Self-Communicating Deep Reinforcement Learning Agents Develop External Number Representations

Silvester Sabathiel[1,2], Flavio Petruzzellis[3], Alberto Testolin[4,5], and Trygve Solstad*[2]

[1]Department of Computer Science, NTNU, NO-7491 Trondheim, Norway
[2]Department of Teacher Education, NTNU, NO-7491 Trondheim, Norway
[3]Department of Mathematics, University of Padova, Italy
[4]Department of Information Engineering, University of Padova, Italy
[5]Department of General Psychology, University of Padova, Italy

Abstract

Symbolic numbers are a remarkable product of human cultural development. The developmental process involved the creation and progressive refinement of material representational tools, such as notched tallies, knotted strings, and counting boards. In this paper, we introduce a computational framework that allows the investigation of how material representations might support number processing in a deep reinforcement learning scenario. In this framework, agents can use an external, discrete state to communicate information to solve a simple numerical estimation task. We find that different perceptual and processing constraints result in different emergent representations, whose specific characteristics can facilitate the learning and communication of numbers.

1 Introduction

Learning numerical symbols is a long and sophisticated process that occupies children for several years during formal education [10]. Some scholars argue that the foundations of mathematical development rest on innate numerical intuitions, such as the ability to estimate approximately the number of objects in a visual scene [24, 11, 14], while others propose that mathematical learning is mostly fostered by the acquisition of number words and counting procedures [6]. The latter approach emphasizes the role of environmental and cultural factors in the acquisition of numerical representations, such as the spatial ordering of quantities along a number line [15]. This perspective is also supported by modern psychological theories that consider human cognition as emerging from the complex interaction between the brain, the body, and all sorts of material representations that can extend our intellectual abilities beyond the limitations imposed by our evolutionary heritage [8, 12].

Compared to other forms of information encoding, such as perceptual and verbal representations, material (external) representations of numerical concepts stand out in several aspects. First, they are persistent and thus reduce working memory demands and attentional load. Second, they allow for a more precise encoding of numerical information: such devices as the abacus and Cuisenaire rods extend our ability to represent exact numbers by exploiting inter-exponential relations to encode large numbers precisely and compactly or to represent compositionality explicitly [16]. External representations can also be shared among individuals, allowing for collaboration within and between generations and thus enabling the incremental development of culture. For example, fingers were among the first external tools used to represent numbers and deploy sequential counting procedures. This representation allows a person to determine the cardinality of a set, even when objects were not all available at the same time [4]. The limited per-

*Corresponding Author: trygve.solstad@ntnu.no

https://doi.org/10.7557/18.6291

sistence and capacity of fingers was overcome by representing numerical information with more sophisticated artifacts, such as pebbles on a string that are restricted to move along a 1D axis. 2D devices enable representations of even more complex numerical relations, necessary for the powerful place-value notation used in modern times.

Deep learning models have provided important insights into the origin of our numerical intuitions, such as demonstrating that approximate numerical representations could emerge in multi-layer neural networks that learn a generative model of their sensory input [19, 22]. Deep networks can simulate both the enumeration and estimation of numerosity [9] and reproduce the developmental trajectories observed in children [23]. However, although the acquisition of counting procedures has been explored in recent models [17], we still lack a computational framework that can stimulate computational research into how numerical representations might emerge from interactions with external representations [21].

The aim of the present work is to shed light on the computational foundations of number learning by developing a framework to investigate how the acquisition of exact numbers could be grounded in sensorimotor experiences. We simulate different scenarios in which deep reinforcement learning agents estimate the number of objects in a visual scene. Crucially, agents can encode numerical information by manipulating an "external tool" according to a predefined set of actions. In line with recent work on emergent multi-agent communication [13, 7], we ask whether the necessity to communicate through a discrete state can lead to the development of efficient numerical representations.

2 Framework and Methods

The goal of this study was to investigate whether numerical representations could emerge when an agent must communicate information about quantity through a discrete state. In this section, we provide the details about our deep reinforcement learning setup. In particular, we describe the design of the environment with which the agent interacts, how the agent was implemented, the task to be solved, and the training procedure.

2.1 The environment

We designed an environment with three components: a visual stimulus, an external representation tool, and a finger that can be used for pointing. These three components correspond to separate perceptual layers that are represented as binary grids with the same dimensions (Fig. 1C).

The numerosity layer represents the visual stimulus that contains a set of items (white grid cells). The agent must encode the numerosity of the items and communicate it in order to solve the task. We simulate two different modalities of presenting numerosity information. In the spatial modality, white rectangles of differing sizes appear in the numerosity layer. In the temporal modality, a white rectangle flashes in the center of the numerosity layer. The rectangle can be of size 1×2 or 2×2 grid cells, depending on the shape of the external representation tool. The rectangle appears for one time step, followed by a random delay of two or three time steps (see Fig. 1A). The temporal setup can be viewed as an approximation of a sequential scanning process of a visual stimulus, such as a counting procedure. Indeed, humans proceed with sequential counting if the number of objects cannot be perceived all at once or is too large to quantify exactly.

The finger layer represents the position of the agent's finger on the tool using one-hot encoding. The agent can move the finger one grid cell at a time in any direction allowed by the shape of the representation tool.

The tool layer represents the neural network's visual input from an external tool that the agent can manipulate and use to communicate numerical information to itself. The tool can be manipulated through the agent's output vector, either directly or depending on the finger position.

We simulate the use of two kinds of tools: the first is a drawing tool intended to emulate the kind of non-restrictive tools available to early humans, such as making notches on a stick or drawing in the sand. The drawing tool allows any grid cell to be turned from black to white or vice versa. We only used a 1D version of the drawing tool in our experiments, but a 2D version can be implemented. The second kind of tool is a model of historically more advanced counting tools, such as counting boards and early abaci, which come with operational re-

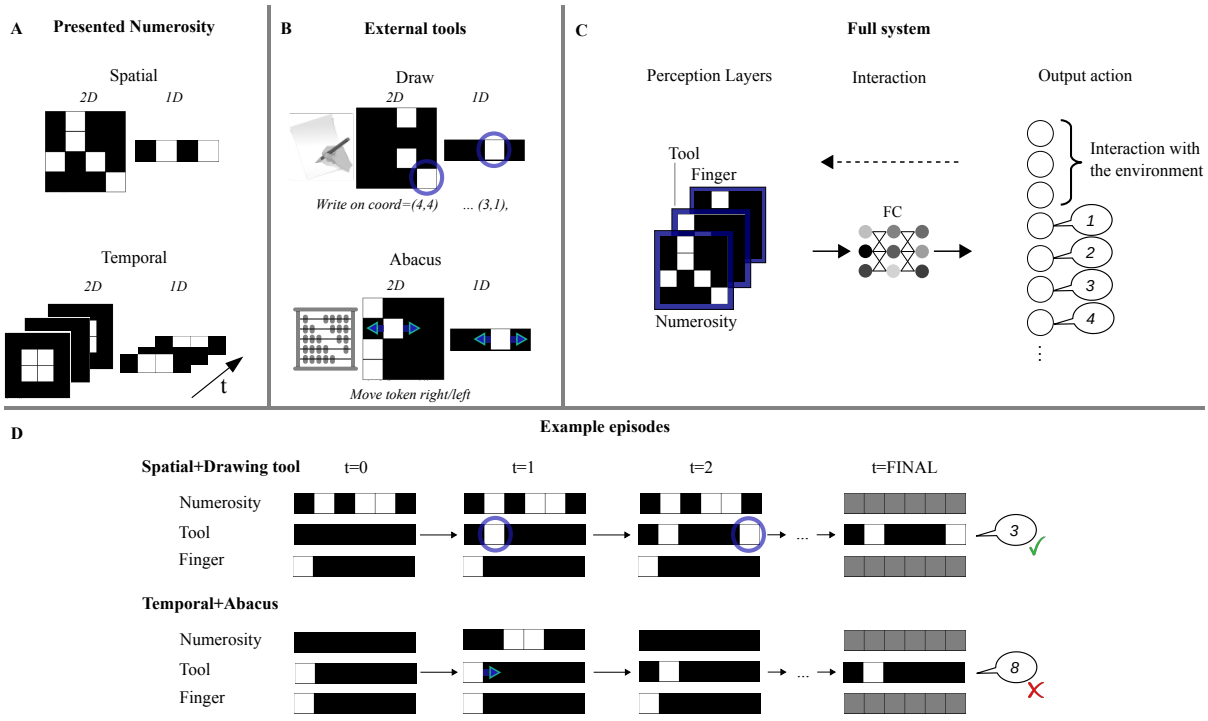

Figure 1: Schematic representation of our modeling framework. **A.** The numerosity can be presented either spatially or temporally. **B.** Agents are provided either with a drawing tool or a simplistic abacus. **C.** The full interactive system consists of the stacked input layers, a fully connected, feed-forward neural network (FC), and a set of output actions. The actions modify the agent's environment or represent a number word as the final answer. **D.** Example episodes of two experimental setups

strictions. They consist of persistent objects (e.g., pebbles), and their only degree of freedom is the position of the pebbles. In our experiments, we use both a 1D and a 2D version of this tool. The 2D tool starts with a column of binary objects placed at the left-most part of the grid. At each time step, the agent can only move the token that is currently in the same row as the finger, either to the left or the right (see Fig. 1B).

2.2 The agent

The agent's processing system is an actor network that takes the three stacked perceptual layers as input, and it consists of three consecutive, fully connected feed-forward layers, each with 64 nodes. The first two layers are followed by tanh activation functions, whereas the final layer is followed by a softmax function encoding a probability distribution over available actions.

The output layer is divided into a verbal output and an action output. The action output feeds back to the environment and the input layers. For each time step, the agent can produce either a motor command to control the tool and finger layers, or a verbal output, conceptually comparable to a number word. The action space depends on the shape and kind of the external representation tool, but it typically includes moving the finger, and drawing, placing or moving tokens on the tool based on the finger position and the tool state (see Fig. 1C).

2.3 The learning task

In each experimental setting, the goal of the agent is to produce the number word associated with the number of items presented in the numerosity layer. The learning task can be framed as a Markov De-

cision Process (MDP) [3]. It consists of a set of possible states for each time step, transition functions between subsequent states depending on an agent's actions, and a reward function that maps any state-action pair to real values. In the following, we describe the environment-specific details of the MDP.

In each episode, the environment is initialized with a random visual scene and a default state for the finger and tool layers. For the spatial setup, a given number of non-overlapping objects is randomly distributed over the grid. In the temporal setup, all grid-cells in the visual scene are initialized to black (no-event state). The agent is then allowed to interact with the environment during subsequent time steps. Each episode ends after three time steps for the static setup, and after three time steps from the last event for the temporal setup. At the final time step of the episode, all layers except for the tool layer are grayed out (uniformly set to values of 0.5). This way, the final answer only depends on the current state of the tool because the agent does not have an internal (e.g., recurrent) memory. The task is said to be performed successfully when the agent outputs the correct number word at the last time step.

For such simplistic models of the environment, neural networks could learn to infer the number of objects directly from the visual input (e.g., through supervised learning mechanisms [9]). However, one of the main purposes of this work is to introduce a novel methodology that can be extended to more complex settings, where using external tools is beneficial or even necessary.

2.4 The training procedure

The agent is trained via reinforcement learning, which uses the formal framework of MDPs to define the interaction between a learning agent and its environment [20]. In our case, it allows the agent to discover optimal action policies to create number representations under the constraints given by the environment[1]. The teaching signal is a scalar reward: when the agent activates the correct number label at the last time step, it receives a reward of 1, in all other cases, there is no reward. We used the Proximal Policy Optimization (PPO) algorithm [18] because preliminary experiments showed that it could successfully learn all the involved tasks without the task-specific fine-tuning of the hyperparameters. This is crucial for our work, as we want to compare different experimental setups of the environment under the same learning conditions. PPO is an on-policy, actor-critic algorithm, so it maintains two models: the actor model to predict the probability of the next action and the critic model to predict the value (expected average discounted reward) of a given state-action pair. For our experiments, we used an adapted version of the implementation in [2]. This version uses the 'Clipped Surrogate Objective'-function, which prevents the updated policy from deviating far from the current policy and allows running multiple training epochs on the collected samples without causing destructively large policy updates. Furthermore, the objective function includes an entropy term that encourages exploration. In our experiments, the actor and critic networks share the same architecture as described in the previous section, with the single difference that in the last layer the actor network uses a softmax function, whereas in the critic network there is no activation function.

An agent is trained with an iterative process of collecting data (states, actions, rewards, next states) and optimizing the neural network on the collected data (see Algorithm 1). At each iteration, 10 new episodes are collected and optimized on over 40 epochs. Every 100 iterations, the agent is tested on 100 episodes, for which the success rate of solving the task is recorded. We designed a curriculum learning approach [5], where the agent learns simpler concepts (small numbers) first and progressively learns harder concepts (larger numbers). In particular, the maximum numerosity is increased by 1 only after the agent masters the current set of input numerosities (i.e., it successfully solves the task in 99% of evaluated episodes).

2.5 Experimental simulations

Here, we describe four experiments designed to address the following research question: how do the form of input and constraints of the available tool affect the representations of numerosity developed by the agent in the tool layer?

[1] The implementation for the environment and the training algorithm, including a list with the hyperparameters used for the experiments, can be found at `https://github.com/ssabathiel/rl-number-agent`.

Algorithm 1 Training procedure

S: Success rate on the evaluation set
n: presented numerosity, drawn uniformly from the interval $[0, N_{max}]$ for each episode.
$S \leftarrow 0\%$
$N_{max} \leftarrow 2$
$itr \leftarrow 0$
while $S < 99\%$ **AND** $N_{max} < 9$ **do**
    for actor=1,2..A **do**
        Run policy on one episode
        Save collected data to buffer B
    end for
    Optimize surrogate loss with K=40 epochs on buffer B
    Clear buffer B
    if $itr \mod 100 = 0$ **then**
        $S \leftarrow$ *Evaluate policy on 100 episodes*
    end if
    if $S \geq 99\%$ **then**
        $N_{max} \leftarrow N_{max} + 1$
    end if
    $itr \leftarrow itr + 1$
end while

In each experimental setup, we only vary one high-level feature of the environment: the availability of the visual stimulus, the kind of representation tool, or its size. Briefly, the experimental setups are: (1) spatial input and 1D drawing tool, (2) temporal input and 1D drawing tool, (3) temporal input and 1D abacus tool, and (4) temporal input and 2D abacus tool. In the 1D setups, each of the perceptual input layers is of dimension 10×1 grid cells, while in the 2D setup, the layers are 4×4 grid cells. In each experiment, the agent is trained to recognize and communicate numerosities from 0 through 9 presented in the numerosity layer.

To compare the outcomes of the simulations quantitatively, we compute two kinds of similarities between the representations of numerosity discovered by the agent in each experimental setup. For each setup, we consider the last state of the external tool from five different episodes for five successfully trained models, obtaining 25 representations. We calculate the *structural similarity* between the representations of numbers from 0 to 9, evaluated with the minimum Hamming distance $D_{H_{min}}$ between tool patterns and rotated reference patterns. Furthermore, we compute the *positional similarity*, determined by the number of rotations that yields the minimum Hamming distance $R(D_{H_{min}})$. We chose the minimum Hamming distance as a similarity measure, because it naturally distinguishes shift variant and shift invariant structures—a foundational distinction in mathematical cognition and humans' development of the place-value notation in our modern symbolic notation. Considering any of the two distances d, we computed its average $\bar{d}$ over the 25 different representations, normalized it ($\bar{d}^* : 0 \leq \bar{d}^* \leq 1$), and then converted it to a similarity measure $1 - \bar{d}^*$. Finally, we plotted the average and normalized similarities as color-coded matrices.

3 Simulations and Results

3.1 Unstructured representations from spatial input

To simulate the visual nature of numerosity perception (pattern recognition), we designed a setup where items are presented as spatially distributed objects and the agent has access to a 1D *drawing tool*. We found that, as the agent learns to solve the task, a unique encoding for each numerosity is generated in the tool layer. However, this code is not structurally related to the numerosity it represents. There is also no systematic structural relationship between codes for different numerosities, so one cannot deduce semantic relations, such as larger or smaller, from the representation (see Fig. 2). This observation is mirrored in the unstructured correlations between representations of different numbers (see Fig. 4). Because there is no systematic relationship between the representations of two successive numerosities, this unstructured representation does not allow for extrapolation beyond the learned range of numbers.

3.2 Structured representations from temporal input

To explore the procedural nature of numerosity perception, we changed the modality of the presented numerosity from spatial to temporal. As in the spatial setup, the agent is provided with an unconstrained *drawing tool*. Because the number of events in the temporal numerosity input cannot be

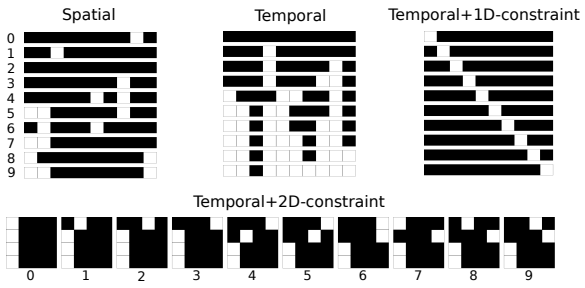

Figure 2: Examples of the last state of the tool layer of a successfully trained model for different experimental setups and presented numerosities.

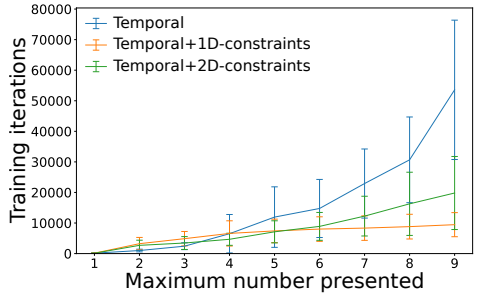

Figure 3: Average number of iterations required to learn the task as a function of the maximum number. Error bars denote standard deviation.

established from a single instance of time and the agent does not have an internal memory, numerosity information must be explicitly tracked and self-communicated using the external tool. Simulations of this setup showed that temporal sequencing of the items fosters external representations that establish a one-to-one correspondence between the counted items and the drawn items in the external tool (see Fig. 2). The external representation can be said to be "structured" in that one can relate each representation to the original number. To some degree, the representations of smaller numbers are a subset of the representations of larger numbers, reminiscent of the compositional nature of natural numbers. Figure 4 shows that the structural similarity of the representations decreases with the difference between the numbers they represent, whereas there is no systematic spatial structure.

3.3 Structured representations from a 1D abacus

The third setup is a version of the temporal setup with a 1D abacus as the external tool. Simulations in this setup show that the agent generates a spatially structured representation of the numbers in the external tool, where smaller to larger numbers are represented from left to right or vice versa (Fig. 2). This spatially structured representation would theoretically allow the agent to extrapolate the production (but not the recognition) of larger numbers beyond the numbers included in the training set.

3.4 Structured representations from a 2D abacus

The final experimental setup is a 2D version of the previous temporal setup. Simulations of this setup show that the agent learns to extend similar update patterns to different rows (Fig. 2). However, the discovered strategies vary between models of different training runs. In particular, the order of exploited rows and the direction in which the agents move the tokens within each row vary from model to model.

3.5 Time course of learning

Figure 3 shows that the setups with the 1D abacus and 2D abacus require fewer training iterations to master larger numbers (> 5) than the corresponding drawing tools. In these more constrained abacus setups, updating the number representation to its successor is independent of the number, and it is therefore a straightforward task to produce new representations for larger numbers. In contrast, for the less constrained drawing-tool setup, the agent has to 'invent' new action patterns for the representations of each number.

4 Discussion

We presented a deep reinforcement learning framework that allows the investigation of the role of external representations when agents must learn to self-communicate observed numerosities. We found that the agents successfully learn to encode numerosity using the external tools, and that the form

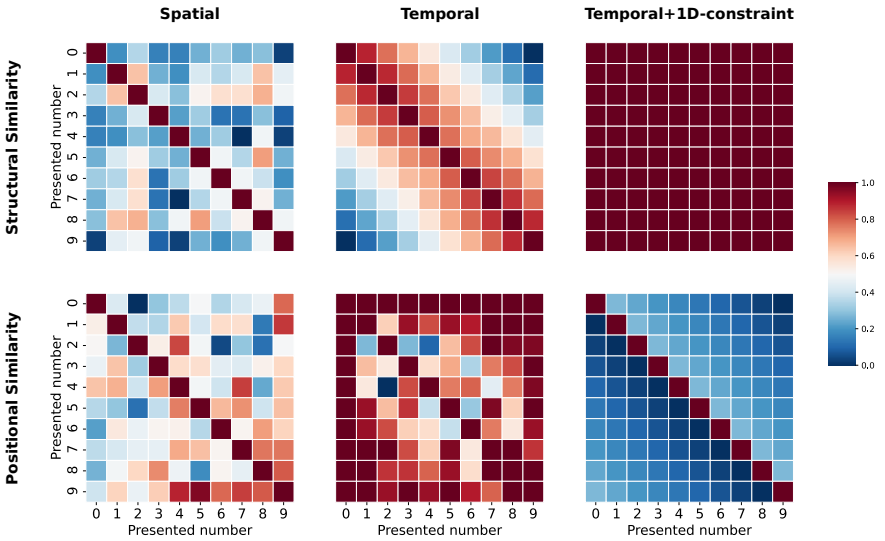

Figure 4: Structural similarity analysis for different setups. The similarity is evaluated using the minimum Hamming distance between the representation of one number and the rotated representation of reference numbers. The positional similarity is determined by the number of rotations that yields the minimum Hamming distance. For details, see the *Methods* section.

of the emergent representations strongly depends on perceptual and tool-specific constraints. The setup with an unrestricted tool and spatially distributed objects leads the agent to develop unstructured representations, whereas numerosities presented as temporally distributed events foster the emergence of more structured representations that incorporate the one-to-one correspondence. Further, restricting the external tool to consist of only movable, persistent objects leads to the emergence of spatially structured representations. The observed properties of the different external representations correspond to number representations developed historically by human cultures, such as symbolic drawings (unstructured), tallies (one-to-one correspondence), and the abacus (spatially structured). This correspondence is consistent with the idea that the development of symbolic numerical systems has been shaped by perceptual constraints and the external tools available.

However, in our model, the agent learns the tasks through random exploration within a simplistic environment. The work leaves open whether the studied constraints are sufficient to lead to similar results in more complex setups or requires the modeled agent to be endowed with more sophisticated cognitive abilities, such as planning and reasoning. Future work should explore under which constraints more advanced number representations, such as place-value systems, could emerge, as well as how more sophisticated external tools could support learning more challenging tasks that require relational inference and higher levels of abstract reasoning (e.g., algebraic problems). This may also incorporate multiple agents acting in a physically grounded environment, which might lead to the emergence of collaborative tool use [1] and would incorporate further aspects of human number representations, such as communication, collaboration, and powerful evolutionary algorithms, on top of individual agent learning.

References

[1] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from

multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019. https://doi.org/10.1016/j.cognition.2012.05.005.

[2] N. Barhate. *Minimal PyTorch Implementation of Proximal Policy Optimization*. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.

[3] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957. https://doi.org/10.1512/iumj.1957.6.56038.

[4] A. Bender and S. Beller. Nature and culture of finger counting: Diversity and representational effects of an embodied cognitive tool. *Cognition*, 124(2):156–182, 2012. https://doi.org/10.1016/j.cognition.2012.05.005.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009. https://doi.org/10.1145/1553374.1553380.

[6] S. Carey and D. Barner. Ontogenetic origins of human integer representations. *Trends in Cognitive Sciences*, 23(10):823–835, 2019. https://doi.org/10.1016/j.tics.2019.07.004.

[7] R. Chaabouni, E. Kharitonov, E. Dupoux, and M. Baroni. Communicating artificial neural networks develop efficient color-naming systems. *Proceedings of the National Academy of Sciences*, 118(12), 2021. https://doi.org/10.1073/pnas.2016569118.

[8] A. Clark et al. *Supersizing the mind: Embodiment, action, and cognitive extension*. OUP USA, 2008. https://doi.org/10.1093/acprof:oso/9780195333213.001.0001.

[9] C. Creatore, S. Sabathiel, and T. Solstad. Learning exact enumeration and approximate estimation in deep neural network models. *Cognition*, 215:104815, 2021. https://doi.org/10.1016/j.cognition.2021.104815.

[10] S. Dehaene. *The number sense: How the mind creates mathematics*. OUP USA, 2011.

[11] S. Dehaene, M. Piazza, P. Pinel, and L. Cohen. Three parietal circuits for number processing. *Cognitive Neuropsychology*, 20(3-6): 487–506, 2003. https://doi.org/10.1080/02643290244000239.

[12] G. Lakoff and R. E. Núñez. Where mathematics comes from: How the embodied mind brings mathematics into being. *AMC*, 10(12): 720–733, 2000.

[13] A. Lazaridou and M. Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.

[14] A. Nieder. Number faculty is rooted in our biological heritage. *Trends in Cognitive Sciences*, 21(6):403–404, 2017. https://doi.org/10.1016/j.tics.2017.03.014.

[15] R. E. Núñez. No innate number line in the human brain. *Journal of Cross-Cultural Psychology*, 42(4):651–668, 2011. https://doi.org/10.1177/0022022111406097. ISSN 00220221. doi: 10.1177/0022022111406097.

[16] K. A. Overmann. Constructing a concept of number. *Journal of Numerical Cognition*, 4(2):464–493, 2018. https://doi.org/10.5964/jnc.v4i2.161. ISSN 2363-8761. doi: 10.5964/jnc.v4i2.161.

[17] S. Sabathiel, J. McClelland, and T. Solstad. A computational model of learning to count in a multimodal, interactive environment. In *CogSci*, 2020.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[19] I. Stoianov and M. Zorzi. Emergence of a'visual number sense'in hierarchical generative models. *Nature Neuroscience*, 15(2):194, 2012. https://doi.org/10.1038/nn.2996.

[20] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[21] A. Testolin. The Challenge of Modeling the Acquisition of Mathematical Concepts. *Frontiers in Human Neuroscience*, 14(March):1–9, 2020. https://doi.org/10.3389/fnhum.

2020.00100. ISSN 16625161. doi: 10.3389/fnhum.2020.00100.

[22] A. Testolin, S. Dolfi, M. Rochus, and M. Zorzi. Visual sense of number vs. sense of magnitude in humans and machines. *Scientific Reports*, 10(1):1–13, 2020. https://doi.org/10.1038/s41598-020-66838-5.

[23] A. Testolin, W. Y. Zou, and J. L. McClelland. Numerosity discrimination in deep neural networks: Initial competence, developmental refinement and experience statistics. *Developmental Science*, 23(5):e12940, 2020. https://doi.org/10.1111/desc.12940.

[24] K. Wynn. An evolved capacity for number. in: Cummins D.D. Allen C. *The evolution of mind*, page 107–126, 1998.