DISCOVERING OBJECTS AND THEIR RELATIONS FROM ENTANGLED SCENE REPRESENTATIONS

D. Raposo,* A. Santoro,* D.G.T. Barrett, R. Pascanu, T. Lillicrap, P. Battaglia
DeepMind
London, United Kingdom
{draposo, adamsantoro, barrettdavid, razp, countzero, peterbattaglia}@google.com

ABSTRACT

Our world can be succinctly and compactly described as structured scenes of objects and relations. A typical room, for example, contains salient objects such as tables, chairs and books, and these objects typically relate to each other by virtue of their correlated features, such as position, function and shape. Humans exploit knowledge of objects and their relations for learning a wide spectrum of tasks, and more generally when learning the structure underlying observed data. In this work, we introduce *relation networks* (RNs) - a general purpose neural network architecture for object-relation reasoning. We show that RNs are capable of learning object relations from scene description data. Furthermore, we show that RNs can act as a bottleneck that induces the factorization of objects from entangled scene description inputs, and from distributed deep representations of scene images provided by a variational autoencoder. The model can also be used in conjunction with differentiable memory mechanisms for implicit relation discovery in one-shot learning tasks. Our results suggest that relation networks are a powerful architecture for solving a variety of problems that require object relation reasoning.

1 INTRODUCTION

The ability to reason about objects and relations is important for solving a wide variety of tasks (Spelke et al., 1992; Lake et al., 2016). For example, object relations enable the transfer of learned knowledge across superficial (dis)similarities (Tenenbaum et al., 2011): the predator-prey relationship between a lion and a zebra is knowledge that is similarly useful when applied to a bear and a salmon, even though many features of these animals are very different.

In this work, we introduce a neural network architecture for learning to reason about – or model – objects and their relations, which we call *relation networks* (RNs). RNs accomplish their goal by adhering to a few fundamental design principles. Firstly, RNs are designed to be invariant to permutations of object descriptions in their input. For example, RN representations of the object set {table, chair, book} will be identical for arbitrary re-orderings of the elements of the set. Secondly, RNs are designed to learn relations *across* multiple objects rather than *within* a single object – a basic defining property of object relations. This design principle manifests itself in the use of shared computations across groups of objects. In designing the RN architecture, we took inspiration from the recently developed *Interaction Network* (IN) (Battaglia et al., 2016) which is more generally concerned with modelling temporal interactions.

In principle, a deep network with a sufficiently large number of parameters and a large enough training set should be capable of matching the performance of a RN. However, such networks would have to learn both the permutation invariance of objects and the relational structure of the objects in the execution of a desired computation. This quickly becomes unfeasible as the number of objects and relations increase.

*Denotes equal contribution.

To test the ability of RNs to discover relations between objects, we turned to the classification of scenes, wherein classification boundaries were defined by the relational structure of the objects in the scenes. There are various ways of encoding scenes as observable data; for example, scene description data can consist of sets of co-occurring objects and their respective features (location, size, color, shape, etc.). A typical room scene, then, might consist of the object set {table, chair, lamp} and their respective descriptions (e.g., the table is large and red). Similar datasets have been used for many decades in cognitive psychology to study human relation learning (Palmer, 1975). Here we consider synthetic scene description data generated by hierarchical probabilistic generative models of objects and object features.

We begin by motivating RNs as a general purpose architecture for reasoning about object relations. Critically, we describe how RNs implement a permutation invariant computation on implicit groups of factored "objects." We then demonstrate the utility of RNs for classification of static scenes, where classification boundaries are defined by the relations between objects in the scenes. Next, we exploit RNs' implicit use of factored object representations to demonstrate that RNs can induce the factorization of objects from entangled scene inputs. Finally, we combine RNs with memory-augmented neural networks (MANNs) (Santoro et al., 2016) to solve a difficult one-shot learning task, demonstrating the ability of RNs' to act in conjunction with other neural network architectures to rapidly discover new object relations from entirely new scenes.

2 MODEL

2.1 DESCRIPTION AND IMPLEMENTATION

RNs are inspired by *Interaction Networks* (INs) (Battaglia et al., 2016), and therefore share similar functional insights. Both operate under the assumption that permutation invariance is a necessary requirement for problems with inputs that can be described with a graphical structure. However, INs use relations between objects as input to determine object *interactions*, mainly for the purpose of reasoning about dynamics. RNs compute object relations, and hence aim to determine object-relational structure from static inputs.

Suppose we have an object, $o = (o^1, o^2, ..., o^n)$, represented as a vector of n features encoding properties such as the object's type, color, size, position, etc. A collection of m objects can be gathered into a $m \times n$ matrix D, called the *scene description*. Although the term *scene description* alludes to visual information, this need not be the case; scenes can be entirely abstract, as can the objects that constitute the scene, and the features that define the objects.

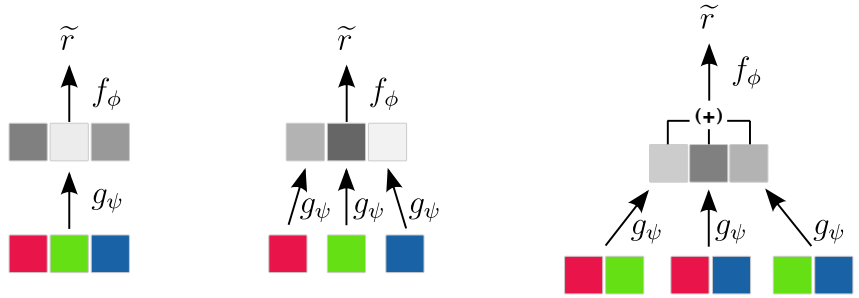
We can imagine tasks (see section 3) that depend on the *relations*, r, between objects. One such task is the discovery of the relations themselves. For example, returning to the predator and prey analogy, the predator-prey relation can be determined from *relative* features between two animals – such as their relative sizes, perhaps. Observation of the size of a single animal, then, does not inform whether this animal is a predator or prey to any other given animal, since its size necessarily needs to be compared to the sizes of other animals.

There are a number of possible functions that could allow for the discovery of object relations (figure 1). Consider a function g, with parameters ψ. The function g_ψ can be defined to operate on a particular factorization of D; for example, $g_\psi(D)$, or $g_\psi(D_{1,2}, ..., D_{i,j,}, ..., D_{m,n})$. We are interested in models defined by the composite function $f \circ g$, where f is a function that returns a prediction $\widetilde{r}$.

One implementation of g_ψ would process the entire contents of D without exploiting knowledge that the features in a particular row are related through their description of a common object, and would instead have to learn the appropriate parsing of inputs and any necessary sub-functions: $\widetilde{r} = f_\phi(g_\psi(o_1^1, o_1^2, ..., o_m^n))$. An alternative approach would be to impose a prior on the parsing of the input space, such that g operates on objects directly: $\widetilde{r} = f_\phi(g_\psi(o_1), g_\psi(o_2), ..., g_\psi(o_m))$. A third middle-ground approach – which is the approach taken by RNs – recognizes that relations necessarily exist in the context of a set of objects that have some capacity to be related. Thus, the computation of relations should entail some common function across these sets. For example, g may operate on *pairs* of objects: $\widetilde{r} = f_\phi(g_\psi(o_1, o_2), g_\psi(o_1, o_3), ..., g_\psi(o_{m-1}, o_m))$.

For RNs, g is implemented as a multi-layered perceptron (MLP) that operates on pairs of objects. The same MLP operates on all possible pairings of objects from D, to produce summaries $s_{i,j}$. Depending on the complexity of the problem, any aggregation function a can be chosen to operate on $s_{i,j}$. However, to ensure the input to f is order invariant, the aggregation function should be commutative and associative, with summation being a natural choice. Thus, $\widetilde{r} = f_\phi(a(s_{1,2}, s_{1,3}, ..., s_{m-1,m})) = f_\phi(\sum_{i,j} s_{i,j}) = f_\phi(\sum_{i,j} g_\psi(o_i, o_j))$. Training therefore entails the optimization of two MLPs: f_ϕ and g_ψ.

The order invariance of the aggregation function is a critical feature of the model, since without this invariance, the model would be unable to take advantage of the shared g_ψ function that operates on all permuted pairs of objects.



(a) No Prior (b) Object prior (c) Object pair prior

Figure 1: **Model types.** RNs are constructed to operate with an explicit prior on the input space (c). Features from all pairwise combinations of objects act as input to the same mlp, g_ψ.

3 Experimental tasks and data

3.1 Datasets

To probe a model's ability to both infer relations from scene descriptions and implicitly use relations to solve more difficult tasks – such as one-shot learning – we first developed datasets of scene descriptions and their associated images. To generate a scene description, we first defined a graph of object relations (see figure 2). For example, suppose there are four types of squares, with each type being identified by its color. A graph description of the relations between each colored square could identify the blue square as being a parent of the orange square. If the type of relation is "position," then this particular relation would manifest as blue squares being independently positioned in the scene, and orange squares positioned in close proximity to blue squares. Similarly, suppose we have triangles and circles as the object types, with color as the relation. If triangles are parents to circles, then the color of triangles in the scene will be randomly sampled, while the color of a circle will be derived from the color of the parent triangle. Datasets generated from graphs, then, impose solutions that depend on relative object features. That is, no information can be used from *within* object features alone – such as a particular coordinate position, or RGB color value.

Graphs define generative models that can be used to produce scenes. For scenes with position relations, root node coordinates (o_x^p, o_y^p), were first randomly chosen in a bounded space. Children were then randomly assigned to a particular parent object, and their position was determined as a function of the parent's position: $(o_x^c, o_y^c) = (o_x^p + d\cos(\theta^c), o_y^p + d\sin(\theta^c))$. Here, $\theta^c \sim \mathcal{U}(\theta^p - \pi/3, \theta^p + \pi/3)$, where θ^p is the angle computed for the parent. For root nodes, then $\theta^p \sim \mathcal{U}(0, 2\pi)$. d is a computed distance: $d = d_0 + d_1$, where d_0 is a minimum distance to prevent significant object overlap, and d_1 is sampled from a half-normal distribution. This same pattern – inherit features from parents and apply noise – is used to generate scenes from graphs that define other relations, such as color. For the case of color, the inherited features are RGB values. Ultimately, scene descriptions consisted of matrices with 16 rows, with each row describing the object type (four rows for each of four types), position, color, and size.

Custom datasets were required to both explicitly test solutions to the task of inferring object relations, and to actively control for solutions that do not depend on object relations. For example,

consider the common scenario of child objects positioned close to a parent object, analogous to chairs positioned around a table. In this scenario, the count information of objects (i.e., that there are more child objects than parent objects) is non-relational information that can nonetheless be used to constrain the solution space; the prediction of the relation between objects doesn't entirely depend on explicitly computed relations, such as the relative distance between the child objects and the parent objects. To return to the kitchen scene analogy, one wouldn't need to know the relative distance of the chairs to the table; instead, knowing the number of chairs and number of tables could inform the relation, if it is known that less frequently occurring objects tend to be parents to more frequently occurring objects. Although information like object count can be important for solving many tasks, here we explicitly sought to test models' abilities to compute and operate on object relations.

The datasets will be made freely available.

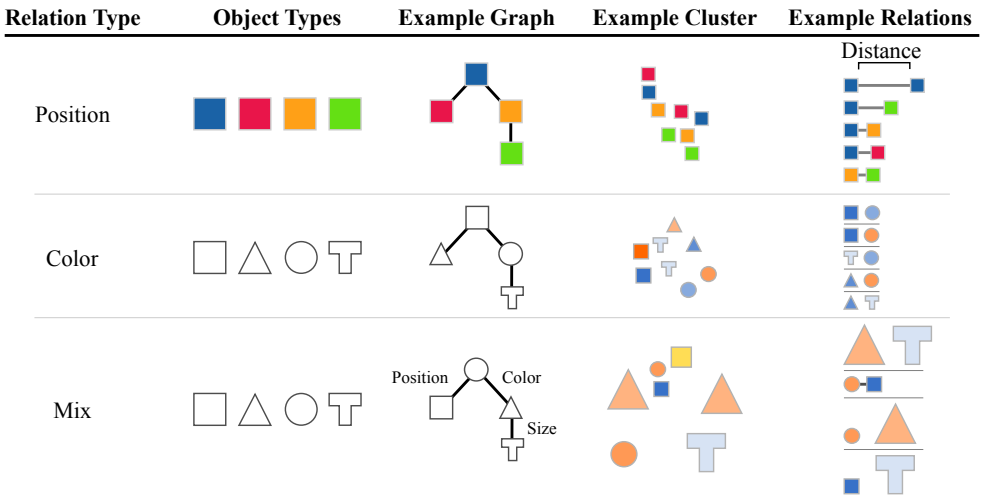


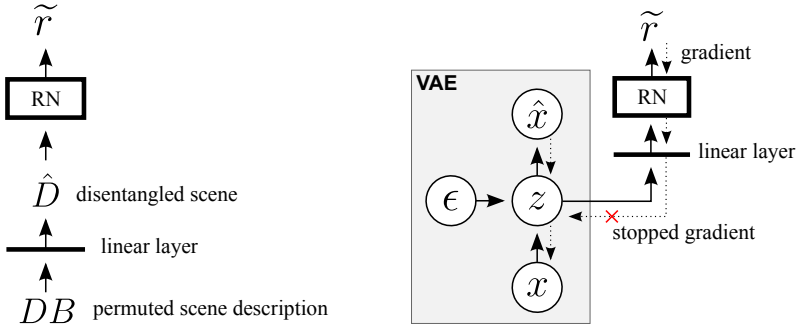
Figure 2: **Objects and relations.** Relation types (column one) between object types (column two) can be described with directed graphs (column three). Shown in the fourth column are cropped clusters from example scenes generated by a model based on the directed graph shown in column three. In the last column are example of relations that can be used to inform class membership; for example, the distances between pairs objects, or the differences in color between pairs of objects that may inform the particular graphical structure, and hence generative model, used to generate the scene.

3.2 TASK DESCRIPTIONS

The tasks on which we assessed the RN's performance fell into three categories. The first category involved the classification of scenes. In this task, the network was given a scene description as input, with the target being a binary vector description of the edges between object types (which constitute the nodes of the graph – see figure 2). Training data consisted of 5000 samples derived from 5, 10, or 20 unique classes (i.e., graphs), with testing data comprising withheld within-class samples. Although the target was a vector description of the generating graph, this task was fundamentally one of classification, and the use of one-hot vector labels could have constituted equivalent training. Nonetheless, since class membership can only be determined from the relational structure of the objects within a particular sample, the ability to classify subsumes the ability to infer the relations.

The second category of tasks tested the ability of the RN to classify scenes – and hence, reason about object relations – from unstructured input domains (see figure 3). Since RNs operate on factored object representations, this task specifically probed the ability of RNs to *induce* the learning of object factorizations from entangled scene descriptions. In the first set of experiments we broke the highly structured scene description input by passing it through a fixed permutation matrix. To decode the entangled state we used a linear layer whose output provided the input to the RN. We then asked the network to classify scenes, as in the previous tasks. In the second set of experiments we pushed

this idea further, and tested the ability of the RN to operate on distributed representations of image depictions of scenes. The images were passed through a variational autoencoder (VAE), and the latent variables were provided as input to a RN.



(a) Entagled scene description (b) Distributed latent representation

Figure 3: **Scene entangling**. To test the ability of the RN to operate on entangled scene representations, we multiplied a flattened vector representation of the scene description by a fixed permutation matrix B (a), or passed image depictions of the scenes through a VAE and used the latent code as input to a RN with an additional linear layer (b).

The final category of tasks tested the implicit use of discovered relations to solve difficult overarching problem: one-shot learning (Vinyals et al., 2016; Santoro et al., 2016; Lake et al., 2015). In the one-shot learning task, sequences of samples were fed to a memory-augmented neural network (MANN) with a relational network pre-processor. Sequences – or *episodes* – consisted of 50 random samples generated from five unique graphs, from a pool of 1900 total classes, presented jointly with time-offset label identifiers, as per Hochreiter et al. (2001) and Santoro et al. (2016). Critically, the labels associated with particular classes change from episode-to-episode. So, the network must depend on within-episode knowledge to perform the task; it must learn the particular label assigned to a class within an episode, and learn to assign this label to future samples of the same class, within the same episode. Labels are presented as input in a time-offset manner (that is, the correct label for the sample presented at time t is given as input to the network at time $t + 1$) to enable learning of an arbitrary binding procedure. Unique information from a sample – which is necessarily something pertaining to the relations of the contained objects – must be extracted, bound to its associated label, and stored in memory. Upon subsequent presentations of samples from this same class, the network must query its memory, and use stored information to infer class membership. There is a critical difference in this task compared to the first. In this task, identifying class labels constantly change episode-to-episode. So, the network cannot simply encode mappings from certain learned relational structures to class labels. Instead, the only way the network can solve the task is to develop an ability to compare and contrast extracted relational structures between samples as they occur within an episode. Please see the appendix for more details on the task setup.

4 Additional model components and training details

4.1 Variational autoencoder

For inferring relations from latent representations of pixels we used a variational autoencoder (VAE) (Kingma & Welling, 2013) with a convolutional neural network (CNN) as the feature encoder and deconvolutional network as the decoder (see figure 3b). The CNN consisted of two processing blocks. The input to each block was sent through four dimension preserving parallel convolution streams using 8 kernels of size 1x1, 3x3, 5x5, and 7x7, respectively. The outputs from these convolutions were passed through a batch normalization layer, rectified linear layer and concatenated. This was then convolved again with a down-sampling kernel of size 3x3, halving the dimension size of the input, and, except for the final output layer, again passed through batch normalization and rectified linear layers. The entire CNN consisted of two of these blocks positioned serially. There-

fore, input images of size 32x32 were convolved to feature-maps of size 8x8. The feature decoder consisted of these same blocks, except convolution operations were replaced with deconvolution operations.

The final feature maps provided by the CNN feature encoder were then passed to a linear layer, whose outputs constituted the observed variables x for the VAE. x, which was decomposed into μ and σ, was then used with an auxiliary Gaussian noise variable ϵ to infer the latent variables $z = \mu + \epsilon\sigma$. These latent variables were then decoded to generate the reconstruction $\hat{x}$, as per the conventional implementation of VAEs. However, z was also fed as input to a linear layer, which projected it to a higher dimensional space – the scene description D (see figure 3b). Importantly, this connection – from z to D – did not permit the backward flow of gradients to the VAE. This prevented the VAE architecture from contributing to the RN's disentangling solution.

4.2 Memory-augmented neural network

For implicit discovery of relations from scene descriptions, the RN was used as a pre-processor for a memory-augmented neural network (MANN). The MANN was implemented as in Santoro et al. (2016), and the reader is directed here for full details on using networks augmented with external memories. Briefly, the core MANN module consists of a controller – a long-short term memory (LSTM) (Hochreiter & Schmidhuber, 1997) – that interacts with read and write heads, which in turn interact with an external memory store. The external memory store consists of a number of memory slots, each of which contains a vector "memory." During reading, the LSTM takes in an input and produces a query vector, which the read head uses to query the external memory store using a cosine distance across the vectors stored in the memory slots, and returns a weighted sum of these vectors based on the cosine distance. During writing, the LSTM outputs a vector that the write head uses to write into the memory store using a least recently used memory access mechanism (Santoro et al., 2016).

4.3 Training details

The sizes of the RN – in terms of number of layers and number of units for both f_ϕ and g_ψ – were $\{200, 200\}$, $\{500, 500\}$, $\{1000, 1000\}$, or $\{200, 200, 200\}$. The MLP baseline models used equivalent sizes. We experimented with different sizes for the summary vector $s_{i,j}$, which is the output from the RN. Performance is generally robust to the choice of size, with similar results emerging for 100, 200, or 500. The MANN used a LSTM controller size of 200, 128 memory slots, 40 for the memory size, and 4 read and write heads.

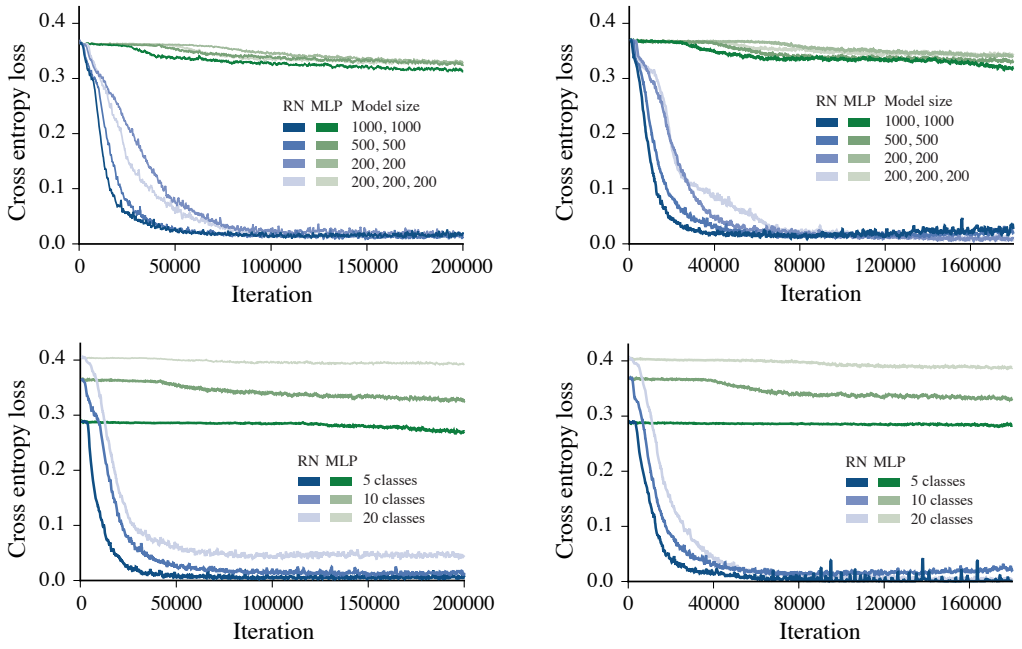
The Adam optimizer was used for optimization (Kingma & Ba, 2014), with learning rate of $1e^{-4}$ for the scene description tasks, and a learning rate of $1e^{-5}$ for the one-shot learning task. The number of iterations varied for each experiment, and are indicated in the relevant figures. All figures show performance on a withheld test-set, constituting 2-5% of the size of the training set. The number of training samples was 5000 per class for scene description tasks, and 200 per class (for 100 classes) for the pixel disentangling experiment. We used minibatch training, with batch-sizes of 100 for the scene description experiments, and 16 (with sequences of length 50) for the one-shot learning task.

5 Results

5.1 Supervised learning of relational structure from scene descriptions

Here, we trained RNs on variations of the classification task described in section 3.2 and contrasted their performance with that of MLPs of different sizes and depths. First, we compared the performance of these models on scenes where the relational structure was defined by position (figure 4a). After 200,000 iterations the RNs reached a cross entropy loss of 0.01 on a withheld test set, with MLPs of similar size failing to match (figure 4a, top). In fact, the smallest RN performed significantly better than the largest MLP. The performance of the MLPs remained poor even after 1 million iterations (cross entropy loss above 0.2 – not shown). This result was consistent for datasets with 5, 10 and 20 scene classes (figure 4a, bottom).

These results were obtained using targets that explicitly described the relations between objects types (binary vector indicating the edges of the graph). Using one-hot vectors as output targets resulted in very similar results, with the RNs' accuracy reaching 97% (see figure 7 in appendix). Our reasoning for using graph descriptions as targets relates to the potential of the RN model to represent relations between individual pairs of object types independent of other relations that might be present in a scene. Explicitly targeting the individual relations could in principle allow the model to learn the particular components that form the overall scene structure. Indeed, when trained to classify hundreds of scene classes, the RN was then able to generalize to unobserved classes (see figure 8 in appendix). (Note: this generalization to unseen classes would be impossible with the use of one-hot labels, since there is no training information that may inform the vector label a particular unseen class). Additionally, the ability to generalize to unobserved classes suggests that the RN is able to generalize in a combinatorially complex object-relation space because of its ability to learn compositional structure. It is able to use pieces (i.e., specific relations) of learned information and combine them in unique, never-before-seen ways, which is a hallmark feature of compositional learning.



(a) Position task

(b) Color task

Figure 4: **Scene classification tasks.** Differently sized RNs performed well when trained to classify 10 scene classes based on position relations (a), reaching a cross entropy loss below 0.01 (a, top), and on tasks that contained 5, 10 or 20 classes (a, bottom). The MLPs performed poorly regardless of network size and the number of classes. When relational structure depended on the color of the objects (*color task*, b) all RN configurations performed well classifying 5, 10 or 20 classes, similar to what we observed on the position task. MLPs with similar number of parameters performed poorly.

We repeated this experiment using relational structure defined by the color of the objects, with position being randomly sampled for each object (figure 4b). Again, RNs reached a low classification error on withheld data (below 0.01), whereas the MLP models did not (error above 0.2; figure 4b, top). This was observed across datasets with 5, 10 and 20 classes (figure 4b, bottom).

5.2 Inferring object relations from non-scene description inputs

There are a number of difficulties that present when inferring object relations. Firstly, one cannot naively apply a deep neural network to infer relations; instead, a particular type of architecture – possibly one that is permutation invariant to some component of its input space – may be necessary.

Secondly, the discovery of the very *things* that can possess relations – which was assumed knowledge in previous experiments – is difficult.

We showed the ability of a RN to solve the first problem, and here, we gain insight on the second problem. A particular feature – or constraint – of RNs is their computation on implicit groups of factored objects. We tested, then, whether this constraint could act as a learning signal to *induce* the factorization of objects from entangled scenes. Specifically, we tested whether RNs can learn with two types of inputs that do not have nicely factored object representations: entangled scene descriptions and pixel images.

5.2.1 INFERRING RELATIONS FROM ENTANGLED SCENE DESCRIPTIONS

In this task we probed the network's ability to classify scenes from entangled scene descriptions. Intuitively, the RN should behave as an architectural bottleneck that can aid the disentangling of objects by a perceptual model to produce factored object representations on which it can operate. To entangle data we started from the scene description D, and reshaped it into a vector of size mn (i.e., a concatenated vector of objects $[o_1; o_2; o_3; ..; o_m]$). This vector was then projected using a random permutation matrix B of size $mn \times mn$. We chose a permutation matrix for two reasons. First, because it preserved all the information of the input without adding any additional scaling between the factors. Second, because it was invertable using a matrix multiplication, and hence produced an interpretable solution found by the model. We used a learnable matrix U of size $mn \times mn$ – implemented as a linear layer without biases, positioned after the permutation and before the RN (see figure 3). Thus, this layer provided the m objects to the RN. Figure 5a shows the performance of the model on this task.

During learning we visualized the absolute value of BU (figure 5a, inset). We noticed a block structure emerging, illustrating the identification of different objects. Note, white pixels denote a value of 0, and black denote a value of 1. BU was of size $mn \times mn$, making the multiplication of a flattened D with BU produce a vector of size mn. This vector was then reshaped into a matrix of size $m \times n$, and provided as input to the RN. The block structure of BU suggests that the object k, as perceived by the network, was a linear transformation (given by the block) of a different object l from the ground truth input in D. Of particular interest is the gradual discovery of new objects over time. Because the model is order invariant, there is no pressure to recover the ground truth order of the objects. Similarly, the particular order of features that define the object were not disentangled, since there is no real pressure for this to happen. Therefore, RNs managed to successfully force the identification of different objects without imposing a particular form, or order of the features *within* objects.

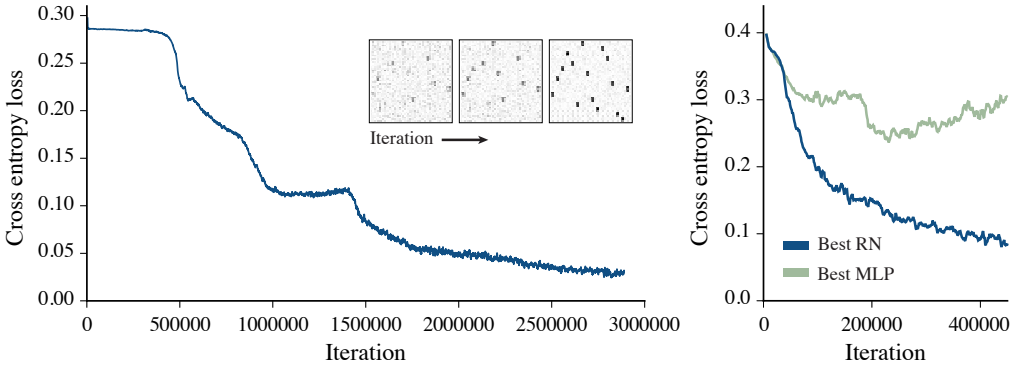
5.2.2 INFERRING RELATIONS FROM PIXELS

Image depictions from 100 unique classes (using the position-relation dataset; see figure 11 in appendix) were passed through a VAE, whose latent variables were provided as input to a linear layer, which acted as a disentangler, and which sent its output to a RN. Both the VAE and RN were trained concurrently – however, gradients from the RN were not propagated to the VAE portion of the model so as to prevent any of the VAE components from contributing to the solution of the RN's task. Thus, this experiment explicitly tested the ability of the RN to operate on distributed representations of scenes. It should be noted that this task is unique from the previous entangling tasks, as the VAE may be capable of providing both disentangled object representations, as well as relational information, in its compressed latent code. Nonetheless, the results from this task suggest a capacity for RNs to operate on distributed representations, which opens the door to the possible conjunction of RNs with perceptual neural network modules (figure 5b).

5.3 IMPLICIT USE OF RELATIONS FOR ONE-SHOT LEARNING

Here, we assess the potential to use RNs in conjunction with memory-augmented neural networks to quickly – and implicitly – discover object relations and use that knowledge for one-shot learning.

We trained a MANN with a RN pre-processor to do one-shot classification of scenes, as described in section 3.2. In order to solve this task, the network must store representations of the scenes (which, if class representations are to be unique, should necessarily contain relational information), and the



(a) Entagled scene description

(b) Distributed latent representation

Figure 5: **Inferring relational structure from entangled scenes.** A RN is able to extract and operate on objects from an entangled scene description (a). Inset shows BU from one example seed. Different seeds produced unique block structures corresponding to different permutations. A RN is also able to extract and operate on objects from a distributed latent representation of the scene (b). Distributed latent representations are inferred latent variables from VAE processing of image depictions of the scenes.

episode-unique label associated with the scene. Once a new sample of the same class is observed, it must use inferred relational information from this sample to query its memory and retrieve the appropriate label.

During training, a sequence of 50 random samples from 5 random classes (out of a pool of 1900) were picked to constitute an episode. The test phase consisted of episodes with scenes from 5 randomly selected and previously unobserved classes (out of a pool of 100). After 500,000 episodes the network showed high classification accuracy when presented with just the second instance of a class (76%) and performance reached 93% and 96% by the 5th and 10th instance, respectively (figure 6a). As expected, since class labels change from episode-to-episode, performance is at chance for the first instance of a particular class.

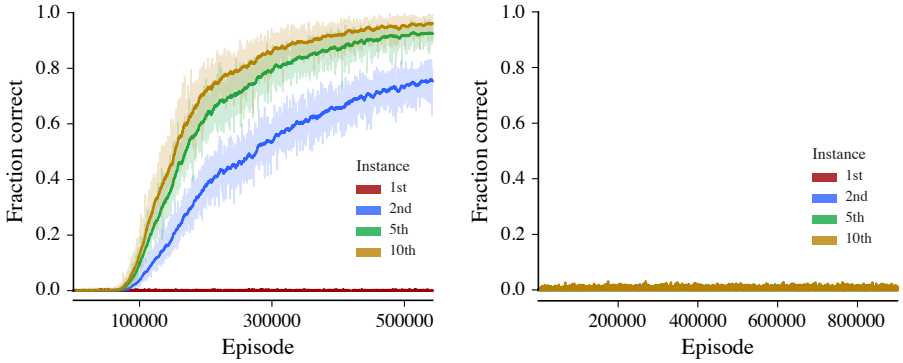
Although samples from a class are visually very dissimilar, a RN coupled to an external memory was able to do one-shot classification. This suggests that the network has the capacity to quickly extract information about the relational structure of objects, which is what defines class membership, and does so without explicit supervision about the object-relation structure. Replacing the RN pre-processor with an MLP resulted in chance performance across all instances (figure 6b), suggesting that the RN is a critical component of this memory-augmented model.

6 Conclusions

RNs are a powerful architecture for reasoning about object-relations. They operate under the assumption that relations exist between pairs of objects, or entities. This assumption affords the ability to classify scenes wherein class boundaries are defined by object relations, as well as the ability to induce factored object representations from entangled scene descriptions.

RNs have the potential to successfully operate with perceptual modules to extract object representations on which to operate. This ability to operate with different neural network modules also extends to the domain of one-shot learning, wherein they are able to pair with MANNs to perform one-shot structure learning.

The utility of RNs as a relation-reasoning module suggests that they have the potential to be useful for solving tasks that require reasoning not only about object-object relations, but also about verb-



(a) MANN with RN pre-processor

(b) MANN with MLP pre-processor

Figure 6: **One-shot classification of scenes.** A MANN with a RN pre-processor is able to accurately classify unobserved scenes after the presentation of a single instance of the same class (a). A MANN with a MLP pre-processor performs at chance (b).

object relations, as in human-object interaction datasets (Chao et al., 2015) or question-answering tasks that involve reasoning between multiple objects (Krishna et al., 2016).

The ability of RNs to induce a disentangling of their input potentially allows for great flexibility in the discovery of the "objects" that are to be factored. In our case, objects were defined *a priori*, but this needn't necessarily be the case. Depending on the task, they may not even constitute physically consistent groups of matter. Combinations of objects can be grouped to become one "cluster" object, or, the notion of what constitutes an object can instead be determined by a deep neural network, which may come up with a very flexible interpretation for "object-ness" in a scene. Developing a way to process various types of input to produce "objects" with which RNs can operate is an exciting direction of future research.

ACKNOWLEDGMENTS

We would like to thank Scott Reed, Daan Wierstra, Nando de Freitas, James Kirkpatrick, and many others on the DeepMind team.

REFERENCES

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *Advances in Neural Information Processing Systems*, 2016.

Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. Hico: A benchmark for recognizing human-object interactions in images. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1017–1025, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*, 2016.

Stephen E Palmer. The effects of contextual scenes on the identification of objects. *Memory & Cognition*, 3:519–526, 1975.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1842–1850, 2016.

Elizabeth S Spelke, Karen Breinlinger, Janet Macomber, and Kristen Jacobson. Origins of knowledge. *Psychological review*, 99(4):605, 1992.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

APPENDIX

ADDITIONAL SCENE CLASSIFICATION EXPERIMENTS

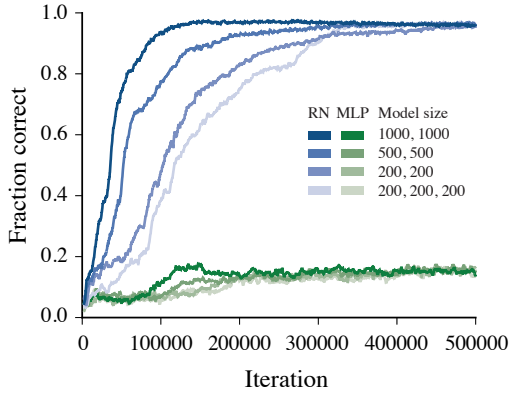


Figure 7: **Scene classification with one-hot vector labels.** Performance of RNs and MLPs on *position task* with 20 classes and using one-hot vectors as output targets.

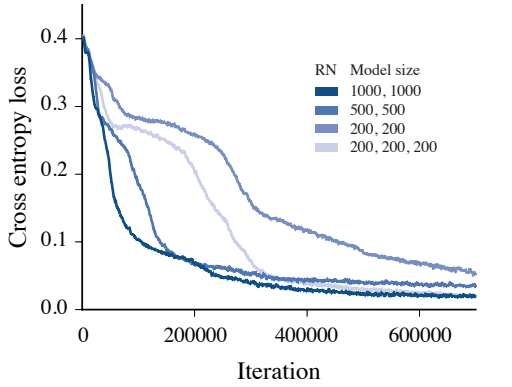


Figure 8: **Scene classification on withheld classes.** Differently sized RNs were trained to classify scenes from a pool of 490 classes. The plot shows the cross entropy loss on a test set composed of samples from 10 previously unseen classes.

ONE-SHOT LEARNING TASK SETUP

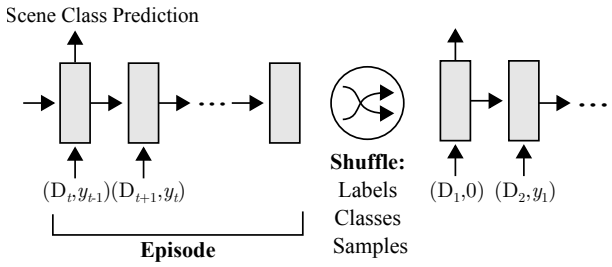
The one-shot learning task proceeds as in Hochreiter et al. (2001) and Santoro et al. (2016). In this task, parameters θ are updated to reduce the expected learning cost across a distribution of datasets $p(\mathcal{D})$:

$$\theta^* = \text{argmin}_\theta \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D})}[\mathcal{L}(\mathcal{D};\theta)]$$

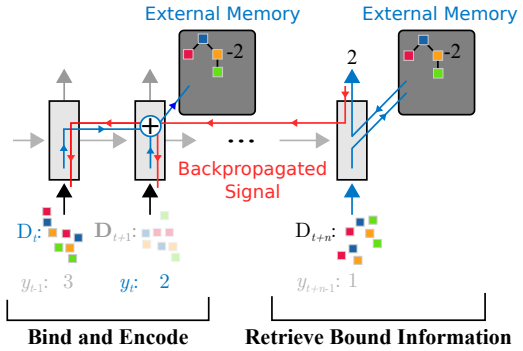
In our case, datasets consist of sequences, or episodes, of input-output pairs, where inputs are scene descriptions D and outputs are target labels y (see figure 9). So, $\{(D_t, y_t)\}_{t=1}^T \in \mathcal{D}$. All D from all datasets are constructed using the same generative process for producing scene descriptions, as described in the main text. The main differentiating factor between datasets, then, is the labels associated with each scene description. From dataset-to-dataset, scenes from a particular scene-class are assigned a unique, but arbitrary target label. This implies that scenes from the same scene class will not necessarily have the same label from episode-to-episode (or equivalently, from dataset-to-dataset), but will indeed have the same label *within* episodes.

In this training setup, labels are provided as input at subsequent timesteps. So, a given input at a particular timestep is (D_t, y_{t-1}), where y_{t-1} is the correct target label for the previous timestep's scene description. This allows the MANN to learn a binding strategy (Santoro et al., 2016) (see figure 9 and 10): it will produce a representation for a particular input and bind it with the incoming label, and will then store this bound information in memory for later use. Importantly, the label is presented in a time offset manner; if labels were instead presented at the same timestep as the corresponding sample, then the network could learn to cheat, and use the provided label information to inform its output.

This particular setup has been shown to allow for a particular meta-learning strategy (Hochreiter et al., 2001; Santoro et al., 2016); the network must learn to bind useful scene representations with arbitrary, episode-specific labels, and use this bound information to infer class membership for subsequent samples in the episode. This task poses a particular challenge for RNs; they must have the capacity to quickly extract useful representations of scene classes to enable rapid, within-episode comparisons to other scene representations. The representation extracted for a single example must contain useful, general information about the class if it is to provide information useful for subsequent classification. As seen in the results, one-shot accuracy is quite high, implying that the RN and MANN combination can extract useful representations of scene classes given a single example, and use this stored representation to successfully classify future examples from the same class. This implies that the network is necessarily extracting relational information from input scenes, since only this information can allow for successful one-shot learning across samples.



(a) Episode structure



(b) Binding strategy

Figure 9: **One-shot learning task setup**

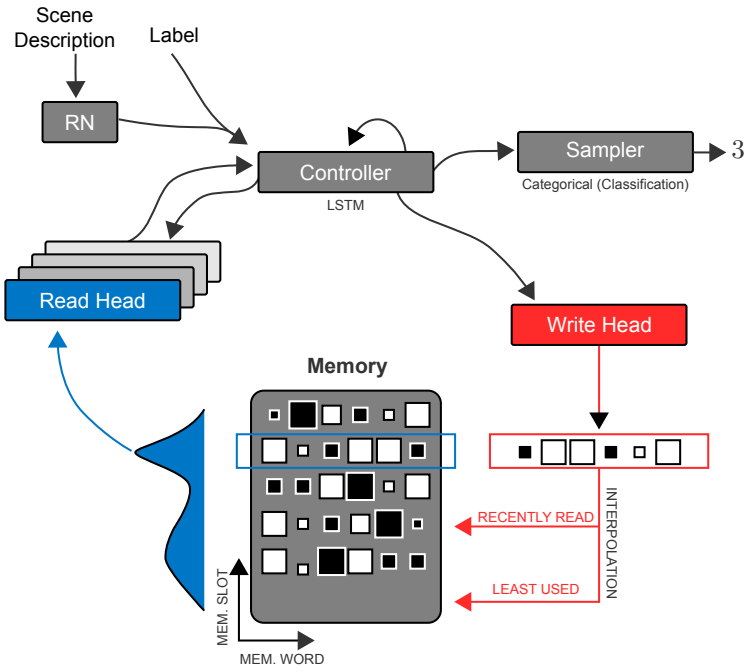


Figure 10: **RN with MANN**

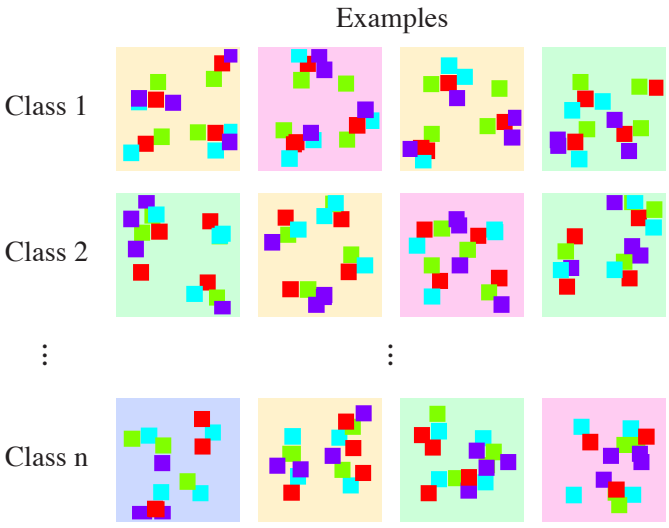


Figure 11: **Examples scene depictions**