise, it is initial settling.'''
    if dream == False:
        self.x_k = ig.create_random_pair(self.vc_dict)

    for _ in range (10):
        if dream == True:
            self.update_x()
        self.update_y(dream)
        self.update_z(dream)

# (Anti-)Hebbian learning phase
def Hebbian(self, anti = False):
    '''This function executes the Hebbian or the anti-Hebbian learning phase.
    When "anti" is true, the network is in the anti-Hebbian leaning phase.'''
    self.update_biases(anti)
    self.update_u(anti)
    self.update_v(anti)

def train_network(self, steps):
    '''This function trains the network by walking through the four learning
         stages a certain amount of times.'''
    for i in range(steps):
        self.initial_dreaming()
        self.Hebbian()
        self.initial_dreaming(dream=True)
        self.Hebbian(anti=True)


def test_network(self, x_in):
    '''This function tests the network for a specific input and returns
         the activations of the bottom two layers.'''
    self.x_k = x_in
    for _ in range(10):
        self.update_y()
        self.update_z()
        self.update_y()
        self.update_x()
    return self.x_k.flatten(), self.y_l.flatten()
```

## Appendix II – Main code for Visualization

```
# Running code
import time
import matplotlib.axes as ax
import matplotlib.pyplot as plt
import varbm as rbm
import variable_input as ig
import numpy as np

# Choose variable or fixed mean burst
setting = 'variable'

# Choose comprehension or production for testing
direction = 'Comprehension'

# Mean vowel and consonant values
a = (13, 19)
i = (7, 25)
u = (7, 13)

if setting == 'variable':
    vc_dict = {"pa": (9, a), "ta": (27, a), "ka": (19, a), "pi": (18, i),
       "ti": (25, i), "ki": (23, i), "pu": (14, u), "tu": (29, u), "ku": (21, u)}
if setting == 'fixed':
    vc_dict = {"pa": (14, a), "ta": (29, a), "ka": (23, a), "pi": (14, i),
       "ti": (29, i), "ki": (23, i), "pu": (14, u), "tu": (29, u), "ku": (23, u)}

# Starting values
steps = 10000
learning_rate = 0.001

# Create model
drbm = rbm.DRBM(40, 30, learning_rate, vc_dict)

drbm.train_network(steps)

test_to_sound = {}
test_to_meaning = {}

# Possible meanings
pair_meanings = ["pa", "pi", "pu", "ka", "ki", "ku", "ta", "ti", "tu"]


# Color-weight relation
def color_choice(weight):
    ''' This function makes positive weights black and negative weights white '''
    if weight >= 0:
        color = "black"
```

```
    else:
        color = "white"
    return color

def line_width(weights, position):
    ''' This function determines the line width of a connection
       at a certain position '''
    width = weights[position[0], position[1]]
    return width

# Loop over all pair meanings that have to be drawn
for pair in pair_meanings:
    # Determine which test has to be taken
    if direction == 'Comprehension':
        comp_test, vow = ig.create_test_from_sound(pair, vc_dict)
        test_to_meaning[pair] = drbm.test_network(comp_test)
    else:
        test_to_sound[pair] = drbm.test_network(prod_test)
        prod_test, vow = ig.create_test_from_meaning(pair)


    # Drawing the network
    model = [drbm.x_k, drbm.y_l, drbm.z_m]
    nodes_per_layer = {}

    # Loop over all layers
    for i, layer in enumerate(model):
        no_of_nodes = len(layer)
        xy_list = []

        # Determine the position of nodes in the plot
        for node in range(no_of_nodes):
            # Ensure spreading of parts in visible layer
            if i == 0:
                # Auditory nodes
                if node < 30:
                    x_position = 0.6 * -(no_of_nodes + (2 * node))
                    - (no_of_nodes * 0.4) - 9
                elif node < 60:
                    x_position = 0.6 * (-no_of_nodes + (2 * node)) –
                    ((no_of_nodes - 10) * 0.4) - 7

                # Meaning nodes
                else:
                    x_position = 2 * (-no_of_nodes + 3.5 * node) –
                    ((no_of_nodes + 5) * 2) - 121.2
```

```
            # Hidden layers
            else:
                x_position = 2 * (-no_of_nodes + (1.7 * node))

            y_position = i
            xy_position = (x_position, y_position)
            xy_list.append(xy_position)

        nodes_per_layer[i] = xy_list


        # draw connections from the top layer down
        if i >= 1:
            low_nodes = nodes_per_layer[i -1]
            high_nodes = nodes_per_layer[i]

            l_node_nr = 0

            for l_node in low_nodes:
                h_node_nr = 0
                for h_node in high_nodes:
                    x_values = l_node[0], h_node[0]
                    y_values = (l_node[1] + 0.05), (h_node[1] - 0.05)
                    width = line_width(drbm.weights_u, (l_node_nr, h_node_nr))
                    if -0.1 > width or width > 0.1:
                        fig = plt.plot(x_values, y_values,
                           color = color_choice(width), linewidth = abs(width))

                    h_node_nr += 1
                l_node_nr += 1

    # draw all nodes
    for layer, nodes in nodes_per_layer.items():
        # draw node activations
        activations = model[layer]
        i = 0

        for node in nodes:
            x, y = node
            activation = activations[i]

            # Make activations in hidden layers more visible
            if layer != 0:
                activation *= 6

            # Determine the color of the activation
            if activation > 0:
                act_color = 'red'
```

```
        else:
            act_color = 'blue'

        plt.plot(x, y, marker = 'o', mec = act_color, mfc = act_color,
          markersize = abs(activation))
        i += 1

    x, y = zip(*nodes)
    plt.plot(x, y, 'ko', mfc = "None", mew = 0.5, markersize = 10)


# general plot info
''' The labeling et cetera is added manually, when the parameters above change,
this piece of code has to be adapted too.'''

plt.yticks(range(3), ["Visual \n layer $(x_{k})$", "Hidden \n layer $(y_{l})$",
   "Hidden \n layer $(z_{m})$"])
plt.xticks(np.arange(-85, 70, 7),
          ["ERB", "[5", 10, 15, 20, 25, "30]",
          "[5", 10, 15, 20, 25, "30]", "",
          "pa", "pi", "pu", "ka", "ki", "ku", "ta", "ti", "tu"])

# remove ticks and show plot
ax = plt.gca()
ax.set_facecolor('darkgray')
plt.tick_params(length = 0)
plt.title(f"{direction} of [{vow}] after {steps} steps")

plt.show()
```

## Appendix III – Input generator code

```
# Script to generate inputs for the network
import numpy as np
import random

# Gaussian input
def Gaussian(node_nr, mean, w = 1.5):
    node_erb = node_nr * (25/29) + 5
    funct = np.exp((-0.5 * ((node_erb - mean) / w) ** 2))
    return funct

# function to create vowels
def create_vowel(formant_tuple, fixed_value = 0):
    F1 = formant_tuple[0]
    F2 = formant_tuple[1]

    if fixed_value == 0:
        mean_vow_F1 = np.random.normal(F1, 1)
        mean_vow_F2 = np.random.normal(F2, 1)
    else:
        mean_vow_F1 = F1 - fixed_value
        mean_vow_F2 = F2 + fixed_value

    # create empty vector and find mean nodes
    vowel_aud = np.linspace(5, 30, 30)

    # adapt the values for the mean and surrounding nodes
    for i in range(len(vowel_aud)):
        node_nr = i
        node_act = 10 * (Gaussian(node_nr, mean_vow_F1) +
              Gaussian(node_nr, mean_vow_F2)) - 1
        vowel_aud[i] = node_act

    return vowel_aud

def create_consonant(burst_mean, fixed_value = 0):
    if fixed_value == 0:
        burst = np.random.normal(burst_mean, 1.5)
    else:
        burst = burst_mean + fixed_value
    cons_aud = np.linspace(5, 30, 30)

    for i in range(len(cons_aud)):
        node_nr = i
        node_act = 12 * Gaussian(node_nr, burst, w=1) - 1
        cons_aud[i] = node_act

    return cons_aud
```

```
def meaning_to_one_hot(meaning):
    vector = np.zeros(9)
    poss_meanings = ["pa", "pi", "pu", "ka", "ki", "ku", "ta", "ti", "tu"]
    for i in range(len(vector)):
        if poss_meanings[i] == meaning:
            vector[i] = 10
    return vector

def create_random_pair(vc_dict):
    pair_meanings = list(vc_dict.keys())

    # select random pair
    random_pair = random.choice(pair_meanings)
    consonant = create_consonant(vc_dict[random_pair][0])
    vowel = create_vowel(vc_dict[random_pair][1])

    # create meaning vector
    meaning = meaning_to_one_hot(random_pair)
    netw_inp = np.concatenate((consonant, vowel, meaning))
    return netw_inp.reshape((len(netw_inp), 1))

def create_specific_pair(pair, vc_dict):
    consonant = create_consonant(vc_dict[pair][0])
    vowel = create_vowel(vc_dict[pair][1])

    # create meaning vector
    meaning = meaning_to_one_hot(pair)

    netw_inp = np.concatenate((consonant, vowel, meaning))
    return netw_inp.reshape((len(netw_inp), 1))

def create_test_from_sound(random_pair, vc_dict, fixed_value = 0):
    # select random pair
    # random_pair = random.choice(pair_meanings)
    elements = vc_dict[random_pair]
    consonant = create_consonant(elements[0], fixed_value)
    vowel = create_vowel(elements[1], fixed_value)

    # create empty meaning vector
    meaning = np.zeros(9)

    # put everything together
    netw_inp = np.concatenate((consonant, vowel, meaning))
    return netw_inp.reshape((len(netw_inp), 1)), random_pair

def create_test_from_meaning(random_pair):
    # select random pair
```

```
# random_pair = random.choice(pair_meanings)
consonant = np.zeros(30)
vowel = np.zeros(30)

# create meaning vector
meaning = meaning_to_one_hot(random_pair)

# put everything together
netw_inp = np.concatenate((consonant, vowel, meaning))
return netw_inp.reshape((len(netw_inp), 1)), random_pair
```

## Appendix IV – Cosine Similarity for production

*Table 7 – Fixed burst cosine similarity for production*

| | pa | pu | pi | ta | tu | ti | ka | ku | ki |
|---|---|---|---|---|---|---|---|---|---|
| **pa** | 100 | 60 | 55 | 67 | 56 | 50 | 66 | 56 | 50 |
| **pu** | 60 | 100 | 59 | 56 | 68 | 53 | 54 | 69 | 54 |
| **pi** | 55 | 59 | 100 | 51 | 55 | 63 | 52 | 55 | 66 |
| **ta** | 67 | 56 | 51 | 100 | 62 | 55 | 66 | 54 | 51 |
| **tu** | 56 | 68 | 55 | 62 | 100 | 60 | 55 | 68 | 54 |
| **ti** | 50 | 53 | 63 | 55 | 60 | 100 | 51 | 54 | 65 |
| **ka** | 66 | 54 | 52 | 66 | 55 | 51 | 100 | 60 | 56 |
| **ku** | 56 | 69 | 55 | 54 | 68 | 54 | 60 | 100 | 59 |
| **ki** | 50 | 54 | 66 | 51 | 54 | 65 | 56 | 59 | 100 |

*Table 8 – Variable burst cosine similarity for production*

| | pa | pu | pi | ta | tu | ti | ka | ku | ki |
|---|---|---|---|---|---|---|---|---|---|
| **pa** | 100 | 54 | 54 | 62 | 58 | 52 | 62 | 57 | 52 |
| **pu** | 54 | 100 | 55 | 57 | 67 | 55 | 56 | 67 | 54 |
| **pi** | 54 | 55 | 100 | 49 | 57 | 65 | 52 | 56 | 64 |
| **ta** | 62 | 57 | 49 | 100 | 58 | 50 | 67 | 56 | 51 |
| **tu** | 58 | 67 | 57 | 58 | 100 | 55 | 55 | 67 | 54 |
| **ti** | 52 | 55 | 65 | 50 | 55 | 100 | 49 | 57 | 67 |
| **ka** | 62 | 56 | 52 | 67 | 55 | 49 | 100 | 58 | 52 |
| **ku** | 57 | 67 | 56 | 56 | 67 | 57 | 58 | 100 | 56 |
| **ki** | 52 | 54 | 64 | 51 | 54 | 67 | 52 | 56 | 100 |