Understanding the Learning Effect of Approximate Arithmetic Training: What was Actually Learned?

Sizhu Cheng[1] **(scheng72@stanford.edu)**
Institute of Computational Mathematical Engineering, Stanford University, Stanford, CA 94305 USA

Arianna Yuan[1] **(xfyuan@stanford.edu)**
Department of Psychology, Department of Computer Science, Stanford University, Stanford, CA 94305 USA

Abstract

Previous studies have documented the learning effect of approximate arithmetic training on symbolic arithmetic skills in both preschoolers and adults. Particularly, Park and Brannon (2013, 2014) have trained participants to add two non-symbolic quantities (dot arrays) and showed that such training improved the participants' symbolic arithmetic skills. They argued that this finding suggested that training subjects to mentally manipulate non-symbolic quantities and practice nonverbal addition would result in enhanced symbolic arithmetic skills. However, we would like to propose an alternative explanation to account for their findings: instead of mentally manipulating visual dot arrays, participants might simply estimate the numerosities represented by the dot arrays and practice exact symbolic arithmetic during the seemingly non-symbolic arithmetic training. In that case, it would also lead to a better performance in subsequent symbolic arithmetic test. To verify our hypothesis, we implemented a neural network model to simulate their experiments. Our simulations confirmed that our explanation was sufficient to reproduce their psychological findings. The current work invites re-interpretation of the benefit of approximate arithmetic training and pushed us to think what cognitive component was improved exactly in approximate arithmetic training and what the relationship between approximate number processing and symbolic number processing really is.

Keywords: neural network model; approximate arithmetic; symbolic arithmetic; cognitive training

Introduction

Many researchers believe that infants exhibit a number sense without any knowledge of mathematical symbols (Feigenson, Dehaene, & Spelke, 2004). This non-symbolic representation of numbers, known as the Approximate Number System (ANS), is believed to be widely present in different species (Feigenson et al., 2004). Previous researchers have documented positive correlations between non-symbolic number processing and symbolic mathematical ability (Halberda, Mazzocco, & Feigenson, 2008; Libertus, Feigenson, & Halberda, 2011).

Particularly, Park and Brannon (2014) investigated the effect of approximate arithmetic training on symbolic arithmetic performance. Approximate arithmetic refers to a process of applying numerical manipulation, such as addition and subtraction, to non-symbolic quantities without counting (Park & Brannon, 2013). It turns out that such nonverbal arithmetic training did improve the exact symbolic arithmetic performance for adults and this result cannot be explained by just better approximate number comparison or a boost of short-term memory (Park & Brannon, 2014). Similar results were found in a recent study by Szkudlarek and Brannon (2018) in which participants were preschoolers.

In Park and Brannon (2013, 2014), participants were first presented with two images of dot array. They were asked to mentally add the two numerical quantities represented by the two dot arrays. Participants were then presented either (1) with a third image of dots and asked to report whether the numerical quantity represented in the third image is more than the sum of the numerical quantities in the first two images (More-or-Less comparison) or (2) with two images of dots and asked to select the one that matched the sum of the numerical quantities in the first two images (Same-or-Different comparison). Note that the participants were not given enough time to count the number of dots and they were not required to report the sum of the numerical quantities in the first two images. Park and Brannon (2013, 2014) found that training participants to perform these tasks lead to significant improvement in symbolic addition and subtraction. The authors took the findings as evidence for a causal link between non-symbolic number processing skills and symbolic number process skills. Many other researchers shared this view as well (for review, see Szkudlarek and Brannon (2017)). Particularly, they argued that nonverbal addition and subtraction was trained in such approximate arithmetic training, which led to improvement in symbolic addition and subtraction. However, we would like to propose an alternative explanation for such learning transfer: participants may simply estimate the numerosities of the first two dot arrays, perform symbolic arithmetic on the extracted numerosities, and compare the sum with the numerosity of the third dot array. In other words, the participants may practice symbolic arithmetic in the seemingly non-symbolic, arithmetic training (Julie & Silke, 2012; Gobel, Watson, Lervag, & Hulme, 2014; Mazzocco, Feigenson, & Halberda, 2011a, 2011b; Kolkman, Kroesbergen, & Leseman, 2013). Under that assumption, it would also lead to a better performance in later symbolic arithmetic test.

In this paper, we explore this alternative explanation by implementing this idea in a deep learning model. In our modeling framework, we simulated a similar approximate arithmetic training experiment as in Park and Brannon (2013, 2014), and evaluated our trained model on symbolic arithmetic problems. For simplicity, we only report our simulations on addition tasks. We compare the model

[1] equal contribution

performance on symbolic addition tasks before and after approximate arithmetic training and showed that the training resulted in significant learning outcome. Our simulations confirmed our hypothesis that an alternative explanation for the transfer learning effect of approximate arithmetic training is plausible, and future work is needed to understand what cognitive component is actually being trained in approximate arithmetic training and what the relationship between approximate number processing and symbolic number processing might be.

Method

Data For non-symbolic number stimuli, we randomly generated 10000 training images of dot arrays (see **Figure 1**, top). The number of dots per image ranged from 0 to 9, and each numerosity class contained 1000 images. We also generated a separate test dataset of 2000 dot array images. For symbolic numbers, we used the training digit images and the test digit images from the MNIST database (LeCun & Cortes, 2010), a dataset of handwritten digits with labels. In both cases, the training dataset and the evaluation dataset had no overlapping images.

For the comparison tasks, we train the model to compare numbers that ranged from 0 to 9. For More-or-Less comparison, the two comparands always differ by one, and the base rates of responding "More" and responding "Less" are the same (except for 0 and 9). In Same-or-Different comparison, every number n is compared with itself, and also with either $n+1$ or $n-1$, so that the base rates of responding "Same" and responding "Different" are the same.

Model Architecture

Approximate Arithmetic

We proposed a neural network architecture to perform the non-symbolic arithmetic task. As mentioned in the **Introduction**, we proposed an alternative account for how participants actually solved the approximate arithmetic problems in Park and Brannon (2013, 2014) experiments. Consistent with our hypothesis, in our model, the approximate arithmetic task was decomposed into a "recognition" part, an "addition" part and a "comparison" part. In our previous work, we have shown that decomposing numeric processing tasks into sub-tasks and building several neural networks that collectively work on the tasks could explain learning transfers in a situated math game environment (Yuan & McClelland, 2019). Inspired by those results, in the current paper our model also consists of several neural networks (**Figure 1**). First, we built an Image Classifier Network to extract the numerosity of the dot arrays. The Image Classifier Network learned to output a one-hot vector which represented the number of dots in the given input image. We built an Addition Network to add two numbers in the form of one-hot vectors/probability vectors to obtain the sum (also represented as a vector). Finally, we had a Comparison Network that learned to compare two numbers. During the approximate arithmetic training, the Addition Network is combined with the Comparison Network to form a so-called Combined Network that could perform the Same-or-Different/More-or-Less comparison. We next describe these networks in detail (**Figure 1**).

Image Classifier Network The Image Classifier Network was trained on dot array images to output a one-hot vector which represented the numerosities (from 0 to 9) of the dot array in the image. The Image Classifier Network processed the input images with a convolutional neural network (CNN). It contained four blocks of (a convolution layer with 3×3 filters and 20 output channels, a ReLU activation layer, a 2×2 max pooling layer), followed by two fully connected layers of size 120 and 84 with ReLU activation function. The activation of the last fully connect layers were then used to predict the one-hot vectors of numerosities. Here the ReLU non-linear activation function is used to restrict the magnitude output to be non-negative, i.e., $\mathrm{ReLU}(x) = \max(0, x)$.

Addition Network The Addition Network was a fully connected neural network of two hidden layers of size 81 (with ReLU activation function), which was trained to output a correct response for the addition task given two addends. Since we are only interested in modeling the relationship between non-symbolic number processing and symbolic number processing, but not multi-column addition, we restricted our scope to single digit processing. Therefore, we only considered numbers that ranged from 0 to 9, and the number pairs whose sum fell into the interval between 0 and 9.

Same-or-Different (SD) Comparison To perform the Same-or-Different comparison, we built a model that learned to differentiate very similar numerosities. The model learned to decide whether two input numbers were the same or different. Our SD Comparison Network is a fully connected layer of two fully connected layers of size 81 (with ReLU activation function) and its output layer contained two nodes, indicating the "Same" and the "Different" responses.

More-or-Less (ML) Comparison We also simulated the More-of-Less comparison. Instead of deciding whether two numerosities were the same or not, the More or Less (ML) Comparison Network needed to decide whether the first image had more dots than the second image or less dots. We trained a different comparison network. The architecture of the ML Comparison Network is exactly the same as the SD Comparison Network, except that the two output nodes now represented the "More" and the "Less" response, rather than the "Same" and the "Different" response.

Combined Network During the approximate arithmetic training, three dot array images were first processed by the Image Classifier Network to obtain three 10-dimensional probability vectors. The vectors of the first two images were then fed to the Addition network. The output of the Addition Network was again a probability vector representing the distribution over the possible sums. This output was concatenated with a third vector and was fed into the comparison network. This was to simulate the experimental procedure in Park and Brannon (2013, 2014). Regarding the output of the Combined Network, in Same-or-Different (SD) Comparison, the output layer had two nodes, indicating whether the number of all dots in the first two images was equal to the number of dots in the third image. In the More-or-Less (ML) Comparison, the output layer also had two nodes, indicating whether the number of all dots in the first two images was greater than the number of dots in the third image. The architect of the model and the information flow during the training process is illustrated in **Figure 1**.

Symbolic Arithmetic Network

When people solve symbolic arithmetic problems, it is reasonable to assume that humans first recognize the digits and then perform the actual addition or subtraction operation. To simulate this process, we decomposed the symbolic addition task into a "recognition" part and an "addition" part. We use the same Image Classifier Network and the same Addition Network as used in the approximate arithmetic training simulation to solve the symbolic arithmetic problems. For symbolic arithmetic task, we trained our Image Classifier Network on the digit images to output a 10-dimensional one-hot vector which represented the digit classes from 0 to 9.

Training Procedure

In this task, we first pre-trained our Image Classifier Network on both digit images and dot array images so that it achieved near-perfect accuracy. Also, it is reasonable to assume that participants in Park and Brannon (2013, 2014) already had some ability to add single-digit numbers. Therefore, we pre-trained our Addition Network to reach a decent level of accuracy. Particularly, the addition network for symbolic arithmetic is pre-trained for 60 epochs to reach ∼60% accuracy. We deliberately leave our Addition Network some room for improvement so that we could explore the role of non-symbolic arithmetic training and avoided a ceiling effect. Similarly, we also assumed that the participants were already equipped with some knowledge about comparison before they underwent the approximate arithmetic training, so we also pre-trained our SD and ML comparison network 7 epochs to reach ∼90% accuracy.

When simulating the comparison tasks, the probability vectors over the possible numerosities are fed to the Comparison Network. Since we pre-trained the Image Classifier to near-perfect performance, the probability vectors approximated 10-dimensional one-hot vectors, where the node standing for the correct label had activation of 1.0 and the rest of the nodes had activation of 0.0.

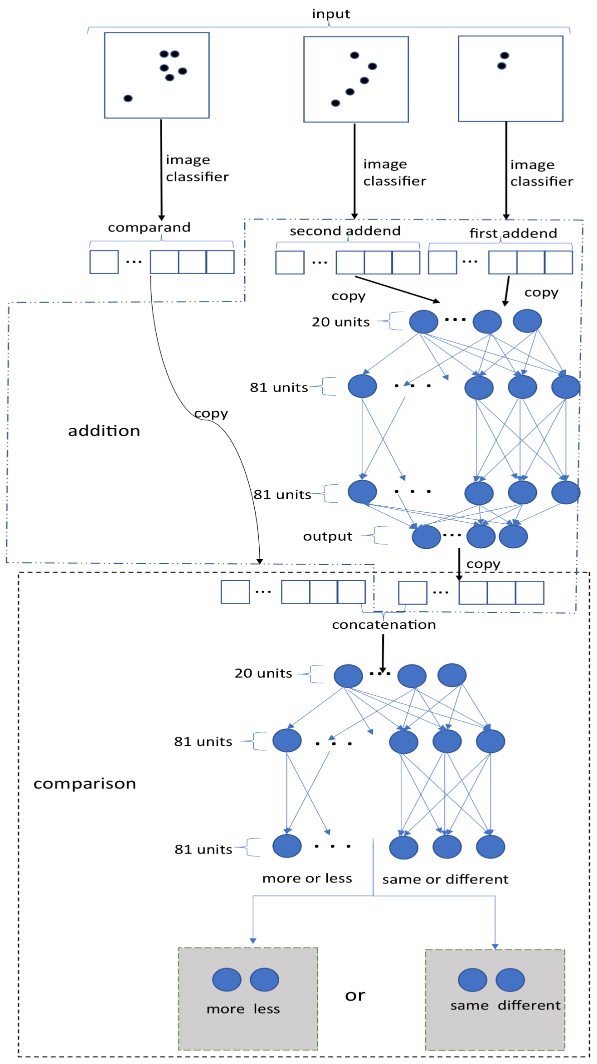


Figure 1: Model Architecture

In the joint-training phase, the Comparison Network is joint trained with the Addition Network. The first comparand was the probability vector obtained from the output layer of the downstream Addition Network, and the second comparand (the referent) was the probability vector obtained from the output layer of the downstream Image Classifier Network when applied to the image of the third dot array. The Combined Network (the Addition Network + the Comparison Network) were trained for 200 epochs, during which time period the network only had access to the "Same/Different" or "More/Less" labels. The model was tested on exact symbolic addition both before and after non-symbolic arithmetic training. The overall experiments were repeated for 10 runs with different random seeds. For both SD comparison and ML comparison, we used cross entropy loss and we ran backpropagation to train the model (Rumelhart, Hinton, Williams, et al., 1988). The learning rate for both SD and ML experiments were the same, as shown in **Table 1**. For evaluation, the accuracy was calculated as the percentage of correct examples among all predictions.

Results

For the sake of clarity, the learning curves of the Addition Network during the single-task training and the joint-training (when the Combined network was trained) are colored differently in the figures in this paper. Moreover, the horizontal axis indicates the cumulative epochs, i.e., the number of epochs that one network has been trained. In other words, since the Addition Network has been trained for 60 epochs before it was recruited in Combined Network for joint-training, then the epoch index of its joint training starts from 60 rather than 0. The arrows and the associated texts in the figures indicate the time point when the training condition changed.

Table 1: Learning Rates in different tasks

Model	Learning Rate
Pre-trained Addition	0.001
Pre-trained Comparison (SD/ML)	0.01
“Combined” Addition	0.0012
“Combined” Comparison	0.0001

Approximate Arithmetic Training with Same-or-Different Comparison

The learning curves of the networks, averaged from 10 runs, are shown in **Figure 2** and **Figure 3**. The error bars indicate standard deviations that are calculated from 10 runs and are plotted every 5 epochs.

It can be seen from **Figure 2** that the addition test loss reached the minimum around epoch 200, i.e. when the combined network was trained for ∼140 epochs. The prediction accuracy and the cross entropy loss were calculated both before the Combined Network was trained (i.e. pre-approximate arithmetic training) and after the Combined Network was trained (i.e. post-approximate arithmetic training). We ran a paired t-test and found that there was a significant improvement in test accuracy of the symbolic arithmetic problems after the approximate arithmetic training, $t = 18.427, p = 0.000$. Both pre-training and post-training test accuracies are shown in **Table 2**.

The learning curves of the Combined Network during approximate arithmetic training are shown in **Figure 4**, with the loss and the accuracy of the Addition Network and the whole Combined Network. We could see that as the training proceeded, both the test accuracies of the Addition Network and the whole Combined Network increased (**Figure 4**, bottom).

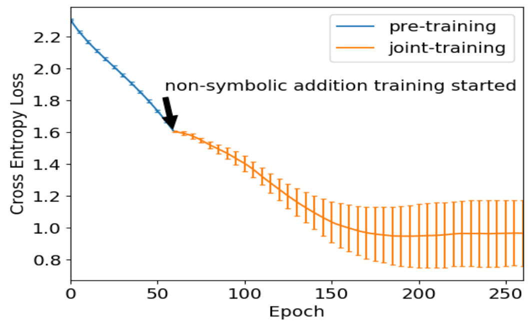


Figure 2: Test loss of the “Addition Network” in Same-or-Different comparison.

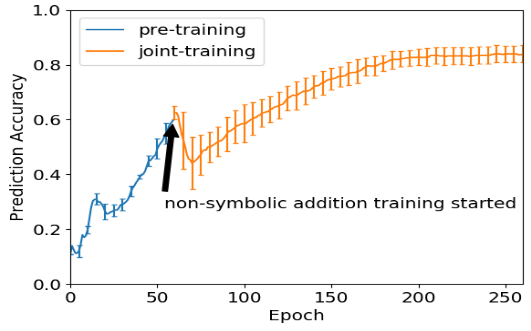


Figure 3: Test accuracy of the “Addition Network” in Same-or-Different comparison.

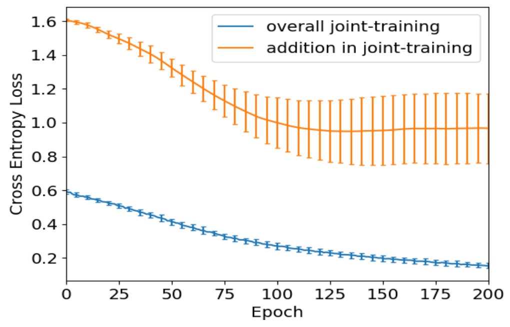
Approximate Arithmetic Training with More-or-Less Comparison

Similarly, in the More-or-Less Comparison case, the overall experiments were repeated 10 times with different random seeds. The learning curves of the networks, averaged from 10 runs, are shown in **Figure 5** and **Figure 6**. **Figure 7** shows the learning curves of the Addition Network and the overall Combined Network during the approximate arithmetic training.

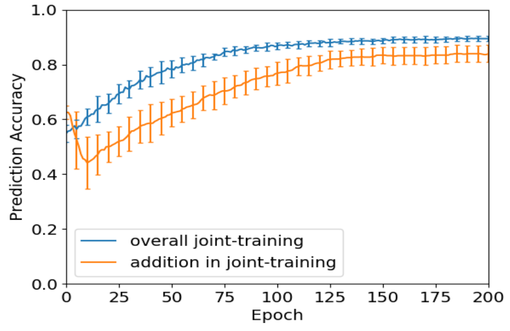
From **Figure 5** we can see that the addition test loss reaches the minimum around epoch 260, i.e. when the combined network was trained for ∼200 epochs. We also ran a paired t-test and found that there was a significant improvement in test accuracy of the symbolic arithmetic problems after the approximate arithmetic training, $t = 17.660, p = 0.000$. Both pre-training and post-training test accuracies are shown in **Table 2**. Though the learning curves demonstrated more variance across different runs in the More-or-Less comparison than in the Same-or-Different comparison, the model’s addition ability did improve in both cases.

Table 2: Symbolic addition performance pre- and post-approximate arithmetic training

Approximate Arithmetic Training	Addition Loss: Pre-training		Addition Loss: Post-training		Pre- vs. Post- *t*-statistics	Pre- vs. Post- *p*-value	Addition Accuracy: Pre-training		Addition Accuracy: Post-training		Pre- vs. Post- *t*-statistics	Pre- vs. Post- *p*-value
	mean	std	mean	std			mean	std	mean	std		
Same-or-Different Comparison	1.612	0.005	0.967	0.206	-9.983	0.000	0.597	0.030	0.837	0.033	18.427	0.000
More-or-Less comparison	1.613	0.007	0.962	0.212	-9.637	0.000	0.607	0.022	0.876	0.038	17.660	0.000



(a) Loss



(b) Accuracy

Figure 4: The test losses and the test accuracies of the Addition Network (orange) and the whole Combined Network (blue) in Same-or-Different comparison.

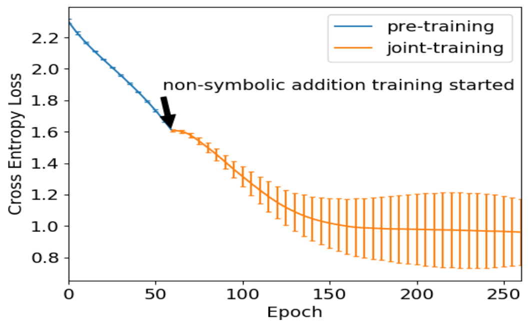


Figure 5: Test loss of the Addition Network in More-or-Less comparison.

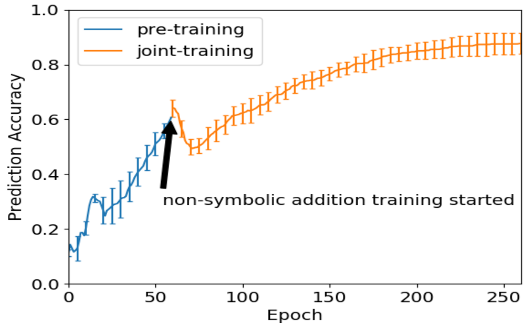
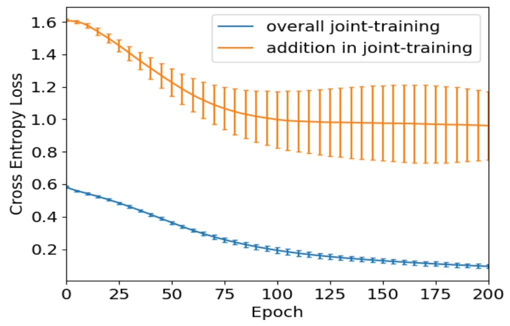


Figure 6: Test accuracy of the Addition Network in More-or-Less comparison.

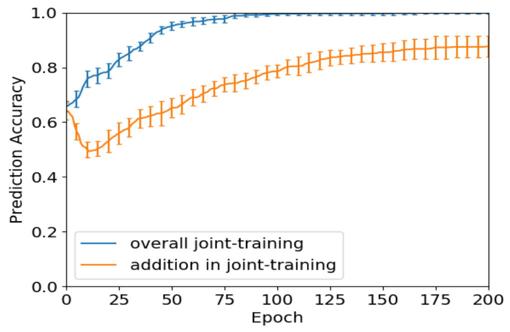
Discussion

Our model demonstrated improved addition accuracies in both the Same-of-Different comparison and the More-or-Less comparison, which is qualitatively aligned with the psychological findings in Park and Brannon (2013, 2014). This confirmed our hypothesis that if participants were doing exact symbolic addition rather than nonverbal addition during the approximate arithmetic training, their symbolic arithmetic skill would still be improved. This alternative explanation forces us to question the original interpretation of the findings in Park and Brannon (2013, 2014). For instance, how did participants mentally added the two dot arrays exactly? Did they combine two dot arrays and compare it with the third dot array, or extract the numerosities first and perform exact symbolic addition? Also, what was being compared in the non-symbolic arithmetic training? Were the non-symbolic combination of dots being compared or the symbolic sums? All these questions call for future investigation.

One of the limitations in the current study is that number distribution in our simulations did not match the one in Park and Brannon (2013, 2014). They used arithmetic problems that involved multi-digit numbers, but in our current study we only limited our scope on single digit addition. It is reasonable to believe that our conclusions should still hold when extended to multi-digit addition, because multi-digit addition does involve single digit addition. Future work is needed to confirm this intuition. Another limitation is that we



(a) Loss



(b) Accuracy

Figure 7: The test losses and the test accuracies of the Addition Network (orange) and the whole Combined Network (blue) in More-or-Less comparison.

did not simulate subtraction in our modeling work. Although computationally subtraction would be analogous to addition for the neural networks and our simulation results won't change if we replace addition with subtraction, in reality the behavioral patterns of human children on addition and subtraction are different. Therefore, it still worth simulating subtraction in future work to see if our model reproduces the learning effect on symbolic subtraction as well.

Finally, we want to note that despite the alternative explanation we proposed for the learning effect in Park and Brannon (2013, 2014), we did not mean to reject the strong connection between non-symbolic number processing and symbolic processing. We only want to advocate for a more cautious interpretation for any learning effect observed in cognitive training and invite other researchers to join our effort in understanding what exactly was learned in different intervention studies.

Acknowledgments

We thank James McClelland for insightful suggestions. A. Y. is funded by the Stanford Interdisciplinary Graduate Fellowship.

References

Feigenson, L., Dehaene, S., & Spelke, E. (2004). Core systems of number. *Trends in Cognitive Sciences*, *8*, 307–314.

Gobel, M. S., Watson, S. E., Lervag, A., & Hulme, C. (2014). Childrens arithmetic development: It is number knowledge, not the approximate number sense, that counts. *Psychological Science*, *25*(3), 789–798. doi: https://doi.org/10.1177/0956797613516471

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

Julie, C., & Silke, G. M. (2012). Impact of high mathematics education on the number sense. *PLoS ONE*, *7*(4:e33832). doi: https://doi.org/10.1371/journal.pone.0033832

Kolkman, M. E., Kroesbergen, E. H., & Leseman, P. P. M. (2013). Early numerical development and the role of non-symbolic and symbolic skills. learning and instruction. *Learning and Instruction*, *25*, 95-103.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Retrieved 2016-01-14 14:24:11, from `http://yann.lecun.com/exdb/mnist/`

Libertus, M. E., Feigenson, L., & Halberda, J. (2011). Preschool acuity of the approximate number system correlates with school math ability. *Developmental Science*, *14*(6), 1292–1300.

Mazzocco, M. M. M., Feigenson, L., & Halberda, J. (2011a). Impaired acuity of the approximate number system underlies mathematical learning disability (dyscalculia). *Child Development*, *82*(4), 1224-1237.

Mazzocco, M. M. M., Feigenson, L., & Halberda, J. (2011b). Preschoolers precision of the approximate number system predicts later school mathematics performance. *PLoS ONE*, *6*(9), 1-8.

Park, J., & Brannon, E. M. (2013). Training the approximate number system improves math proficiency. *Psychological Science*, *24*.

Park, J., & Brannon, E. M. (2014). Improving arithmetic performance with number sense training: An investigation of underlying mechanism. *Cognition*, *133*, 188–200.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, *5*(3), 1.

Szkudlarek, E., & Brannon, E. M. (2017). Does the approximate number system serve as a foundation for symbolic mathematics? *Language Learning and Development*, *13*(2), 171–190.

Szkudlarek, E., & Brannon, E. M. (2018). Approximate arithmetic training improves informal math performance in low achieving preschoolers. *Frontiers in Psychology*, 1–12.

Yuan, A., & McClelland, J. (2019). Modeling cross-task transfer by integrating verbal, visual, and grounded action components of natural number. In *Proceedings of the Annual Meeting of the Cognitive Science Society.*