A Number Sense as an Emergent Property of the Manipulating Brain

Supplementary Material

N. Kondapaneni and P. Perona
California Institute of Technology

November 16, 2022

A Additional Experiments

A.1 Controlling for spurious correlates of 'number'

One may be concerned that image properties other than the abstraction of 'object number' may be driving the quantity estimate. Indeed, *confound variables* , such as the count of pixels that are not black, are correlated with object number and might play a role in the model's ability to estimate the number of objects in the scene.

We controlled for this hypothesis by exploiting the natural variability of our test set images. We postulated that overall brightness, the area of the envelope of the objects in the image, and total pixel area of the objects would be good candidates as confound variables that might affect estimation.

To explore this possibility we analyzed close-call relative estimate tasks (e.g. 16 vs 18 objects), where errors are frequent both for our model and for human subjects, and, while holding the numbers of objects constant in the two scenes, we studied the behavior of error rates as a function of fluctuations in the confound variables. One would expect more errors when comparing image pairs where quantities that typically correlate with the number of objects are anticorrelated in the specific example (**Fig. S1**). Conversely, one would expect lower error rates when the confound variables are positively correlated with number.

In **Fig. S2** we plotted error rates vs each one of the confound variables. As is clear from the figure, we could not find large systematic biases even for extreme variations in the confound variables. In conclusion, we are unable to construct a convincing argument that any of the confound variables we studied is strongly implicated in the estimate of quantity.

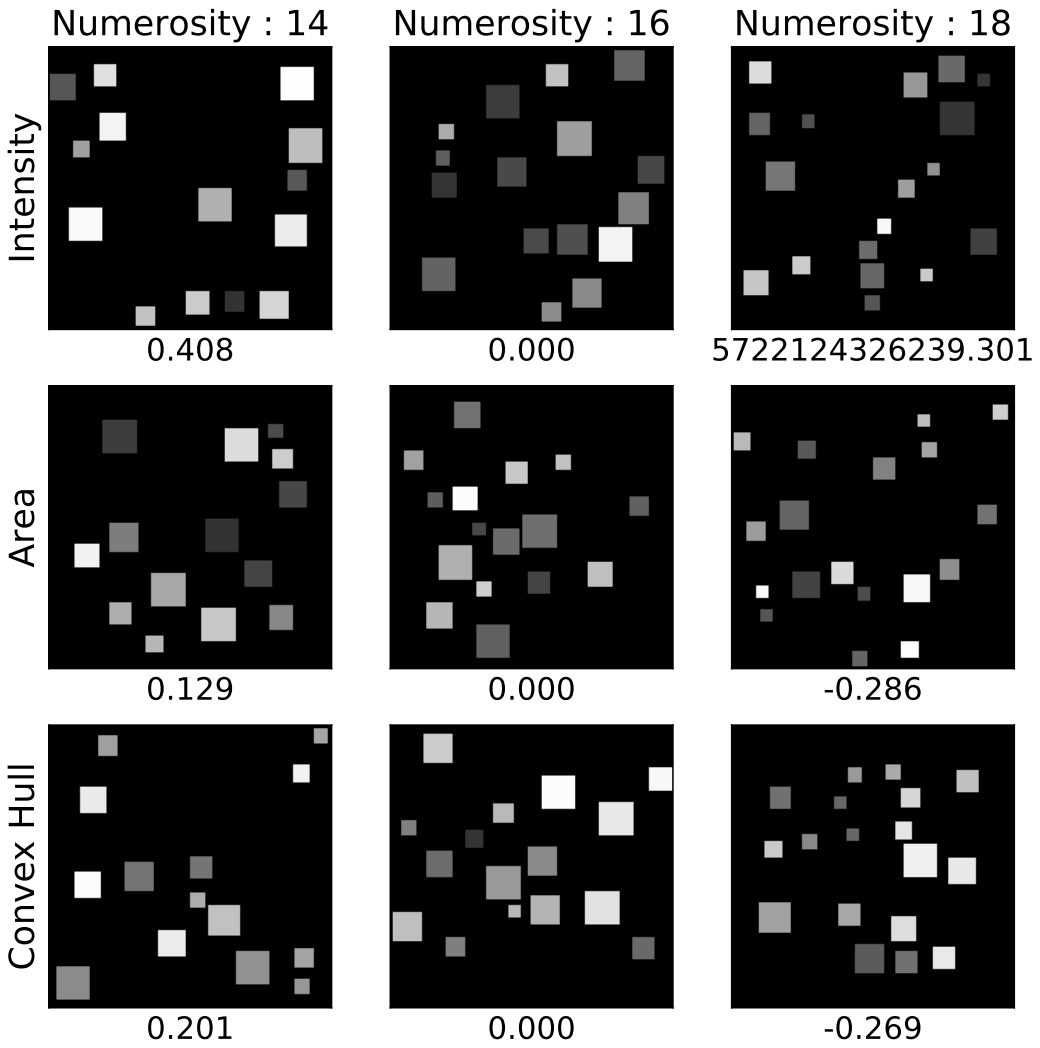

Figure S1: Sample images where covariates are anticorrelated with number. Examples of test images used in the experiments shown in **Figure S2**. In these examples the change in numerosity is anticorrelated with the three covariates we study (one covariate per row). The number below each plot shows the fractional difference from the value of the covariate in the reference image (center column). For example, in the top left, there is a 37.1% increase in average image intensity when compared to the intensity in the reference image (center column). In the last row, the scene with 18 objects has a smaller convex hull than the corresponding scenes with 14 and 16 objects. Our model correctly classifies all these examples in relative numerosity judgements (left column compared to center and right column compared to center). For each row, from the lowest numerosity to the highest, the model predicts a perceived numerosity of 12.65, 14.91, and 15.61 (Intensity); 13.66, 13.87, 15.68 (Area); 12.65, 14.88, 15.92 (Convex Hull). As explained in **Figure 5B**, perceived numerosity generally slightly underestimates the true numerosity and is not necessarily an integer value.

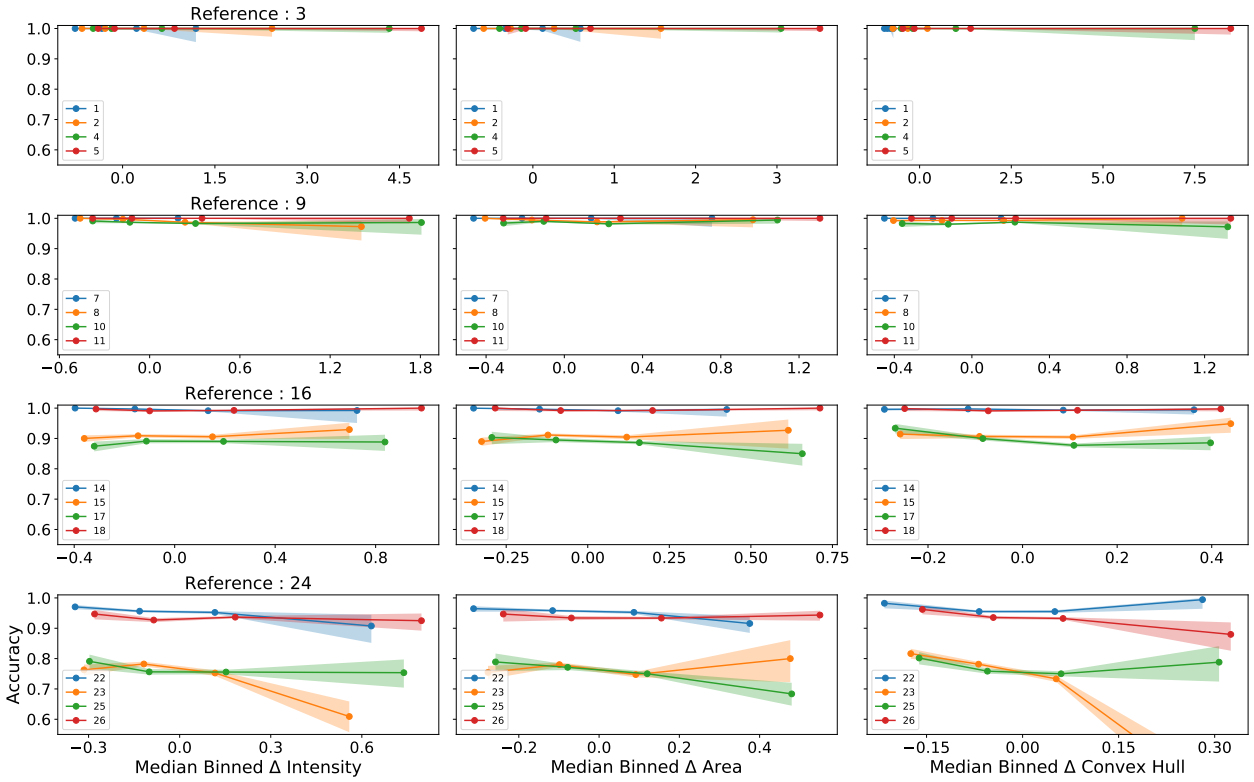

Figure S2: Effects of covariates of numerosity. Three covariates of the number of objects in the scene are explored for possible influence on our model's estimate of numerosity. These are average image intensity **(left column)**, the sum of the areas of the objects **(middle column)**, and the area of the objects' convex hull **(right column)**. Each plot shows the error rates in a relative quantity discrimination task like the one in **Figure 5A**. For each plot we chose a reference image containing respectively 3, 9, 16 and 24 objects (rows of the figure) and had our model judge relative numerosity w.r. to test images containing a different but similar number of objects (indicated in the legend and associated with colors). We used 4650 test images, 150 images per test number per condition. Given the stochastic nature of the images, the covariates vary over a wide range for each number of objects (see **Fig. S1**). For each number of objects, we plot the model's error rates (y axis) as a function of the value of the covariate quantity (x axis) which is expressed as fractional difference from the reference image (the values are binned). Shadows display 95% Bayesian confidence intervals ($N > 100$). Horizontal error lines indicate no correlation of numerosity estimation with the covariate quantity. A few lines have slopes that differ slightly from zero indicating a possible correlation. However, some of the slopes indicate a negative correlation (i.e. the better the signal, the higher the error rate). From this evidence it is difficult to conclude that that the model is exploiting anything but 'number' to estimate numerosity.

A.2 Interpreting the Embedding Space

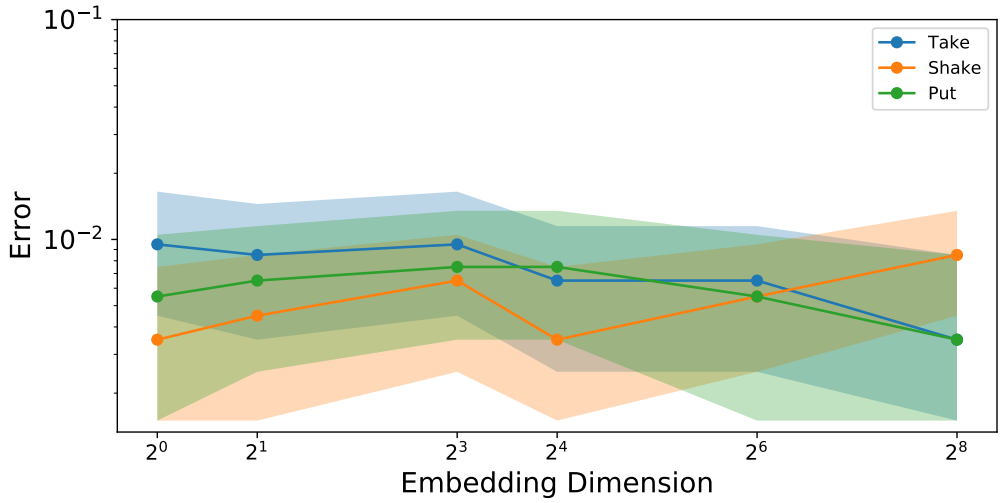

Figure S3: The embedding dimension effects on error. Classification errors for Model B, averaged over the number of items in the scene (0 - 3) are plotted as a function of the dimension of the embedding (a free parameter in our model). Since the effect is minimal we arbitrarily picked a dimension of two for ease of visualization (**Figs. 4, S4)**. The shadows show 95% Bayesian confidence intervals ($287 \leq N \leq 355$).

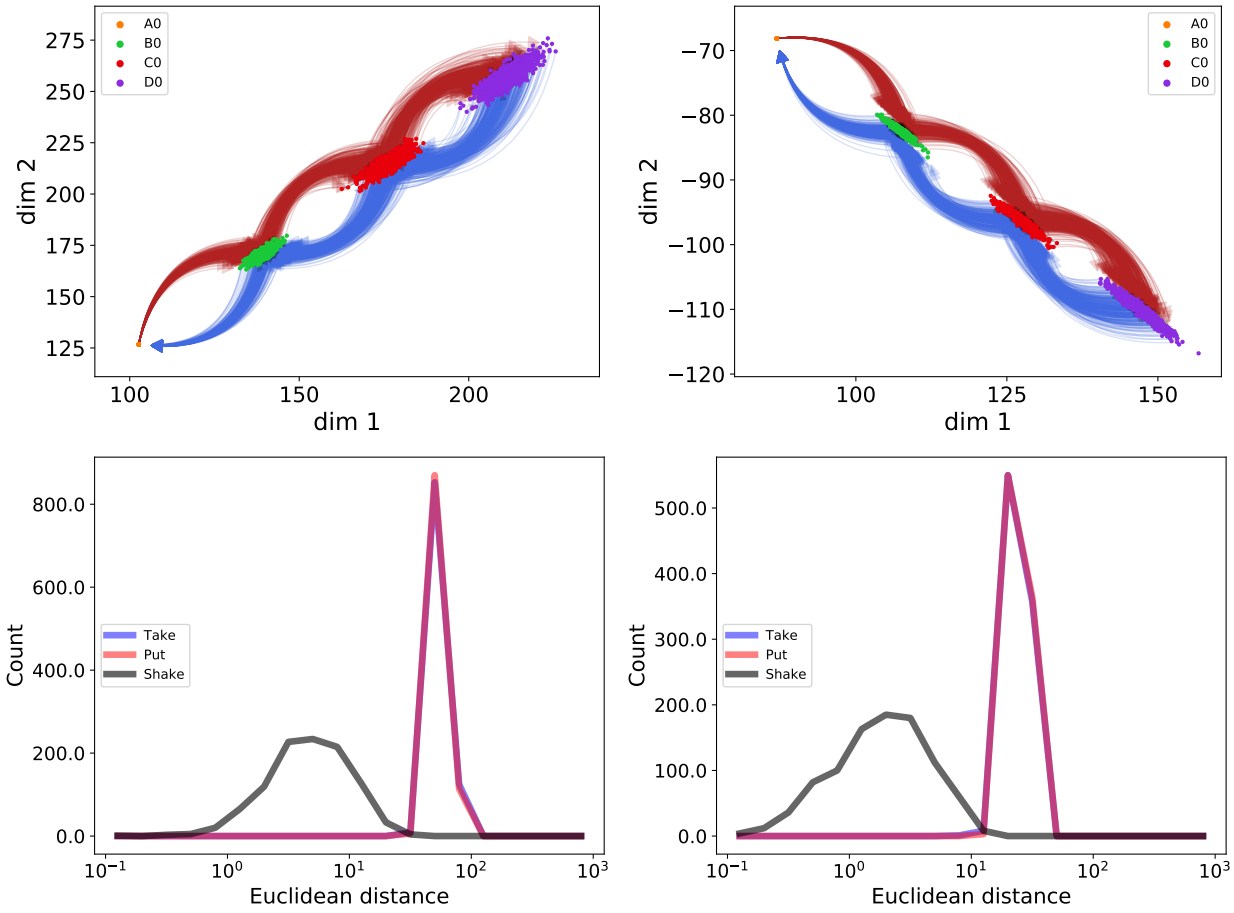

Figure S4: Embeddings with topology for Model A and Model B. A close-up look at the embedding space within the training limit. The left side are plots from Model A and the right side from Model B. **(A), (B)** Unsupervised clustering is performed on the embedding space. Each embedding is colored by it's cluster. Each cluster A0 - D0 correspond to images with numerosities 0 - 3. The clusters are well-separated. The "zero" clusters, for both Model A and Model B, are immediately recognizable as they have no variance (orange dot). As numerosity increases, Model A clusters remain well-separated, whereas Model B clusters begin to come closer to each other. We also overlay a topology from the training actions (P), (T), (S). Blue arrows joining a pair of points represent take actions, red arrows represent put actions. Arrows representing shake actions are under the point clouds and are mostly not visible. **(C), (D)** Distances between pairs of points in the embedding space are histogrammed by action. The histograms show the clearly different distribution for shake actions in comparison to take and put actions. Furthermore, the overlap between shake and non-shake actions is smaller for Model A than Model B, explaining the higher performance in action classification for Model A.

A.3 Varying Training Limit

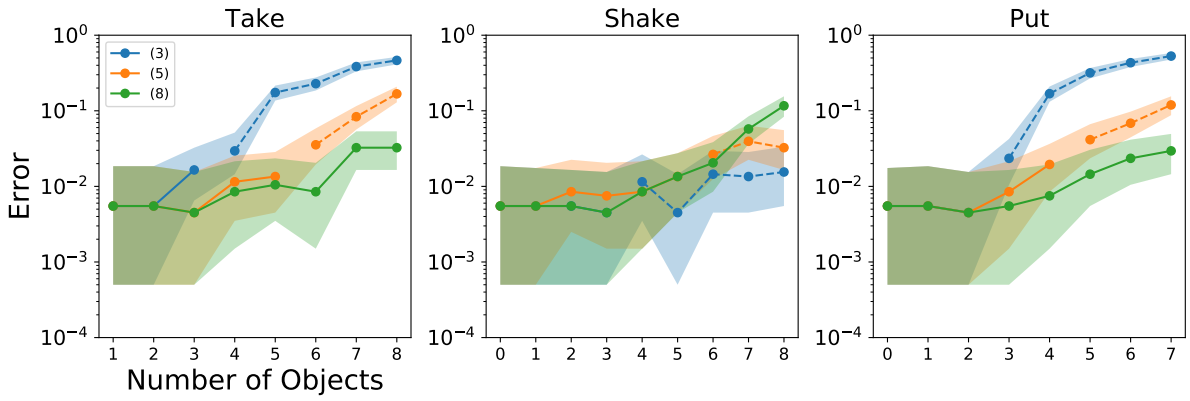

Figure S5: Effect of modifying the training limit. In order to explore the effect of the number of objects during training, we trained the network to predict actions using a maximum of 3, 5, or 8 objects with images like those in dataset B (**Fig. 2B**). We tested the network on 8 objects. Each panels shows errors on the training task and are in the same style as **Figure 3**. The line-breaks and dashed lines mark where the training limit ends and the testing region begins, and the legend shows the training limit in parentheses. The shadows provide 95% confidence intervals ($287 \leq N \leq 355$). As expected, the error is lower when the training limit is higher.

A.4 Reproducibility of Results

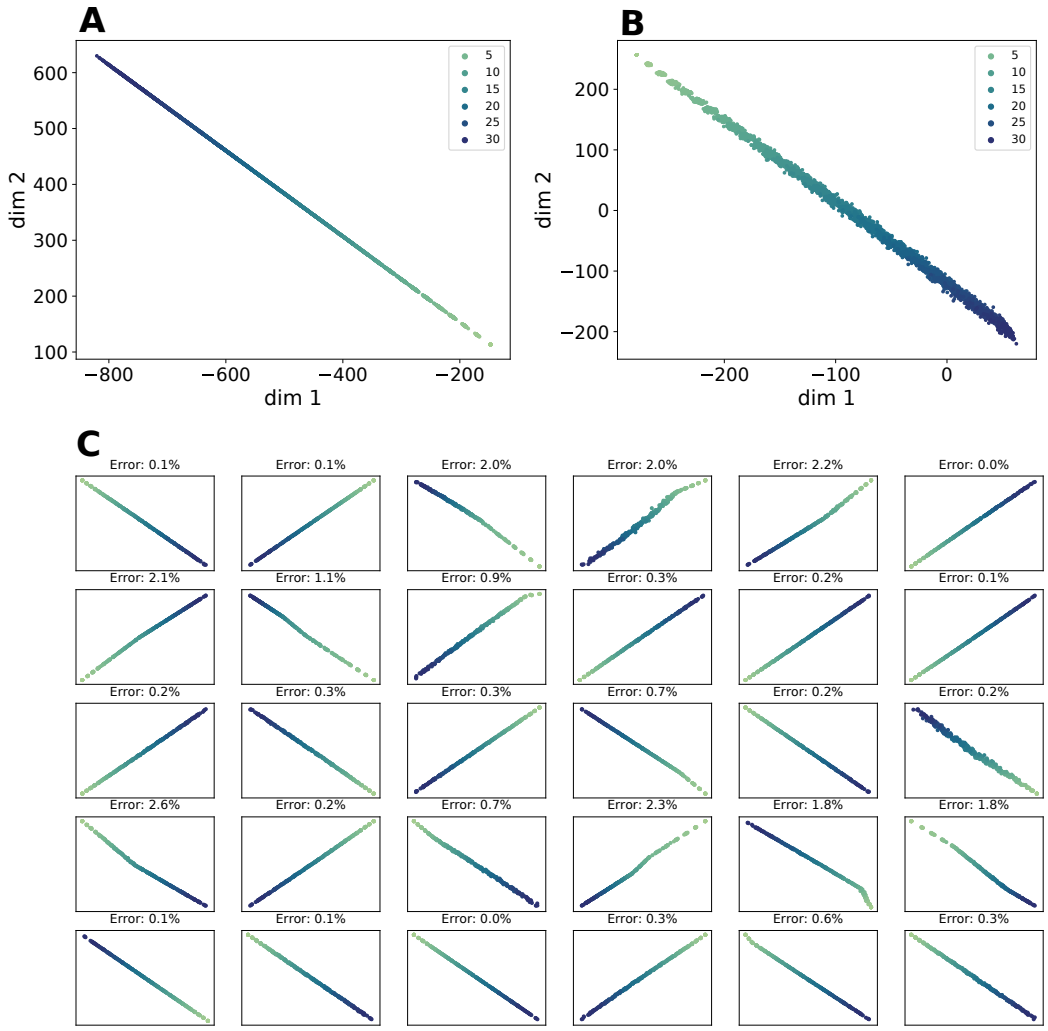

Figure S6: Miscellaneous embedding spaces (A) Embedding space for the network trained on dataset B, with up to five objects. **(B)** Embedding space for the network trained on dataset B, with up to eight objects. **(C)** Embedding spaces for 30 different random initializations. We repeated the training procedure 30 times on different random initializations of dataset B, with a training limit of 3 objects. Qualitatively, 21 embedding spaces look like a straight line, six initializations present a slight kink in the line, and three instances either present a large kink or two kinks. The linear approximation error (Methods - Interpreting the Embedding Space) is provided above each subplot and measures the approximate deviation from a purely linear model. An error below 4% predicts an approximately linear embedding line.

A.5 Restricting Dataset Variability

In this experiment, we explore if the randomness in object representation is necessary for the representation discovered in Model B. We restrict the randomness by jittering certain properties of the objects. We find that jitter is sufficient to produce models with the same key properties as in Model B, however, they are more sensitive to the initial seed. We refer to this dataset as the *jitter dataset* and model's trained by this dataset as *Jitter Models*.

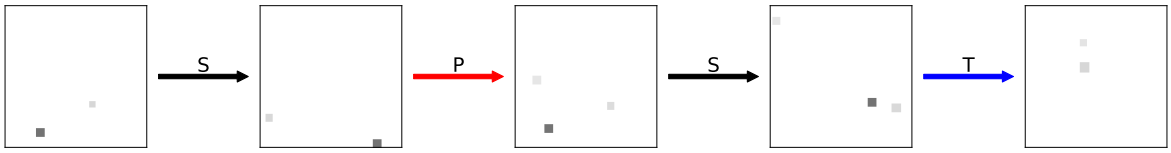

Figure S7: In this dataset, we restrict the change in size and contrast an object may undergo due to an action. After each action, the size (diagonal) of an object will be allowed to jitter by up to 3 pixels and the contrast by $\pm 0.02\%$ of the maximum contrast. We find that these small perturbations in object representations are sufficient to recreate similar results to those seen with Model B.

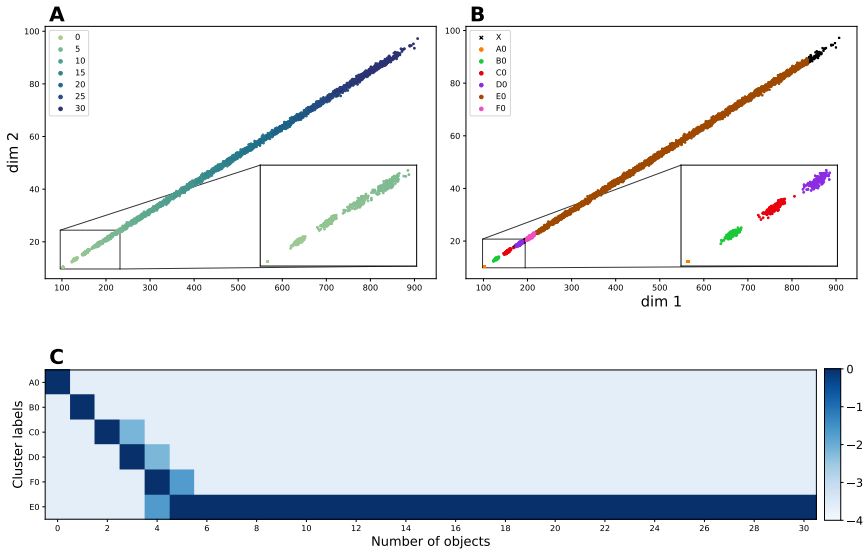

Figure S8: We find that the important properties of the Model B representation arise with Jitter Models. The model representations are linear, monotonic, with the early numbers easily separable. We set the minimum cluster size to 30 (HDBSCAN), in order to produce the most concise plots. Note the Jitter Model representations are more sensitive to minimum cluster size.

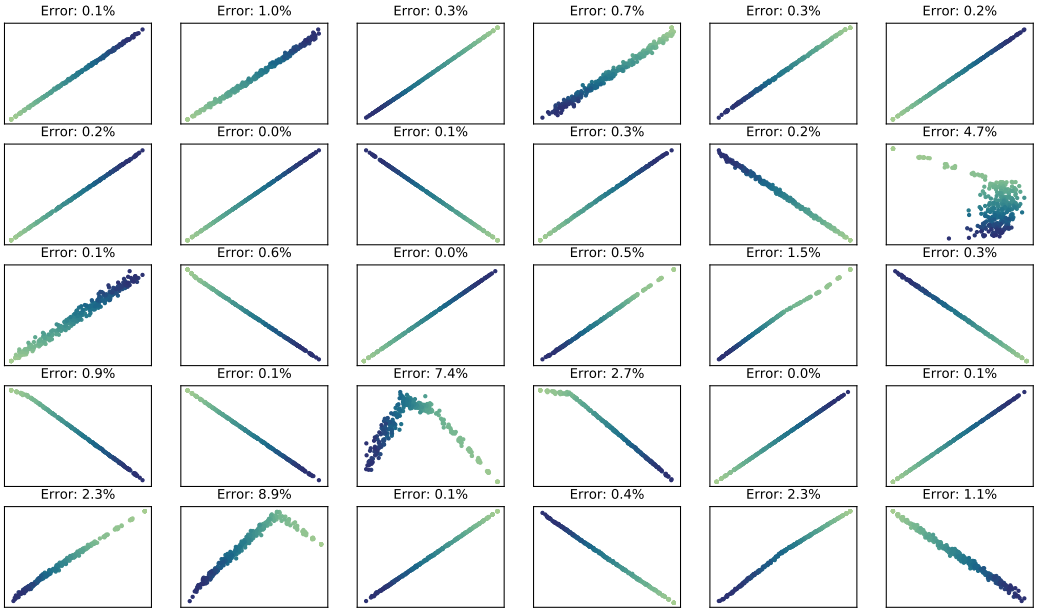

Figure S9: We vary the initial seed to determine how reproducible the results are. We find model's trained with the jitter dataset learn mostly linear representations, however, certain seeds do result in large kinks.

A.6 Imprecise Action Sizes

In this experiment, we explore if the precision of the action sizes (the agent always performing an action on a single object) is necessary for the properties of interest to emerge. We find that while precise actions help in building distinct clusters in the subitization range, it is not necessary. We refer to this dataset as the *imprecise actions dataset* and model's trained by this dataset as *Imprecise Action Models*.

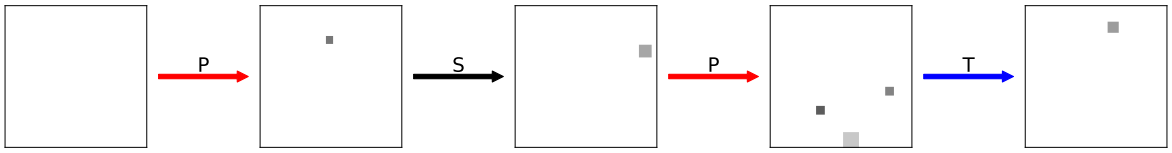

Figure S10: In this dataset, we allow the number of actions taken or placed during an action to be 0-3 (limited by the number of objects in the visual scene). This dataset mimics a situation in which the agent is imprecise with their actions and does not always select one object. The object's size and contrast are randomized between actions (like in dataset B).

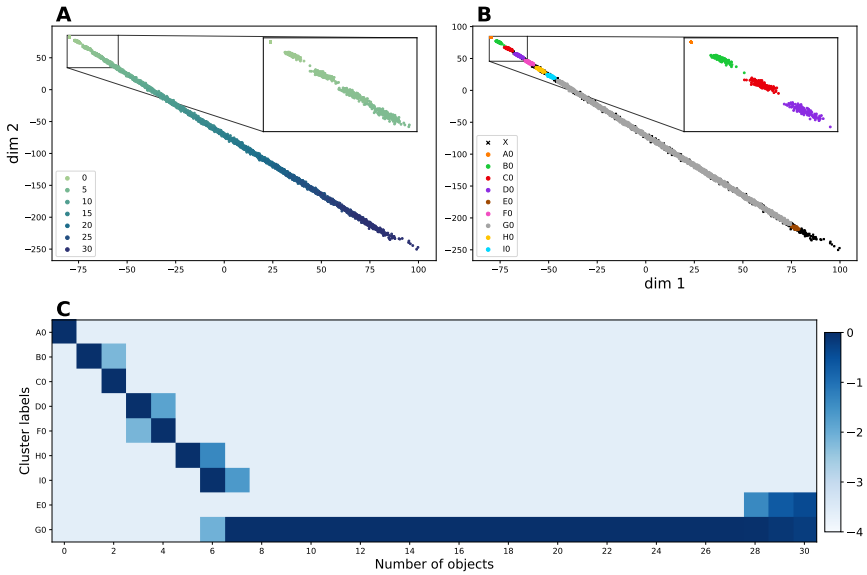

Figure S11: We find that the important properties of the Model B representation arise with Imprecise Action Models. The model representations are linear, monotonic, with the early numbers easily separable. However, the separability of the early clusters is rougher than with precise action sizes. We set the minimum cluster size to 50 (HDBSCAN), in order to produce the most concise plots. Note the Imprecise Action Model representations are more sensitive to minimum cluster size.

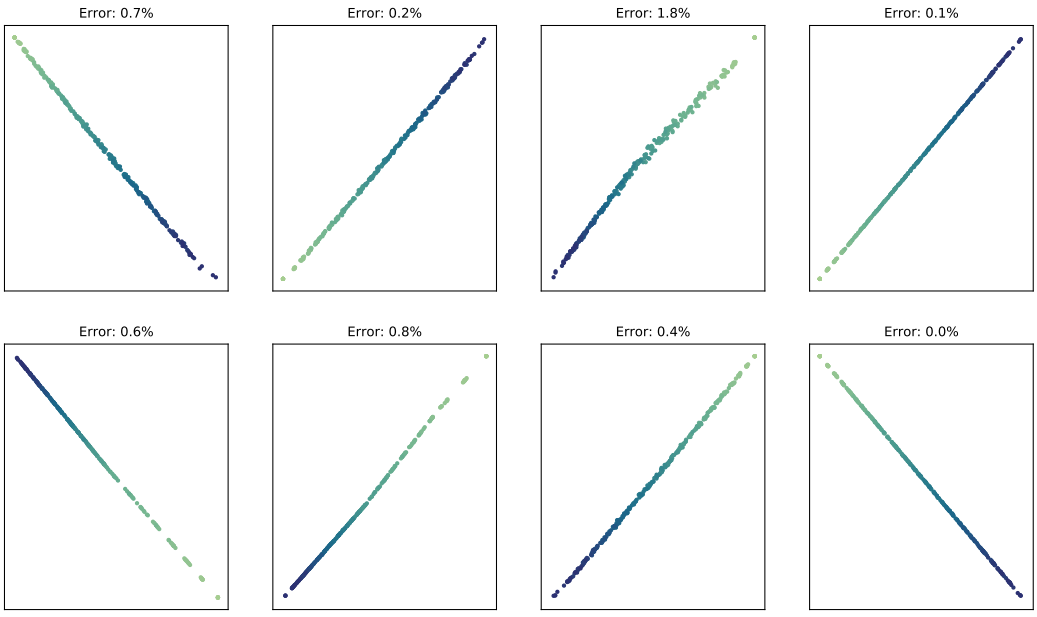

Figure S12: We vary the initial seed to determine how reproducible the results are. We find model's trained with the imprecise action sizes learn mostly linear representations.

B Dataset Statistics

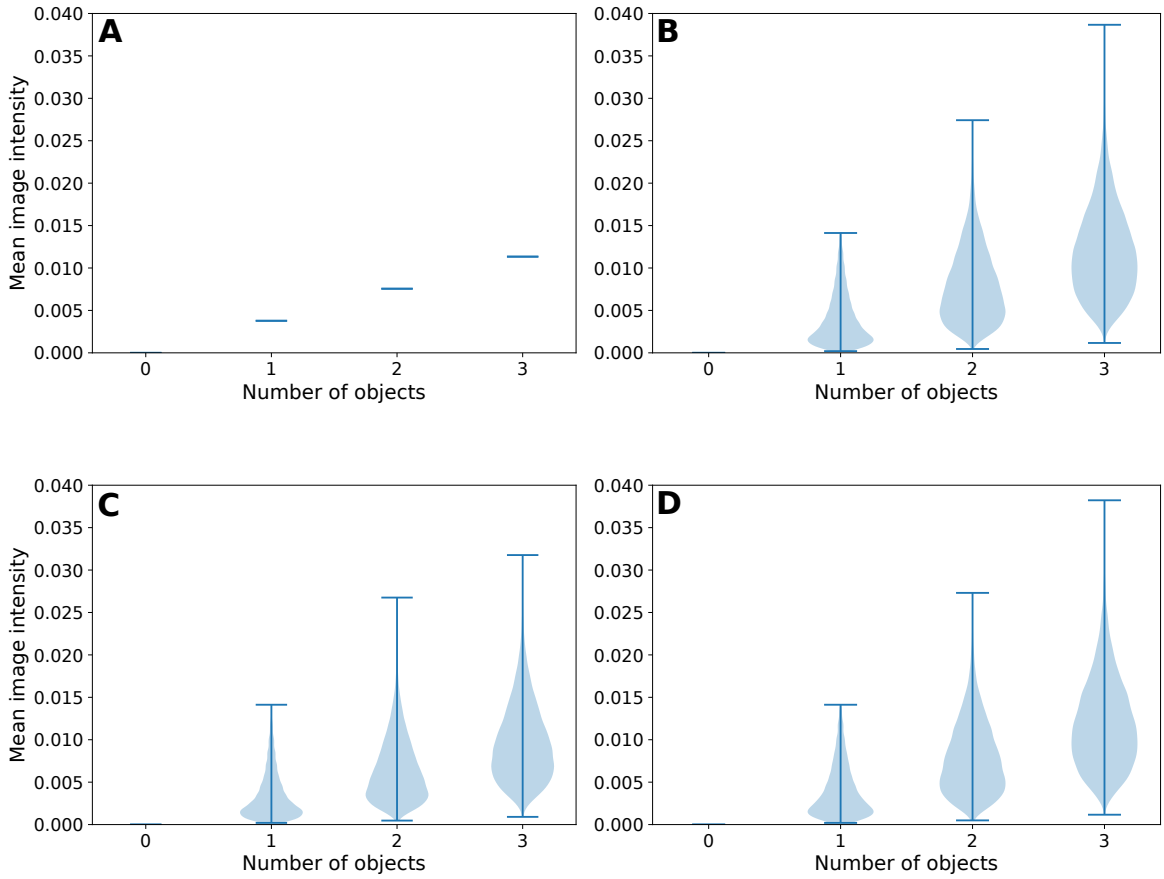

Figure S13: Training set statistics. **(A)** In dataset A (**Fig. 2A**) objects have the same size and contrast. Thus, the number of objects predicts the mean image intensity and vice-versa. **(B)** Objects in dataset B (**Fig. 2B**) have variable sizes and variable contrast, thus mean image intensity is not sufficient to predict the number of objects. **(C)** Objects in the jitter datasets (**Fig. S10**) have a restricted, but variable size and contrast. We see the image statistics are similar to that of dataset B, but have a smaller amount of variability. **(D)** Objects in the imprecise actions datasets (**Fig. S10**) have random numbers of objects manipulated in an action. We see the image statistics are effectively the same as that of dataset B.

C Network Details

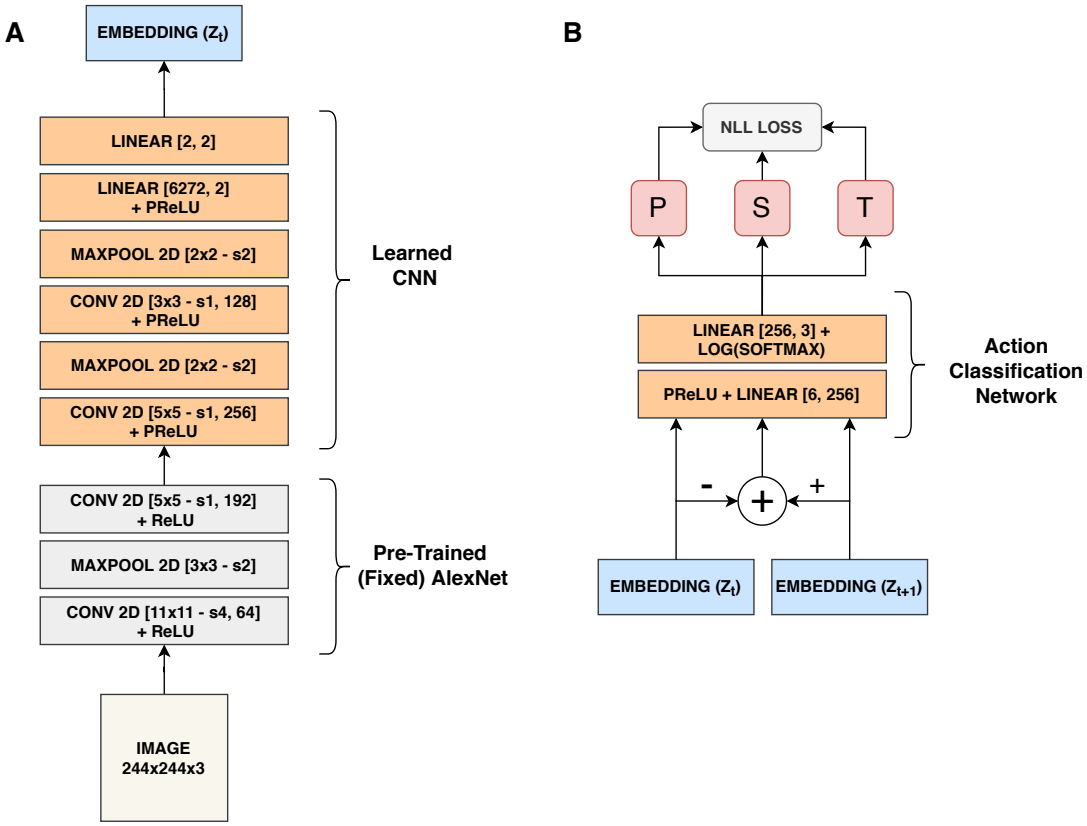

Figure S14: Detailed diagram of the network structure.
(A) The feature extraction / embedding network. The gray layers are pre-trained on ImageNet [48, 24] and remain fixed throughout the course of training. The orange layers are randomly seeded and trained simultaneously with the classifier in (B). The details of the layer are described within the brackets. For example, [11x11 - s4, 64] is an 11x11 kernel with a stride of 4 and 64 filters. During a training step, the embedding network accepts an image (x_t) of the visual scene and generates a lower-dimensional feature embedding (z_t) of the visual scene. An action: (P), (T), or (S) modifies the visual scene and the "after" image (x_{t+1}) is passed through the embedding network as well. The outputs of the embedding network, (z_t) and (z_{t+1}) are treated as inputs to the action classification network. The shared embedding network is trained together with the classifier (B), in a Siamese configuration. **(B)** The action classification network is a 2-layer classifier network and is composed of two fully connected layers with a log-softmax transformation on the output. The input is the representation of the visual scene before and after an action is performed. The negative log-likelihood (NLL) loss function is used to train both the action classification network and the embedding network simultaneously. An overview of the entire training paradigm is shown in **Figure 1**.