A neural network for multi-modal category formation in infants

Zackary Gibson
11576448

Bachelor Thesis - Linguistics
Supervised by Paul Boersma
2020

Introduction

Modelling experimental data and results with neural networks remains a relatively unexplored area of linguistic research, and is a particularly attractive method for its biological plausibility. Some areas in which neural networks have been applied successfully are in modeling auditory dispersion and category formation (Boersma, Benders & Seinhorst, 2020). This paper will attempt to model results from Plunkett, Hu, and Cohen (2008), which demonstrated that, for infants, category creation based on visual stimuli can be disrupted in the presence of incongruous spoken labels. In their study, 10-month olds were presented with line drawings (figure 1) of an imaginary creature with corresponding features in four of five experiments; e.g. spread ears would predict a short tail and vice versa. This was the narrow condition, in which the statistical distribution of the drawings led to the formation of two distinct categories of figures by the infants, each with diametrically opposed features: 1111 and 5555. The narrow continuum invites the participant to form two categories because the four leftmost drawings of figure 1's narrow condition each have bunched ears, large tails, long necks, and short legs which sits in contrast to the right four figures, which tend to have longer legs, shorter necks, smaller tails, and spread ears. Category formation was measured via the infants' novel looking preference. The broad condition was used only in a single experiment as a control. In the broad condition, the values of the legs and tail predicted opposing values for the neck and ears. Likewise, neck and ear values predicted opposing leg and tail values. For example, in the first figure of the broad condition, 1155, the values of 1 for the legs(1 = short legs) and tail (1= thin tail) predict values of 5 for the neck (5 = long neck) and ears (5 = spread ears). This pattern of distribution led the infants to form a single, general category: 3333, which can be observed by noting that there is less consistency within the broad condition creatures than within the narrow condition creatures. When spoken labels were introduced to the narrow condition stimuli, it had a disruptive and overriding effect on visual category formation. When two spoken labels were congruous with the visual category cues, there was no change in category formation. However, when a single spoken label was given for all figures, the infants formed only a single category, as in the broad condition control experiment. When two labels were assigned at random, no category formation occurred.

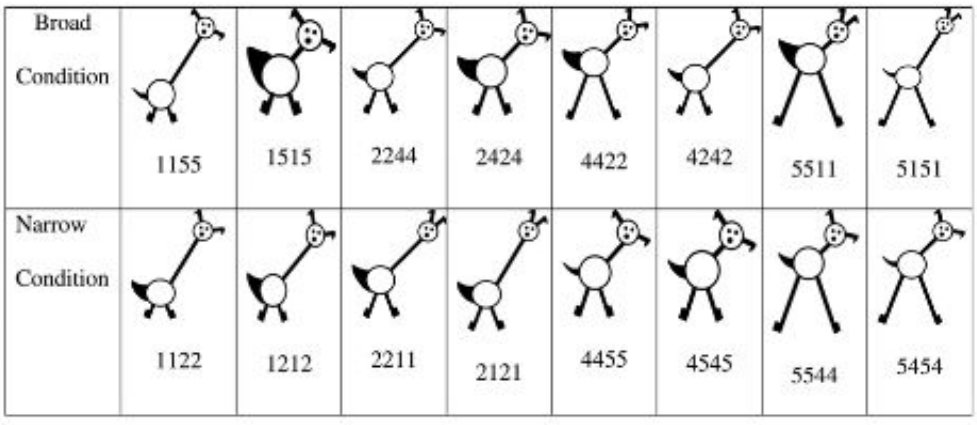

Figure 1: Broad and narrow condition line drawings (Plunkett et al., 2008).

A bimodal, parallel, neural network would be needed to model the effect shown by Plunkett et al. (2008). Similar models have already been constructed that can model the McGurk effect, which shows the

opposite effect: an overriding influence of the visual input over the audible input when presented simultaneously (McGurk & MacDonald, 1976). At this point, there seems to be two dominant approaches to bimodal modeling: Boltzmann machines and self-organizing map (SOM) networks.

SOM networks map inputs into two dimensional space without supervision based on patterns found within the data. One such network constructed by Gustaffson, Jantvik & Paplinski (2014) generated SOMs based on visual and auditory similarity of spoken segments. Video recordings of spoken segments were grouped into equivalence classes before being mapped together onto a grid of nodes, the SOM — three equivalence classes were created: labiodental, bilabial, and open mouth segments. The auditory cues underwent a similar process, but similarity was determined by mel-frequency cepstrum measurements, a measure of spectral power. The network itself featured parallel inputs for each modality which then fed into an output. The output provided a partial input to the auditory modality via a feedback loop. When an auditory /ba/ and visual /ga/ were supplied as inputs, /da/ was the likely bimodal winner due to its visual similarity to /ga/ and its auditory similarity to /ba/, as indicated on the SOMs. This network was able to reliably reproduce the McGurk effect from bimodal visual-audio inputs.

As a testament to the efficacy of the SOM approach, Plunkett et al. (2008) was successfully modeled using two SOM networks (Gliozzi, V., Mayor, J., Hu, J., & Plunkett, K. 2008). In this follow-up research, one of the networks had a single SOM that mapped both visual and audio cues; the other network featured three SOMs: visual, acoustic, and bimodal. Infant looking time was realized as "a function of the quantisation error." Their models were able to recreate infant looking time for each of the five experiments of Plunkett et al. (2008, p 401). Interestingly, the authors also note that after further training the model with stimuli from experiment five (two visual categories and a single spoken label) results in a transient effect: eventually the model rejects the label in favor of the two visual categories. This has interesting implications for infants — they may exhibit similar transient categorization behavior. Additionally, the shift in perception from single category to two categories could allow for their model to represent hierarchies of categorization.

Another contemporary approach to neural modeling is to use a Boltzmann machine. Boltzmann machines consist of layers of connected nodes whose activation probabilities are stochastically determined by statistical distributions within the input data. Ngiam, Khosla, Kim, Nam, Lee, and Ng (2011) demonstrate the usefulness of Boltzmann machines for modeling bimodal inputs. Using a video-only auto-encoder consisting of input nodes, hidden layers, and a reconstructed output, they were able to supply a visual input and reliably recreate both a visual and audio output. Their bimodal auto-encoder worked similarly, except it could receive audio or visual data as inputs, but was somewhat less reliable in its reconstructions. This model was also able to reproduce the McGurk effect. Visual /ga/ and auditory /ba/ inputs often resulted in the model perceiving /da/, despite /da/ not being in the input data set. This study, and those following it, provide proof of concept that Boltzmann machines can model bimodal inputs.

While Plunkett et. al (2008) has been modeled with a SOM network, it has not yet been modeled with a Boltzmann machine. Thus, this study will attempt to construct a bimodal Boltzmann machine to model the results of Plunkett et. al (2008). More specifically, a restricted deep Boltzmann machine will be constructed based on a modified version of Boersma's (2019) network for modeling emergent category formation. As with that network, this too will be constructed with the speech processing and analysis software Praat (Boersma & Weenink, 2020).

Methods

A three-layered network consisting of 42 input nodes, 50 intermediary nodes, and 40 output nodes was constructed. Each node of the input layer, with its respective activation (x_k, where x represents the activity level and k is the index specifying an individual node), is connected to each node of the intermediate layer (y_l) by a weighted connection (u_{kl}). Again, each intermediate layer node possesses an individual activation level. The intermediate level is also connected (v_{lm}) to each output layer node, again with every output layer node having its own activity level (z_m). The values of k, l, and m range from 1 to the number of nodes in their respective level. Furthermore, each node layer has its own bias, represented by a_k for the input layer, b_l for the middle layer, and c_m for the output layer. A positive bias value adds to the activity level of the affected node, increasing the likelihood of activation. Inversely, a negative bias is subtractive and decreases the likelihood of activation. Node connections exist only between layers, and never within a layer. The network is bidirectional — data moves between node layers; values in one layer will affect the values of the other layers during the period between the initial input and the final output (Boersma, 2019). Training the network occurs in four stages: initial settling, Hebbian learning, dreaming, and anti-Hebbian learning.

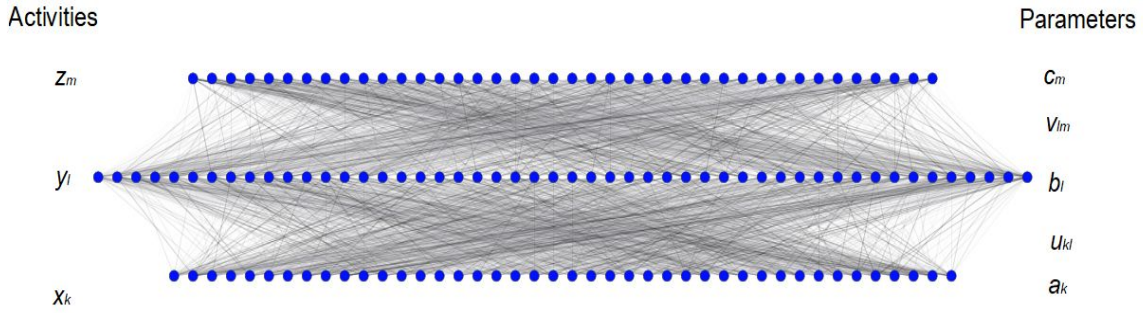

Figure 2: A visual representation of the entire network.

Initial settling

The activity of the middle layer (y_l) is determined by holding the activities of the input (x_k) and output nodes (z_m) constant, and allowing activities to spread to the middle layer. Here, k and m represent the activation of a single node in the input and output respectively while K and M are the total number of nodes in the input and output layers, 42 and 40 respectively. This is shown in the following equation. In this instance, σ represents the standard logistic function, and the output of σ is a probability representing the likelihood of node l being activated

$$y_l \leftarrow \sigma(b_l + \Sigma_{k=1}^{K} x_k u_{kl} + \Sigma_{m=1}^{M} v_{lm} z_m) \text{ , where } \sigma(x) := 1/(1 + exp(-x))$$

After this, the network resonates until it reaches a near-final state. Resonation generates activation values for every node in the output layer (z_m) while still holding input layer activities (x_k) constant. Here, l represents the activation of a single node in the middle layer while L represents the total of the middle layer nodes, 50.

$$z_m \leftarrow \sigma(c_m + \Sigma_{l=1}^{L} y_1 v_{lm})$$

This entire process of initial settling is repeated ten times to bring the network to a state of near equilibrium (Boersma, 2019).

Hebbian Learning

In the Hebbian stage, biases ($\boldsymbol{a_k}$, $\boldsymbol{b_l}$, $\boldsymbol{c_m}$) and all existing connections between active nodes are strengthened ($\boldsymbol{u_{kl}}$, $\boldsymbol{v_{lm}}$) to ensure that co-activated nodes associate with each other and fire together. Here, η represents the learning rate of 0.001 (Boersma, 2019).

$$\begin{aligned}
a_k &\leftarrow a_k + \eta x_k \\
b_l &\leftarrow b_l + \eta y_l \\
c_m &\leftarrow c_m + \eta z_m \\
u_{kl} &\leftarrow u_{kl} + \eta x_k y_l \\
v_{lm} &\leftarrow v_{lm} + \eta y_l z_m
\end{aligned}$$

Dreaming

Next, the network stochastically generates patterns. During the initial settling phase, following a single input into the input layer, an activity for the intermediate layer ($\boldsymbol{y_l}$) is calculated. Then $\boldsymbol{y_l}$ is used to calculate activities in the input layer ($\boldsymbol{x_k}$) Next, the output layer ($\boldsymbol{z_m}$) is stochastically determined. This stochasticity is achieved through the use of a deviate of the random Bernoulli function ($\mathcal{B}$). Finally, the new activities of the input and output layers are used to stochastically recalculate the intermediate layer. The dreaming sequence repeated ten times (Boersma, 2019).

$$\begin{aligned}
x_k &\leftarrow a_k + \Sigma_{l=1}^{L} u_{kl} y_l \\
z_m &\sim \mathcal{B}(\sigma(c_m + \Sigma_{l=1}^{L} y_1 v_{lm})) \\
y_1 &\sim \mathcal{B}(\sigma(b_l + \Sigma_{k=1}^{K} x_k u_{kl} + \Sigma_{m=1}^{M} v_{lm} z_m))
\end{aligned}$$

Anti-Hebbian Learning

The anti-Hebbian stage weakens all connections and biases.

$$\begin{aligned}
a_k &\leftarrow a_k - \eta x_k \\
b_l &\leftarrow b_l - \eta y_l \\
c_m &\leftarrow c_m - \eta z_m \\
u_{kl} &\leftarrow u_{kl} - \eta x_k y_l \\
v_{lm} &\leftarrow v_{lm} - \eta y_l z_m
\end{aligned}$$

Once the network reaches a stable state, the connection weights and the biases for the input level nodes are updated again in a second, identical anti-Hebbian stage (Boersma, 2019).

The Input Layer

The input layer was divided into four slabs of 10 consecutive nodes each to represent the visual inputs, with the remaining two nodes representing the binary spoken labels. The first slab, nodes 1 through 10, represent the legs of Plunkett et al.'s (2008) creatures. Nodes 1 and 2 correspond with Plunkett's number 1, the shortest legs, nodes 3 and 4 correspond with number 2, for slightly less short legs, and so on until nodes 9 and 10, which represent Plunket number 5 — the longest legs (fig. 1) . The mean values of the input data were centered on two peaks: the first at node 2.5 and the second at node 8.5. A peak width of 3 nodes and a standard deviation of 1 node ensured that the first peak covered nodes 1 to 4, Plunkett's numbers 1 and 2. Likewise, the second peak was distributed over nodes 7 to 10, accounting for Plunkett's values of 4 and 5. This configuration of the distributional peaks allows the network to account for variability of the input while still distinguishing between two groups necessary for category formation. Each of the other three slabs featured identical architecture and accounted for the tails, neck, and ears of the creature, with node numbers adjusted accordingly. A drawing of the first visual input slab is shown in figure 3.

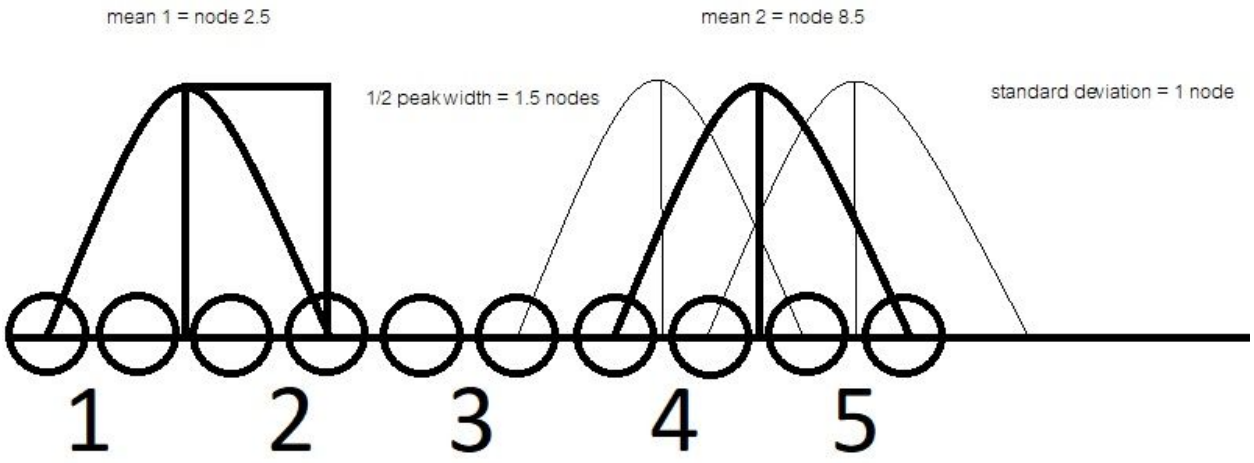

Figure 3: The first 10 visual input nodes of the network, representing the legs of Plunkett et al.'s (2008) creatures.

Nodes 41 and 42 represent the binary labels assigned to the creatures in experiments 3, 4, and 5 of Plunket et al. (2008). When no labels are present, these nodes are not activated at all. In experiments where labelling is present, they are activated with a greater activation strength than the nodes of the visual input slabs to represent the dominating influence of the audible inputs over the visual inputs. Unlike in the visual slabs, no input variability was present in the label nodes.

After the network has been trained on the input data, it is tested by passing a prototypical input in sequence with small intervals through the entirety of the input layer, and allowing the network to deterministically resonate. Then the state of the input layer is drawn. These drawings showing the end state of the input layer are the network's outputs.

Experiment 1: Broad condition

In the original experiment's broad condition, a leg value of 1 or 2 would predict an ear value of 5 or 4 respectively. Inversely, an ear value of 1 or 2 would predict a respective leg value of 5 or 4. Thus, in the network, the input activations in the leg and ear slabs were anti-correlated with each other for this experiment. For each network input, a random selection is made between the low mean and the high mean for the slab. If the low mean was chosen for the legs slab, then a high mean would be selected for the ears slab. If the high leg mean was selected for the input, then a low mean input would be provided for the ears. An identical relationship to the one just described exists between the tails and necks within the original experiment, and the model. This relationship is illustrated in figure 4, which shows the four major permutations of the input node layer for the broad condition stimuli. The two nodes representing spoken labels were not used in this condition.

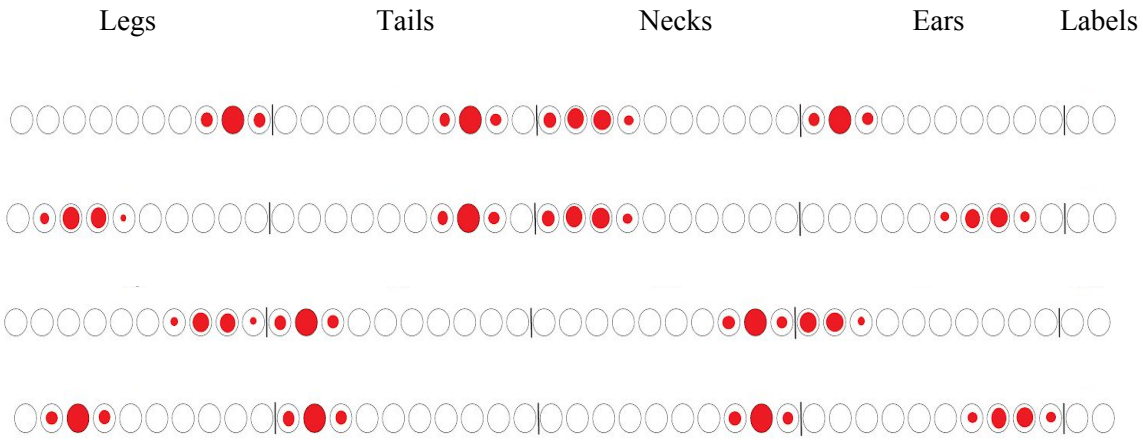

Figure 4: The four major input permutations of the broad condition.

Experiment 2: Narrow condition

The narrow condition differs from the broad condition only in that all four visual input features are correlated with one another. A randomly selected low mean input for the first slab, the legs, will lead to low mean inputs for the ears, tails, and neck for that given learning cycle. An example, along with potential input variability, is shown in figure 5. A randomly chosen high mean input for the legs slab will deterministically result in all other visual input slabs having high mean inputs. See figure 6 for an example. This mirrors the line drawings in figure 1, which led to the formation of two distinct categories when the infants were presented with the narrow condition stimuli. There, each creature has feature values that are either entirely low, 1 or 2, or entirely high, 3 or 4.

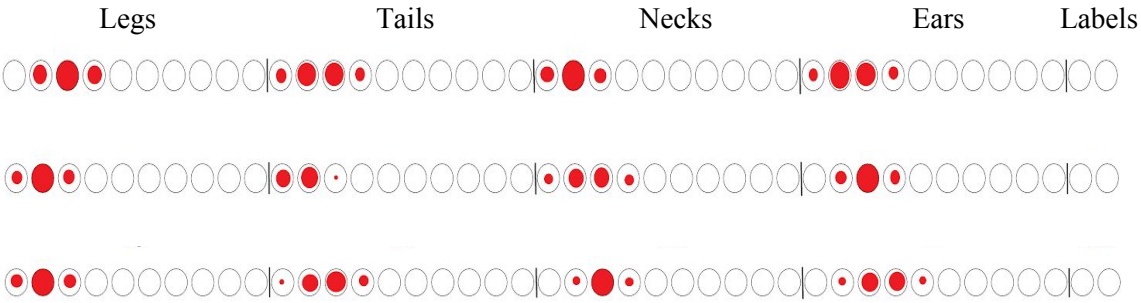

Figure 5: Input layer activation patterns for the narrow condition's first archetype. The three images show possible input variability that could occur in three separate learning cycles.

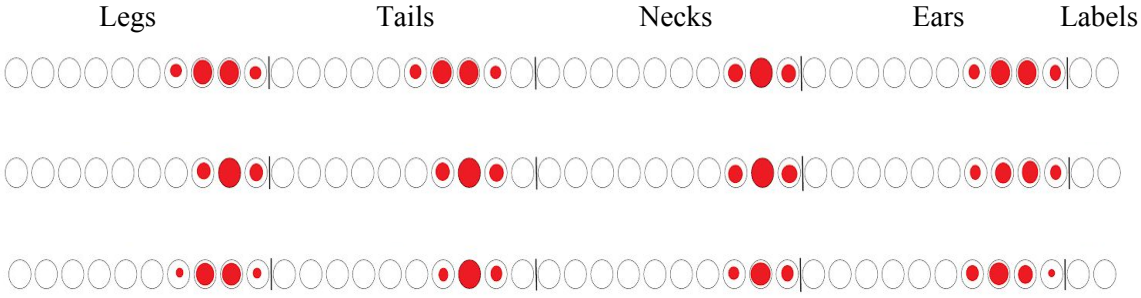

Figure 6: Input layer activation patterns for the narrow condition's second archetype with examples of possible variability.

Experiment 3: Narrow condition with congruent labels

The visual inputs, represented by the first 40 nodes, were identical to experiment 2. Nodes 41 and 42, the nodes representing the binary labels, were utilized in this experiment. The binary label nodes were activated in the same correlational manner as the visual inputs in the narrow condition. If the lower mean value was randomly selected for the leg slab, then node 41 was activated for that input (fig. 7). Conversely, if the higher mean was selected for the leg slab, then node 42 was activated (fig. 8). In simpler terms, when the network generated an input like that on the left side of the narrow condition continuum (fig. 1), node 41 was activated. Node 42 was activated when the network generated an input like that on the right half of figure 1's narrow continuum. This mirrors Plunkett et al.'s (2008) experiment where the spoken category labels given to the children corresponded with the visual categories that the children formed on the basis of the stimuli (fig. 1).

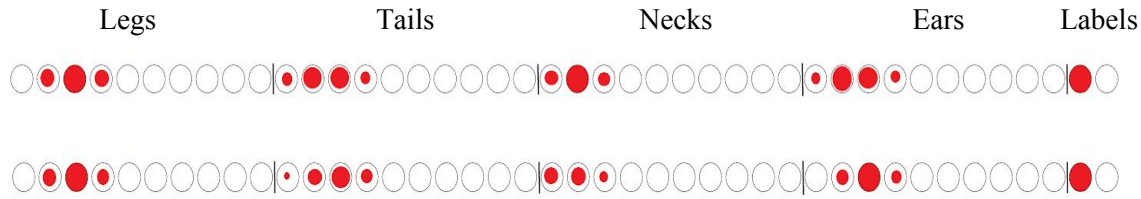

Figure 7: Possible input layer activation patterns for the narrow condition's first archetype with a matching label.

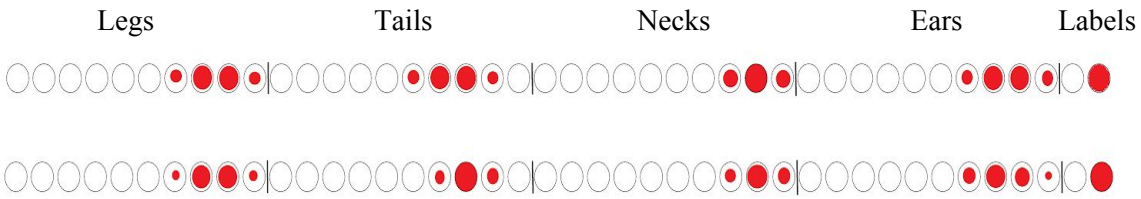

Figure 8: Possible input layer activation patterns for the narrow condition's second archetype with a matching label.

Experiment 4: Narrow condition with a single label

In experiment 4, label node 41 was activated for every input while label node 42 was never activated (figs. 9 and 10). Experiment 4 was identical to experiment 3 in all other aspects. In Plunkett et al. (2008), a single, unchanging, spoken label was given to every stimuli.

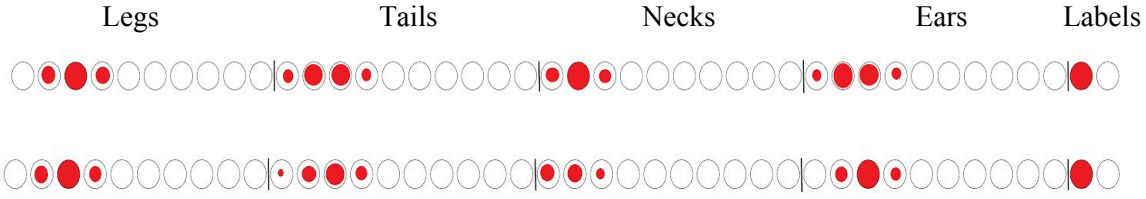

Figure 9: Potential input layer activation patterns for the narrow condition's first archetype with a single, unchanging label for each input.

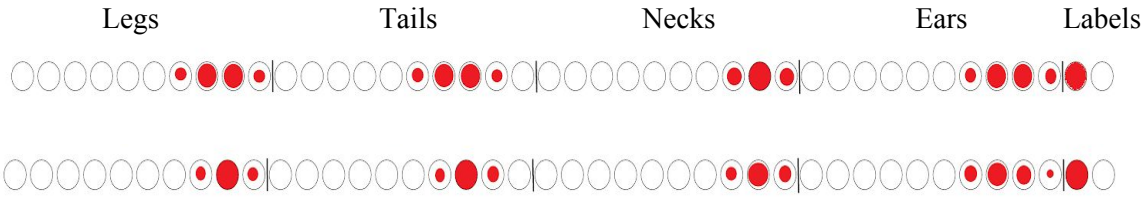

Figure 10: Potential input layer activation patterns for the narrow condition's second archetype with a single, unchanging label for each input.

Experiment 5: Narrow condition with random labels

One of the two binary label nodes were randomly activated during each input sequence (figs. 11 and 12). Otherwise, experiment 5 was identical to experiments 3 and 4. The original study gave the children one of the two labels at random for each stimuli, with no correlation between the visual stimuli and the label that was given (Plunkett et al., 2008).

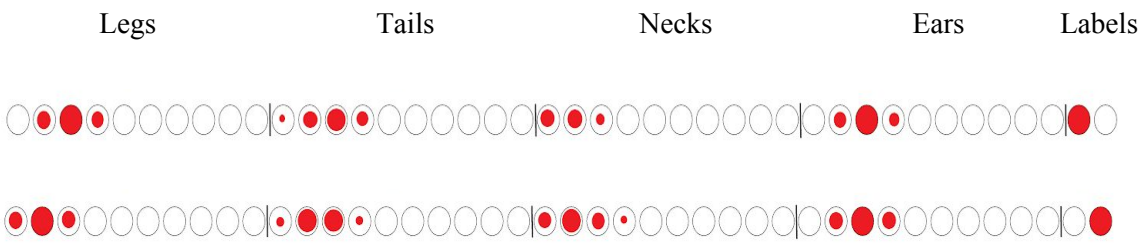

Figure 11: Input layer activation pattern for the narrow condition's first archetype with a single, random label for each input. Each label had an equal probability of being the sole activated label node for a given input sequence.

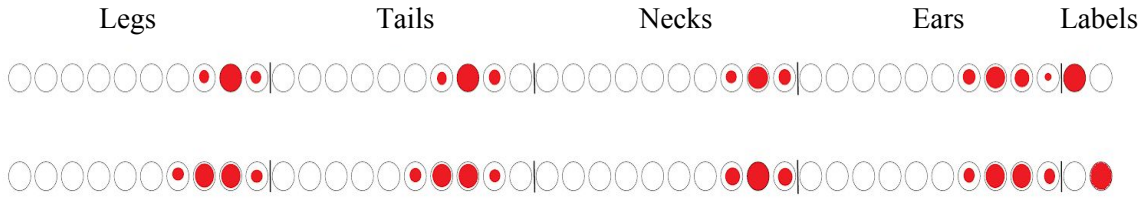

Figure 12: Input layer activation pattern for the narrow condition's second archetype with a single, random label for each input. Each label had an equal probability of being the sole activated label node for a given input sequence.

Results

Testing the Network

The outputs from the experiments are shown in the following subsections (figs. 16 - 20). The Y axis, ranging from 1 to 42, represents the nodes of the input layer, while the X axis represents the sequential activation of individual input layer nodes. The leftmost point on the X axis is the activation of the first node and the rightmost point shows the activation of the 42nd input node. Darkened areas indicate high levels of activation, while white space indicates an absence, or trace levels, of node activity. These darkened regions of activation are also representative of category formation within a given input slab, and the network as a whole. Each output image represents the state of the input layer.

Figure 13 shows the input sweep, where each node is scanned in a gradient sequence subdivided into 407 steps. These 407 steps are actually 407 prototypical inputs given at evenly spaced intervals along the input layer. Each of the network's output plots (figs. 13-20) can therefore be subdivided into 407 vertical columns, with each column representing a testing input that activates a small slice of the input layer. Since each prototypical test input occurs at a certain place along the input continuum, associated nodes within that vertical slice will be activated as well. These patterns of activation were learned during the training phase. While this is happening, the middle and upper layers are allowed to activate freely. The network is then allowed to resonate deterministically and reflect upon the prototypical inputs before generating an output. Mid and output layer nodes that co-activate with the input nodes are activated as well, but are not relevant for demonstrating categorization, so they are omitted from the generated output drawings. This procedure tests the network's learning of the training inputs. In figure 13's sweep, resonance has yet to occur, therefore the training patterns, with their activations and co-activations of associated nodes, have no bearing on this output and only the sequential activation of input layer nodes is shown. Examples of post resonance outputs can be seen in figures 14 - 20.

If too many training inputs are given, the network's output (fig. 14) begins to resemble the input sweep of figure 13, as seen in the jagged diagonal activation band running from node 1 to 42. Again, these outputs show the state of the input layer. Figure 14, and the remaining outputs in the results section (figs. 15 - 20) are post-resonance outputs where the network has been allowed to reflect upon the prototypical inputs in the testing phase. In figure 14, categorical behavior has broken down and the network is beginning to unlearn categories. To see this contrast, notice that categorical behavior can be seen clearly in figure 18 where there is distinct movement along the central diagonal showing the sequential activation of each input node and its correlates, and by the dark checkered boxes along this diagonal which remain distinct from each other, and do not merge. This is not so in figure 14 where there is a meshing of feedback from the visual prototypical test inputs. Eventually, with enough inputs, the network's output will come to nearly match the form of the input sweep (fig. 13).

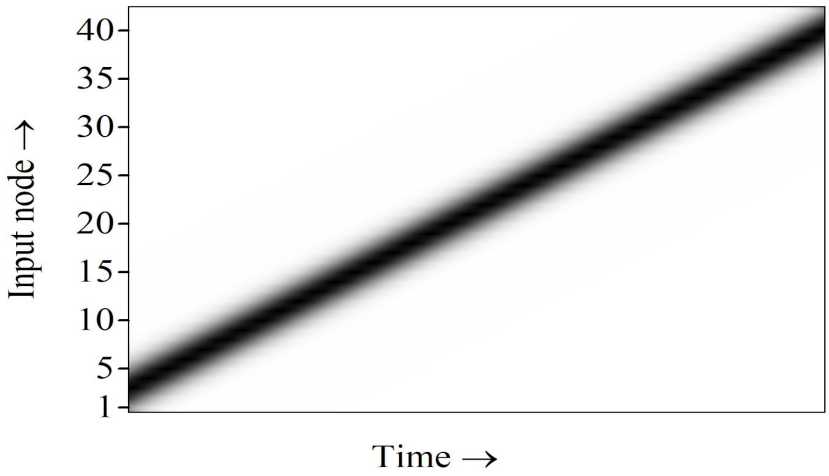

Figure 13: Input sweep with zero training inputs

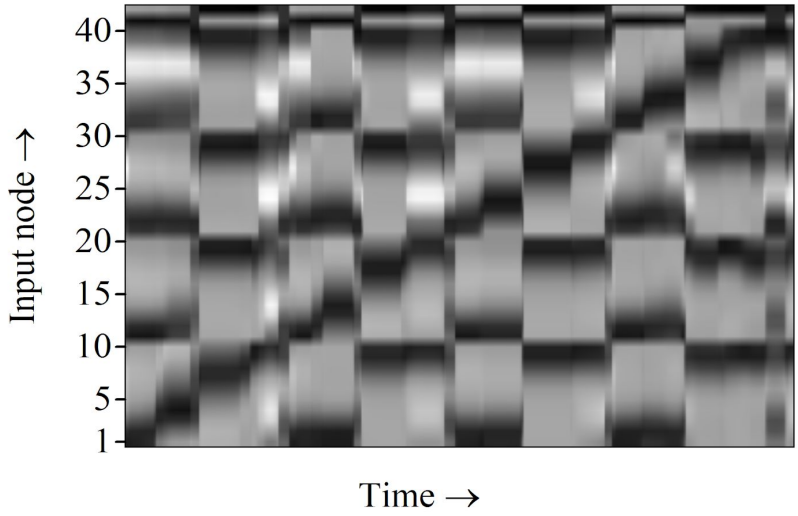

Figure 14: Output of the narrow condition with congruent labels after 30,000 inputs

If there are too few inputs then category formation cannot occur. Figure 15 shows the network after only 30 inputs. We can see that the network is registering the various entities of the inputs, but without drawing the correlations necessary to form categories. All experimental outputs (figs. 16 - 20) were the result of 3,000 inputs, although stable categorization does occur in as few as 400 inputs.

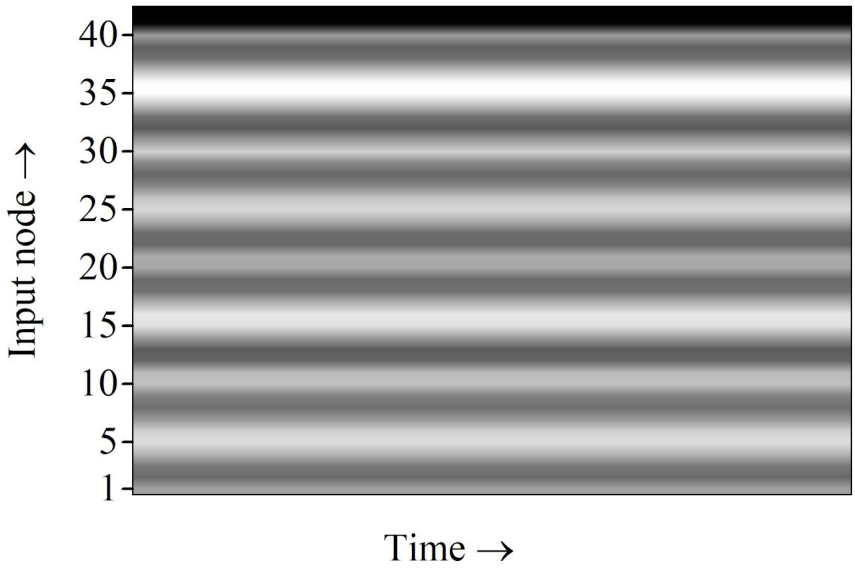

Figure 15: Output of the narrow condition with congruent labels after 30 inputs

Very faintly shaded categorization bands and the stretching of categorization bands on the right edge of the output visualizations, seen most clearly in figure 16 below, are remnants from the stochastic patterns generated during the dreaming phase.

Experiment 1: Broad condition

The dark bands in figure 16 replicate the conditions of Plunkett et al's (2008) broad condition. Slab 1 (nodes 1-10), representing the creature's legs in the visual inputs, and slab 2 (nodes 11-20), which represents the tails, are independent of each other. Thus, an activation of nodes 1-5, very short or short legs, will activate simultaneously with nodes 11-15 (a small or very small tail) or 16-20 (a large or very large tail), with no influence of one slab upon the other. The first slab of visual input nodes is anti-correlated with the nodes of slab 4 (nodes 31-40), which represents the ears. An activation of nodes 1-5, or short legs, will always activate nodes 36-40 — very spread ears or spread ears. Following the same logic, an activation of nodes 6-10 will invariably activate nodes 31-35; i.e. long legs will always predict bunched ears. The same anti-correlational relationship exists between slabs 2 and 3 (neck length). An activation in the region of nodes 11-15, a small tail, will activate nodes 26-30, a long neck. An activation of nodes 16-20, a large tail, predicts an activation of nodes 21-25, a short neck. The network's output aligns with the features of the broad condition drawings (fig. 1), which led the infant participants to form a single general category centered on Plunkett values 3333. The empty space at the top of figure 3 are nodes 41 and 42, the binary spoken labels. Labels were not used in experiment 1 or 2, so there are no activations in that region.

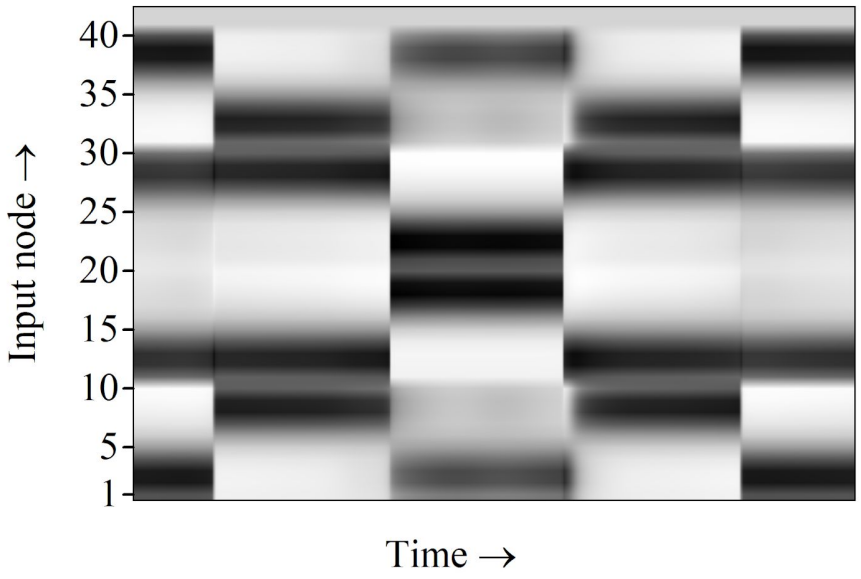

Figure 16: Broad condition output

Experiment 2: Narrow condition

The network conditions for the narrow condition were mechanically similar to experiment 1, except the activations in each slab were correlated with the first slab, nodes 1 to 10. If slab 1 received an input in nodes 1 to 5, the other slabs received inputs in their respective lower halves. In plain terms, a visual input of a creature with short legs would generate a small tail, a short neck, and bunched ears for the other features. A visual input of long legs would create a creature with long legs, a large tail, a long neck, and spread ears. This relationship is clearly shown in the output (fig. 17). At any given time, only the first or the second half of the nodes in every visual input slab are activated. In Plunkett et al.'s (2008) experiment, every narrow condition creature had Plunkett feature values that were all 1 or 2 (a dark band in the first five nodes of a slab), or which were all 4 or 5 (a dark band in the last five nodes of a slab), leading the infants to form two perceptual categories. This network adequately replicates their input and generates two sets of categories, as seen by the two differing patterns in the columns of figure 17. The first category, shown in the leftmost column and every second column thereafter, represents a short legged, small tailed, short necked, and bunched eared creature. The

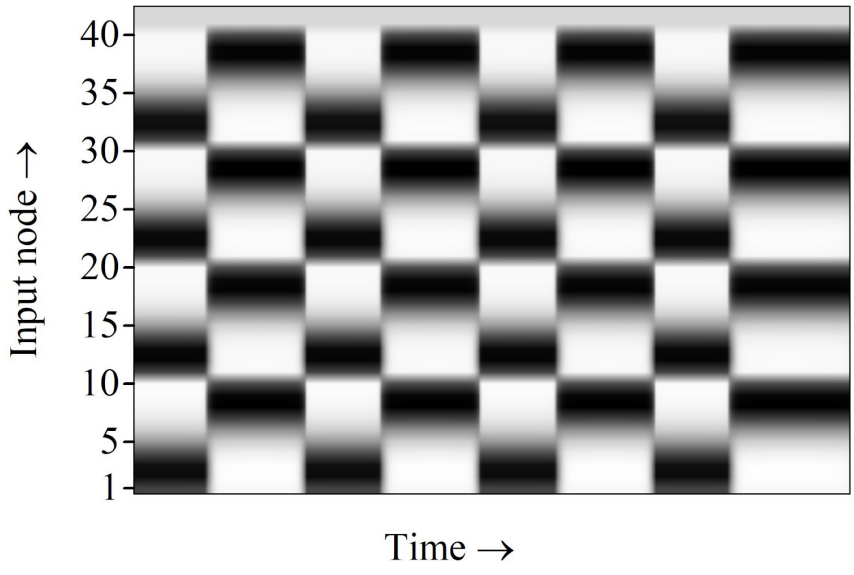

Figure 17: Narrow condition output

second category, present in the second column from the left and every second column thereafter, represents a long legged, large tailed, long necked, and spread eared creature. These match exactly with the two perceptual categories from Plunket et al.'s experiment (fig. 1).

Experiment 3: Narrow condition with congruent labels

The results of experiment 3 (fig. 18) were identical to experiment 2 for the 40 visual input nodes. Experiment 3 introduced the binary spoken labels, represented by nodes 41 and 42. In this experiment, the binary labels followed the same activation pattern as the visual input nodes, which produced the same checkerboard pattern in nodes 41 and 42 that was present in the 40 visual nodes. In this instance, there is no difference between the perception of visual categories and the perception of audible categories, matching the findings of Plunkett et al. (2008). The activation band is intentionally darker for the label nodes because they were programmed to receive higher activity levels than the visual nodes in order to mimic the overriding effects of the spoke inputs upon the visual inputs. The label nodes received activations valued at 5.0 while the visual input nodes received activation values of 4.5 at their respective peaks. This is in line with the results of Plunkett et al. (2008), which found that congruent spoken labels had no impact on visual category formation in the infant test subjects.

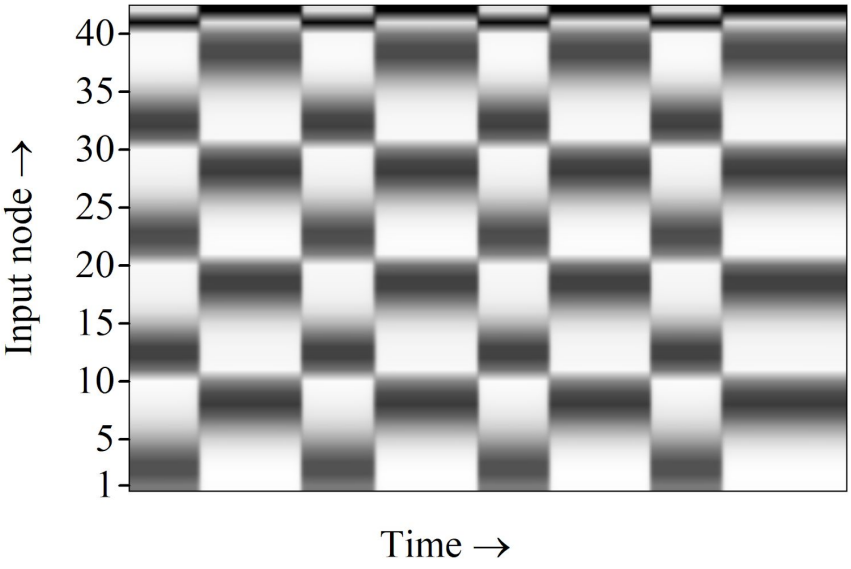

Figure 18: Narrow condition with congruent labels output

Experiment 4: Narrow condition with a single label

Experiment 4, the output of which is shown in figure 19, was identical to experiment 3 except that a single binary spoken label node was activated for every input. This is shown by the dark activation band at node 41 extending across the entire width of the x-axis. The activity level in node 41 is greater than the activity levels in the 40 visual input nodes, indicating that the single category from the spoken label is the preferred output. Plunket et al.'s (2008) findings that a single spoken label led to a single category, despite presence of two visual categories, were not replicated here because the presence of the single label node had no discernible influence on the output of the 40 visual input nodes.

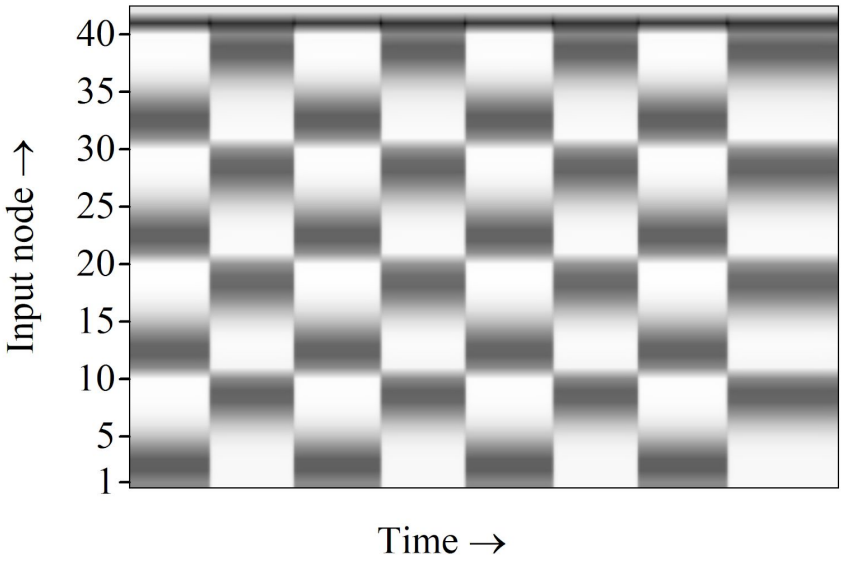

Figure 19: Narrow condition with a single label output

Experiment 5: Narrow condition with random labels

The fifth experiment, with the output shown in figure 20, was nearly identical to experiments 3 and 4. In experiment 5, one randomly selected label was activated for each visual input. This produced a thick, continuous activation band in nodes 41 and 42 with no discernible pattern of activation. Plunkett et al. (2008) found that random label assignments interfered with the infant's perception of categories, and led to the infants forming no categories. As in experiment 4, the activations of the label nodes had no influence on the outputs of the 40 visual nodes, and thus the results from Plunkett et al. (2008) were not replicated.

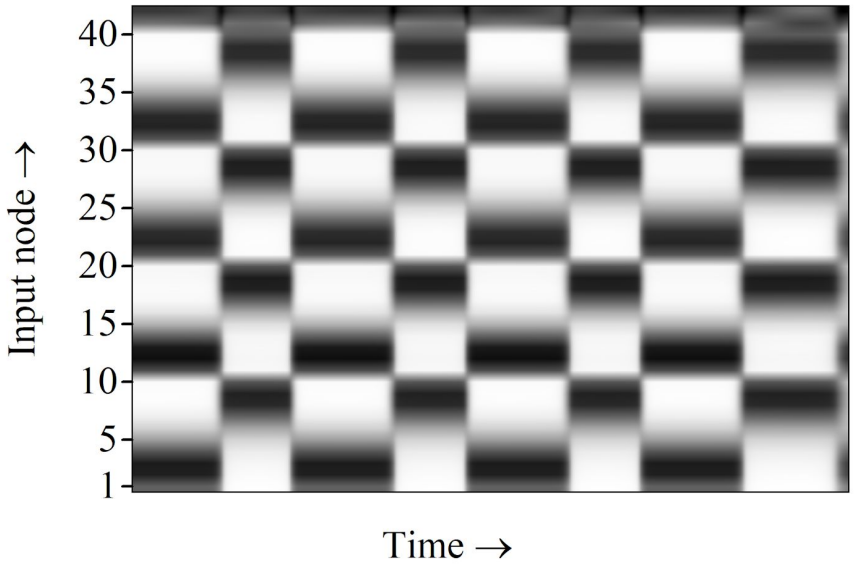

Figure 20: Narrow condition with random labels output

Conclusion and Discussion

The attempt to replicate the results of each of Plunkett et al.'s (2008) experiments with a restricted Boltzmann machine was only partially successful. The broad condition, narrow condition, and narrow condition with congruent labels were successfully modelled. The final two experiments with their novel stimuli, the narrow condition with a single label and with a random label, were not successfully modelled. The label nodes and their deviant activation patterns had no influence over the formation of visual categories. This model also did not take into account infant looking time, which was accounted for in the successful modelling of Plunkett et al. (2008) by Gliozzi et al. (2008). Gliozzi et al's (2008) SOM model was also able to incorporate one-shot learning to model the presentation of Plunkett et al.'s (2008) stimuli to the infant participants. One-shot learning has proven to be extremely difficult to achieve with Boltzmann machines. For example, this paper's mechanical Boltzmann machine children require approximately 400 inputs to form categories, whereas the human infants required only a single viewing of each stimuli.

Figure 14 is perhaps the most linguistically interesting and plausible of this network's outputs. If we trace the progression of the diagonal testing sweep from beginning to end, we see the visual representations activating the spoken labels, and the visual inputs. In this regard, the network has convincingly imitated natural speech, albeit in a simplistic manner. The output of figure 14 is the result of overtraining the network, as mentioned briefly in the results section. Overtraining breaks down the strictly defined categories within the network's code, allowing them to distort, mesh, and merge while retaining the same general features of a network that has been trained on a more *ideal* number of training inputs. Once again, this mimics real speakers. As we know, human linguistic categories, whether they be vowel sounds, associated lexical items, or anything else, are rarely so neatly delimited. There is often ambiguity when we speak that is resolved by context (McGurk & MacDonald, 1976). In that sense, the output of the overtrained network shows these grey areas, both figuratively and literally. Essentially, the network described in this paper is a simple brain featuring a rudimentary auditory cortex and a visual cortex, and possesses the ability to categorize a limited set of objects and speech sounds in its environment based on statistical regularities of the input. Functionally, this network is very similar to its immediate predecessor created by Boersma (2019), which created vowel categories based on *vibrations* in a simulated basilar membrane.

There are several ways in which this network could be improved. It was previously discussed that the network was unable to reproduce Plunkett et al's (2008) results from the experiments with randomized labels and the single label because the visual categorization was not affected by the novel label schemes. A possible solution, suggested by Boersma, is to use three label nodes while maintaining the binary labels. The newly added node could freely associate with either label in the hope that it would merge with one and add a greater net activity and then, hopefully, influencing the visual section of the input continuum to organize itself based on the inputs fed to the label nodes, rather than their own nodes. Another solution that could be used independently or in conjunction with the addition of a third label node, is to train the network on other categorization tasks beforehand. The children in Plunkett's (2008) study were 11 months old, with all the language inputs and experience that comes with that. They were not the blank slate, artificial children created by this network to analyze statistical distributions of inputs. By giving the network more and diverse categorization tasks beforehand, it may come to more closely resemble the categorization behavior of real children. Last, is the problem of one-shot learning. The

infants were able to reliably form categories after a single exposure to each line drawing, but the Boltzmann machine requires several hundred inputs for categories to form. If a novel method could be devised for one-shot learning in a Boltzmann machine, it would improve the authenticity of the neural network's simulation, and would likely have further and greater implications for the field of machine learning. However, at present, there are no suggestions for how to accomplish this.

Citations

Boersma, P. (2019). *Simulated distributional learning in deep Boltzmann machines leads to the emergence of discrete categories*. Proceedings of the 19th International Congress of Phonetic Sciences, 1520-1524. Melbourne, Australia.

Boersma, P. & Weenink, D. (2020). *Praat: doing phonetics by computer* [Computer program]. Version 6.1.10, retrieved 25 March 2020 from http://www.praat.org/

Boersma, P., Benders, T. & Seinhorst, K. (2020). *Neural networks for phonology and phonetics*. Manuscript, University of Amsterdam

Gliozzi, V., Mayor, J., Hu, J., & Plunkett, K. (2008). *The Impact of Labels on Visual Categorization: a Neural Network Model*. Proceedings of the Annual Meeting of the Cognitive Science Society, 30(30).

Gustafsson, L., Jantvik, T. & Paplinski, A. (2014). *A Self-organized artificial neural network architecture that generates the McGurk effect*. Proceedings of the International Joint Conference on Neural Networks, 3974-3980.

McGurk, H. and MacDonald, J. (1976). *Hearing lips and seeing voices*. Nature, 264(5588), 746–748.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. (2011). *Multimodal Deep Learning*. Proceedings of the 28th International Conference on Machine Learning. Bellevue, WA, United States of America.

Plunkett, K., Hu, J., & Cohen, L. (2008). *Labels can override perceptual categories in early infancy*. Cognition 106(2), 665-81.

Appendix

```
#
#Paul Boersma and Zackary Gibson (2020)
#

form Sweep categories in a deep-belief network
	integer Number_of_data 0 (= input sweep)
	choice experiment: 3
		button broad condition
		button narrow condition
			button narrow condition with congruent labels
			button narrow condition with single label
			button narrow condition with random label
endform

numberOfInputNodes = 42
numberOfMiddleNodes = 80
numberOfOutputNodes = 50
sigma = 1.0
peakWidth = 1.5
learningRate = 0.001
edge = 2
numberOfMeanFieldEchoes = 10
numberOfGibbsEchoes = 10
semf.offsetNode = 0

numberOfLegs = 2
		legMean [1] = 2.5
		legMean [2] = 8.5
numberOfTails = 2
		tailMean [1] = 12.5
		tailMean [2] = 18.5
numberOfNecks = 2
		neckMean [1] = 22.5
		neckMean [2] = 28.5
numberOfEars = 2
		earMean [1] = 32.5
		earMean [2] = 38.5
numberOfLabels = 2
		meanLabel [1] = 41.0
		meanLabel [2] = 42.0
```

```
x# = zero# (numberOfInputNodes)
y# = zero# (numberOfMiddleNodes)
z# = zero# (numberOfOutputNodes)
a# = x#
b# = y#
c# = z#
u## = zero## (numberOfInputNodes, numberOfMiddleNodes)
v## = zero## (numberOfMiddleNodes, numberOfOutputNodes)

#
# Spread up
#

procedure setFourInputs: .formantLegs, .formantTails, .formantNecks, .formantEars
	x# ~ if col >=1 and col <=10 then 5 * exp (-0.5 * ((col - .formantLegs) / peakWidth) ^ 2) - 0.5
else self fi
	x# ~ if col >=11 and col <=20 then 5 * exp (-0.5 * ((col - .formantTails) / peakWidth) ^ 2) - 0.5
else self fi
	x# ~ if col >=21 and col <=30 then 5 * exp (-0.5 * ((col - .formantNecks) / peakWidth) ^ 2) - 0.5
else self fi
	x# ~ if col >=31 and col <=40 then 5 * exp (-0.5 * ((col - .formantEars) / peakWidth) ^ 2) - 0.5
else self fi
endproc

procedure setFiveInputs: .formantLegs, .formantTails, .formantNecks, .formantEars, .label
	x# ~ if col >=1 and col <=10 then 5 * exp (-0.5 * ((col - .formantLegs) / peakWidth) ^ 2) - 0.5
else self fi
	x# ~ if col >=11 and col <=20 then 5 * exp (-0.5 * ((col - .formantTails) / peakWidth) ^ 2) - 0.5
else self fi
	x# ~ if col >=21 and col <=30 then 5 * exp (-0.5 * ((col - .formantNecks) / peakWidth) ^ 2) - 0.5
else self fi
	x# ~ if col >=31 and col <=40 then 5 * exp (-0.5 * ((col - .formantEars) / peakWidth) ^ 2) - 0.5
else self fi
	x# [41] = 0
	x# [42] = 0
	x# [.label] = 5
endproc

procedure spreadUp
	# GBC2016: 661
	z# = zero# (numberOfOutputNodes)   ; or to random values
```

```
        for iecho to numberOfMeanFieldEchoes
                y# = sigmoid# (mul# (x#, u##) + mul# (v##, z#) + b#)
                z# = sigmoid# (mul# (y#, v##) + c#)
        endfor
endproc

procedure resonate
        for iecho to numberOfGibbsEchoes
                x# = mul# (u##, y#) + a#
                z# = randomBernoulli# (sigmoid# (mul# (y#, v##) + c#))
                y# = randomBernoulli# (sigmoid# (mul# (x#, u##) + mul# (v##, z#) + b#))
    endfor
endproc

procedure hebbianLearning: .learningRate
        a# += .learningRate * x#
        b# += .learningRate * y#
        c# += .learningRate * z#
        u## += .learningRate * outer## (x#, y#)
        v## += .learningRate * outer## (y#, z#)
endproc

procedure antiHebbianLearning: .learningRate
        @hebbianLearning: - .learningRate
endproc

Erase all
Font size: 10
oversampling = 10
numberOfTimes = (numberOfInputNodes - 5) * oversampling + 1
image = Create simple Matrix: "inputImage", numberOfInputNodes, numberOfTimes, "0"

procedure drawImage
        Select outer viewport: 0, 3.7, 0, 2.5
        selectObject: image
        Paint image: 0, 0, 0, 0, 0, 0
        One mark left: 1, "yes", "yes", "no", ""
        Marks left every: 1, 5, "yes", "yes", "no"
        if number_of_data = 300
                White
        endif
        Select outer viewport: 0.1, 3.3, 0.1, 2.4
        Text bottom: "yes", "Time \->"
```

```
        Black
        Text left: "yes", "Input node \->"
        Select outer viewport: 0.1, 3.3, 0.3, 2.4
        Remove
endproc

if number_of_data = 0
        for itime to numberOfTimes
                formant = 1 + edge + (itime - 1) / oversampling
                @spreadUp: formant
                selectObject: image
                Formula: ~ if col = itime then x# [row] else self fi
        endfor
        @drawImage
    exitScript ()
endif

#
# Train the network.
#
if experiment = 1
        for ipattern to number_of_data
                legs = randomInteger (1, 2)
                tails = randomInteger (1, 2)
                necks = if tails = 1 then 2 else 1 fi
                ears = if legs = 1 then 2 else 1 fi

                formantLegs = randomGauss (legMean [legs], sigma)
                formantTails = randomGauss (tailMean [tails], sigma)
                formantNecks = randomGauss (neckMean [necks], sigma)
                formantEars = randomGauss (earMean [ears], sigma)

                @setFourInputs: formantLegs, formantTails, formantNecks, formantEars
                @spreadUp
                @hebbianLearning: learningRate
                @resonate
                @antiHebbianLearning: learningRate
                endfor
endif
if experiment = 2
        for ipattern to number_of_data
                legs = randomInteger (1, 2)
                tails = if legs = 1 then 1 else 2 fi
```

```
                necks = if legs = 1 then 1 else 2 fi
                ears = if legs = 1 then 1 else 2 fi

                formantLegs = randomGauss (legMean [legs], sigma)
                formantTails = randomGauss (tailMean [tails], sigma)
                formantNecks = randomGauss (neckMean [necks], sigma)
                formantEars = randomGauss (earMean [ears], sigma)

                @setFourInputs: formantLegs, formantTails, formantNecks, formantEars
                @spreadUp
                @hebbianLearning: learningRate
                @resonate
                @antiHebbianLearning: learningRate
        endfor
endif
if experiment = 3
        for ipattern to number_of_data
                legs = randomInteger (1, 2)
                tails = if legs = 1 then 1 else 2 fi
                necks = if legs = 1 then 1 else 2 fi
                ears = if legs = 1 then 1 else 2 fi

                formantLegs = randomGauss (legMean [legs], sigma)
                formantTails = randomGauss (tailMean [tails], sigma)
                formantNecks = randomGauss (neckMean [necks], sigma)
                formantEars = randomGauss (earMean [ears], sigma)
                formantCongruentLabel = if legs = 1 then meanLabel [1] else meanLabel [2] fi

                @setFiveInputs: formantLegs, formantTails, formantNecks, formantEars,
formantCongruentLabel
                @spreadUp
                @hebbianLearning: learningRate
                @resonate
                @antiHebbianLearning: learningRate
        endfor
endif
if experiment = 4
        for ipattern to number_of_data
                legs = randomInteger (1, 2)
                tails = if legs = 1 then 1 else 2 fi
                necks = if legs = 1 then 1 else 2 fi
                ears = if legs = 1 then 1 else 2 fi
```

```
                formantLegs = randomGauss (legMean [legs], sigma)
                formantTails = randomGauss (tailMean [tails], sigma)
                formantNecks = randomGauss (neckMean [necks], sigma)
                formantEars = randomGauss (earMean [ears], sigma)
                formantSingleLabel = meanLabel [1]

                @setFiveInputs: formantLegs, formantTails, formantNecks, formantEars,
formantSingleLabel
                @spreadUp
                @hebbianLearning: learningRate
                @resonate
                @antiHebbianLearning: learningRate
        endfor
endif
if experiment = 5
        for ipattern to number_of_data
                legs = randomInteger (1, 2)
                tails = if legs = 1 then 1 else 2 fi
                necks = if legs = 1 then 1 else 2 fi
                ears = if legs = 1 then 1 else 2 fi

                formantLegs = randomGauss (legMean [legs], sigma)
                formantTails = randomGauss (tailMean [tails], sigma)
                formantNecks = randomGauss (neckMean [necks], sigma)
                formantEars = randomGauss (earMean [ears], sigma)
                formantRandomLabel = randomInteger (41.0, 42.0)

                @setFiveInputs: formantLegs, formantTails, formantNecks, formantEars,
formantRandomLabel
                @spreadUp
                @hebbianLearning: learningRate
                @resonate
                @antiHebbianLearning: learningRate
        endfor
endif

#
# Test the network.
#
for itime to numberOfTimes
        #
        #initializing x
        #
```

```
        formant = 1 + (itime - 1) / (numberOfTimes - 1) * (numberOfInputNodes - 1)
        x# ~ 5* exp (-0.5 * ((col - formant) / peakWidth) ^ 2) - 0.5
        @spreadUp
        for iecho to numberOfGibbsEchoes
                x# = mul# (u##, y#) + a#
                z# = sigmoid# (mul# (y#, v##) + c#)
                y# = sigmoid# (mul# (x#, u##) + mul# (v##, z#) + b#)
        endfor
        selectObject: image
        Formula: ~ if col = itime then x# [row] else self fi
endfor

@drawImage
```