Developing the Knowledge of Number Digits in a child like Robot

DI NUOVO, Alessandro <http://orcid.org/0000-0001-5308-7961> and MCCLELLAND, James L.

Title

Developing the Knowledge of Number Digits in a child-like Robot

Authors:

Alessandro Di Nuovo[1]*, James L. McClelland[2]

[1] Sheffield Robotics, Department of Computing, Sheffield Hallam University, UK

[2] Department of Psychology, Center for Mind, Brain and Computation, Stanford University, Stanford, CA, US

* Corresponding author. Email: a.dinuovo@shu.ac.uk

Abstract

Number knowledge can be initially boosted by embodied strategies, such as the use of fingers. This article explores the perceptual process of grounding number symbols in artificial agents, particularly the iCub robot - a child-like humanoid with fully functional five-fingered hands. It studies the application of convolutional neural network models in the context of cognitive developmental robotics, where the training information is likely to be gradually acquired while operating rather than being abundant and fully available as in many machine learning scenarios. The experimental analyses show increased efficiency of the training and similarities with studies in developmental psychology. Indeed, the proprioceptive information from the robot hands can improve accuracy in the recognition of spoken digits by supporting a quicker creation of a uniform number line. In conclusion, these findings reveal a novel way for the humanization of artificial training strategies, where the embodiment can make the robot's learning more efficient and understandable for humans.

Introduction

The embodied cognition theory affirms that human intelligence is formed not only by the brain but shaped also by the body and the experiences acquired through it, such as manipulatives, gestures, and movements[1–4]. Indeed, research in developmental psychology has shown that embodied experiences help children in the learning of various cognitive skills by using limbs and senses to interact with the surrounding environment and other human beings[5].

Among the human cognitive skills that can be extended through bodily experiences, number processing is particularly valuable because it can provide a window into the neuronal mechanisms of high-level brain functions[6]. Numbers constitute the building blocks of mathematics, a language of the human mind that can express fundamental properties of the physical world and make the universe intelligible[7]. Therefore, understanding how the artificial sensorimotor system embodies numerical processes can also help to answer the wider question of how bodily (real or artificial) systems support and scaffold the development of abstract cognitive skills[8].

Within the embodied mathematics framework, fingers are spontaneous tools that play a crucial role in developing number cognition until a level of basic arithmetic is achieved (for details see recent reviews[9,10]). In particular, Gunderson et al.[11] observed that young children can better communicate their knowledge about numbers using hand gestures rather than words, particularly for numbers that they have not yet learned in speech. In fact, one of the most evident embodied interactions with cognition is the use of fingers to convey both cardinal and ordinal aspects of numbers: finger *montring*[12] refers to the use of finger configurations to represent cardinal number information; finger *counting* and *pointing* gestures are used to support ordinal representation for counting quantities or doing basic arithmetic operations[13,14]. In a short review[15], Di Luca and Pesenti have shown that "*finger-counting/montring activities, especially if practised at an early age, can contribute to a fast and deep understanding of number concepts, which has an impact during the entire cycle of life by providing the sensory-motor roots onto which the number concept grows*". The essential role of motor contribution was validated by Sixtus et al.[16], who compared visual images versus actively produced finger postures (motor priming), showing that only canonical motor finger posing has a significant positive effect on number processing. The concept of embodied cognition extends the role of fingers beyond just another external material (e.g. blocks, Cuisenaire rods) for learning how to process numbers. Instead, internal finger-based representations provide a natural numerical representation that facilitates the development of initial mathematical cognition[17]. More specifically, Butterworth[18] suggested that "*without the ability to attach number representations to the neural representations of fingers and hands in their normal locations, the numbers themselves will never have a normal representation in the brain*". These ideas from behavioural research were confirmed and extended by recent neuroimaging research in the area of embodied mathematics (a recent literature review can be found in[19]) where empirical studies have suggested a neural link or even a common substrate for the representation of fingers and numbers in the human brain[20]. In the neuroimaging data, neural correlates of finger and number representations can be located in neighbouring or even overlapping cortex areas, e.g.[21]. Therefore, it is suggested that finger processing may play a role in setting up the biological neural networks on which more advanced mathematical computations are built[22].

Importantly, numerous studies also showed a permanent neural link between finger configurations and their cardinal number meaning in adulthood. For instance, researchers have found that adult humans still activate the same motor cortex areas that control fingers while processing digits and number words, even if motor actions are inhibited[23]. Tschentscher et al.[24] hypothesized the link is the result of an association in the early stages of number learning when finger configurations are used by both teachers and children to represent numbers while explaining mathematical concepts. Indeed, hand gestures are often observed when teaching mathematical concepts as a way to scaffolding students' understanding[25], especially when communicating new material[26]. Several authors, e.g.[27,28], show that children who observe gesture while learning mathematics perform better than children who do not, and that gesture during teaching encourages children to produce gestures of their own, which, in turn, can enhance the training allowing them to consolidate and transfer the learning of abstract concepts. However, while children often use fingers to support their early mathematical learning and this habit correlates with better performance in initial stages, they do not need gestures in later stages after they have successfully learned the basic concepts[29]. The use of fingers while learning about numbers has also generated a debate between researchers in neurocognition and education, with the latter concerned that relying on fingers can be detrimental for the later numerical development. These authors recommend the use of finger representations only at early stages, then to be replaced by concrete structured representations and, finally, mental representations of numbers to perform numerical operations[30].

An innovative approach for studying the embodied learning is Cognitive Developmental Robotics (CDR), which was defined as the "interdisciplinary approach to the autonomous design of behavioural and cognitive capabilities in artificial agents (robots) that takes direct inspiration from the developmental principles and mechanisms observed in natural cognitive systems (children)"[31]. The application of embodied theory in artificial agents is among the motivations for designing new robotic platforms for research to resemble the shape of a human body, known as "humanoids", e.g. ASIMO[32], and in particular that of a child, notably iCub[33]. One of the postulates of CDR is that the humanization of the learning process can help to make artificial intelligence more understandable for humans and may increase the acceptance of robots in social environments[34]. CDR is still making its first steps, but it has already been successfully applied in the modelling of embodied word learning as well as in the development of perceptual, social, language and numerical cognition[35,36], and recently extended as far as the simulation of embodied motor and spatial imagery[37,38].

Yet, only a few attempts have been made so far to simulate embodied number learning in robots[39], mostly aimed at investigating finger counting with synthetic datasets. For example, inspired by the earlier work by Alibali and Di Russo[14], Ruciński et al.[40] presented a model in which pointing gestures significantly improve the counting accuracy of the humanoid robot iCub. De La Cruz, Di Nuovo et al.[41–43] investigated artificial models for learning finger counting (motor), Arabic digit recognition (visual) and spoken digits (auditory), to explore whether finger counting and its association with spoken or Arabic digits could serve to bootstrap number cognition. Results of these experiments show that learning number word sequences together with finger sequencing speeds up the building of the neural network's internal representations resulting in patterns that better capture the similarities between numbers. In fact, the internal representations of finger configurations can represent the ideal basis for the building of an embodied number representation in the robot. Subsequently, Di Nuovo et al.[44] presented a deep learning model that was validated in a simulation of the embodied learning behaviour of bi-cultural children, using different finger counting habits to support their number learning. Recently, Di Nuovo[45] presented a "shallow" embodied model for handwritten digit recognition, which incorporates the link hypothesized by Tschentscher et al.[24]. Simulations showed how the robot fingers could boost the performance and be as effective as the cardinal numerosity magnitude that has been proposed to be the ideal computational representation for artificial mathematical abilities[46]. Moving further to arithmetic, Di Nuovo[47] investigated a Long Short-Term Memory (LSTM) architecture for modelling the addition operation of handwritten digits using an embodied strategy. The results confirm an improved accuracy in performing the simultaneous recognition and addition of the digits, also showing a higher frequency of split-five errors in line with what has been observed in studies with humans[48].

All these studies provided valuable information about the simulation of artificial learning and demonstrated the value of the CDR approach to study aspects of numerical cognition. However, even if they apply machine intelligence methods, they lack generalization and applicability in this field. Indeed, like many other CDR studies, those presented above are based on simple "shallow" models trained on synthetic data, which was often created ad-hoc for the aim of the study. For instance, early models[41–43] were a simple recurrent network trained and tested on the same a database of just 10 synthesized spoken number words and 5x2 black and white pixels visual digits, also no alternate representations were compared. Vice versa, two recent studies[45,47] made use of the popular MNIST database of real handwritten digits, but they made a somewhat implausible setting in the context of early cognitive development, where speech usually precedes and accompany writing. In order to significantly contribute to the progress of the state of the art in machine intelligence, novel research is needed to properly contextualize the simulations in developmental learning in deeper neural network architectures while showing applicability to real datasets and problems.

In this article, we apply the CDR approach to the recognition of real spoken digits by presenting a deep Convolutional Neural Network (CNN) architecture designed to apply the embodiment principles by using the sensory-motor information from an artificial humanoid body, i.e. the child-like robot iCub, which is one of the few platforms that has fully functional five-fingered hands[49]. The spoken digits are taken from a novel open database for speech recognition, created by Google to facilitate new applications[50]. Simulating the developmental plasticity of the human brain, the models are trained using a two-stage approach, known as "transfer learning"[51], in which the robot learns first to associate spoken digits and finger representations, i.e. motor patterns specifying the state of each of the robot's fingers (extended or open vs closed or retracted). Then the network is extended with new layers to perform the classification into the number classes by building upon the previously learned association. In a first scenario, the training procedure simulates how children initially behave while learning to recognize symbolic numerals (in the form of spoken digits), in particular when learning number words by repeating them together with the corresponding finger sequence to help the transition from preverbal to verbal counting and computation[52]. In a second scenario, we present a longitudinal analysis that gives useful insights on how biologically inspired strategies can improve deep CNNs performance in the context of applied robotics, where the training information is likely to be gradually acquired while operating rather than being abundant and fully available as in the majority of machine learning scenarios.

Recognising spoken digits in a cognitive developmental robot

This section presents the experimental results of the CNN architecture designed to simulate the embodied learning to recognize spoken number digits in comparison with a standard non-embodied baseline. Figure 1 presents the schematics of the baseline (left) and embodied (right) networks. Three possible internal representations, two embodied and one control, are considered and compared: (i) The cardinal numerosity using a thermometer representation. In this representation, the first neuron in a set of nine is active for the number 1, the first two are active for the number 2, etc. (ii) The iCub robot encoder values of the right and left hand when displaying the finger representations of digits (See inset in *Figure 1*, and *Supplementary Figure 1*). These representations indicate the number magnitude by the number of open fingers, though the numbers 3 and 4 as well as 8 and 9 involve only partially overlapping sets of fingers. (iii) Random numbers in the range [0,1], which are used as the control for validation. As an additional control condition, we also applied the transfer learning approach to the baseline model by pre-training the convolutional blocks using the same random values as targets. Details about the models, the embodied representations and the spoken digits database are in the Methods section.

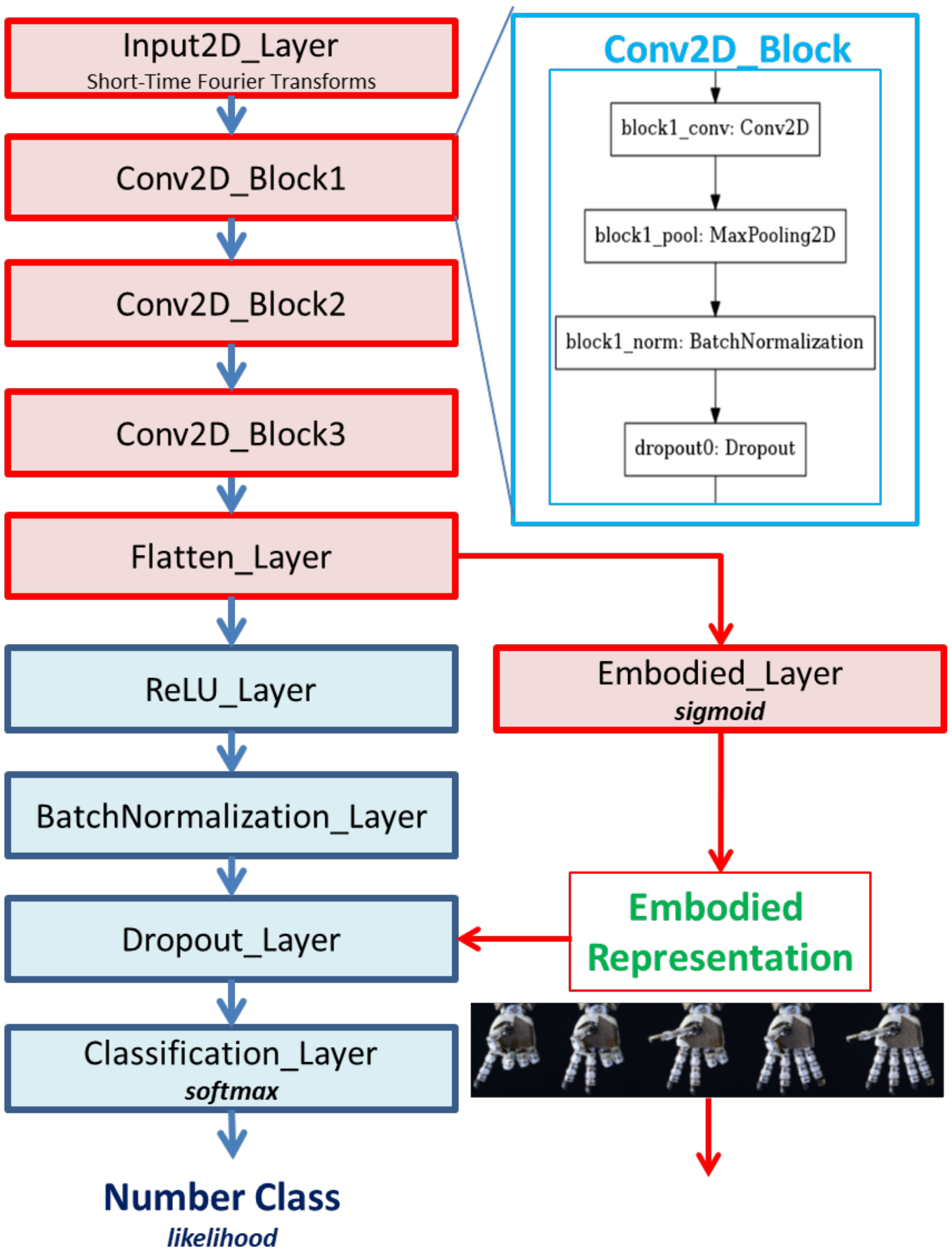

Fig. 1. Schematics of the artificial neural network architectures.
The baseline CNN architecture is on the left. The detail of the Conv2D blocks is on the top right. The embodied architecture is created including the Embodied Layer (on the right). In the first stage, the layers in red are those pre-trained to reproduce the embodied representation, e.g. output is the positions for the robot's finger motors. After the pre-training, the embodied model is completed by linking, the Embodied Layer to the final Dropout Layer, thus the full embodied architecture can be trained both to classify the spoken digits and to reproduce the embodied representations.

Table 55 (Methods section) gives the summary of the layers with detail of the parameters, arguments and initialization.

In the following, we label *embodied models* when the architecture in Figure 1 is trained with the Cardinal Numerosity or the iCub robot fingers as targets for the Embodied layer. Cardinal Numerosity can be considered the ideal embodied representation of number magnitude, while the fingers are its real-world implementation. Instead, we define *control model* if the targets are the random values. Two other conditions are: the *simple baseline*, which is the one that goes straight from the Input to the Classification Layer on the left of Figure 1; the *pre-trained baseline*, which is the same baseline architecture, where the CNN blocks are pre-trained similarly to the control model using random values as targets. The baselines and the control model are considered also as *control conditions*.

Scenario 1: Learning to process spoken digits while acquiring counting principles

This subsection presents a simplified simulation of the early number processing when children initially learn the number digits while repeating them by rote together with the finger representations[52]. The first phase includes the acquisition of the one-to-one principle, i.e. assigning one counting word to each item in a set[53]. Indeed, in this scenario, the models are initially pre-trained using a smaller subset of uniformly distributed examples. Examples were grouped in batches formed by four sequences of the nine digits in their cardinal order.

Next, inspired by the children using finger representations while communicating number words, this simulated number learning scenario continues by training the robot to classify the spoken digits while reproducing the corresponding finger representations. This second training phase can be associated to the acquisition of the cardinal principle, which is defined as learning that "the last number spoken, when counting a set of items, tell how many items are in the set"[53]. For a proper simulation of digit learning at this developmental stage, the appropriate distribution of the training examples should follow the Zipf's empirical law that the frequency of any word is inversely proportional to its frequency rank[54]. Therefore, as explained in the methods section, we created an ad-hoc dataset to match the Zipfian distribution. Furthermore, to simulate a gradual education like in the case of children, we considered three quotas (25%, 50%, 100%) of the Zipfian training sample from which we extracted three uniform sub-sets for the pre-training.

The models' performance during the second training phase is presented in Figure 2, where the graphs show the accuracy rate on the test set at the end of each training epoch. They include the pre-trained baseline (blue lines), the embodied models with the iCub robot fingers (purple), the Cardinal Numerosity (red), and the control model with random values (green). For all three training sample sizes, we see a significant increase in the accuracy for the models using either the Cardinality Numerosity or the iCub robot representation compared to either of the other two control conditions, with a stronger initial effect for the smaller sample sizes where Cohen's is $d>1$. Figure 2a,b,c show also that the accuracy on the test set grows fast until after around 22 epochs (median) when it starts to oscillate, as usual for CNN, with little or no improvement but without significant overfitting. For this reason, we decided to stop the training after 25 epochs and average the accuracies of the epochs with the lowest loss.

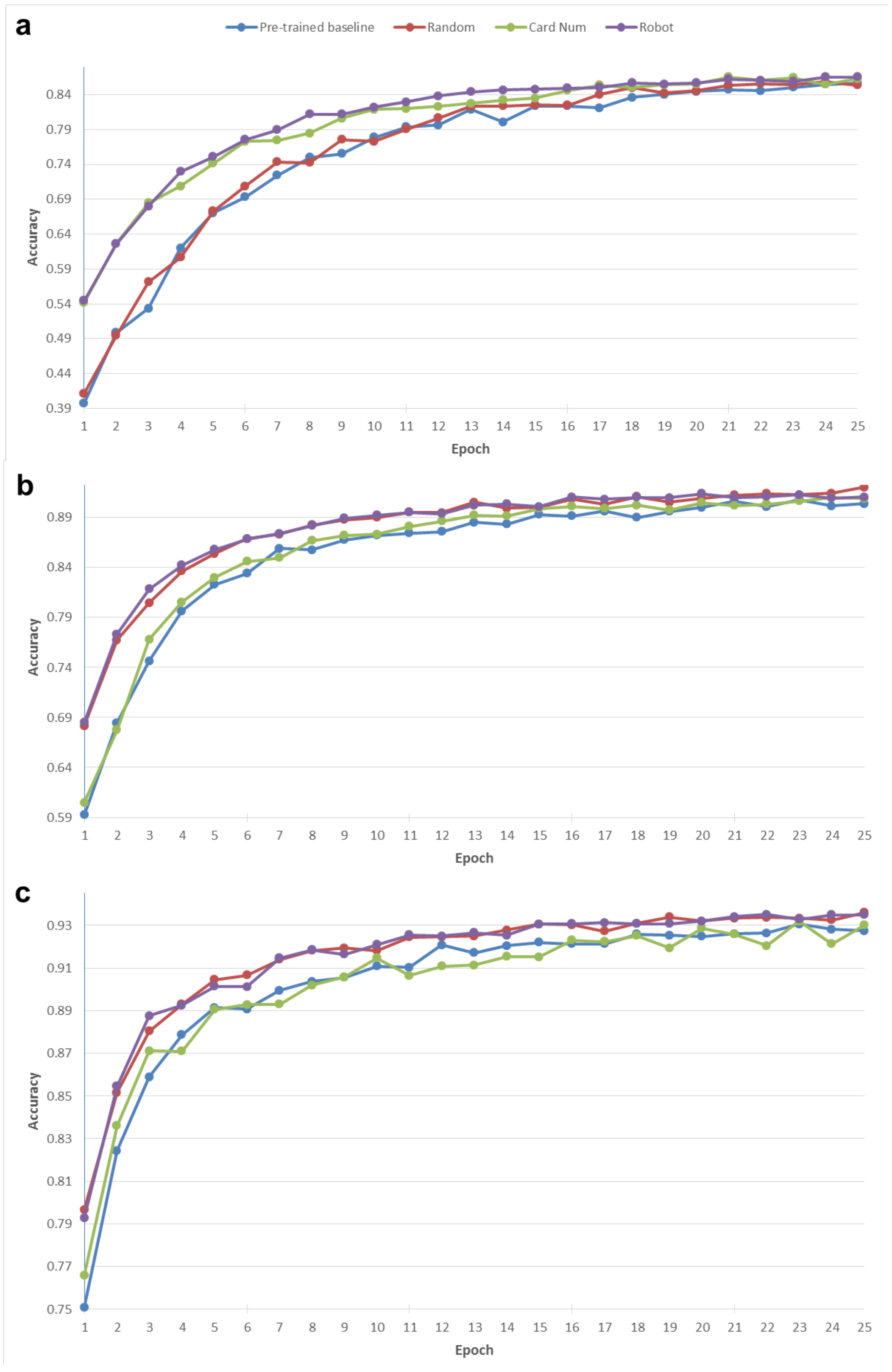

Fig. 2. Accuracy rate on the test set over epochs.

The accuracy rate of the embodied models (purple: robot; red: Cardinal Numerosity), pre-trained control conditions (blue: randomly pre-trained baseline; green: random control model). ***a****, small: pre-training 774 uniformly distributed examples; full-training 2193 examples with Zipfian distribution.* ***b****, medium: pre-training 1548 uniformly distributed examples; full-training 4386 with Zipfian distribution.* ***c****, large: pre-training 3096 uniformly distributed examples; full-training 8773 with Zipfian distribution.*

Table 1 gives a comparative report of results after the first and the last epoch in the training. Accuracy rates, standard deviations (SD) on the test set are shown for the embodied and control conditions along with the Cohen's *d* for comparison against the pre-trained baseline. After the first epoch (Table 1), the performance of the control model with random values was always significantly inferior to the other two representations, indeed even if its average accuracy was typically higher, the control model was not significantly better than the pre-trained baseline and Cohen's always indicated a small effect size (*d*<0.5).

Table 1. Accuracy rates for varying training example sizes and different representations.
Average accuracies (Acc) on the test set, with standard deviations (SD) and Cohen's d. Values in bold are significantly (p<0.05) better (black) than the baseline. The best overall accuracy rate for each row is highlighted in green (multiple in case of no statistical difference).

Training examples (pre/full)	**Baseline (pre-trained)**		**Random values**			**Cardinal Numerosity**			**iCub robot fingers**		
	Acc	**SD**	**Acc**	**SD**	***d***	**Acc**	**SD**	***d***	**Acc**	**SD**	***d***
Average after Epoch 1											
774/2193	0.3950	0.0867	0.4009	0.0680	0.08	**0.5386**	0.0539	1.99	**0.5396**	0.0459	2.08
1548/4386	0.5811	0.0526	0.5926	0.0623	0.20	**0.6766**	0.0394	2.05	**0.6798**	0.0411	2.09
3096/8773	0.7592	0.0683	0.7692	0.0844	0.13	**0.7946**	0.0309	0.67	**0.7940**	0.0321	0.65
Final (average of epochs with lowest training loss)											
774/2193	0.8535	0.0191	0.8508	0.0180	-0.14	**0.8638**	0.0147	0.60	**0.8665**	0.0176	0.71
1548/4386	0.8990	0.0170	0.9055	0.0120	0.44	**0.9126**	0.0076	1.03	**0.9071**	0.0097	0.58
3096/8773	0.9279	0.0098	0.9299	0.0098	0.20	**0.9347**	0.0110	0.65	**0.9361**	0.0053	1.04

The Cardinal Numerosity and the iCub Fingers represent the magnitude of the digits, which can explain the better performance because they contribute to the acquisition of a more linear number line, indeed they are faster at improving accuracy for bigger digits, which are more difficult because they are less represented in the Zipfian distribution of the training set. The correlation among improved numerical categorization, increasingly linear number line estimation, and numerical magnitude in children was shown by Laski and Siegler[55]. To exemplify the advantage, Table 2 summarizes the development of average accuracy rates for the groups of smaller (1-4) and bigger (5-9) digits. This analysis allows relating to experimental data with children, who can label small set sizes exactly (1–4) and larger set sizes approximately (5–9) while learning the cardinal principle[56]. Without pretending to replicate the study, we note that all our models shown a progression similar to what observed in children, who progress their knowledge starting from the smaller numbers, then gradually improving the others following the number line.

Table 2. Accuracy progression for smaller and bigger digits.
The table reports the average accuracy rates for smaller digits, from 1 to 4, bigger digits, from 5 to 9, after epoch 1 and 25. Higher accuracies are in bold.

Pre/Full Train Sizes	**Baseline (pre-trained)**		**Random Values**		**Cardinal Numerosity**		**iCub Fingers**	
	(1-4)	**(5-9)**	**(1-4)**	**(5-9)**	**(1-4)**	**(5-9)**	**(1-4)**	**(5-9)**
Average after epoch 1								
774/2193	0.578	0.172	0.601	0.181	**0.707**	**0.360**	**0.723**	**0.350**
1548/4386	0.776	0.388	0.758	0.424	**0.806**	**0.540**	**0.815**	**0.537**
3096/8773	0.865	0.638	0.868	0.664	**0.879**	**0.713**	**0.879**	**0.704**
Average after epoch 25								
774/2193	0.904	0.799	0.902	0.793	**0.905**	**0.810**	**0.903**	**0.813**
1548/4386	0.934	0.862	0.935	0.871	**0.941**	**0.890**	0.928	**0.878**
3096/8773	0.950	0.905	0.952	0.910	**0.955**	**0.919**	**0.952**	**0.917**

It is interesting to note that we didn't found relevant differences between the Cardinal Numerosity representation and the iCub finger configurations, except for the final performance with medium training size, in which there is a medium effect (*d*=0.6312) in favour of the Cardinal Numerosity. However, they both contributed equally in modelling a more uniform number line, even if, in the case of the robot, there are the same numbers of simulated motor activations for 3 and 4 or 8 and 9. Besides, pertaining to any of the four kinds considered provides a jump start for subsequent learning compared to the baseline with no pre-training (results not shown for conciseness), as expected. A comparison with the simple baseline is discussed in detail in the next subsection.

Scenario 2: A Longitudinal study of spoken digits recognition in embodied artificial agents

In this subsection, we present results of longitudinal experimentation on the development of learning in artificial agents, i.e. how the performance evolves with the training and the number of examples available for it. Indeed, to analyze the gradual development of spoken digits recognition, we split the training examples and investigated the models' performance with varying number of examples. For simplicity, in the following, we will refer to the groups as small (128, 512, 1024), medium (2788, 5576), large (13942, 27884). The training and testing sets have a pseudo-uniform distribution as specified in Table 4 (Methods section). In this experimental scenario, the artificial learner used a portion (25%) of the training dataset for a quick pre-training of the CNN blocks.

Figure 3 presents the history of the average accuracy rate on the test set at the end of each epoch. The graph (a) is for the small group, (b) for the medium and (c) for the large. As seen in the previous experiment, we avoided significant overfitting thanks to the use of common strategies like mini-batches, batch normalization layers, and dropout layers.

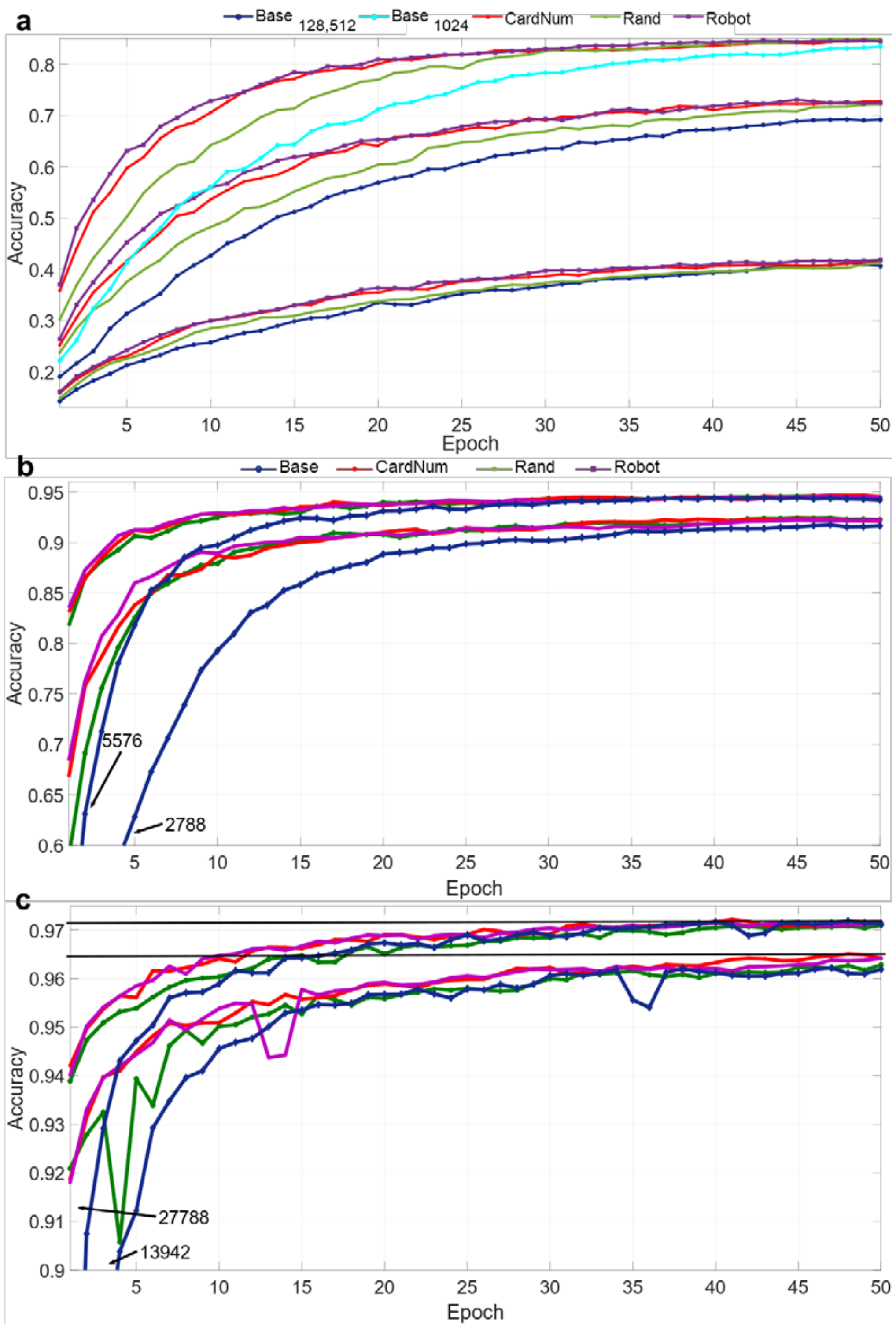

Fig. 3. Accuracy rate on the test set over epochs.
The accuracy rate of the embodied models (purple: iCub Robot fingers; red: Cardinal Numerosity), and control conditions (blue: Simple Baseline; green: Control Model with random values). ***a****, small: pre-training: 32, 128, 256; training 128, 512, 1024 examples.* ***b****, medium: pre-training 697, 1394; full-training 2788,5576.* ***c****, large: pre-training 3485, 6971; full-training 13942, 27884. In the groups, training with bigger sets always achieved higher accuracy and there is no overlap among the lines of different groups. The only exception is the simple baseline, which is in light blue in* ***a*** *because the line for 1024 starts below the previous case (512). For clarity, in* ***b*** *and* ***c*** *it is specified the training set size for the blue lines (baseline). In* ***c*** *the black lines at the top identify the best overall results.*

Results of the longitudinal experiment are summarized in Table 3, which presents the embodied models in a comparison against two control conditions: control model with the random values (first columns), and the simple baseline (last columns). Accuracies on the test set are calculated averaging the results of the epochs with lowest training loss at half-way (25 epochs) and the end of the training (50 epochs). The last part of Table 3 reports the first epoch when average accuracy was greater than 99% of the baseline's final accuracy. This is a measure of how fast the training converges. We see that control conditions reached the same accuracy of the embodied models after more training repetitions (epochs). Exceptions were in the medium group, Figure 3b, where the control model was as accurate as the embodied models earlier, i.e. after 10 epochs (2788) and it is almost as good as the embodied models since the beginning (5576).

Table 3. Summary of the results on the test set.
Average accuracy rates (Acc) on the test set, with Standard Deviations (SD) and Cohen's d. Accuracy rates are highlighted in green when significantly (p<0.05) better than baseline, in bold when significantly (p<0.05) better (black) or worse (red) than the control model with random values. The final rows of this table show the median epochs when test accuracy was greater than 99% of the baseline's final average accuracy. Supplementary Table 2 reports the p-values for all the pairs considered in this table.

Training examples (pre/full)	Random values		Cardinal Numerosity			iCub robot fingers			Baseline		
	Acc	STD	Acc	SD	*d*	Acc	SD	*d*	Acc	SD	*d*
After Epoch 25 (average of testing after epochs with lowest training loss)											
32/128	0.3602	0.035	**0.3800**	0.028	0.625	**0.3828**	0.024	0.748	0.3558	0.027	-0.140
128/512	0.6462	0.030	**0.6816**	0.026	1.252	**0.6847**	0.017	1.571	**0.6148**	0.034	-0.985
256/1024	0.8095	0.017	**0.8243**	0.010	1.039	**0.8255**	0.016	0.975	**0.7663**	0.016	-2.604
697/2788	0.9143	0.007	0.9126	0.008	-0.234	0.9139	0.006	-0.056	**0.9003**	0.008	-1.796
1394/5576	0.9384	0.006	**0.9426**	0.005	0.745	**0.9424**	0.005	0.723	**0.9317**	0.006	-1.125
3485/13942	0.9584	0.004	**0.9607**	0.002	0.727	**0.9613**	0.003	0.872	0.9587	0.003	0.084
6971/27884	0.9677	0.002	**0.9698**	0.002	0.861	**0.9694**	0.002	0.690	0.9688	0.002	0.483
Final (average of testing after epochs with lowest training loss)											
32/128	0.4093	0.027	0.4186	0.019	0.402	0.4203	0.029	0.394	0.4000	0.028	-0.341
128/512	0.7213	0.018	**0.7308**	0.018	0.525	**0.7327**	0.025	0.524	**0.6960**	0.025	-1.172
256/1024	0.8487	0.011	0.8484	0.010	-0.024	0.8472	0.014	-0.119	**0.8340**	0.013	-1.263
697/2788	0.9230	0.006	0.9222	0.005	-0.133	0.9230	0.005	-0.007	**0.9166**	0.006	-1.060
1394/5576	0.9463	0.004	0.9470	0.004	0.195	0.9474	0.004	0.285	**0.9419**	0.005	-0.977
3485/13942	0.9630	0.003	**0.9648**	0.002	0.676	0.9639	0.002	0.339	0.9625	0.003	-0.150
6971/27884	0.9714	0.002	0.9721	0.001	0.455	0.9716	0.002	0.125	0.9716	0.002	0.120
Epoch when testing accuracy was greater than 99% of the Baseline's Final Average Accuracy											
32/128	44		36			35			43		
128/512	36		29			27			45		
256/1024	31		27			29			46		
697/2788	17		19			19			34		
1394/5576	16		15			16			22		
3485/13942	13		11			10			14		
6971/27884	12		6			9			14		

In summary, the longitudinal experiments confirmed that the embodied models were more effective learners than the control conditions, indeed, they achieved higher recognition accuracies in fewer epochs, especially with the smaller training sets. The embodied models were significantly more successful than the baseline, with exceptions in the larger group, when they achieved a higher accuracy but there was no statistical difference. The embodied models were often more accurate than the control model, with some exceptions, e.g. when training with 1024 and 2788 examples after 50 epochs. However, while embodied models were the best until around 25 epochs, their advantage usually decreased with the training, often lacking statistically significant difference if compared to the control model final accuracy. These results can be linked to the transition from early to mature mathematical cognition in children, who initially perform better when they can use finger representations, then, gradually abandon them for other strategies[29].

Comparing the embodied models' representations, the two were statistically equivalent in terms of performance (see Supplementary Table 2). This confirmed that the physically embodied representation is as good as the pure cardinality representation, while it captures the real motor activation data of the robot. Supplementary Table 3 shows a comparison among the control conditions. As seen in the previous scenario, the comparison evidenced that the control model has often a higher final accuracy, but it was not significantly different than the pre-trained baseline. However, the pre-trained baseline was slower than the control model, i.e. it often reached the peak accuracy later.

Conclusions

Recent studies in developmental psychology and cognitive neuroscience demonstrated a pivotal role of fingers in developing number cognition. Inspired by these studies, this article investigated the perceptual process of recognizing spoken digits in deep, convolutional neural networks by embodying them in the humanoid robot iCub's fingers in the training. In particular, finger representations replicated activations in motor cortex when processing numbers that reflect the hand used for counting as seen in humans[24].

Simulation results showed that the robot's fingers boost the performance by setting up the network and augmenting the training examples when these were numerically limited. This is a common scenario in robotics, where robots will likely learn from a small amount of data. Results can be related to some behaviours also observed in several human studies in developmental psychology and neuroimaging. Overall, the hand-based representation provided our artificial system information about magnitude representations that improved the creation of a more uniform number line, as seen in children[55,56]. Interestingly, our results also indicate that accuracy can be increased by pre-training convolutional blocks with a uniform subset taken from a non-uniform training set. Furthermore, longitudinal experimentation showed that the performance improvement from the representation of the robot's fingers was reduced with experience, similarly to the transition from early to mature mathematical cognition in children, who initially perform better when they can use fingers, but, after they grow in experience, gradually abandon finger representations without affecting accuracy[29].

Comparative analyses showed that the embodied strategy can represent a novel approach to increase efficiency in training deep neural networks also outside the contexts of robotics.

Importantly, this is the first time that cognitive developmental robotics is demonstrated to be effective against the standard approach in benchmark machine learning problem. Indeed, we saw performance improvements with other synthetic representations too, like the Cardinal Numerosity or, in some conditions, even vectors of randomly generated values. While the Cardinal Numerosity showed similar performance to the iCub fingers, the control model with random values often underperformed and was not significantly different from the other control conditions.

However, like their biological counterparts, the robot's fingers seem better suited than other synthetic representations for simulating early mathematical education in interactive scenarios with a child-like robot. Indeed, they are more likely to be presented and intuitively understood by humans without requiring advanced communication by the robot. For instance, examples of spoken digits can be proactively acquired by the robot by showing finger representations and asking: "what number is this?" Also, human teachers may simply open and close the robot's fingers to instruct the robot or correct the representation in case of error.

In conclusion, we believe that these findings validate the cognitive developmental robotics approach as a tool for implementing the embodied cognition ideas, and for further developing machine intelligence while making artificial learning more intuitive for humans.

Methods

The Google Tensorflow Speech commands dataset

To provide a realistic numerical challenge to our models, we used a new publicly available benchmark in machine learning: the Google Tensorflow Speech commands dataset[50]. The accompanying paper reports a basic benchmark of 88.2% (on the whole database of spoken commands) and the best result reported in the Leaderboard of the 2017 TensorFlow Speech Recognition Challenge[57] was 91.06% on the first version of the database.

Here, we used the second version of the database, which contains 105,829 one-second long utterances of 35 short words, by thousands of different people. The digits are around one-third of the database, which includes 34,856 spoken digits from 1 to 9 that we randomly split into 80% (27,884) training set and 20% (6,972) testing set. For Scenario 1, we aimed at standard child development scenario, where analysis of number word frequencies in natural corpora[54,58] suggests that smaller numbers are more frequent than larger numbers. This implied that frequencies of digits should decrease proportionally to their numerical magnitude. For this reason, we created an ad-hoc training dataset with a Zipfian distribution by extracting examples with frequency 1/N, where N is the numerical value of the digit. The distribution of the examples for each scenario is presented in Table 4.

The original files are 16-bit little-endian PCM-encoded in the WAVE format at a 16KHz sample rate. For our experiments, these were preprocessed using a standard approach the makes use of the short-time Fourier transform (STFT). The resulting samples are 90x63 STFT spectrograms, which were rescaled to be in the range [0,1], which is optimal for training artificial neural networks.

Note that in this study we didn't include the zeros because there is no finger representation that can be associated. This is coherent with all empirical studies about embodied arithmetic in the literature, where tasks usually don't include the zero, e.g.[11,24], because of its special status among numbers.

Table 4. Dataset distributions.
The table gives the number of the spoken digits from 1 to 9 in the test set and train sets for each Scenario. In Scenario 1, the train is created extracting examples from the original dataset in such a way the distribution was Zipfian, then the pre-train was derived from the train using the same number of examples for each digit. In Scenario 2, from the full training dataset, we derived various subsets with the same distribution of the original.

Digit	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**TOT**
Test	*788*	*708*	*763*	*733*	*811*	*821*	*794*	*742*	*812*	***6,972***
Scenario 1: Initial learning (reduced dataset)										
Pre-train (uniform)	*344*	*344*	*344*	*344*	*344*	*344*	*344*	*344*	*344*	***3,096***
Train (Zipfian)	*3102*	*1551*	*1034*	*775*	*620*	*517*	*443*	*387*	*344*	***8,773***
Scenario 2: Original semi-uniform distribution (full dataset)										
Train (100%)	*3102*	*3172*	*2964*	*2995*	*3241*	*3039*	*3204*	*3045*	*3122*	***27,884***

In our experiments, the database is split into smaller sets in order to simulate a gradual course of education typical for the children, by investigating the models' performance of varying size of training examples. This also allows gaining information on the efficacy and efficiency of the proposed embodied strategy in scenarios where examples are scarce. The division is obtained simply by taking a sequence of consecutive examples from the main database. The sequences are varied among the 32 runs, **Algorithm 1** describes the procedure. For Scenario 1, the sizes considered were 25%, 50% and 100% of the Zipfian dataset. For Scenario 2, we aimed at a more fine-grained analysis with 7 set sizes. The 3 smallest sizes were selected as multipliers of the minibatch size (32), while the others were respectively 10%, 20%, 50% and 100% of the training dataset.

The source code of the implementation can be found in the GitHub repository (files: generator.py; dataset.py; zipfian.py - link in the acknowledgements). Supplementary Figure 2 shows examples of the spectrograms.

Simulated internal representations

Three fixed codes are used to simulate the embodied representations of the digits from 1 to 9:

- The *cardinal numerosity*, which represents a cardinal number N with the same quantity of ones. If the number of available digits for the representation is greater than N, then zeros are included to fill. In our case, we used 9 digits to represent the numbers, with 1 represented as *100000000* and 9 as *111111111* and, for instance, the representation of N=4 is *111100000*. The cardinal numerosity has cognitive plausibility and it has been shown to facilitate learning for symbolic and ordinal representations [59]. Indeed, neural network models based on the numerosity representation can account for a wide range of empirical data [46]. In the context of this article, the cardinal numerosity is an abstract representation, however, it could synthetically represent a set of objects that the robot can produce, which an alternate method to the use of fingers while learning about numbers.
- *The iCub robot encoder values for the finger representations.* The iCub is an open-source humanoid robot platform designed to facilitate embodied artificial intelligence research[33]. The iCub provides motor proprioception (joint angles) of the fingers' motors, for a total of 7 degrees of freedom (DoF) for each hand as follows: 2 DoF for the thumb, index, and middle fingers, and one for controlling both ring and pinky fingers, which are coupled together[60]. However, this limitation is also common in human beings, who often can't freely move these two fingers independently[61]. To overcome the possible distortion by unbalanced representations, the contribution of the motors controlling two fingers is double; therefore, we have 16 inputs, we normalized in the [0,1] range. Pictures of the iCub finger representations are in Supplementary Figure 1, which shows the right hand. Note that the finger configurations of each hand are replicating the American Sign Language number representation from 1 to 5. Indeed, the representations with the left hand are specular, and they are used in addition to the fully open right hand to represent numbers from 6 (5+1) to 9 (5+4). The finger representations of the American Sign Language were selected to represent the embodied internal representation as an appropriate solution to a limitation of the iCub hand. Also, some physical limitation prevents some fingers to be fully opened or closed, e.g. the thumb, see the supplementary video of the iCub counting from 1 to 10. The numerical values of the encoders can be found in the file named "robot.cvs" in the "database" folder of the GitHub repository (the link is in the acknowledgements).
- *Random numbers* in the range [0,1] as "control" representations. In this case, 9 vectors of 16 random numbers are created and associated with the numbers. These representations are generated for each run and remain stable for the entire training. Random representations are included as a "control group" to confirm the performance contribution is due to the embodied signals rather than other factors.

It should be noted that arbitrary random gestures could be suitable in computer simulations for control conditions, but it is unlikely that they can be successful in realistic scenarios because they will require preliminary training to be executed and it is unlikely human teachers can be precise in repeating them, i.e. there will be significant noise and systematic errors to disrupt the training.

Deep Learning architecture for simulating embodied learning

To explore the embodied learning of numbers in the iCub robot, we designed a baseline and an embodied connectionist model for classifying spoken digits. These models are based on a Deep CNN classifier with 19 layers, of which the first 13 layers are shared between spoken digits recognition and embodied motor control. CNN is an essential part of the network for selecting the right features to present to the actual classifier, i.e. the hidden and classification layers, but they only account for 20-25% of the trainable parameters of our models.

Architectures based on CNN networks are naturally fit to implement the "transfer learning" approach because the convolutional layers can extract inherent properties from examples, which can be independent on the problem and, therefore, be generalized and used as a base for different problems. This strategy saves computational resources (time and memory) because the convolutional blocks have 73,632 parameters, which represent just 20.83% the full embodied model, which in total has 353,545 parameters when trained with the iCub fingers.

Table 5. Summary of the CNN architecture.

The rows reports the type, the size of the output, input and output links, the number of trainable parameters, the arguments and the initialization function for each layer. The baseline includes all the layers, except the 15 ("embodied"), which is part of the Embodied network only.

Layer	Type	Output Shape	Input Layer(s)	Output Layer(s)	N. Param.	Arguments	Initialization
1	Inputs	90x63	-	2		Range=[0,1]	
2	Conv2D	90x63	1	3	640	filters=64, size=3x3;	He uniform
3	Pooling	32x32	2	4		size=3x3; stride=3x2	
4	BatchNorm	30x32	3	5	256		
5	Dropout	30x32	4	6		probability=0.25	
6	Conv2D	15x16	5	7	36928	filters=64, size=3x3;	He uniform
7	Pooling	15x16	6	8		size=3x3; stride=2x2	
8	BatchNorm	15x16	7	9	256		
9	Dropout	15x16	8	10		probability=0.25	
10	Conv2D	8x8	9	11	18464	filters=32, size=3x3;	He uniform
11	Pooling	8x8	10	12		size=3x3; stride=2x2	
12	BatchNorm	8x8	11	13	128		
13	Dropout	8x8	12	14		probability=0.25	
14	Flatten	2048	13	15&16			
15	Embodied	9 or 16	14	18		function=**Sigmoid**	Glorot uniform
16	Dense	128	14	17	262272	function=**ReLU**	Glorot uniform
17	BatchNorm	128	16	18	512		
18	Dropout	128	15&17	19		probability=0.5	
19	Dense	9	19	-	1161-1305	function=**Softmax**	Glorot uniform

The baseline is a relatively simple but effective deep CNN architecture that includes a sequence of 3 classical two-dimensional convolutional blocks, which altogether have 73,632 trainable parameters, while the baseline network includes a total of 320,041 trainable parameters. The embodied model is created by extending the baseline by adding a dense layer named "Embodied" (15 in Table 5), which serves both as an output for the embodied representations associated to the spoken digits and provide these representations as an input to the final classification layer (19). With the additional layer, there are two weighted connections, which extend the number of trainable parameters to 353,545. The embodied model is trained in two steps: first, the shared CNN blocks and the "embodied layer" (red in Figure 1, Layers 1-15 in Table 5) are trained to associate digit images to embodied representations, then the remaining layers (blue in Figure 1, 16-19 in Table 5) are connected and the full model is tuned to classify the spoken digits. In the full training phase, the loss is the weighted sum of the losses for the two outputs, both weighted 1.0. Unless otherwise stated, the layers are regular densely connected layers, where all units are connected to the others.

From the machine learning point of view, the embodied strategy could be also seen as a bio-inspired alternative to the "auxiliary" classifiers that were introduced in the Google Inception network to prevent the middle part of the network from "dying out" because of the limitations of backpropagation algorithms in propagating the error through the many layers of deep CNN[62].

The network parameters, e.g. number of units for each layer, were set on the baseline via a trial-and-error procedure using the final performance (accuracy) as a criterion for the selection. In fact, the baseline model, when fully trained, can achieve a final accuracy of over 97% with the test set.

Neural Network Implementation details

To improve the understanding of the article, we give an overview of the layers that compose the architectures and the methods used for learning in the following subsections. The overview is not intended to be exhaustive; the aim is to facilitate the general understanding of the methods used in this work and to point the inexperienced reader towards the relevant sources. The models were implemented, trained and tested using python and Keras 2.2.4[63] high-level APIs running on top of TensorFlow 1.8.0[64]. Greater detail can be found in the documentation of these tools available from the respective websites.

The ReLU layer

The label ReLU is commonly used to identify a layer with Rectified Linear Units, which apply a non-saturating activation function:

$$ReLU(x) = \max(0, x)$$

It increases the nonlinear properties of the decision function and of the overall network. In our models, the ReLU layers proved to be more effective than the classical sigmoid.

The Sigmoid layer

A sigmoid layer is formed of units with the most common transfer function for artificial neural networks, the sigmoid:

$$sigmoid(x) = \frac{e^x}{e^x + 1}$$

The Convolutional layer

Convolutional layers characterize the Convolutional Blocks, they are one of the most successful instruments in building deep learning architectures[65,66], which represent the current state of the art in computer vision and they are inspired by biological organization and process of the visual cortex in animals and humans[67]. The convolutional layers enable artificial neural networks to extract the main features from an image and recognize patterns by learning about the shapes of objects.

In a convolutional layer, each unit is repeatedly activated by a receptive field (typically rectangular), which is connected via a weight vector (a filter) to single input sensory neurons. The receptive field is shifted step by step across a 2-dimensional array of input values, e.g. the frequency for a time step.

The Max Pooling layer

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling amongst which max pooling is the most common because it has been shown that max pooling can give a better performance than other pooling operations[68]. The Pooling layer partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. The pooling operation provides another form of translation invariance.

The Classification layer

The final layer of the models ("Classification_Layer") uses the *softmax* transfer function that naturally ensures all output values are between 0 and 1, and that their sum is 1. The output of a *softmax* classifier is a probability/likelihood; a classification output layer is also trained to transform the probabilities into one of the classes. The total number of classes considered in our experiment is 9, which corresponds to the digits from 1 to 9.

The *softmax* function used is as follows:

$$softmax(\boldsymbol{x}, i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

where the vector ***x*** is the net input to a *softmax* node, and *n* is the number of nodes in the *softmax* layer.

The Other Layers

The other layers included in our models are:

- **The Dropout layer**, which operates by randomly dropping a fraction of input at each update at training time. Dropout layers help to prevent overfitting[69]. The drop rate of the dropout layers in the three Convolutional blocks is 0.25, while the last drop rate is 0.5.
- **The Flatten layer**, which reshapes multidimensional inputs into one-dimensional output vector. This layer doesn't apply a transfer function and it is transparent to the learning, but it is needed to enable the transition from Convolutional layers to standard layers.
- **The Batch Normalization (BatchNorm) layer,** which scales the output of the previous layer by standardizing the activations of each input variable per mini-batch. This has the effect of inducing a more predictive and stable behaviour of the gradients, which allows faster training[70].

The dropout and batch normalization layers are inserted to reduce overfitting and improve generalization performance.

Initializers

The layer initializers used were:

- **He Uniform**[71], which uses a uniform distribution within [$-\sqrt{6/Nw_{in}}$, +$\sqrt{6/Nw_{in}}$] where Nw_{in} is the number of inputs of the layer.
- **Glorot Uniform**[72], which draws samples from a uniform distribution within [$-\sqrt{6/(Nw_{in}+Nw_{out})}$, +$\sqrt{6/(Nw_{in}+Nw_{out})}$] where Nw_{in} is the number of inputs of the layer, while Nw_{out} is the number of outputs.

Algorithms for training the networks

After some preliminary tests with the optimization algorithms included in the Keras framework, we selected two adaptive learning methods, based on stochastic gradient descent, for training the models: RMSprop and Adam. As recommended, we left the parameters of this optimizer at their default values, which follow those provided in the original publications cited below. The training was executed in mini-batches of 16 or 32 examples, full and pre-training respectively. The use of mini-batches proved to improve the generalization of the network, i.e. the accuracy in the test set.

The Root Mean Square Propagation (RMSprop) method[74] is a gradient-based method that maintains per-parameter learning rates, which are divided by a moving average $\hat{v}(\theta,t)$ of the squared gradient for each model parameter θ:

$$\theta(t+1)=\theta(t)-\frac{\eta}{\sqrt{\hat{v}(\theta,t)}}\cdot\frac{\partial L}{\partial\boldsymbol{\theta}}(t)$$

Where $\frac{\partial L}{\partial\boldsymbol{\theta}}(t)$ is the gradient of the loss function $L(t)$ at epoch *t*, η is the learning rate, which, in our experiments, has been set as *0.001*. The moving average $\hat{v}(\theta,t)$ is calculated as:

$$\hat{v}(\theta, t) = \gamma \cdot \hat{v}(\theta, t-1) + (1-\gamma) \cdot \left(\frac{\partial L}{\partial \theta}(t)\right)^2$$

where γ is 0.9 as suggested in [73]. RMSprop can be seen as a mini-batch version of Rprop[74].

The Adaptive Moment Estimation algorithm (Adam)[75] combines the advantages of RMSprop and Adagrad. In fact, Adam is widely used in the field of deep learning because it is fast and achieves good results. Like the RMSprop, Adam also makes use of a moving average of the squared gradient $\hat{v}(\theta, t)$, but it keeps an exponentially decaying average of past gradients $\hat{m}(\theta, t)$, similar to the momentum. Adam's parameter update is given by:

$$\theta(t+1) = \theta(t) - \eta \frac{\hat{m}(\theta, t)}{\sqrt{\hat{v}(\theta, t)}}$$

Specifically, $\hat{v}(\theta, t)$and $\hat{m}(\theta, t)$ are calculated using the parameters β_1 and β_2 to control the decay rates of these moving averages:

$$\hat{m}(\theta, t+1) = \frac{m(\theta,t)}{1-\beta_1^t} \text{ where } m(\theta, t) = \beta_1 \cdot m(\theta, t-1) + (1-\beta_1) \cdot \frac{\partial L}{\partial \theta}(t)$$

$$\hat{v}(\theta, t+1) = \frac{v(\theta,t)}{1-\beta_2^t} \text{ where } v(\theta, t) = \beta_2 \cdot v(\theta, t-1) + (1-\beta_2) \cdot \left(\frac{\partial L}{\partial \theta}(t)\right)^2$$

Note that β_1^t and β_2^t denote the parameters β_1 and β_2 to the power of t.

Good default settings are $\eta = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. These values are used in our experiments.

In our experiments, the Adam algorithm was used in training the final classifiers, while the RMSprop was used in the regression tasks where it showed the best performance, i.e. when the learning target was the embodied representations to pre-train the CNN layers.

Loss function

The Loss function $L(t)$ was the Cross-entropy function, which computes the performance given by network outputs and targets in such a way that extremely inaccurate outputs are heavily penalized, while a very small penalty is given to almost correct classifications.

The calculation of the Cross-entropy depends on the task: Categorical H_C when classifying into the number classes; Binary H_B predicting the embodied representations.

In the case of classification, the output $\boldsymbol{p}$ is a categorical vector of N probabilities that represent the likelihood of each of the N classes with $\sum \boldsymbol{p} = 1$, while $\widetilde{\boldsymbol{y}}$ is a one-hot encoded vector (1 for the target class, 0 for the rest). The Categorical cross-entropy H_C is calculated as the average of the cross-entropy of each pair of output-target elements (classes):

$$H_C = \frac{1}{N} \sum_{i=1}^{N} -\widetilde{y_i} \cdot \log(p_i)$$

When the target is the embodied representation, the output is a vector $\boldsymbol{z}$ of K independent elements. The cross-entropy can be calculated considering 2 binary classes: one corresponds to the target value, zero otherwise. In this case, the loss function is calculated using the binary cross-entropy expression:

$$H_B = \frac{1}{K} \sum_{i=1}^{K} -\widetilde{y_i} \cdot \log(z_i) - (1 - \widetilde{y_i}) \cdot \log(1 - z_i)$$

Training and testing procedures

We ran the training 32 times with random parameters initializations. The stopping criterion was a fixed number of epochs (25 for the first experiment and all the pre-training, 50 for the second). The final performance is calculated as the average of the accuracies on the test set after the epoch with the lowest loss for each run. The following pseudo-code summarizes the training and testing procedure for our experiments.

For clarity, details on the statistical analysis used are given in the Supplementary Information.

Algorithm 1. Pseudo-algorithm of the training procedure.

```
N = 27884 #number of training examples

For $i \in [1,32]$ #number of runs for each model was 32

Random = generate_random_normal_distribution(9;[0,1]) #generates 9 representations
                                                       each has 16 random values in [0,1]

    For each $K \in \{\{2193,4386,8773\}$ #Scenario 1: Zipfian distribution
|{128,512,1024,2788,5576,13942,27884}} #Scenario 2: standard semi-uniform distribution

        train_interval = [(K*((i-1)%(int(N/K))),K*(i%int(N/K))}] #the train interval covers
                                                                  as much of the training
                                                                  dataset as possible.

        train(convolutional_blocks,
                optimizer=rmsprop,
                epochs=25,
                mini-batches_size=32
                input=MNIST_TRAIN[pre-train_interval],
                output=(random|num_mag|robot)) #embodied architecture only

        train(full_model, optimizer=adam,
                epochs=(25|50) #25 for experiment 1; 50 for experiment 2
                mini-batches_size=16
                input= SPOKEN_DIGITS_TRAIN[train_interval],
                main_output=(classes),embodied_output=(random|num_mag|robot),
                loss=1.0*classifier_loss+1.0*embodied_loss) #full training

        $accuracy_i(k, epoch)$ =evaluate(full_model, input= SPOKEN_DIGITS_TEST)
```

Note that, in the case of 128,512 and 1024 examples, all runs had a different portion of the training set, while in the other cases they cycle among 10,5,2 folds of the training set.

References

1. Glenberg, A. M. Embodiment as a unifying perspective for psychology. *Wiley Interdiscip. Rev. Cogn. Sci.* **1,** 586–596 (2010).
2. Wilson, M. Six Views of Embodied Cognition. *Psychon. Bull. Rev.* **9,** 625–636 (2002).
3. Pfeifer, R., Bongard, J. & Grand, S. *How the body shapes the way we think: a new view of intelligence*. (MIT press, 2007).
4. Shapiro, L. *The Routledge handbook of embodied cognition*. (Routledge, 2014).
5. Dackermann, T., Fischer, U., Nuerk, H. C., Cress, U. & Moeller, K. Applying embodied cognition: from useful interventions and their theoretical underpinnings to practical applications. *ZDM - Math. Educ.* **49,** 545–557 (2017).
6. Nieder, A. The neuronal code for number. *Nat. Rev. Neurosci.* **17,** 366 (2016).
7. Barrow, J. D. *New theories of everything: the quest for ultimate explanation*. (Oxford University Press, 2008).
8. Lakoff, G. & Nuñez, R. *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. (Basic Books, 2000).
9. Soylu, F., Lester Jr., F. K. & Newman, S. D. You can count on your fingers: The role of fingers in early mathematical development. *J. Numer. Cogn.* **4,** 107–135 (2018).
10. Goldin-Meadow, S., Levine, S. C. & Jacobs, S. *Gesture's role in learning arithmetic* (eds. Edwards, L. D., Ferrara, F. & Moore-Russo, D.) 50-64 (Information Age Publishing, 2014).
11. Gunderson, E. A., Spaepen, E., Gibson, D., Goldin-Meadow, S. & Levine, S. C. Gesture as a window onto children's number knowledge. *Cognition* **144,** 14–28 (2015).
12. Di Luca, S. & Pesenti, M. Masked priming effect with canonical finger numeral configurations. *Exp. Brain Res.* **185,** 27–39 (2008).
13. Domahs, F., Kaufmann, L. & Fischer, M. H. *Handy numbers: Finger counting and numerical cognition*. (Frontiers E-books, 2014).
14. Alibali, M. W. & DiRusso, A. A. The function of gesture in learning to count: More than keeping track. *Cogn. Dev.* **14,** 37–56 (1999).
15. Di Luca, S. & Pesenti, M. Finger numeral representations: more than just another symbolic code. *Front. Psychol.* **2,** 272 (2011).
16. Sixtus, E., Fischer, M. H. & Lindemann, O. Finger posing primes number comprehension. *Cogn. Process.* **18,** 237–248 (2017).
17. Klein, E., Moeller, K., Willmes, K., Nuerk, H.-C. & Domahs, F. The Influence of Implicit Hand-Based Representations on Mental Arithmetic. *Front. Psychol.* **2,** 197 (2011).
18. Butterworth, B. *The Mathematical Brain*. (McMillan, 1999).
19. Peters, L. & De Smedt, B. Arithmetic in the developing brain: A review of brain imaging studies. *Dev. Cogn. Neurosci.* **30,** 265–279 (2018).
20. Andres, M., Michaux, N. & Pesenti, M. Common substrate for mental arithmetic and finger representation in the parietal cortex. *Neuroimage* **62,** 1520–1528 (2012).
21. Kaufmann, L. *et al.* A developmental fMRI study of nonsymbolic numerical and spatial processing. *Cortex* **44,** 376–385 (2008).
22. Gracia-Bafalluy, M. & Noël, M.-P. Does finger training increase young children's numerical performance? *Cortex* **44,** 368–375 (2008).
23. Sato, M., Cattaneo, L., Rizzolatti, G. & Gallese, V. Numbers within our hands: Modulation of corticospinal excitability of hand muscles during numerical judgment. *J. Cogn. Neurosci.* **19,** 684–693 (2007).
24. Tschentscher, N., Hauk, O., Fischer, M. H. & Pulvermüller, F. You can count on the motor

cortex: finger counting habits modulate motor cortex activation evoked by numbers. *Neuroimage* **59,** 3139–48 (2012).

25. Alibali, M. W. & Nathan, M. J. Embodiment in mathematics teaching and learning: Evidence from learners' and teachers' gestures. *J. Learn. Sci.* **21,** 247–286 (2012).
26. Alibali, M. W. *et al.* How Teachers Link Ideas in Mathematics Instruction Using Speech and Gesture: A Corpus Analysis. *Cogn. Instr.* **32,** 65–100 (2014).
27. Cook, S. W. & Goldin-Meadow, S. The Role of Gesture in Learning: Do Children Use Their Hands to Change Their Minds? *J. Cogn. Dev.* **7,** 211–232 (2006).
28. Cook, S. W., Duffy, R. G. & Fenn, K. M. Consolidation and Transfer of Learning After Observing Hand Gesture. *Child Dev.* **84,** 1863–1871 (2013).
29. Jordan, N. C., Kaplan, D., Ramineni, C. & Locuniak, M. N. Development of number combination skill in the early school years: When do fingers help? *Dev. Sci.* **11,** 662–668 (2008).
30. Moeller, K., Martignon, L., Wessolowski, S., Engel, J. & Nuerk, H.-C. Effects of Finger Counting on Numerical Development – The Opposing Views of Neurocognition and Mathematics Education. *Front. Psychol.* **2,** 328 (2011).
31. Cangelosi, A. & Schlesinger, M. *Developmental robotics: From babies to robots*. (MIT Press, 2015).
32. Sakagami, Y. *et al.* The intelligent ASIMO: system overview and integration. in *IEEE/RSJ International Conference on Intelligent Robots and Systems,* **3,** 2478–2483 (IEEE, 2002).
33. Sandini, G., Metta, G. & Vernon, D. The iCub Cognitive Humanoid Robot: An Open-System Research Platform for Enactive Cognition. in *50 years of artificial intelligence* (eds. Lungarella, M., Pfeifer, R., Iida, F. & Bongard, J.) 358–369 (Springer-Verlag, 2007).
34. Theodorou, A., Wortham, R. H. & Bryson, J. J. Designing and implementing transparency for real time inspection of autonomous robots. *Conn. Sci.* **29,** 230–241 (2017).
35. Asada, M. *et al.* Cognitive developmental robotics: A survey. *IEEE Trans. Auton. Ment. Dev.* **1,** 12–34 (2009).
36. Cangelosi, A. *et al.* Embodied language and number learning in developmental robots. in *Conceptual and Interactive Embodiment: Foundations of Embodied Cognition* (eds Fischer, M. H. and Coello, Y.) vol. **2,** 275–293 (Routledge, 2016).
37. Di Nuovo, A., Marocco, D., Di Nuovo, S. & Cangelosi, A. Autonomous learning in humanoid robotics through mental imagery. *Neural Networks* **41,** 147–155 (2013).
38. Di Nuovo, A., Marocco, D., Di Nuovo, S. & Cangelosi, A. Embodied Mental Imagery in Cognitive Robots. in *Springer Handbook of Model-Based Science* (eds. Magnani, L. & Bertolotti, T.) 619–637 (Springer International Publishing, 2017).
39. Di Nuovo, A. & Jay, T. The development of numerical cognition in children and artificial systems: a review of the current knowledge and proposals for multi-disciplinary research. *IET Cogn. Comput. Syst.* **1**, 2 – 11, (2019).
40. Rucinski, M., Cangelosi, A., and Belpaeme, T. Robotic model of the contribution of gesture to learning to count. in *Proceedings of IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-Epirob),* 1-6 (IEEE, 2012).
41. De La Cruz, V. M., Di Nuovo, A., Di Nuovo, S. & Cangelosi, A. Making fingers and words count in a cognitive robot. *Front. Behav. Neurosci.* **8,** 13 (2014).
42. Di Nuovo, A., De La Cruz, V. M. & Cangelosi, A. Grounding fingers, words and numbers in a cognitive developmental robot. in *IEEE Symposium on Cognitive Algorithms, Mind, and Brain (CCMB)* 9–15 (IEEE, 2014).
43. Di Nuovo, A., De La Cruz, V. M., Cangelosi, A. & Di Nuovo, S. The iCub learns numbers: An

embodied cognition study. in *International Joint Conference on Neural Networks (IJCNN 2014)* 692–699 (IEEE, 2014).
44. Di Nuovo, A., De La Cruz, V. M. & Cangelosi, A. A Deep Learning Neural Network for Number Cognition: A bi-cultural study with the iCub. in *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* 320–325 (2015).
45. Di Nuovo, A. An Embodied Model for Handwritten Digits Recognition in a Cognitive Robot. in *IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB)* 1–6 (IEEE, 2017).
46. Zorzi, M., Stoianov, I., Umiltà, C. & Umilta', C. Computational modeling of numerical cognition. In *The handbook of mathematical cognition* (ed. Campbell, J.) 67–84 (Psychology Press, 2005).
47. Di Nuovo, A. Long-short term memory networks for modelling embodied mathematical cognition in robots. in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)* 1–7 (IEEE, 2018).
48. Domahs, F., Krinzinger, H. & Willmes, K. Mind the gap between both hands: Evidence for internal finger-based number representations in children's mental calculation. *Cortex* **44,** 359–367 (2008).
49. Davis, S., Tsagarakis, N. G. & Caldwell, D. G. The initial design and manufacturing process of a low cost hand for the robot icub. in *8th IEEE-RAS International Conference on Humanoid Robots* 40–45 (IEEE, 2008).
50. Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. Preprint at https://arxiv.org/abs/1804.03209 (2018).
51. Sharif Razavian, A., Azizpour, H., Sullivan, J. & Carlsson, S. CNN features off-the-shelf: an astounding baseline for recognition. in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* 806–813 (IEEE, 2014).
52. Gallistel, C. R. & Gelman, R. Preverbal and verbal counting and computation. *Cognition* **44,** 43–74 (1992).
53. Gelman, R. & Gallistel, C. R. *The child's understanding of number*. (Harvard University Press, 1986).
54. Piantadosi, S. T. Zipf's word frequency law in natural language: a critical review and future directions. *Psychon. Bull. Rev.* **21,** 1112–1130 (2014).
55. Laski, E. V & Siegler, R. S. Is 27 a Big Number? Correlational and Causal Connections Among Numerical Categorization, Number Line Estimation, and Numerical Magnitude Comparison. *Child Dev.* **78,** 1723–1743 (2007).
56. Gunderson, E. A., Spaepen, E. & Levine, S. C. Approximate number word knowledge before the cardinal principle. *J. Exp. Child Psychol.* **130,** 35–55 (2015).
57. Tensorflow speech recognition challenge. Retrieved from https://www.kaggle.com/c/tensorflow-speech-recognition-challenge (*2018)*
58. Dehaene, S. & Mehler, J. Cross-linguistic regularities in the frequency of number words. *Cognition* **43,** 1–29 (1992).
59. Stoianov, I., Zorzi, M., Becker, S. & Umilta, C. Associative arithmetic with Boltzmann Machines: The role of number representations. In *International Conference on Artificial Neural Networks (ICANN)* 277-283 (Springer, 2002).
60. Schmitz, A. *et al.* Design, realization and sensorization of the dexterous iCub hand. in *10th IEEE-RAS International Conference on Humanoid Robots* 186–191 (IEEE, 2010).
61. Lang, C. E. & Schieber, M. H. Human Finger Independence: Limitations due to Passive Mechanical Coupling Versus Active Neuromuscular Control. *J. Neurophysiol.* **92,** 2802–

2810 (2004).

62. Szegedy, C. *et al.* Going deeper with convolutions. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 1–9 (IEEE, 2015).
63. Chollet François. Keras: The Python Deep Learning library. (2018). Available at: http://keras.io.
64. GoogleResearch. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. (2018). Available at: https://www.tensorflow.org.
65. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks* **61,** 85–117 (2015).
66. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015).
67. Fukushima, K. Artificial vision by multi-layered neural networks: Neocognitron and its advances. *Neural Networks* **37,** 103–119 (2013).
68. Scherer, D., Müller, A. & Behnke, S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. in *International Conference on Artificial Neural Networks (ICANN)* 92–101 (Springer, 2010).
69. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15,** 1929–1958 (2014).
70. Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. How does batch normalization help optimization? in *Advances in Neural Information Processing Systems* (NIPS) 2483–2493 (2018).
71. He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. in *Proceedings of the IEEE international conference on computer vision* 1026–1034 (IEEE, 2015).
72. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. in *International Conference on Artificial Intelligence and Statistics* 249–256 (2010).
73. Ruder, S., An overview of gradient descent optimization algorithms. Preprint available at https://arxiv.org/abs/1609.04747.
74. Riedmiller, M. & Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks* 586–591 (IEEE, 1993).
75. Kingma, D. P. & Ba, J. L. Adam: a Method for Stochastic Optimization. Preprint available at https://arxiv.org/abs/1412.6980.

Acknowledgements

Funding: Alessandro Di Nuovo acknowledges the support of EPSRC through project grant EP/P030033/1 (NUMBERS). Di Nuovo also gratefully acknowledges the support of NVIDIA Corporation with the donation of the GeForce Titan X and the Tesla K40 GPUs used for this research.

Author contributions: ADN conceptualized the experiment, developed the methodology and designed the baseline artificial neural network architecture. ADN and JLM collaborated on the design of the embodied model. ADN implemented the source code, ran the simulations, validated the results, and wrote the first draft of the article. JLM provided relevant ideas from cognitive psychology and neuroscience and contributed to the discussion.

Competing interests: The authors declare that they have no competing interests.

Data and code availability: The data and source code for the models presented in this paper can be found in the GitHub repository: https://github.com/EPSRC-NUMBERS/EmbodiedCNN-Speech. A Supplementary Video of the iCub counting from 1 to 10 is also provided. The Google Tensorflow Speech Command database can be downloaded from http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz.