From Size Perception to Counting: An Evolutionary Computation Point of View

Gali Barabash Katz
Dept. of Cognitive Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
barabash@post.bgu.ac.il

Amit Benbassat
Dept. of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
amitbenb@cs.bgu.ac.il

Liana Diesendruck
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Illinois, USA
ldiesend@illinois.edu

Moshe Sipper
Dept. of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
sipper@cs.bgu.ac.il

Avishai Henik
Dept. of Psychology
Zlotowski Center for Neuroscience
Ben-Gurion University of the Negev
Beer-Sheva, Israel
henik@bgu.ac.il

ABSTRACT
The ability to perceive size is shared by humans and animals. Babies present this basic ability from birth, and it improves with age. Counting, on the other hand, is a more complex task than size perception. We examined the theory that the counting system evolved from a more primitive system of size perception (the leading alternative being that the two systems evolved separately). By using evolutionary computation techniques, we generated artificial neural networks (ANNs) that excelled in size perception and presented a significant advantage in evolving the ability to count over those that evolved this ability from scratch. This advantage was observed also when evolving from ANNs that master other simple classification tasks. We also show that ANNs who train to perceive size of continuous stimuli present better counting skills than those that train with discrete stimuli.

Categories and Subject Descriptors
I.2.0 [**Artificial Intelligence**]: General - Cognitive simulation; I.2.6 [**Artificial Intelligence**]: Learning - Connectionism and neural nets, Knowledge acquisition

General Terms
Algorithms, Theory

Keywords
Numerical Cognition, Size Perception, Genetic Algorithms, Neural networks, NEAT

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

1. INTRODUCTION
The field of numerical cognition encompasses many aspects of numerical processing including the processing of symbolic stimuli (e.g., Arabic numbers) versus non-symbolic ones (e.g., an array of dots) and the processing of discrete stimuli versus continuous ones. Within this field, we focus on the differences between size perception and actual counting. The ability to perceive size (i.e., to estimate the area of a blob) and to count (i.e., how many blobs are presented) is shared by humans [1] and animals [2].

It has been proposed that the basic numerical intuitions (e.g., "the number sense") [4] are supported by an evolutionarily, ancient, approximate number system (ANS) [3] which enables the representation of approximate number of items in visual or auditory arrays without verbally counting [5]. This core system for numerical processing might be the root for high-level human numerical abilities, such as arithmetic [4].

We consider two potential hypotheses regarding the development of the counting system. One is that the counting system has evolved from a more primitive system, designed to perceive and evaluate size or amount of substance [3, 6]. The other is that size perception and counting evolved independently, most likely in different epochs of time [6].

1.1 The Current Study
In order to test our hypothesis (i.e., the counting system evolved from a more primitive system of size perception) we used genetic algorithms. We first evolved ANNs that can perceive size successfully and then further evolved these same networks to count. Our main goal was to examine if these ANNs have an advantage in evolving the ability to count over new learners (i.e., ANNs that did not first evolve to perceive size).

We used NEAT[1], a method for evolving artificial neural networks using genetic algorithms, in order to evolve ANNs that could perceive size and count. NEAT simulates evolution by starting

[1] *NeuroEvolution of Augmenting Topologies*. We used the NEAT4J Java implementation.

with small, simple networks which become increasingly complex through evolution [9]. The complexity of the final networks were analyzed by examining the number of inner nodes added to the networks as the task became more complex.

2. METHODS

2.1 Stimuli

We used a 2 by 4 two-dimensional Boolean array to represent the visual input. A total of 256 (2^8) possible stimuli can be produced by the array; some of these are discrete and some are continuous. We define a stimulus as continuous if there is a path from each visible cell in the array to every other visible cell that passes through visible cells (in single up/down/left/right steps). According to this definition, of the 256 possible stimuli, 147 are discrete and 109 are continuous (see Figure 1).

Figure 1. Examples of continuous (a) and discrete (b) 2X4 arrays.

Three sets of stimuli were generated in order to accommodate all the experiments planned in this study. The "Continuous" and the "Discrete" stimuli collections were composed each of 108 stimuli randomly divided into training and test sets of 54 stimuli[2] each (with no repetitions). A combined collection of continuous and discrete stimuli ("Both") was also generated; its training set contained a total of 128 (256/2) stimuli, while its test set included the other 128 stimuli.

Prior to being inserted into the NEAT system, the stimulus was flattened into a one-dimensional array of 0's and 1's.

2.2 Procedure

Several different types of evolutionary procedures were tested during the study. Every *evolutionary run* was performed 30 times using each of the 3 stimuli collections (*continuous*, *discrete*, and *both*), resulting in a total of 90 runs per run type.

The procedure is similar for all run types. Every run begins with a training stage, where the population is trained until its average fitness score exceeds the value 0.999[3]. Next, an evaluation of the population takes place. During this test stage, each individual is evaluated in relation to each of the tasks relevant to the fitness functions used; an additional test on counting accuracy was performed in all runs. For example, the "Size Perception" population trained to perceive size was tested on size perception but also on counting.

In order to avoid overfitting to the training set, the algorithm observes the variation of the population's fitness over time. After each generation is evaluated, the difference between the best fitness score of the current and the previous generation is calculated. Similar values for the last 100 generations are kept in the system. If the sum of these values drops below 10% improvement in fitness over 100 generations, the algorithm terminates and the best individual from that point is saved in order to be tested later.

2.3 Genetic Algorithm Parameters

In all evolutionary runs we used the following parameters

- Population size of 100 individuals
- Fitness termination condition of 0.999 or 0.998 (see Footnote 3).
- Mutation probability $P_m = 0.25$
- Crossover probability $P_c = 1$

2.4 Fitness Calculation

Each run type defines the expected output it should get from NEAT. When a result from NEAT is received, the fitness function checks if the expected output is equal to the observed one. If so, it assigns a 100% score, otherwise, it calculates a score according to the distance of each digit from the expected array to the observed one (after sorting both arrays) as follows:

$$fitness = \sum_{i=1}^{inputs} score\ , \qquad max = output.length$$

$$score = \frac{max - \sum_{i=0}^{max-1} |expected_{sorted}[i] - observed_{sorted}[i]|}{max}$$

3. SIMULATIONS

3.1 Simulation 1: From Size Perception to Counting

We created five evolutionary run types in order to test our hypothesis. The goal was to train a set of ANNs in a certain task and then switch to a different task by changing the fitness function in mid-run. The algorithm performs the switch from one fitness function to the other after a predefined number of generations chosen experimentally passes. We opted for this approach instead of waiting for the networks to excel in the tasks, in order to avoid the "overfitting" phenomenon (bloated networks with too many inner-nodes that excel specifically on the training inputs rather than solving the more general problem) in the consecutive runs. Thus, we switched from the classification tasks to more complex tasks after 25 generations (see Table 1).

Table 1. The size perception, counting and control groups genetic algorithms with their fitness functions and outputs.

#	Run type	Fitness and Output
1	Size Perception (SP)	A given input is classified as BIG if its number of ones ≥ array.length/2, and otherwise as SMALL. The same logic is applied to the output. BIG is an output with a number of ones which is ≥ [array.length/2]. SMALL is an output of a number of ones which is < [array.length/2].
2	Counting (C)	For a given input, the exact number of ones given - is expected to be in the output (without order considerations).
3	Size Perception and Counting (SP-C)	The size perception fitness function is switched to the counting fitness function in mid-run after 25 generations (the entire run takes in average 48.5 generations).

[2] This number was chosen because there were 147 discrete and 109 continuous stimuli out of 256, and we wanted to keep an equal number of stimuli in the training and test groups; thus we chose the minimal group (i.e., 109) and divide it into two equal groups of 54 stimuli (one for training and one for testing).

[3] 0.999 in Size Perception and in Counting and 0.998 in Subitizing as it was found to be sufficient in order to excel in this task in the testing stage.

4	Control 1 (C1)	The set of inputs was divided randomly into two groups, one group expected a BIG output, and the other an SMALL one, similar to SP (see Table item 1).
5	Control 1 and Counting (C1-C)	The control 1 fitness function switched to the counting fitness function after 25 generations. We expected an output of the exact number of ones, similar to C (see Table item 3).

3.1.1 Results

We conducted 3 different two-way ANOVA (Analysis of Variance), for 3 different dependent variables (counting score, generations, inner nodes). The design was between subjects (since each algorithm condition produced a different population of ANNs) of 3 (Stimuli: both/continuous/discrete) X 5 (Run: SP/C1/C/SP-C/C1-C). In the following analyses we present the planned comparisons relevant to our theory considerations, with significance level of p<.05.

3.1.1.1 Counting Score

The counting score (based on the fitness calculation in section 2.4) was higher in networks that first evolved to perceive size or to perform other classification tasks than of those that evolved to count independently (e.g., **SP-C VS C**: F (1,435) = 58.49, MSE =.0062, p<.01, η_p^2 =.118).

In addition, when the tasks were easy (i.e., single runs: **SP**, **C1**), score for discrete stimuli was higher than for continuous stimuli (e.g., **SP**: F (1,435) = 84.47, MSE = .0062, p<.01, η_p^2 =.162) but when the tasks became more complex (in **C1-C**: F (1,435) = 6.146, MSE = .0062, p<.05, η_p^2 =.0139) the opposite pattern was observed and the ANNs with continuous stimuli got better at counting (in **C** and **SP-C** the differences were not significant, see Figure 2).

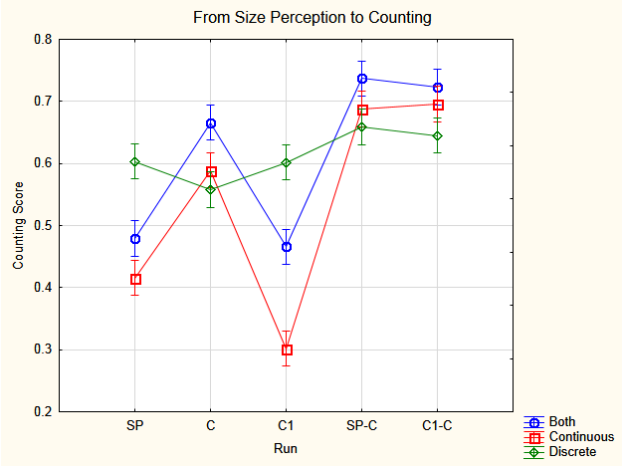

Figure 2. Counting score of simulation 1 testing sets.

3.1.1.2 Generations

SP and **C1** runs were faster than all other tasks (**SP**: M = 48.5, SD = 9.21, **C1**: M = 44.9, SD = 5.15, **C**: M = 308.22, SD = 4.32, **SP-C**: M = 330.7, SD = 16.33, **C1-C**: M = 334.7, SD = 24.33) .More generations were required to evolve the networks to count after evolving to perform any other task (**C vs. C1-C**: F (1,435) = 183.49, MSE = 172.51, p<.01, η_p^2 =.296).

3.1.1.3 Inner Nodes

Complex runs (i.e., **C**, **SP-C**, **C1-C**) generated ANNs with more inner nodes than the easier ones (**SP**, **C1**): **SP**: M = 3.64, SD = 1.64 and **SP-C**: M = 12.39, SD = 4.6, **C1**: M = 3.96, SD = 1.37 and **C1-C**: M =12.05, SD =5.04, and **C**: M = 11.81, SD = 4.71.

In addition, during the complex runs the ANNs evolved on discrete stimuli contained more inner nodes than the ones dealing with continuous ones (**C1-C**: F (1,435) = 25.25, MSE = 13.87, p<.01, η_p^2 =.05 and **SP-C**: F (1,435) = 5.55, MSE = 13.87, p<.05, η_p^2 = .002, but C was not significant).

3.2 Simulation 2: Continuous vs. Discrete

In the runs discussed in the previous subsection, we trained the ANNs on a certain stimuli type and tested on the same type (e.g., trained on continuous and tested on continuous), which seems to imply that continuous stimuli are more suitable to the task of evolving counting ability. In order to examine if this is so, or if the high score for continuous stimuli in the counting test was just due to continuous stimuli being less complex to process than discrete stimuli, we proceeded to train on continuous stimuli and test on discrete stimuli and vice versa (see Table 2).

Table 2. The genetic algorithms of the ANNs that trained on continuous stimuli and tested on discrete, and vice versa, with their fitness functions and outputs.

#	Run type	Fitness and Output
1	SP	The same as SP in Table 1 but trained on continuous stimuli and tested on discrete stimuli (and vice versa).
2	C	The same as C in Table 1 but trained on continuous stimuli and tested on discrete stimuli (and vice versa).

3.2.1 Results

The same ANOVA as in 3.2.1, with an array of 2 (Stimuli: continuous/discrete) X 2 (Run: SP/C). Similar to previous analyses, we present the planned comparisons relevant to our theory considerations, with significance level of p<.05.

3.2.1.1 Counting Score

In both selected runs in the current simulation, we received significant differences in the counting scores (based on the fitness calculation in section 2.4) when training the ANNs on continuous stimuli and tested on discrete than the other way around: F(1,116) = 6.923, MSE =.009, p <.01, η_p^2 = .056 (see Figure 3).

3.2.1.2 Generations

As expected, training ANNs on continuous stimuli and testing on discrete ones requires the same number of generations as training ANNs on discrete stimuli and testing on continuous ones (for **SP** it took about 46 generations to learn discrete stimuli and be tested on continuous and vice versa: M = 46.13, SD = 7.41, for **C**: M = 308.45, SD = 4.09).

3.2.1.3 Inner nodes

In the complex run (i.e., **C**) more inner nodes were produced when the ANNs trained on discrete stimuli (**C**: M = 15.66, SD = 5.99) than when trained on continuous stimuli (**C**: M = 11.46, SD = 3.702, F (1,116) = 9.07, MSE = 14.12, p<.05, η_p^2 = .07). In the SP run the number of inner nodes stayed the same in both stimuli types (**SP**: M = 3.3, SD = 1.84).

4. RESULTS SUMMARY

The trends seem to indicate that counting skills evolve to a higher level if ANNs are first evolved to perform another, simpler, classification task and then evolved further to count. This holds true even if the simple task is not related to size

perception/counting. In addition, training with continuous stimuli resulted in significantly better counting skills than training with discrete stimuli, despite the reasonable assumption that discrete stimuli would lend themselves better to the counting task.

According to our results it is possible that the counting system in extant animals might have evolved on the back of some primitive system, instead of being evolved independently. A certain division between continuous and discrete stimuli appears to be useful in training ANNs to improve their counting skills.

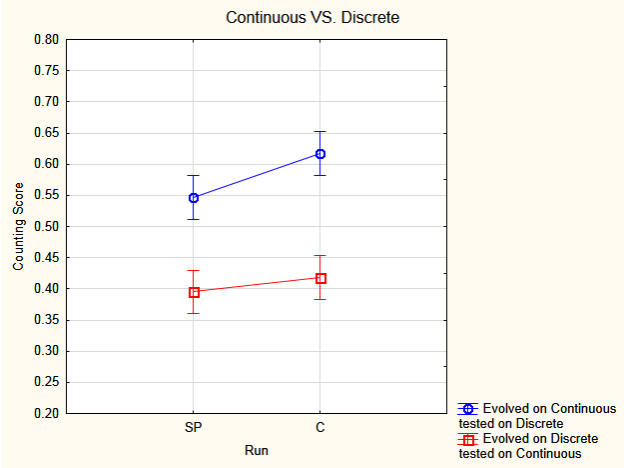

Figure 3. Counting score of simulation 3 testing sets.

5. DISCUSSION

In this research, we examined two hypotheses: (1) The counting system developed through evolution from a more primitive size perception system, and (2) the size perception and counting systems evolved independently in different periods of time. Our results indicate that processing discrete stimuli in complex tasks leads to larger and more complex networks. Interestingly, training on continuous stimuli and testing on discrete stimuli led to better counting scores than the other way around. It is possible (and should be further tested) that the successful ANNs trained on continuous stimuli developed a unique structure that is not only economical (less inner nodes) but also well organized (modular).

For example see Figures 4 and 5 that show the structures of two ANN individuals from the counting (C) runs in Simulation 2 (Both runs present the best result for their run-type). The bottom row is the input layer, the top row is the output layer, and the inner nodes are in between. Note that each individual ANN at the beginning of the evolutionary process has only 8 inputs and outputs. The hidden nodes and the network's connections are manipulated during the evolutionary process resulting in the final networks.

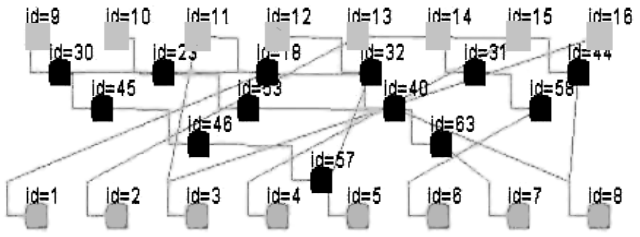

Figure 4. The structure of the best ANN individual from the C runs that trained on discrete stimuli and tested on continuous stimuli, achieving a score of 57.4 in counting.

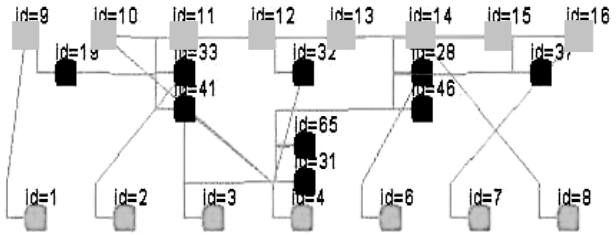

Figure 5. The structure of the best ANN individual from the C runs that trained on continuous stimuli and tested on discrete stimuli, achieving a score of 98.1 in counting. In this case, there was no need for 8 input nodes, thus this individual survived with a mutation which removed input node #5 through evolution.

It is important to mention that the division between continuous and discrete stimuli might not reflect the reality in Nature, and that it should be tested with larger arrays of stimuli. Nevertheless, it might have implications to treatment methods later on. For example, people with dyscalculia [8] could be trained in size perception tasks on continuous stimuli possibly resulting in improved counting skills.

6. ACKNOWLEDGMENTS

We thank Desiree Meloul for her valued input to this paper.

This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement n° 295644.

Amit Benbassat is partially supported by the Lynn and William Frankel Center for Computer Sciences. This research was supported by the Israel Science Foundation (grant n° 123/11).

7. REFERENCES

[1] Brannon, E. M., Lutz, D., and Cordes, S. 2006. The development of area discrimination and its implications for number representation in infancy. *Developmental Sci.* 9, F59-F64.

[2] Cantlon, J. F., and Brannon, E. M. 2007. How much does number matter to a monkey? *J. Exp. Psychol.Anim. B.* 33, 32-41.

[3] Cantlon, J. F., Platt, M. L., and Brannon, E. M. 2009. Beyond the number domain. *Trends Cogn. Sc.* 13, 83-91.

[4] Dehaene, S. 2001. Is the number sense a patchwork? *Mem. Lang.* 16, 89–100.

[5] Halberda, J., Mazzocco, M. M. M., and Feigenson, L. 2008. Individual differences in non-verbal number acuity correlate with maths achievement. *Nature* 455(7213), 665-668.

[6] Henik, A., Leibovich, T., Naparstek, S., Diesendruck, L., and Rubinsten, O. 2012. Quantities, amounts, and the numerical core system. *Frontiers Hum. Neurosci.* 5, 186.

[7] Kashtan, N., Noor, E., and Alon, U. 2007. Varying environments can speed up evolution, *P. Natl. Acad. Sci.* 104, 13711-13716.

[8] Molko, N. et al. 2003. Functional and structural alterations of the intraparietal sulcus in a developmental dyscalculia of genetic origin. *Neuron* 40(4), 847–858.

[9] Stanley, K. O., and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. *Evol. Comput.* 10(2), 99–127.