Neural scene representation and rendering

S. M. Ali Eslami*†, Danilo J. Rezende†, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, Demis Hassabis

[1]DeepMind, 5 New Street Square, London EC4A 3TW, UK

*Corresponding author. Email: aeslami@google.com

†These authors contributed equally to this work.

Abstract

Scene representation – the process of converting visual sensory data into concise descriptions – is a requirement for intelligent behaviour. Recent work has shown that neural networks excel at this task when provided large labelled datasets. However, removing the reliance on human labelling remains an important open problem. To this end, we introduce the Generative Query Network (GQN), a framework within which machines learn to represent scenes using only their own sensors. The GQN takes as input images of a scene taken from different viewpoints, constructs an internal representation, and uses this representation to predict the appearance of that scene from previously unobserved viewpoints. The GQN demonstrates representation learning without human labels or domain knowledge, paving the way towards machines that autonomously learn to understand the world around them.

Introduction

Modern artificial vision systems are based on deep neural networks that consume large, labelled datasets to learn functions that map images to human-generated scene descriptions. They do so by, for example, categorizing the dominant object in the image (*1*), classifying the scene type (*2*), detecting object bounding boxes (*3*), or labelling individual pixels into pre-determined categories (*4, 5*). In contrast, intelligent agents in the natural world appear to require little to no explicit supervision for perception (*6*). Higher mammals including human infants learn to form representations that support motor control, memory, planning, imagination and rapid skill acquisition without any social communication, and generative processes have been hypothesised to be instrumental for this ability (*7–10*). It is thus desirable to create artificial systems that learn to represent scenes by modeling data that agents can directly obtain while processing the scenes themselves (e.g., 2D images and the agent's position in space), and without recourse to semantic labels that would have to be provided by a human (e.g., object classes, object locations, scene types, or part labels) (*11*).

To that end, we present the Generative Query Network (GQN). In this framework, as an agent navigates a 3D scene i, it collects K images $\boldsymbol{x}_i^k$ from 2D viewpoints $\boldsymbol{v}_i^k$, which we collectively refer to as its observations $\boldsymbol{o}_i = \{(\boldsymbol{x}_i^k, \boldsymbol{v}_i^k)\}_{k=1,\dots,K}$. The agent passes these observations to a GQN composed of two main parts: a representation network f and a generation network g (Fig. 1). The representation network takes as input the agent's observations and produces a neural scene representation $\boldsymbol{r}$, which encodes information about the underlying scene (we omit scene subscript i where possible, for clarity). Each additional observation accumulates further evidence about the contents of the scene in the same representation. The generation network then predicts the scene from an arbitrary query viewpoint $\boldsymbol{v}^q$, using stochastic latent variables $\boldsymbol{z}$ to create variability in its outputs where necessary. The two networks are trained jointly, in an end-to-end fashion, to maximize the likelihood of generating the ground-truth image that would be observed from the query viewpoint. More formally, (i) $r = f_\theta(\boldsymbol{o}_i)$, (ii) the deep generation network g defines a probability density $g_\theta(\boldsymbol{x}|\boldsymbol{v}^q, \boldsymbol{r}) = \int g_\theta(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{v}^q, \boldsymbol{r})\, d\boldsymbol{z}$ of an image $\boldsymbol{x}$ being observed at query viewpoint $\boldsymbol{v}^q$ for a scene representation $\boldsymbol{r}$ using latent variables $\boldsymbol{z}$, and (iii) the learnable parameters are denoted by θ. Although the GQN training objective is intractable owing to the presence of latent variables, we can employ variational approximations and optimize with stochastic gradient descent.

The representation network is unaware of the viewpoints that the generation network will be queried to predict. As a result, it will produce scene representations that contain all information necessary for the generator to make accurate image predictions (e.g., capturing object identities, positions, colours, counts and room layout). In other words, the GQN will learn by itself what these factors are, as well as how to extract them from pixels. Moreover, the generator internalizes any statistical regularities that are common across different scenes (e.g., typical colours of the sky, object shape regularities and symmetries, patterns and textures). This allows the GQN to reserve its representation capacity for a concise, abstract description of the scene, with the generator filling in the details where necessary. For instance, instead of specifying the precise shape of a robot arm, the representation network can succinctly communicate the configuration of its joints, and the generator knows how this high-level representation manifests itself as a fully rendered arm with its precise shapes and colours. In contrast, voxel (*12–15*) or point-cloud (*16*) methods (as typically obtained by classical structure-from-motion) employ literal representations and therefore typically scale poorly with scene complexity and size and are also difficult to apply to non-rigid objects (e.g., animals, vegetation, or cloth).

Rooms with multiple objects

To evaluate the feasibility of the framework, we experimented with a collection of environments in a simulated 3D environment. In the first set of experiments, we consider scenes in a square room containing a variety of objects. Wall textures – as well as the shapes, positions and colours of the objects and lights – are randomised, allowing for an effectively infinite number of total scene configurations; however, we used finite datasets to train and test the model [section 4 of (17) for details]. After training, the GQN computes its scene representation by observing one or more images of a previously unencountered, held-out test scene. With this representation, which can be as small as 256 dimensions, the generator's predictions at query viewpoints are highly accurate and mostly indistinguishable from ground-truth (Fig. 2A). The only way in which the model can succeed at this task is by perceiving and compactly encoding in the scene representation vector $\boldsymbol{r}$: the number of objects present in each scene, their positions in the room, the colours in which they appear, the colours of the walls and the indirectly observed position of the light source. Unlike in traditional supervised learning, GQNs learn to make these inferences from images without any explicit human labelling of the contents of scenes. Moreover, the GQN's generator learns an approximate 3D renderer (in other words, a program that can generate an image when given a scene representation and camera viewpoint) without any prior specification of the laws of perspective, occlusion or lighting (Fig. 2B). When the contents of the scene are not uniquely specified by the observation (e.g., because of heavy occlusion), the model's uncertainty is reflected in the variability of the generator's samples (Fig. 2C). These properties are best observed in real-time, interactive querying of the generator, (movie S1, https://youtu.be/G-kWNQJ4idw).

Notably, the model only ever observes only a small number of images from each scene during training (in this experiment, fewer than 5), yet it is capable of rendering unseen training or test scenes from arbitrary viewpoints. We also monitored the likelihood of predicted observations of training and test scenes (fig. S3) and found no noticeable difference between values of the two. Taken together, these points rule out the possibility of model over-fitting.

Analysis of the trained GQN highlights several desirable properties of its scene representation network. Two-dimensional t-distributed stochastic neighbour embedding (t-SNE) (18) visualisation of GQN scene representation vectors shows clear clustering of images of the same scene despite marked changes in viewpoint (Fig. 3A). In contrast, representations produced by auto-encoding density models such as variational auto-encoders (VAE) (*19*) apparently fail to capture the contents of the underlying scenes [section 5 of (17)]; they appear to be representations of the observed images instead. Furthermore, when prompted to reconstruct a target image, GQN exhibits compositional behaviour as it is capable of both representing and rendering combinations of scene elements it has never encountered during training (Fig. 3B) despite learning that these compositions are unlikely. To test whether the GQN learns a factorized representation, we investigated whether changing a single scene property (e.g., object colour) whilst keeping others fixed (e.g., object shape and position), leads to similar changes in the scene representation (as defined by mean cosine-similarity across scenes). We found that object colour, shape, and size; light position; and, to a lesser extent, object positions are indeed factorized [Fig. 3C; sections 5.3 and 5.4 of (17)]. We also found that the GQN is able to carry out 'scene algebra' [akin to word embedding algebra (20)]. By adding and subtracting representations of related scenes, we found that object and scene properties can be controlled, even across object positions [Fig. 4A; section 5.5 of (17)]. Finally, because it is a probabilistic model, GQN also learns to integrate information from different viewpoints in an efficient and consistent manner, as demonstrated by a reduction in its Bayesian 'surprise' at observing a held-out image of a scene as the number of views increases [Fig. 4B; section 3 of (17)]. We include analysis on the GQN's ability to generalise to out-of-distribution scenes, as well as further results on modelling of Shepard-Metzler objects in Sections 5.6 and 4.2 of (17).

Control of a robotic arm

Representations that succinctly reflect the true state of the environment should also allow agents to learn to act in those environments more robustly and with fewer interactions. Therefore, we considered the canonical task of moving a robotic arm to reach a coloured object, to test the GQN representation's suitability for control. The end-goal of deep reinforcement learning is to learn the control policy directly from pixels; however, such methods require a large amount of experience to learn from sparse rewards. Instead, we first trained a GQN and used it to succinctly represent the observations. A policy was then trained to control the arm directly from these representations. In this setting, the representation network must learn to communicate only the arm's joint angles, the position and colour of the object, and the colours of the walls for the generator to be able to predict new views. Because this vector has much lower dimensionality than the raw input images, we observed substantially more robust and data-efficient policy learning, obtaining convergence-level control performance with approximately one-fourth as many interactions with the environment than a standard method using raw pixels [Fig. 5; section 4.4 of (17)]. The 3D nature of the GQN representation allows us to train a policy from any viewpoint around the arm and is sufficiently stable to allow for arm joint velocity control from a freely moving camera.

Partially observed maze environments

Finally, we considered more complex, procedural maze-like environments to test GQN's scaling properties. The mazes consist of multiple rooms connected via corridors, and the layout of each maze and the colours of the walls are randomised in each scene. In this setting any single observation provides a small amount of information about the current maze. As before, the training objective for GQN is to predict mazes from new viewpoints, which is possible only if GQN successfully aggregates multiple observations to determine the maze layout (i.e., the wall and floor colours, the number of rooms, their positions in space, and how they connect to one another via corridors). We observed that GQN is able to make correct predictions from new first-person viewpoints (Fig. 6A). We queried the GQN's representation more directly by training a separate generator to predict a top-down view of the maze and found that it yields highly accurate predictions (Fig. 6B). The model's uncertainty, as measured by the entropy of its first-person and top-down samples, decreases as more observations are made [Fig. 6B; section 3 of (17)]. After about only five observations, the GQN's uncertainty disappears almost entirely.

Related work

GQN offers key advantages over prior work. Traditional structure-from-motion, structure-from-depth and multi view geometry techniques (*12–16, 22*) prescribe the way in which the 3D structure of the environment is represented (for instance as point clouds, mesh clouds or a collection of pre-defined primitives). GQN, by contrast, learns this representational space, allowing it to express the presence of textures, parts, objects, lights and scenes concisely and at a suitably high level of abstraction. Furthermore, its neural formulation enables task-specific fine-tuning of the representation via back-propagation, e.g. via further supervised or reinforced deep learning.

Classical neural approaches to this learning problem – e.g., auto-encoding and density models (*23–28*) – are required to capture only the distribution of observed images, and there is no explicit mechanism to encourage learning of how different views of the same 3D scene relate to one another. The expectation is that statistical compression principles will be sufficient to enable classical networks to discover the 3D structure of the environment; however, in practice, they fall short of achieving this kind of meaningful representation and instead focus on regularities of colours and patches in the image space.

Viewpoint transformation networks do explicitly learn this relationship; however, they have thus far been non-probabilistic and limited in scale, e.g., to only rotation around individual objects where a single view is sufficient for prediction (*15, 29–34*), or to small camera displacements between stereo cameras (e.g., (*35–37*)).

By employing state-of-the-art deep, iterative, latent variable density models (*26*), GQN is capable of handling free agent movement around scenes containing multiple objects. In addition, owing to its probabilistic formulation, GQN can account for uncertainty in its understanding about a scene's contents in the face of severe occlusion and partial observability. Notably, the GQN framework is not specific to the particular choice of architecture of the generation network, and alternatives such as generative adversarial networks (GANs, e.g., (*38*)) or auto-regressive models (e.g., (*39*)) could be employed.

A closely related body of work is that of discriminative pose estimation (e.g., (*40–42*)) in which networks are trained to predict camera motion between consecutive frames. The GQN formulation is advantageous, as it allows for aggregation of information from multiple images of a scene (see maze experiments); it is explicitly probabilistic, allowing for applications such as exploration through Bayesian information gain; and, unlike the aforementioned methods where scene representation and pose prediction are intertwined, the GQN architecture admits a clear architectural separation between the representation and generation networks. The idea of pose estimation is complementary, however – the GQN can be augmented with a second 'generator' that, given an image of a scene, predicts the viewpoint from which it was taken, providing a new source of gradients with which to train the representation network.

Outlook

In this work, we have shown that a single neural architecture can learn to perceive, interpret and represent synthetic scenes without any human labelling of the contents of these scenes. It can also learn a powerful neural renderer that is capable of producing accurate and consistent images of scenes from new query viewpoints. The GQN learns representations that adapt to and compactly capture the important details of its environment (e.g., the positions, identities and colours of multiple objects, the configuration of the joint angles of a robot arm, and the layout of a maze), without any of these semantics being built into the architecture of the networks. GQN uses analysis-by-synthesis to perform 'inverse graphics', but unlike existing methods (*43*) which require problem-specific engineering in the design of their generators, GQN learns this behaviour by itself and in a generally applicable manner. However, the resulting representations are no longer directly interpretable by humans.

Our experiments have thus far been restricted to synthetic environments for three reasons: (i) a need for controlled analysis, (ii) limited availability of suitable real datasets, and (iii) limitations of generative modeling with current hardware. Although the environments are relatively constrained in terms of their visual fidelity, they capture many of the fundamental difficulties of vision – namely severe partial observability and occlusion as well as the combinatorial, multi-object nature of scenes. As new sources of data become available (e.g., (*42*)) and advances are made in generative modeling capabilities (e.g., (*38, 44*)) we expect to be able to investigate application of the GQN framework to images of naturalistic scenes.

Total scene understanding involves more than just representation of the scene's 3D structure. In the future, it will be important to consider broader aspects of scene understanding – e.g., by querying across both space and time for modeling of dynamic and interactive scenes – as well as applications in virtual and augmented reality, and exploration of simultaneous scene representation and localisation of observations, which relates to the notion of 'Simultaneous Localisation and Mapping' (SLAM) in computer vision.

Our work illustrates a powerful approach to machine learning of grounded representations of physical scenes, and of the associated perception systems that holistically extract these representations from images, paving the way towards fully unsupervised scene understanding, imagination, planning and behaviour.

References and Notes:

1. A. Krizhevsky, I. Sutskever, G. E. Hinton, NIPS (2012), pp. 1–9, ImageNet classification with deep convolutional neural networks (2012).

2. B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, NIPS (2014), pp. 487–495, Learning deep features for scene recognition using places database (2014).

3. S. Ren, K. He, R. Girshick, J. Sun, NIPS (2015), pp. 91–99, Faster R-CNN: towards real-time object detection with region proposal networks (2015).

4. R. Girshick, J. Donahue, T. Darrell, J. Malik, CVPR (2014), pp. 580–587, Rich feature hierarchies for accurate object detection and semantic segmentation (2014).

5. M. C. Mozer, R. S. Zemel, M. Behrmann, NIPS (1992), pp. 436–443, Learning to segment images using dynamic feature binding (1992).

6. J. Konorski, Science 160, 652 (1968), Learning, perception, and the brain (Book reviews: integrative activity of the brain. An interdisciplinary approach), vol. 160 (1968).

7. D. Marr, Vision: A computational investigation into the human representation and processing of visual information (Henry Holt and Co., Inc., New York, 1982).

8. D. Hassabis, E. A. Maguire, Trends in cogn. sci. 11, 299 (2007), Deconstructing episodic memory with construction, vol. 11 (2007).

9. D. Kumaran, D. Hassabis, J. L. McClelland, Trends in cogn. sci. 20, 512 (2016), What learning systems do intelligent agents need? Complementary learning systems theory updated, vol. 20 (Elsevier, 2016).

10. B. M. Lake, R. Salakhutdinov, J. B. Tenenbaum, Science 350, 1332 (2015), Human-level concept learning through probabilistic program induction, vol. 350 (American Association for the Advancement of Science, 2015).

11. S. Becker, G. E. Hinton, Nature 355, 161 (1992), Self-organizing neural network that discovers surfaces in random-dot stereograms, vol. 355 (1992).

12. Z. Wu, et al., CVPR (2015), pp. 1912–1920, 3D ShapeNets: a deep representation for volumetric shapes (2015).

13. J. Wu, C. Zhang, T. Xue, W. Freeman, J. Tenenbaum, NIPS (2016), pp. 82–90, Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling (2016).

14. D. J. Rezende, et al., NIPS (2016), pp. 4996–5004, Unsupervised learning of 3D structure from images (2016).

15. X. Yan, J. Yang, E. Yumer, Y. Guo, H. Lee, NIPS (2016), pp. 1696–1704, Perspective transformer nets: learning single-view 3D object reconstruction without 3D supervision (2016).

16. M. Pollefeys, et al., IJCV 59, 207 (2004), Visual modeling with a hand-held camera, vol. 59 (Springer, 2004).

17. See supplementary materials on *Science* Online.

18. L. v. d. Maaten, G. Hinton, JMLR 9, 2579 (2008), Visualizing data using t-SNE, vol. 9 (2008).

19. I. Higgins, et al., ICLR (2016), β-VAE: learning basic visual concepts with a constrained variational framework (2016).

20. T. Mikolov, et al., NIPS (2013). Distributed representations of words and phrases and their compositionality (2013).

21. A. A. Rusu, et al., arXiv:1610.04286 (2016), Sim-to-real robot learning from pixels with progressive nets (2016).

22. Y. Zhang, W. Xu, Y. Tong, K. Zhou, ACM Transactions on graphics 34, 159 (2015), Online structure analysis for real-time indoor scene reconstruction, vol. 34 (2015).

23. D. P. Kingma, M. Welling, ICLR (2013), Auto-Encoding variational Bayes (2013).

24. D. J. Rezende, S. Mohamed, D. Wierstra, ICML (2014), vol. 32, pp. 1278–1286, Stochastic back-propagation and variational inference in deep latent Gaussian models, vol. 32 (2014).

25. I. Goodfellow, et al., NIPS (2014), pp. 2672–2680, Generative adversarial nets (2014).

26. K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, D. Wierstra, NIPS (2016), pp. 3549–3557, Towards conceptual compression (2016).

27. P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, ICML (2008), Extracting and composing robust features with denoising autoencoders (2008).

28. P. Dayan, G. E. Hinton, R. M. Neal, R. S. Zemel, Neu. comp. 7, 889 (1995), The Helmholtz machine, vol. 7 (1995).

29. G. E. Hinton, A. Krizhevsky, S. D. Wang, ICANN (Springer, 2011), pp. 44–51, Transforming auto-encoders (2011).

30. C. B. Choy, D. Xu, J. Gwak, K. Chen, S. Savarese, ECCV (2016), vol. 1, pp. 628–644, 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction, vol. 1 (2016).

31. M. Tatarchenko, A. Dosovitskiy, T. Brox, LNCS (2016), vol. 9911, pp. 322–337, Multi-view 3D models from single images with a convolutional network, vol. 9911 (2016).

32. F. Anselmi, et al., Theor. comput. scI. 633, 112 (2016), Unsupervised learning of invariant representations, vol. 633 (2016).

33. D. F. Fouhey, A. Gupta, A. Zisserman, CVPR (2016), 3D shape attributes (2016).

34. A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, T. Brox, IEEE trans. pattern anal. 39, 692 (2017), Learning to generate chairs, tables and cars with convolutional networks, vol. 39 (2017).

35. C. Godard, O. Mac Aodha, G. J. Brostow, CVPR (2017), Unsupervised monocular depth estimation with left-right consistency (2017).

36. T. Zhou, S. Tulsiani, W. Sun, J. Malik, A. A. Efros, ECCV (2016), pp. 286–301, View synthesis by appearance flow (2016).

37. J. Flynn, I. Neulander, J. Philbin, N. Snavely, CVPR (2016), pp. 5515–5524, DeepStereo: Learning to predict new views from the world's imagery (2016).

38. T. Karras, T. Aila, S. Laine, J. Lehtinen, arXiv:1710.10196 (2017), Progressive growing of GANs for improved quality, stability, and variation (2017).

39. A. v. d. Oord, et al., NIPS (2016), Conditional image generation with PixelCNN Decoders (2016).

40. D. Jayaraman, K. Grauman, ICCV (2015), Learning image representations tied to egomotion (2015).

41. P. Agrawal, J. Carreira, J. Malik (2015), Learning to see by moving (2015).

42. A. R. Zamir, et al., ECCV (2016), pp. 535–553, Generic 3D representation via pose estimation and matching (2016).

43. T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, V. Mansinghka, CVPR (2015), pp. 4390–4399, Picture: A probabilistic programming language for scene perception (2015).

44. Q. Chen, V. Koltun, ICCV (2017), Photographic image synthesis with cascaded refinement networks (2017).

45. T. T. S. Jaakkola, M. M. I. Jordan, Statistics and computing 10, 25 (1999), Bayesian parameter estimation via variational methods, vol. 10 (Springer, 1999).

46. D. P. Kingma, J. L. Ba, ICLR (2015), pp. 1–15, Adam: a method for stochastic optimization (2015).

47. J. Schmidhuber, Trans. autonomous mental dev. 2, 230 (2010), Formal theory of creativity, fun, and intrinsic motivation, vol. 2 (IEEE, 2010).

48. D. J. C. MacKay, Neural comput. 4, 590 (1992), Information-based objective functions for active data selection, vol. 4 (MIT Press, 1992).

49. E. Todorov, T. Erez, Y. Tassa, IROS (2012), pp. 5026–5033, MuJoCo: a physics engine for model-based control (2012).

50. R. N. Shepard, J. Metzler, Science 171, 701 (1971), Mental rotation of three-dimensional objects, vol. 171 (American Association for the Advancement of Science, 1971).

51. C. Beattie, et al., arXiv:1612.03801 (2016), DeepMind Lab (2016).

52. V. Mnih, et al., ICML (2016), pp. 1928–1937, Asynchronous methods for deep reinforcement learning (2016).

Acknowledgments: We thank Murray Shanahan, Andrew Zisserman, Peter Dayan, Joel Leibo, Peter Battaglia and Greg Wayne for helpful discussions and advice; Georg Ostrovski, Nicolas Heess, Daniel Zoran, Vinod Nair, David Silver for reviewing the paper; Keith Anderson for help creating environments; and the rest of the DeepMind team for their support and ideas.

Funding: This research was funded by DeepMind.

Author contributions: S.M.A.E. and D.J.R. conceived the model. S.M.A.E., D.J.R., F.B. and F.V. designed and implemented the model, datasets, visualisations, figures and videos. A.S.M. and A.R. designed and performed analysis experiments. M.G. and A.A.R. performed robot arm experiments. I.D., D.P.R., O.V. and D.R. assisted with maze navigation experiments. L.B. and T.W. assisted with Shepard-Metzler experiments. H.K., C.H., K.G., M.B., D.W., N.R., K.K. and D.H. managed, advised and contributed ideas to the project. S.M.A.E. and D.J.R. wrote the paper.

Competing interests: The authors declare no competing financial interests. DeepMind has filed a U.K. patent application (GP-201495-00-PCT) related to this work.

Datasets and materials availability: Datasets used in the experiments have been made available to download at https://github.com/deepmind/gqn-datasets.

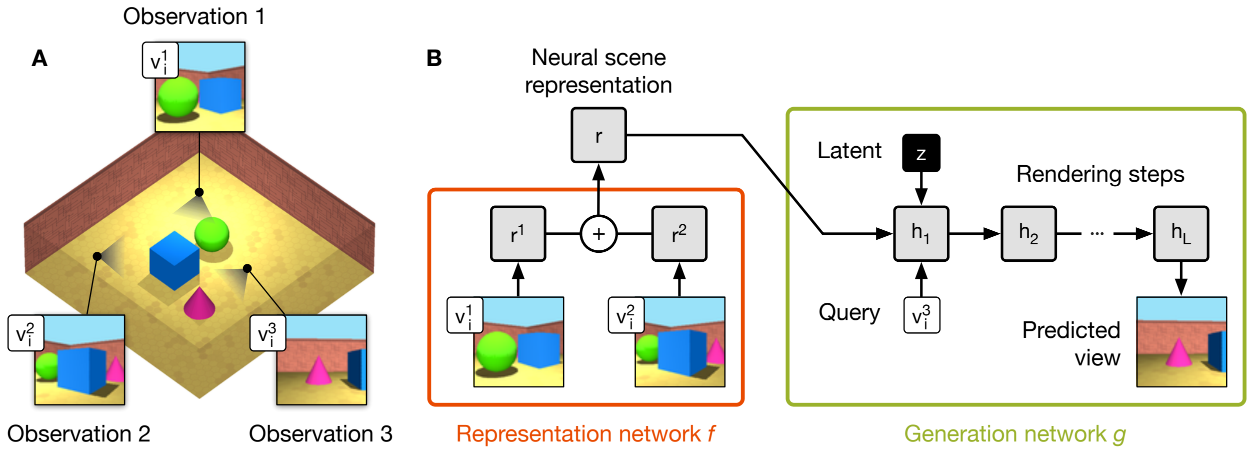


Fig. 1. **Schematic illustration of the Generative Query Network. (A)** The agent observes training scene i from different viewpoints (in this example from $\boldsymbol{v}_i^1$, $\boldsymbol{v}_i^2$ and $\boldsymbol{v}_i^3$). **(B)** The inputs to the representation network f are observations made from viewpoints $\boldsymbol{v}_i^1$ and $\boldsymbol{v}_i^2$, and the output is the scene representation $\boldsymbol{r}$, which is obtained by element-wise summing of the observations' representations. The generation network, a recurrent latent variable model, uses the representation to predict what the scene would look like from a different viewpoint $\boldsymbol{v}_i^3$. The generator can only succeed if $\boldsymbol{r}$ contains accurate and complete information about the contents of the scene (e.g., the identities, positions, colours and counts of the objects, as well as the room's colours). Training via back-propagation across many scenes, randomizing the number of observations, leads to learned scene representations that capture this information in a concise manner. Only a handful of observations need to be recorded from any single scene to train the GQN. h_1, h_2, … h_L are the L layers of the generation network.

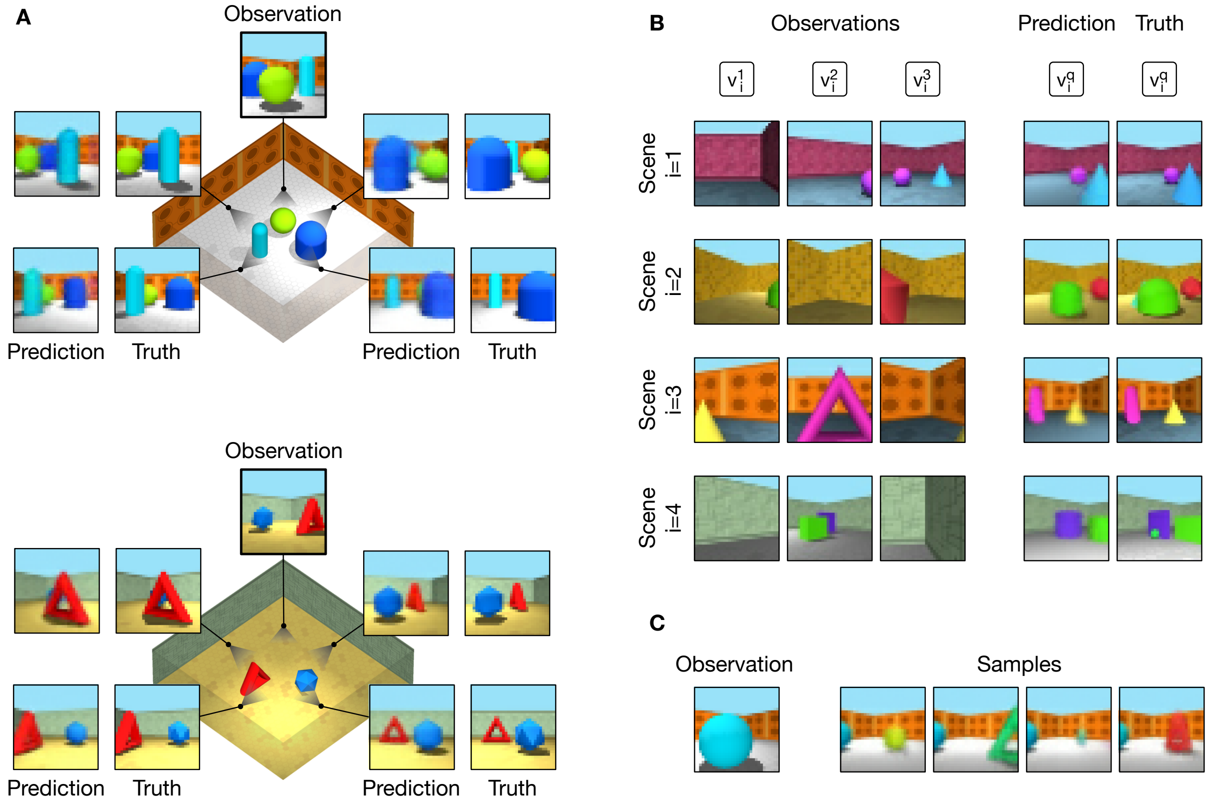


Fig. 2. Neural scene representation and rendering. (A) After having made a single observation of a previously unencountered test scene, the representation network produces a neural description of that scene. Given this neural description, the generator is capable of predicting accurate images from arbitrary query viewpoints. This implies that the scene description captures the identities, positions, colours, and counts of the objects, as well as the position of the light and the colours of the room. **(B)** The generator's predictions are consistent with laws of perspective, occlusion, and lighting (e.g., casting object shadows consistently). When observations provide views of different parts of the scene, the GQN correctly aggregates this information (scenes 2 and 3). **(C)** Sample variability indicates uncertainty over scene contents (in this instance, owing to heavy occlusion). Samples depict plausible scenes, with complete objects rendered in varying positions and colours (see fig. S7 for further examples). The model's behaviour is best visualised in movie format; see movie S1 for real-time, interactive querying of GQN's representation of test scenes (https://youtu.be/G-kWNQJ4idw).

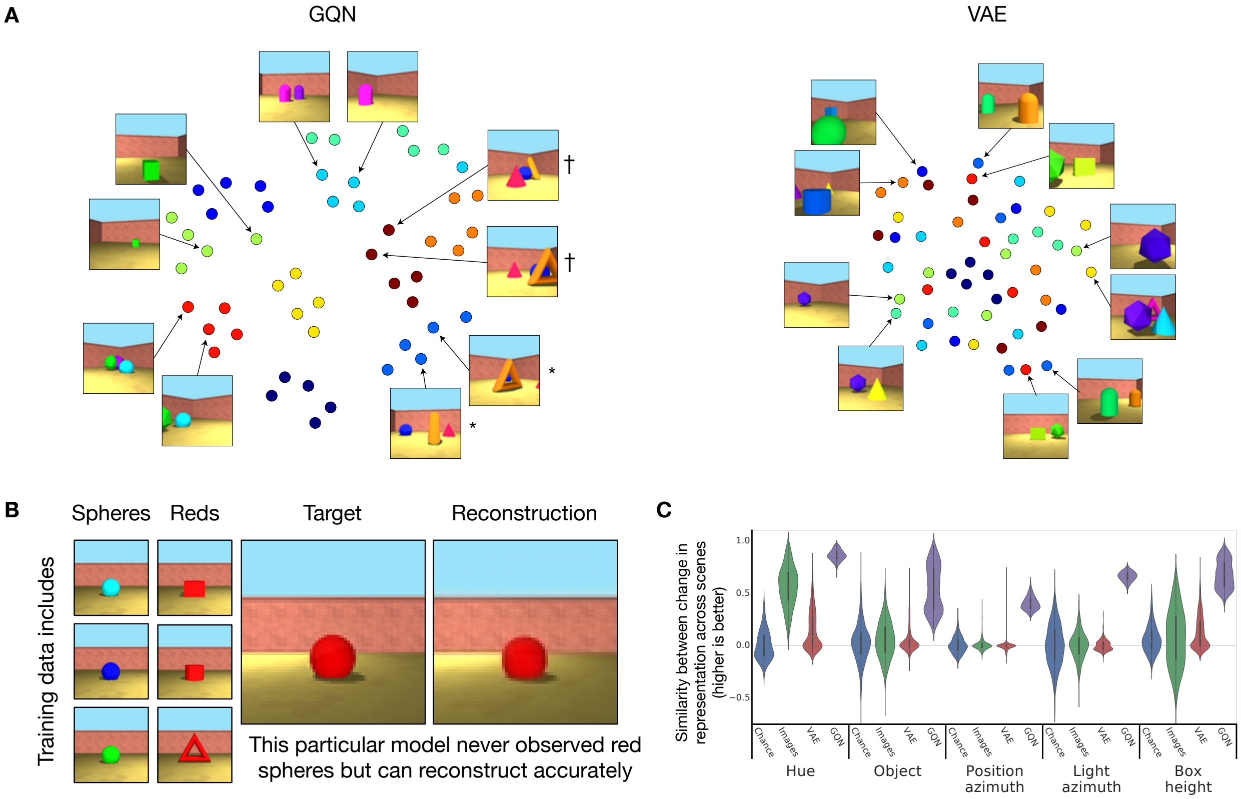


Fig. 3. Viewpoint invariance, compositionality and factorization of the learned scene representations. (A) t-SNE embeddings. t-SNE is a method for non-linear dimensionality reduction that approximately preserves the metric properties of the original high-dimensional data. Each dot represents a different view of a different scene, with colour indicating scene identity. Whereas the VAE clusters images mostly on the basis of wall angles, GQN clusters images of the same scene, independent of view (scene representations computed from each image individually). Two scenes with the same objects (represented by * and †) but in different positions are clearly separated. **(B)** Compositionality demonstrated by reconstruction of holdout shape-colour combinations. **(C)** GQN factorizes object and scene properties, because the effect of changing a specific property is similar across diverse scenes (as defined by mean cosine-similarity of the changes in the representation across scenes). For comparison, we plot chance factorization, as well as the factorization of the image-space and VAE representations. See section 5.3 of (17) for details.

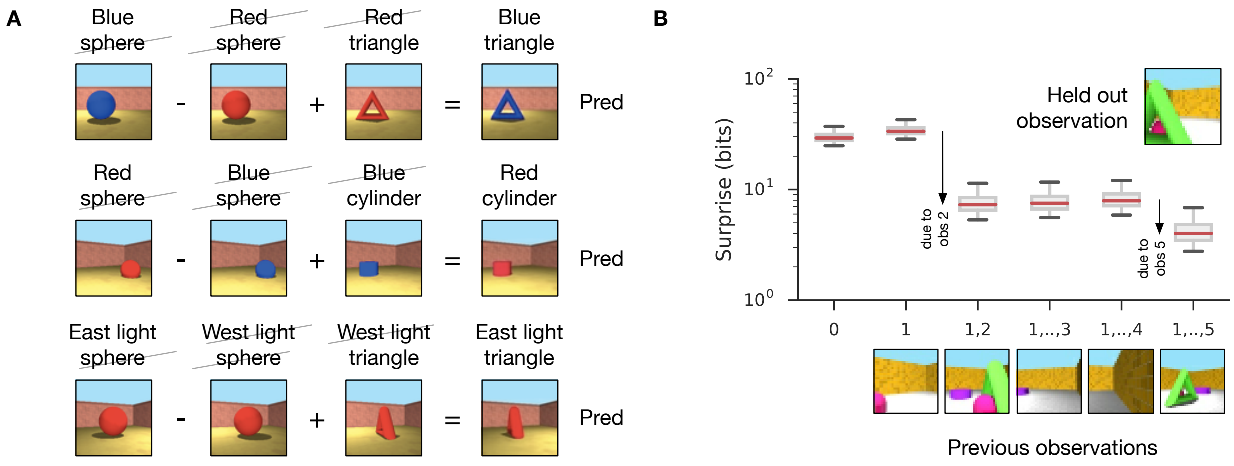


Fig. 4. Scene algebra and Bayesian surprise. (A) Adding and subtracting representations of related scenes enables control of object and scene properties via 'scene algebra', and indicates factorization of shapes, colours and positions. Pred, prediction. **(B)** Bayesian surprise at a new observation after having made observations 1 to k for k in 1 to 5. When the model observes images that contain information about the layout of the scene, its surprise (defined as the Kullback-Leibler divergence between conditional prior and posterior) at observing the held-out image decreases.

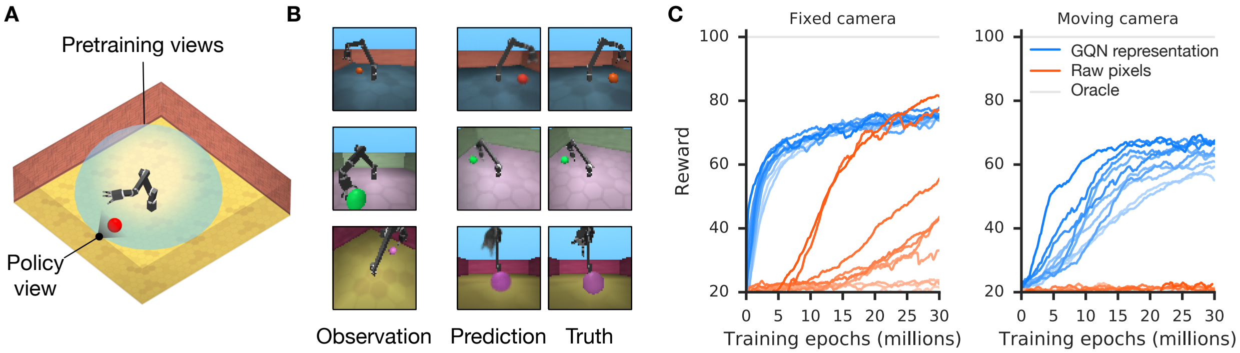


Fig. 5. GQN representation enables more robust and data-efficient control. (A) The goal is to learn to control a robotic arm to reach a randomly positioned coloured object. The controlling policy observes the scene from a fixed or moving camera (grey). We pretrain a GQN representation network by observing random configurations from random viewpoints inside a dome around the arm (light blue). **(B)** The GQN infers a scene representation that can accurately reconstruct the scene. **(C)** (Left) For a fixed camera, an asynchronous advantage actor-critic (A3C) reinforcement learning agent (*21*) learns to control the arm using roughly one-fourth as many experiences when using the GQN representation, as opposed to a standard method using raw pixels (lines correspond to different hyper-parameters; same hyper-parameters explored for both standard and GQN agents; both agents also receive viewpoint coordinates as inputs). The final performance achieved by learning from raw pixels can be slightly higher for some hyper-parameters, because some task-specific information might be lost when learning a compressed representation independently from the RL task as GQN does. Right: The benefit of GQN is most pronounced when the policy network's view on the scene moves from frame to frame, suggesting viewpoint invariance in its representation. We normalize scores such that a random agent achieves 0 and an agent trained on 'oracle', ground-truth state information achieves 100.

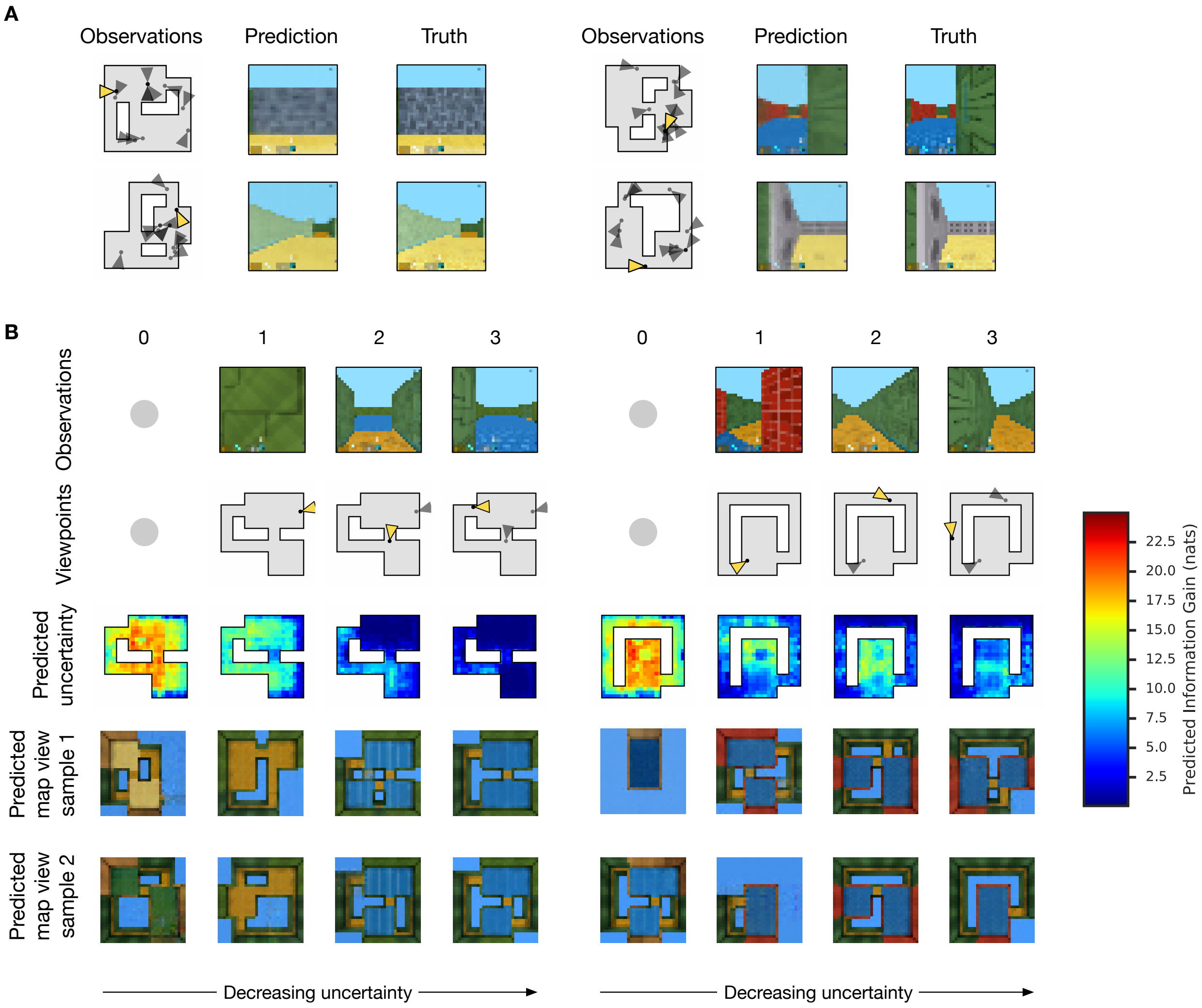


Fig. 6. Partial observability and uncertainty. (A) The agent (GQN) records several observations of a previously unencountered test maze (indicated by grey triangles). It is then capable of accurately predicting the image that would be observed at a query viewpoint (yellow triangle). It can accomplish this task only by aggregating information across multiple observations. **(B)** In the kth column, we condition GQN on observations 1 to k, and show GQN's predicted uncertainty, as well as two of GQN's sampled predictions of the top-down view of the maze. Predicted uncertainty is measured by computing the model's Bayesian surprise at each location, averaged over three different heading directions. The model's uncertainty decreases as more observations are made. As the number of observations increases, the model predicts the top-down view with increasing accuracy. See section 3 of (17), fig. S8 and movie S1

for further details and results (https://youtu.be/G-kWNQJ4idw). nats, natural units of information.

Supplementary Materials

Figures S1 – S16

Algorithms S1 – S3

References (S45) – (S52)

Table S1

Movie S1 (https://youtu.be/G-kWNQJ4idw)

Neural Scene Representation and Rendering
Supplementary Materials

1 Model details

1.1 Conditional generative models

Conditional latent variable models implicitly describe densities $g_\theta\left(\mathbf{x}|\mathbf{y}\right)$ over datapoints $\mathbf{x}$, given the conditioning variables $\mathbf{y}$, through a marginalisation over a set of latent variables $\mathbf{z}$:

$$g_\theta\left(\mathbf{x}|\mathbf{y}\right) = \int g_\theta\left(\mathbf{x}|\mathbf{z},\mathbf{y}\right)\pi_\theta\left(\mathbf{z}|\mathbf{y}\right)\,d\mathbf{z}, \tag{S1}$$

where $g_\theta\left(\mathbf{x}|\mathbf{z},\mathbf{y}\right)$ is a conditional density referred to as the observation model, $\pi_\theta\left(\mathbf{z}|\mathbf{y}\right)$ is a conditional prior, and θ is the set of parameters of the model. Training this model on a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ entails minimising the negative log-likelihood

$$\begin{aligned}\mathcal{L}\left(\theta\right) &= -\sum_i \ln g_\theta\left(\mathbf{x}_i|\mathbf{y}_i\right) &\text{(S2)}\\ &= -\sum_i \ln \int g_\theta\left(\mathbf{x}_i|\mathbf{z}_i,\mathbf{y}_i\right)\pi_\theta\left(\mathbf{z}_i|\mathbf{y}_i\right)\,d\mathbf{z}_i &\text{(S3)}\end{aligned}$$

with respect to θ. For most generative models of interest, it is intractable to optimize the negative log-likelihood $\mathcal{L}\left(\theta\right)$ directly due to the required integral over the high-dimensional latent variables $\mathbf{z}$, and we must resort to approximations. In this work we employ variational approximations (*45*) by instead minimising an upper-bound $\mathcal{F}$ to the negative log-likelihood ($-\mathcal{F}$ is also known as the evidence lower bound, or ELBO):

$$\begin{aligned}\mathcal{F}\left(\theta,\phi\right) &= \sum_i \int q_\phi\left(\mathbf{z}_i|\mathbf{x}_i,\mathbf{y}_i\right)\ln\frac{q_\phi\left(\mathbf{z}_i|\mathbf{x}_i,\mathbf{y}_i\right)}{g_\theta\left(\mathbf{x}_i|\mathbf{z}_i,\mathbf{y}_i\right)\pi_\theta\left(\mathbf{z}_i|y_i\right)}\,d\mathbf{z}_i, &\text{(S4)}\\ &= -\mathcal{L}\left(\theta\right) + \sum_i \mathrm{KL}[q_\phi\left(\cdot|\mathbf{x}_i,\mathbf{y}_i\right)|p_\theta\left(\cdot|\mathbf{x}_i,\mathbf{y}_i\right)], \\ &\geq -\mathcal{L}\left(\theta\right)\end{aligned}$$

where the density $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ is an approximation to the true posterior density and is parametrised by the vector ϕ and $\mathrm{KL}[q_\phi(\cdot|\mathbf{x}_i,\mathbf{y}_i)\,|p_\theta(\cdot|\mathbf{x}_i,\mathbf{y}_i)]$ is the KL-divergence between the aproximate posterior $q_\phi(\cdot|\mathbf{x}_i,\mathbf{y}_i)$ and the true posterior $p_\theta(\cdot|\mathbf{x}_i,\mathbf{y}_i)$. Learning in this formulation corresponds to jointly optimising the model parameters θ and variational parameters ϕ to minimise $\mathcal{F}(\theta,\phi)$.

The variational formulation allows for a straightforward optimization algorithm, where the gradients of $\mathcal{F}(\theta,\phi)$ with respect to θ and ϕ are approximated in an unbiased manner by drawing a small number of samples from $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$. This can be done in a computationally cheap and unbiased manner due to the integrals being outside the $\ln(\cdot)$ non-linearity, and also due to the re-parametrisation trick (22, 23).

1.2 Generative Query Networks

In the GQN setup we consider training datasets of the form $D = \left\{\left(\mathbf{x}_i^k,\mathbf{v}_i^k\right)\right\}$ with $i \in \{1,\ldots,N\}$ and $k \in \{1,\ldots,K\}$, where N is the number of scenes in the dataset, K is the number of recorded views of each scene, and $\mathbf{x}_i^k$ is an RGB image captured from viewpoint $\mathbf{v}_i^k$. Viewpoints are parametrised by a 5-dimensional vector $(\mathbf{w},\mathbf{y},\mathbf{p})$, where $\mathbf{w}$ is the three-dimensional position of the camera, $\mathbf{y}$ its yaw and $\mathbf{p}$ its pitch, however other parametrisations are also possible. Position, yaw and pitch are measured with respect to a fixed reference frame. We are interested in the task of predicting the image $\mathbf{x}_i^q$ that would be recorded from an arbitrary viewpoint $\mathbf{v}^q$, given a set of M observations $(\mathbf{x}_i^{1,\ldots,M},\mathbf{v}_i^{1,\ldots,M})$ from the same scene, for arbitrary $M \geq 0$. That is, the model should support prediction given no observations (i.e., sampling from its prior), a single observation, or even a larger number of observations than it has encountered during training.

In the general case, for any finite set of M observations $(\mathbf{x}_i^{1,\ldots,M},\mathbf{v}_i^{1,\ldots,M})$, it may be impossible to precisely predict an arbitrary view of a scene, due to the fact that objects occlude themselves and one another, and that each 2D observation only has finite coverage over 3D space. We address this challenge by using the framework of conditional generative modelling to train powerful stochastic generators. Through training, the models will form prior knowledge about probable configurations of object positions, shapes, lighting, textures and shadows and will use this knowledge to sample plausible images.

In our setting, the conditioning variables comprise the collection of all observed images $\mathbf{x}_i^{1,\ldots,M}$, their respective viewpoints $\mathbf{v}_i^{1,\ldots,M}$ and the query viewpoint $\mathbf{v}_i^q$. The target variable is the image $\mathbf{x}_i^q$ that would be observed from viewpoint $\mathbf{v}_i^q$: i.e., $\mathbf{x} = \mathbf{x}_i^q$.

With this notation we write the generator, prior and inference models as $g_\theta(\mathbf{x}|\mathbf{z},\mathbf{v}^q,\mathbf{r})$, $\pi_\theta(\mathbf{z}|\mathbf{v}^q,\mathbf{r})$ and $q_\phi(\mathbf{z}|x^q,\mathbf{v}^q,\mathbf{r})$ respectively, where $\mathbf{r} = f\left(\mathbf{x}^{1,\ldots,M},\mathbf{v}^{1,\ldots,M}\right)$ is effectively a summary of the observations and is computed by the scene representation network. The representation network $f\left(\mathbf{x}^{1,\ldots,M}, \mathbf{v}^{1,\ldots,M}\right)$ is defined by the following set of equations:

$$\hat{\mathbf{v}}^k = (\mathbf{w}^k, \cos(\mathbf{y}^k), \sin(\mathbf{y}^k), \cos(\mathbf{p}^k), \sin(\mathbf{p}^k)) \tag{S5}$$

$$\mathbf{r}^k = \psi\left(\mathbf{x}^k, \hat{\mathbf{v}}^k\right) \tag{S6}$$

$$\mathbf{r} = \sum_{k=1}^{M} \mathbf{r}^k, \tag{S7}$$

where $\psi\left(\mathbf{x}^k, \hat{\mathbf{v}}^k\right)$ is typically a convolutional network.

The additive aggregation function was found to work well in practice, despite its simplicity. Since the representation and generation networks are trained jointly, gradients from the generation network encourage the representation network to encode each observation independently *in such a way* that when they are summed element-wise, they form a valid scene representation.

For instance, one strategy that might arise, is for $\psi\left(\mathbf{x}^k, \hat{\mathbf{v}}^k\right)$ to transform the content of $\mathbf{x}^k$ to form a top-down 'map' of the contents of the scene as seen from $\mathbf{v}^k$. As evidence for the presence of objects is summed element-wise across views, the map becomes more confident and refined about the contents of the scene.

An additional benefit of this aggregation function is that it is permutation invariant, meaning the order in which observations are made has no effect on the final scene representation. The additive aggregation function may struggle due to interference, however, if the number of observations increases beyond a certain point. In our experiments, a plateau in model performance was observed when the number of context images exceeded 30.

1.3 Representation architecture

We define three possible choices of architecture for $\psi\left(\mathbf{x}^k, \hat{\mathbf{v}}^k\right)$ in Fig. S1. We consistently found the 'tower' representation architecture to learn fastest across datasets, which was therefore used in all experiments unless noted otherwise. Interestingly, the three architectures do not have the same factorisation and compositionality properties; we identified the 'pool' architecture to be more likely to exhibit view-invariant, factorised and compositional characteristics, and is therefore the architecture analysed in Fig. 3. See Section 5 for further details.

1.4 Generation architecture

We parametrise the conditional densities $g_\theta(\mathbf{x}|\mathbf{z}, \mathbf{v}^q, \mathbf{r})$ and $\pi_\theta(\mathbf{z}|\mathbf{v}^q, \mathbf{r})$ with deep neural networks inspired by recurrent latent Gaussian models (25), where the vector of latent variables $\mathbf{z}$ is split into L groups of latent variables $\mathbf{z}_l$, $l = 1, \ldots, L$ and the density over the variable of interest is constructed sequentially. Due to this sequential architecture, the prior $\pi_\theta(\mathbf{z}|\mathbf{v}^q, \mathbf{r})$ can be written as an auto-regressive density:

$$\pi_\theta(\mathbf{z}|\mathbf{v}^q, \mathbf{r}) = \prod_{l=1}^{L} \pi_{\theta_l}(\mathbf{z}_l|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}), \tag{S8}$$

where θ_l refers to the subset of parameters θ that are used by the conditional density at step l. The resulting model can be defined by a sequence of conditional computations expressed by the following equations:

$$\begin{aligned}
&\text{Scene encoder} & \mathbf{r} &= f\left(\mathbf{x}^{1,\ldots,M}, \mathbf{v}^{1,\ldots,M}\right) & \text{(S9)}\\
&\text{Initial state} & (\mathbf{c}_0^g, \mathbf{h}_0^g, \mathbf{u}_0) &= (\mathbf{0}, \mathbf{0}, \mathbf{0}) & \text{(S10)}\\
&\text{Prior factor} & \pi_{\theta_l}(\cdot|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}) &= \mathcal{N}\left(\cdot\middle|\eta_\theta^\pi(\mathbf{h}_l^g)\right) & \text{(S11)}\\
&\text{Prior sample} & \mathbf{z}_l &\sim \pi_{\theta_l}(\cdot|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}) & \text{(S12)}\\
&\text{State update} & \left(\mathbf{c}_{l+1}^g, \mathbf{h}_{l+1}^g, \mathbf{u}_{l+1}\right) &= C_\theta^g(\mathbf{v}^q, \mathbf{r}, \mathbf{c}_l^g, \mathbf{h}_l^g, \mathbf{u}_l, \mathbf{z}_l) & \text{(S13)}\\
&\text{Observation sample} & \mathbf{x} &\sim \mathcal{N}\left(\mathbf{x}^q\middle|\mu = \eta_\theta^g(\mathbf{u}_L), \sigma = \sigma_t\right), & \text{(S14)}
\end{aligned}$$

where the convolutional networks $\eta_\theta^\pi(\mathbf{h}_l^g)$ map its respective inputs to the sufficient statistics of a Gaussian density (i.e., means and standard deviations) and $\eta_\theta^g(\mathbf{u}_L)$ maps its inputs to the mean of Gaussian density, and the bulk of the computation at every layer is performed by the core C_θ^g, which is a skip-connection convolutional LSTM network defined by the equations

$$\begin{aligned}
&\text{Convolutional LSTM state update} & \left(\mathbf{c}_{l+1}^g, \mathbf{h}_{l+1}^g\right) &= \text{ConvLSTM}_\theta^g(\mathbf{v}^q, \mathbf{r}, \mathbf{c}_l^g, \mathbf{h}_l^g, \mathbf{z}_l) & \text{(S15)}\\
&\text{Skip connection state update} & \mathbf{u}_{l+1} &= \mathbf{u}_l + \Delta\left(\mathbf{h}_{l+1}^g\right), & \text{(S16)}
\end{aligned}$$

and $\mathbf{c}_l^g$ and $\mathbf{h}_l^g$ are the standard LSTM state variables (output and cell), ConvLSTM_θ^g is a size-preserving convolutional LSTM network and $\Delta\left(\mathbf{h}_{l+1}^g\right)$ is a transposed convolution which has the effect of up-sampling the image. Note that we use spatial $\mathbf{c}_l^g$ and $\mathbf{h}_l^g$ variables, to take advantage of the natural structure of images, and empirically we find this to outperform a fully-connected architecture. For all variables, the superscript g indicates that the corresponding variable is specific to the generative process, as opposed to the superscript e which will indicate below that the variable belongs to the encoder network in the inference process.

In practice we find it beneficial to anneal the per-pixel variance of the observation likelihood, Eq. (S14), over the duration of training (see Table S1), encouraging the model to focus on large-scale aspects of the prediction problem in the beginning and only later on the low-level details.

Due to the fact that we do not learn per-pixel variances, in figures, we show the mean value of each pixel conditioned on the sampled latent variables. We specify further implementation details visually, see Fig. S2.

1.5 Inference architecture

The variational posterior density $q_\phi\left(\mathbf{z}|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}\right)$ is also parametrised by a sequential neural network, specifically one that shares some of its parameters with the generative network. In other words, θ is a subset of ϕ. In analogy to the prior model, $q_\phi\left(\mathbf{z}|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}\right)$ is written as an auto-regressive density $q_\phi\left(\mathbf{z}|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}\right) = \prod_{l=1}^{L} q_{\phi_l}\left(\mathbf{z}_l|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}\right)$, where ϕ_l refers to the subset of parameters ϕ that are used by the conditional density at step l. The variational posterior can be expressed by the following equations:

$$\text{Scene encoder} \qquad \mathbf{r} = f\left(\mathbf{x}^{1,\ldots,M}, \mathbf{v}^{1,\ldots,M}\right) \tag{S17}$$

$$\text{Generator initial state} \qquad \left(\mathbf{c}^g_0, \mathbf{h}^g_0, \mathbf{u}_0\right) = \left(\mathbf{0}, \mathbf{0}, \mathbf{0}\right) \tag{S18}$$

$$\text{Inference initial state} \qquad \left(\mathbf{c}^e_0, \mathbf{h}^e_0\right) = \left(\mathbf{0}, \mathbf{0}\right) \tag{S19}$$

$$\text{Inference state update} \qquad \left(\mathbf{c}^e_{l+1}, \mathbf{h}^e_{l+1}\right) = C^e_\phi\left(\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{c}^e_l, \mathbf{h}^e_l, \mathbf{h}^g_l, \mathbf{u}_l\right) \tag{S20}$$

$$\text{Posterior factor} \qquad q_{\phi_l}\left(\cdot|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}\right) = \mathcal{N}\left(\cdot|\eta^q_\phi\left(\mathbf{h}^e_l\right)\right) \tag{S21}$$

$$\text{Posterior sample} \qquad \mathbf{z}_l \sim q_{\phi_l}\left(\cdot|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}\right) \tag{S22}$$

$$\text{Generator state update} \qquad \left(\mathbf{c}^g_{l+1}, \mathbf{h}^g_{l+1}, \mathbf{u}_{l+1}\right) = C^g_\theta\left(\mathbf{v}^q, \mathbf{r}, \mathbf{c}^g_l, \mathbf{h}^g_l, \mathbf{u}_l, \mathbf{z}_l\right) \tag{S23}$$

Here C^e_ϕ is a computational core dedicated to the inference process defined by a standard convolutional LSTM network. The convolutional network $\eta^q_\phi\left(\mathbf{h}^e_l\right)$ maps the inference network state to the sufficient statistics of the variational posterior $q_{\phi_l}\left(\cdot|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}\right)$ for the latent variables $\mathbf{z}_l$. Note that, through the dependence of C^e_ϕ on $\mathbf{h}^g_l$, the variational posterior defined by this architecture constitutes an auto-regressive density over $\mathbf{z}$ and is therefore capable of approximating very complex, multi-modal distributions.

2 Optimisation

As standard in variational approximations, the bound in Eq. (S4) can be decomposed into two main terms: the reconstruction likelihood and a regularization term,

$$\mathcal{F}\left(\theta, \phi\right) = \mathbb{E}_{(\mathbf{x},\mathbf{v})\sim D, \mathbf{z}\sim q_\phi}\left[-\ln \mathcal{N}\left(\mathbf{x}^q|\eta^g_\theta\left(\mathbf{u}_L\right)\right) + \sum_{l=1}^{L} \text{KL}\left[\mathcal{N}\left(\cdot|\eta^q_\phi\left(\mathbf{h}^e_l\right)\right) || \mathcal{N}\left(\cdot|\eta^\pi_\theta\left(\mathbf{h}^g_l\right)\right)\right]\right]. \tag{S24}$$

Due to the auto-regressive architecture of the model, the individual contributions of each computational step to the KL term are computed sequentially via Eqs. (S9) to (S23). Note that this equation is exact, and if we use a finite set of samples to evaluate the expectation, it provides an unbiased estimator of the bound.

In practice, to produce a numerical value for the bound, we sample from q_ϕ in a sequential manner, obtaining a chain of L samples for each term of the posterior. Although we cannot compute the sum of all KL terms contributing to the bound analytically, for each lth conditional KL term, we can compute its value analytically by conditioning on the $l-1$ preceding latent samples. This procedure retains the unbiased nature of the estimator, but has lower variance than estimating the conditional KL terms using only the samples.

Detailed pseudo-code for an unbiased estimator of Eq. (S24) is provided in Algorithm S2. Optimization is performed via adaptive gradient descent (*46*). Each gradient step is computed by first sampling a mini-batch of B scenes, each with a random number of M observations (between 0 and K) from the dataset D. Then a single sample $\mathbf{z} \sim q_\phi\left(\cdot|\mathbf{x}, \mathbf{y}\right)$ is drawn from the variational posterior defined by Eqs. (S17) to (S23) for every datapoint at every optimisation step. This procedure is described in detail in Algorithm S1. The procedure for generating conditional samples from GQN is detailed in Algorithm S3.

We train each GQN model simultaneously on 4 NVidia K80 GPUs for 2 million gradient steps. The values of the hyper-parameters used for optimisation are detailed in Table S1, and we show the effect of model size on final performance in Fig. S4.

3 Bayesian surprise

Bayesian surprise or *Information gain* measures the number of bits necessary to encode a new observation given previous observations (*47, 48*). Formally, given a conditional latent variable model of the form $g_\theta\left(\mathbf{x}|\mathbf{z}, \mathbf{y}\right)\pi_\theta\left(\mathbf{z}|\mathbf{y}\right)$ with posterior density $p\left(\mathbf{z}|\mathbf{x}, \mathbf{y}\right)$, the information gain about the latent variable $\mathbf{z}$ conditioned on previous available information $\mathbf{y}$ provided by a new observation $\mathbf{x}$ is defined as

$$
\begin{aligned}
\mathrm{IG}\left(\mathbf{x}, \mathbf{y}\right) &= \mathrm{KL}\left[p\left(\cdot|\mathbf{x}, \mathbf{y}\right)||\pi_\theta\left(\cdot|\mathbf{y}\right)\right] && \text{(S25)} \\
&\approx \mathrm{KL}\left[q_\phi\left(\cdot|\mathbf{x}, \mathbf{y}\right)||\pi_\theta\left(\cdot|\mathbf{y}\right)\right] && \text{(S26)} \\
&\approx \sum_{l=1}^{L} \mathrm{KL}\left[\mathcal{N}\left(\cdot|\eta_\phi^q\left(h_l^e\right)\right)||\mathcal{N}\left(\cdot|\eta_\theta^\pi\left(h_l^g\right)\right)\right]. && \text{(S27)}
\end{aligned}
$$

$\mathrm{IG}\left(\mathbf{x}, \mathbf{y}\right)$ in the GQN can be approximated by sampling from the inference model using Eqs. (S9) to (S23) multiple times and averaging. Information gain is an important tool to quantitatively analyse how much information the model can extract from a set of observations, and answers the question “How surprised is the model at observing $\mathbf{x}$ given it has already observed $\mathbf{y}$”.

In Fig. 4B and Fig. S5 we compute the information gain of GQN as a function of the number of context views for a single 3D scene and for a fixed new observation. We use 1000 samples from the inference network per configuration to generate the plots.

The reduction of information gain as new relevant observations are made demonstrates that GQN can efficiently integrate scene information as it becomes available, taking into account the rich scene prior learned by its generation network. We also compute the information gain of GQN as a function of the number of context views for a collection of 50 scenes in Fig. S6, demonstrating that the reduction of surprise as we increase the number of observations is a general effect.

A drawback of the information gain measure is that it must be computed for a particular, known target observation $\mathbf{x}$, which restricts its applicability in practice. A more widely applicable quantity is the *Predicted Information Gain*, defined as the expectation of $\mathrm{IG}\left(\mathbf{x}, \mathbf{y}\right)$ under the model:

$$\mathrm{PIG}\left(\mathbf{y}\right) = \mathbb{E}_{g_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})\pi_\theta(\mathbf{z}|\mathbf{y})}\left[\mathrm{IG}\left(\mathbf{x}, \mathbf{y}\right)\right]. \tag{S28}$$

In Fig. 6B and Fig. S8 we compute the predicted information gain of GQN at every location in a random maze by arranging test viewpoints on a uniform 30×30 grid covering the maze. For every point on the grid, we consider 3 different heading directions. The PIG is approximated at every point by averaging over 50 samples per heading directions. These results quantitatively demonstrate that GQN is consistently extracting and integrating spatial information about the layout of the mazes from the provided 2D observations. The reduction of model uncertainty as it receives more observations is also verified by the reduction in the variability of samples (Fig. S8).

Our results using IG and PIG suggest that both quantities can reliably measure and detect surprise in GQNs. For instance, these quantities could be used for active vision or spatial exploration, guiding the agent to maximally informative locations in the maze.

4 Experiment details

4.1 Rooms

We consider scenes of a variable number of random objects captured in a square room of size 7×7 units. Wall textures, floor textures as well as the shapes of the objects are randomly chosen within a fixed pool of discrete options. There are 5 possible wall textures (red, green, cerise, orange, yellow), 3 possible floor textures (yellow, white, blue) and 7 possible object shapes (box, sphere, cylinder, capsule, cone, icosahedron and triangle). Each scene contains 1, 2 or 3 objects.

The positions, sizes and colours of the objects and lights are randomised within a fixed continuous set. Object positions are sampled uniformly randomly at any real-valued location in a 3×3 square in the centre of the room, and the objects rotate by a real-valued amount around their vertical axis uniformly at random. Object colours are randomised in HSV space, with hue sampled uniformly between $[0, 1]$, saturation sampled uniformly between $[0.75, 1]$ and value set to 1. The light is positioned at a height of 15 units and its real-valued x and y position is sampled uniformly inside a 8×8 square centred at the centre of the room.

Images are rendered using MuJoCo's default OpenGL renderer (*49*). In order to capture an image for each random scene, we sample two points within the room, position the camera at the first and point it at the second. We sample 2 million scenes and 5 images per scene at a resolution of 64×64 in order to construct the dataset. The model is trained by conditioning on M observations, with M being randomly chosen between 1 and 5 in each mini-batch. We did not experiment with these numbers and it is likely that the same results could be obtained with a smaller number of scenes and context images. The dataset is split into train and test scenes at a 9 to 1 ratio. Further results of the model's performance on this dataset are shown in Fig. S7.

4.2 Shepard-Metzler objects

We also consider scenes consisting of a single 3D object composed of multiple parts (*50*), in order to test GQN's ability to represent larger combinatorial spaces and to model complex 3D object shapes. In these experiments, each object is composed of 7 randomly coloured cubes that are positioned by a self-avoiding random walk in 3D grid. As before, the camera is parametrised by its position, yaw and pitch, however it is constrained to only move around the object at a fixed distance from its centre.

Images are rendered using MuJoCo's default OpenGL renderer (*49*). We sample 2 million scenes and 15 images per scene at a resolution of 64×64 in order to construct the dataset. The model is trained by conditioning on M observations, with M being randomly chosen between 1 and 15 in each mini-batch. We did not experiment with these numbers and it is likely that the same results could be obtained with a smaller number of scenes and context images. The dataset is split into train and test scenes at a 9 to 1 ratio.

The performance of the model on this dataset is shown in Figs. S9 to S10. The GQN is capable of inferring the 3D structure of the object from even a single image, and is capable of re-

rendering the object from any viewpoint with a high degree of accuracy – in most cases the samples are indistinguishable from ground truth images. When the full configuration of the object is not uniquely determined by the observation, GQN samples consistent and plausible explanations. See supplementary video for further results.

4.3 Mazes

We create random mazes using an OpenGL-based DeepMind Lab game engine (*51*). Each maze is constructed out of an underlying 7 by 7 grid, with walls falling on the boundaries of the grid locations. However, the agent can be positioned at any continuous position in the maze. The mazes contain 1 or 2 rooms, with multiple connecting corridors. The walls and floor textures of each maze are determined by random uniform sampling from a predefined set of textures.

We sample 2 million scenes and 300 images per scene at a resolution of 64×64 in order to construct the dataset. The model is trained by conditioning on M observations, with M being randomly chosen between 1 and 20 in each mini-batch. We did not experiment with these numbers and it is likely that the same results could be obtained with a smaller number of scenes. The dataset is split into train and test scenes at a 9 to 1 ratio.

The environment additionally allows us to render the maze from above. We capture these images and train a *separate* generator network to produce top-down views of the maze from first-person observations. That is, the top-down view of the maze is never fed to the network as an observation, and gradients from the top-down view of the maze are never used to train the representation network. Further results of the model's performance on this dataset are shown in Fig. S8 and in the supplementary video.

4.4 Jaco arm

The Jaco arm reaching task is embedded into the MuJoCo (*49*) room environment. A MuJoCo reproduction of the robotic Jaco arm is placed in the middle of the room along with one spherical target object. The arm has nine joints. As in the previous setup, the appearance of the room is modified for each episode by randomly choosing a different texture for the walls and floor from a fixed pool of options. In addition, we modify both colour and position of the target randomly. Finally, the joint angles of the arm are also initialised at random within a range of physically sensible positions.

The goal of the RL task is for the hand to reach the target and remain close to it for the remaining duration of the episode. The reward obtained at every step is a decreasing function of the distance from the hand to the target:

$$d_{final} = 1 - \tanh^2\left(\max\left(0, \left(\frac{d_{palm} + d_{pinch}}{2} - 0.15\right) \times 10\right)\right), \tag{S29}$$

where d_{palm} and d_{pinch} are the distance from the target to either the palm of the hand or the pinch site weighted by $[1.41, 1.41, 1]$ along the x, y and z axes, respectively.

In order to carry out the reaching experiments we train two models: first we pre-train a GQN model on the scenes of the room containing the Jaco arm. We then use the representations from this model to train an RL agent separately. To ensure that GQN learns a complete representation of the Jaco-arm space we generate a dataset with a variety of arm positions. We achieve this by selecting random points in 3D space as targets for a proprioception-driven agent and recording one random intermediate state from the resulting trajectory. We do this with 50 independently trained agents to ensure diversity. We sample 4 million scenes and 20 images per scene to construct the dataset. The model is trained by conditioning on M observations, with M being randomly chosen between 1 and 7 in each mini-batch. Finally, to avoid having a very large state space for reinforcement learning we modify the representation network by adding two fully connected layers after the convolutional layers. These layers reduce the representation size from $8 \times 8 \times 128$ to 64×1.

Once we have trained the GQN model, we train a feed-forward A3C (*52*) agent from pixels using nine independent policies for each of the nine arm joints. The only difference to the previously published setup, apart from the change in environment, is that we modify the architecture of the A3C baseline input network to be identical to the representation network architecture of GQN for comparison. Crucially, while GQN is trained using several input images at each step, we only feed one image at every step during RL training in order to remain close to current experimental protocols. When training the agent, the pre-trained weights of the representation network are not updated. We compare our agent to standard A3C without pre-trained weights by randomly initialising the weights of the input network and updating them during RL training. To normalise the resulting scores we bound the performance with a random agent from below and an agent trained on oracle state information from above. The same hyper-parameters were used to train both groups of agents.

Because the GQN's scene representation vector has much lower dimensionality than the raw input images, we observe substantially more robust and data-efficient policy learning, obtaining convergence-level control performance with approximately 4 times fewer interactions with the environment than the standard A3C agent without pre-training of weights. Note, in particular, the sensitivity of the agent to the choice of hyper-parameters when the representation is learned from scratch, and only using RL. Training using the GQN representation, by comparison, is significantly more robust to the choice of hyper-parameters.

5 Analysis of scene representations

All analyses are performed in the room setting and unless otherwise noted, using the ‘pool’ representation network (see Fig. S1).

5.1 VAE

As a baseline for unconditional image compression, we use the representation learned by a convolutional ReLU variational autoencoder (22, 23). The VAE encoder network outputs a diagonal Gaussian density and is defined by a sequence of down-sizing convolutional layers: $64 \times 64 \times 3 \rightarrow 32 \times 32 \times 64 \rightarrow 16 \times 16 \times 128 \rightarrow 8 \times 8 \times 512 \rightarrow 1 \times 1 \times 256$. Similarly, the VAE decoder network is defined by a sequence of up-sizing convolutional layers: $1 \times 1 \times 256 \rightarrow 16 \times 16 \times 128 \rightarrow 32 \times 32 \times 512 \rightarrow 64 \times 64 \times 512 \rightarrow 64 \times 64 \times 3$. The VAE prior is also chosen to be a diagonal Gaussian density. The training procedure and hyper-parameters are the same as for the GQN model. After training the unconditional VAE on the same datasets as the GQN model, we use the representation learned by the encoder network for the t-SNE analysis in Fig. 3A, the trajectory analysis in Fig. 3C and Fig. S11A, and the view dependence analysis in Fig. S11B.

5.2 View dependence

If the GQN learns a view-invariant representation, the representations generated by different views of the same scene should be similar. It is challenging to interpret similarity metrics in high-dimensional spaces, however, and therefore we ask instead whether changes in scene or changes in viewpoint have a greater impact on the scene representation.

To evaluate this property, we first compute the representations resulting from single views of randomly generated room scenes drawn from the training distribution. Holding all other room properties constant (floor/wall texture, object shapes/colours/sizes, camera positions), we randomise the positions of all objects, creating a ‘shuffled scene’. Representations of the shuffled scenes are then computed. As a baseline, representations are also computed using the VAE model.

As a qualitative test of the scene representation’s view dependence, we reduce the dimensionality of the embeddings and visualise them using t-SNE (Fig. 3A). Data is pre-processed by reducing the dimensionality to 20 using principal components analysis. t-SNE embedding is performed using a perplexity of 15, early exaggeration of 100, and cosine similarity as the distance metric.

To test the scene representation's view dependence quantitatively, we measure the cosine distance between representations of the same scene at different viewpoints ('intra-scene'), and between representations of the original scenes and the shuffled scenes at the same and different viewpoints ('inter-scene'). We then plot these distances separately for both the inter- and intra-scene cases as a function of the angle between the viewpoints (Fig. S11B). This analysis demonstrates that, for the representations of the VAE and 'tower' GQN, the impact of changing the viewpoint by approximately 40 degrees is equivalent to changing the structure of the scene itself (e.g., by randomising object positions). In contrast, for the representation of the 'pool' GQN, and to a lesser extent, the 'pyramid' GQN, the impact of changing scene structure is consistently greater than that resulting from a change in the viewpoint. We note, however, that even in the average 'pool' GQN, changing the viewpoint still has a significant impact on the scene representation. Together, these results demonstrate that, while none of the GQN models are entirely view-invariant, for some GQN models, the configuration of the scene itself has a greater effect on the representation than the viewpoint.

5.3 Trajectory Analysis

If the representation of different object and scene properties is factorised in GQN, the effect of changing a single object property should be similar, regardless of other object and scene properties. To test this, we analyse a series of room images containing a single object, in which one object property is systematically varied, whilst all others are held constant. For example, to analyse object colour, we generate a series of room images in which a sphere of a fixed size at a fixed position with fixed views gradually changes colour. We then generate similar series for objects with different sets of fixed object properties and views. Importantly, the property of interest is varied identically across all scenes. The scene representation of each of these images is then computed, resulting in a one-dimensional 'trajectory' through representation space for each series. Representations are computed for each GQN representation network as well as for the VAE baseline.

If the representation is factorised, the shapes of these trajectories should be similar. Therefore, we next approximate the local gradient empirically at each point in the trajectory by simply calculating the first-order discrete difference as the property of interest is varied. We then calculate the mean pairwise cosine distance across trajectories. If two trajectories have identical shapes, the mean cosine distance between their local gradients would be 0, while if their shapes are uncorrelated, the mean cosine distance would be 1. Importantly, this analysis only measures the shape of the trajectories, and is invariant to differences in the absolute values of each representation.

We perform two critical controls to determine whether the resulting distances are meaningful. First, to determine chance similarity in representation space, we perform a permutation test by randomly shifting each trajectory along the property axis by a different amount, thereby misaligning the trajectories. We then calculate the similarity as above ('Chance' in Fig. 3C and 'Shuffled model' in Fig. S11A). Second, to determine whether the GQN representation network factorises object properties or merely maintains the factorisation present in the input images, we perform the above analysis in pixel space as well ('Images' in Fig. 3C and Fig. S11A). To match the summing operation across representations performed by the GQN representation network, images are summed prior to this analysis.

We find that neither the VAE nor the 'tower' GQN representation network factorise any of the object properties. Additionally, with the exception of object hue, object properties are not factorised in image-space. However, both the 'pool' and 'pyramid' GQN representation networks factorise all object properties to varying extents (Fig. S11A).

5.4 Compositionality

If the GQN learns a factorised, compositional representation, it should exhibit compositional behaviour. We therefore test GQN's ability to combine observed object primitives to generate novel objects. We train an instance of the GQN on a dataset containing red objects and spheres of various colours, but no red spheres. If the GQN learns to 'understand' colour and shape independently, it should be able to reconstruct views of scenes containing red spheres at inference time. We find that GQN is able to generate samples containing red spheres, providing an existence proof that both GQN's representation and generation networks can exhibit compositional behaviour (Fig. 3B). Importantly, however, this effect is not completely robust, as red cylinders are often generated in place of red spheres (in roughly 30% to 50% of samples).

5.5 Scene Algebra

To perform 'scene algebra', we compute the GQN representation resulting from multiple independent scenes. Unless otherwise specified, all representations are generated from the same set of views. We perform arithmetic in representation space, adding and subtracting representations to generate representations which should modify an object in a predictable fashion. For example, starting with the representation resulting from a red sphere, we subtract the representation resulting from a blue sphere and add the representation resulting from a blue cylinder. In this case, the sphere property and the blue property should each be cancelled out, leaving a representation of a red cylinder. Samples are then drawn from the generation network, conditioned on the new representation.

We find that scene algebra generates the correct object modifications for a variety of object properties, and is able to recombine properties even across object positions (Fig. 4A). However, scene algebra also fails in several interesting ways. For example, our choice of representation network architecture appears to not support scene algebra across scenes with different sets of views, nor can it add scenes with different objects together (Fig. S12).

5.6 Generalisation Failure Modes

In Fig. S13, we train a GQN on objects of varying sizes, colours and shapes as before; however now we test its performance on a number of out-of-distribution scenes, specifically scenes containing previously unseen objects (half-cylinders, walls and coloured floors). In some cases (half-cylinders) the model's performance is surprisingly good, generalising to great effect. However its renders are inconsistent for strongly out-of-distribution scenes (walls which are taller than any previously seen object).

In Fig. S14, we investigate the degree to which this generalisation capability is dependent on the number of context observations. Interestingly, we find that while the GQN's ability to produce previously seen objects is largely unaffected by the number of context observations, its ability to generalise to novel objects is highly dependent on the number of context observations, as the GQN's ability to generalise decreases substantially when it has fewer opportunities to observe the out-of-distribution scene.

In Fig. S15, we add increasing amounts of noise to the images that are provided to the GQN as observations. We find that, perhaps unsurprisingly, for models trained exclusively on noiseless images, performance degrades as the degree of noise increases. We expect, however, that the model could easily overcome this sensitivity by training or fine-tuning on noisy data.

Finally, in Fig. S16, we train GQNs on up to 3 objects as before, but now test their performance on a number of strongly out-of-distribution scenes with 4 or 7 objects each. We observe that, depending on the choice of representation network architecture, the model generalises to varying degrees. Interestingly, while the 'tower' representation, which contains a spatially arranged scene representation, extrapolates quite well, the 'average pool' representation, which does not preserve spatial information and appears to bind object properties in a manner which assumes a maximum number of objects, struggles most as the number of objects increases.

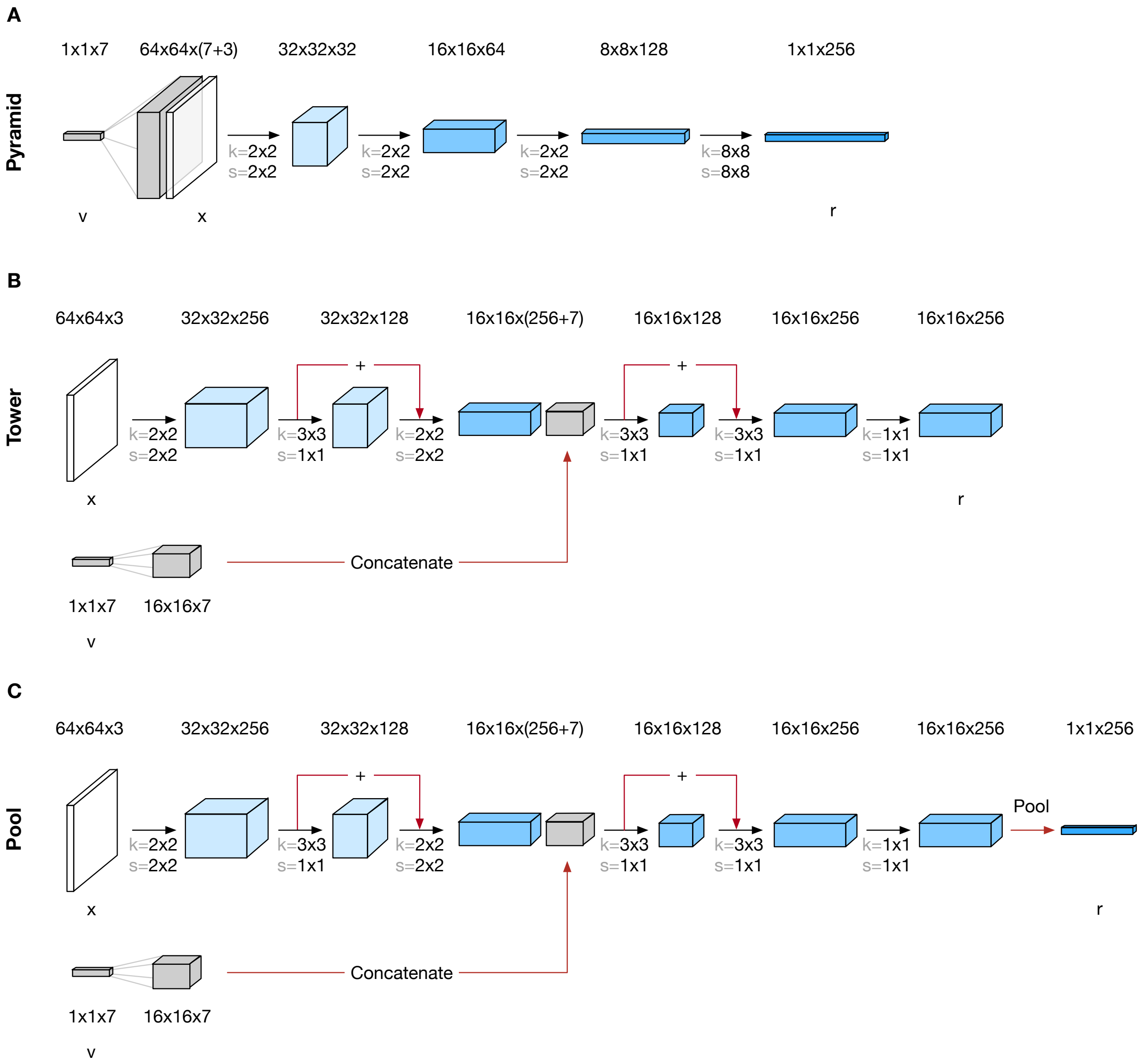


Figure S1: **Representation network architecture**. Implementation details of three possible architectures for the representation network, which given an image $\mathbf{x}$ and corresponding viewpoint $\mathbf{v}$, produces a representation $\mathbf{r}$: **(A)** Pyramid. **(B)** Tower. **(C)** Pool (like Tower, but followed by an average pooling layer that reduces the representation size to 1×1). All black arrows represent convolutional layers followed by rectified linear activations (ReLUs), with kernel and stride indicated by k and s. Convolutions of stride 1×1 are size preserving, whilst all others are 'valid'. Red arrows marked with '+' indicate residual connections. When concatenating the viewpoint $\mathbf{v}$ to an image or feature map, its values are 'broadcast' in the spatial dimensions to obtain the correct size. In all cases, when more than one observation is available, the resulting representations are summed element-wise to form an aggregate representation of the scene.

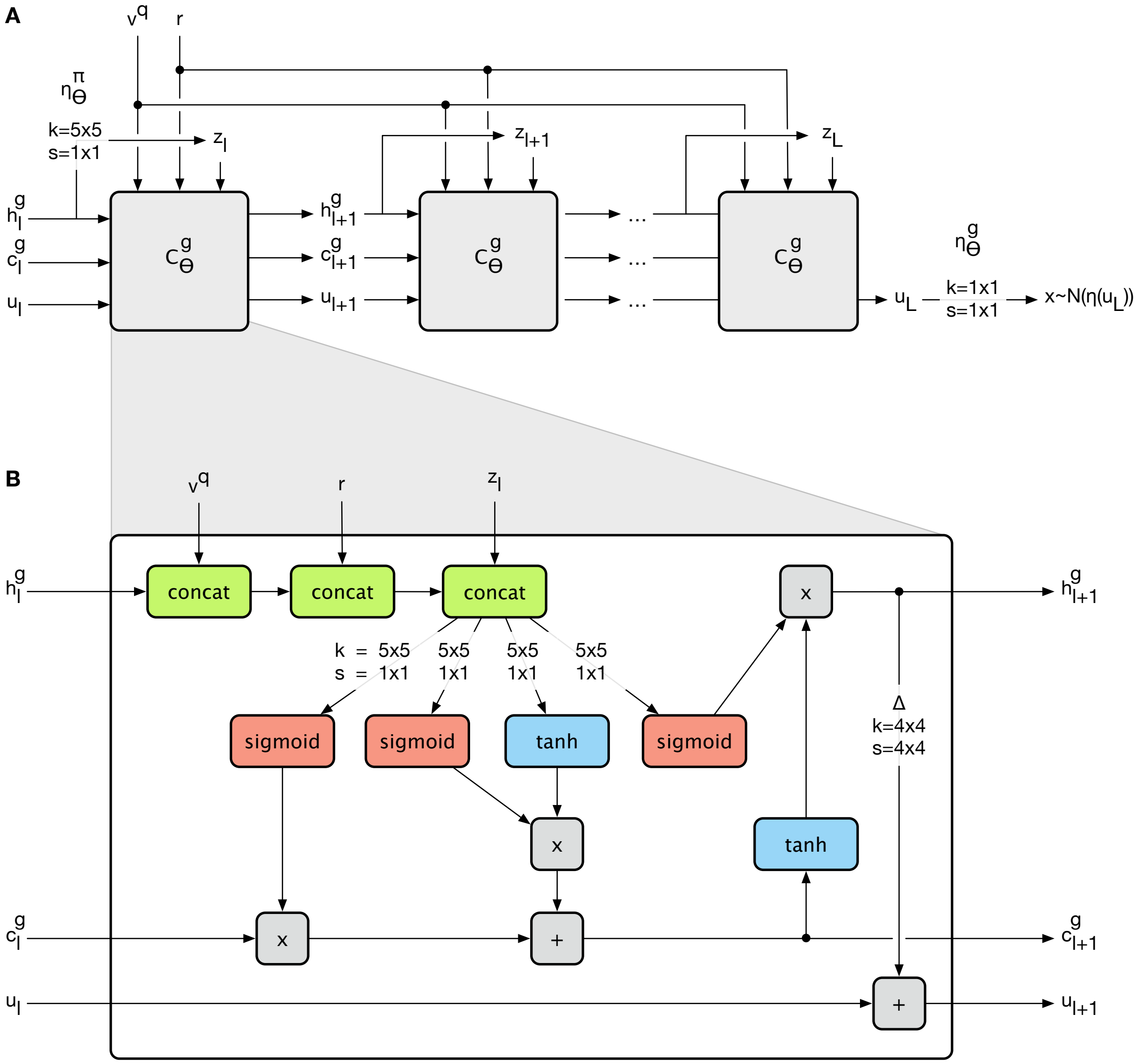


Figure S2: **Generation network architecture**. Implementation details of one possible architecture for the generation network, which given query viewpoint $\mathbf{v}^q$ and representation $\mathbf{r}$ defines the distribution $g_\theta\left(\mathbf{x}^q|\mathbf{v}^q, \mathbf{r}\right)$ from which images can be sampled. Convolutional kernel and stride sizes are indicated by k and s respectively. Convolutions of stride 1×1 are size preserving, whilst all others are 'valid'. **(A)** The architecture produces the parameters of the output distribution through the application of a sequence of computational cores C^g_θ that take $\mathbf{v}^q$ and $\mathbf{r}$ as input. At each iteration l, a distribution over the latents $\mathbf{z}_l$ is computed as a function of $\mathbf{h}^g_l$, sampled from, and fed as an additional input to the core. **(B)** Each core is a skip-convolutional LSTM network, with output $\mathbf{h}^g_l$, cell state $\mathbf{c}^g_l$ and $\mathbf{u}_l$ acting as the skip-connection pathway.

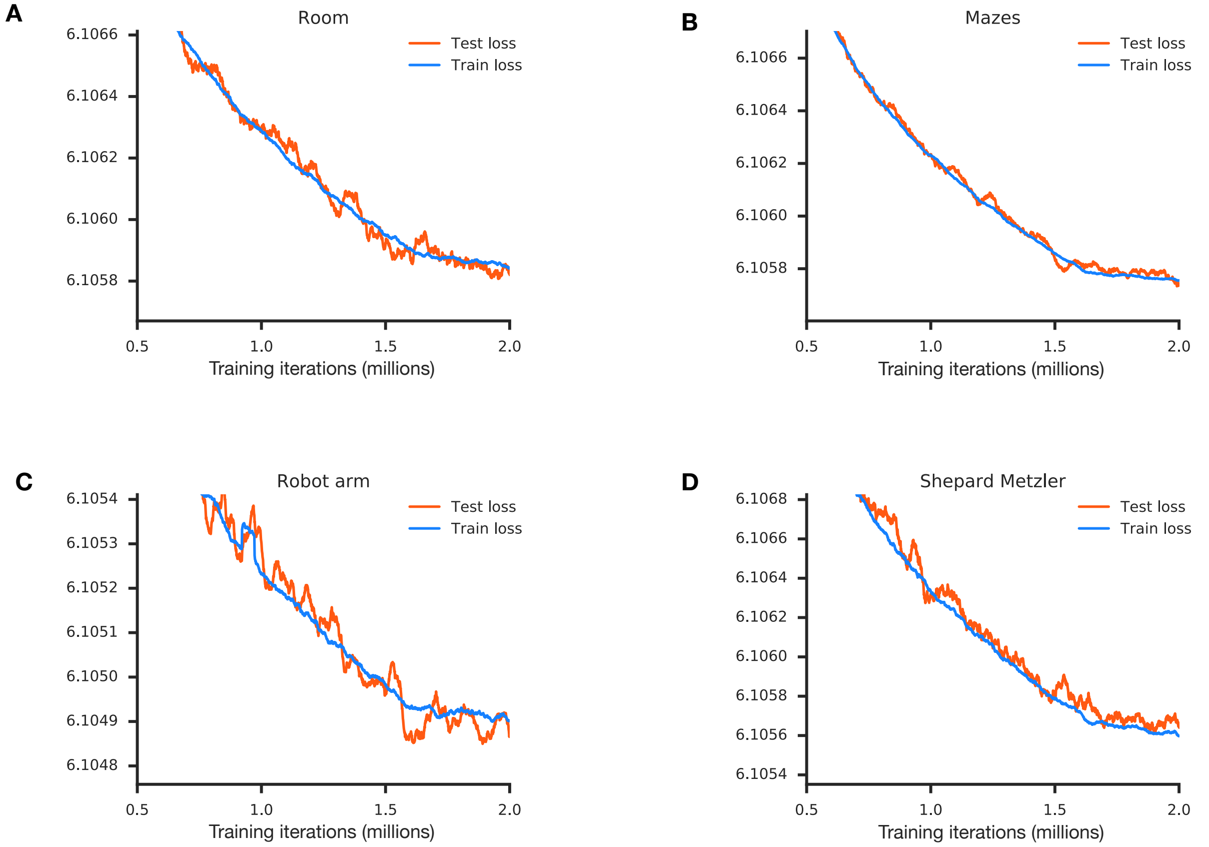


Figure S3: **Model generalisation**. Each dataset is split into train and test subsets at a ratio of 9 to 1, i.e., a whole scene (e.g., configuration of objects, room layout) and all of its observations are either present in the train set, or in the test set, but not both. The GQN's loss is monitored on the train and test datasets throughout optimisation. Train and test losses closely match for **(A)** room **(B)** maze **(C)** robot arm **(D)** Shepard-Metzler environments, ruling out the possibility of overfitting to particular scene configurations. Note that generalisation is further demonstrated by GQN's ability to generate accurate novel viewpoints, despite only ever observing any particular training scene from a handful of positions.

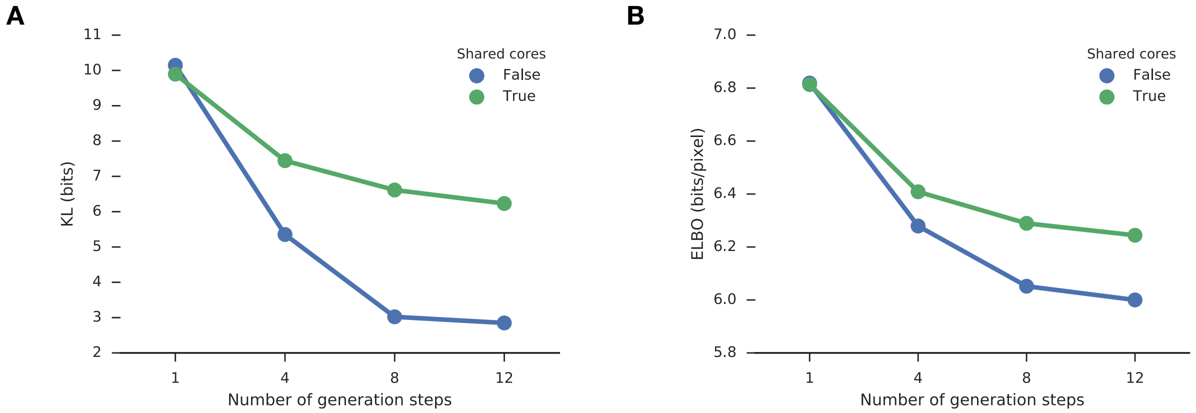


Figure S4: **Effect of generator size on model performance**. We compare several GQN variants after a fixed number of training iterations. Deeper models perform best, obtaining **(A)** higher likelihood (lower ELBO), and **(B)** lower KL between posterior and conditional prior upon observing ground-truth images at query viewpoints, however of course they are slower to train. We also observe that not sharing the weights of the cores across generation steps slightly improves overall performance. In separate experiments, we found that a GQN trained with a variational autoencoder (VAE) as generator (5 convolutional encoding layers and 5 convolutional decoding layers) achieves 6.71 (bits/pixel) after the same number of training iterations, i.e., only marginally stronger than a single-step iterative generator.

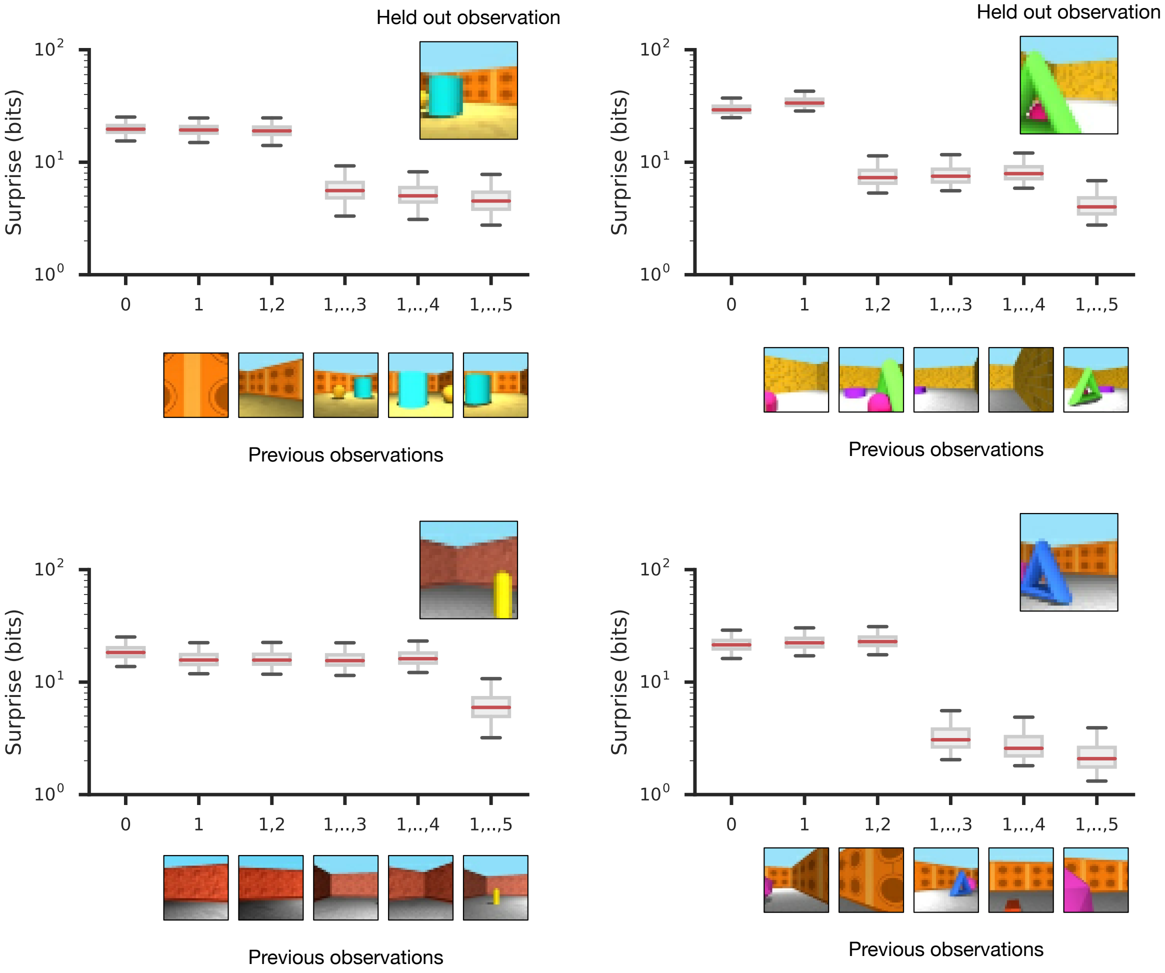


Figure S5: **Information gain**. For each scene, we plot the model's Bayesian surprise of a new observation after having made observations 1 to k for each k in 1 to 5. The model's surprise of the held-out observation drops most sharply when it views the scene from a position that informs it about the position, identity and colour of the object in view. Additional observations reduce the surprise as the model determines these properties with higher precision by aggregating information across views. For instance in the first scene (top left), the model is surprised about the held out observation after having observed 0, 1 or 2 images, however the third image which contains information about the blue cylinder reduces its uncertainty. In the second scene (top right), observations 2 and 5 both contribute to a reduction in surprise. Errors bars are computed by sampling multiple times from the generator's posterior.

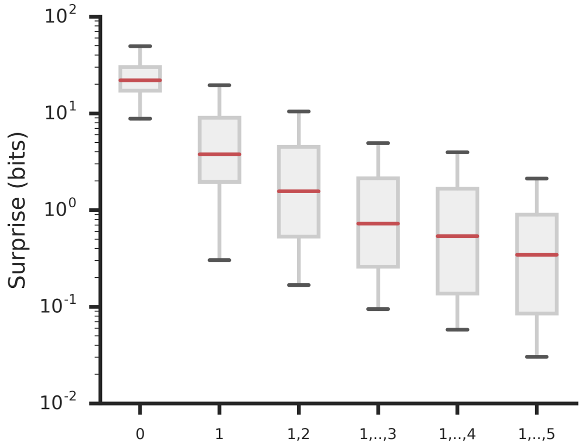


Figure S6: **Average information gain as a function of number of observations**. We show the distribution of information gain estimates averaged across a collection of 50 random scenes in the room. This demonstrates a general trend towards a reduction in the model's uncertainty as the number of observations grows.

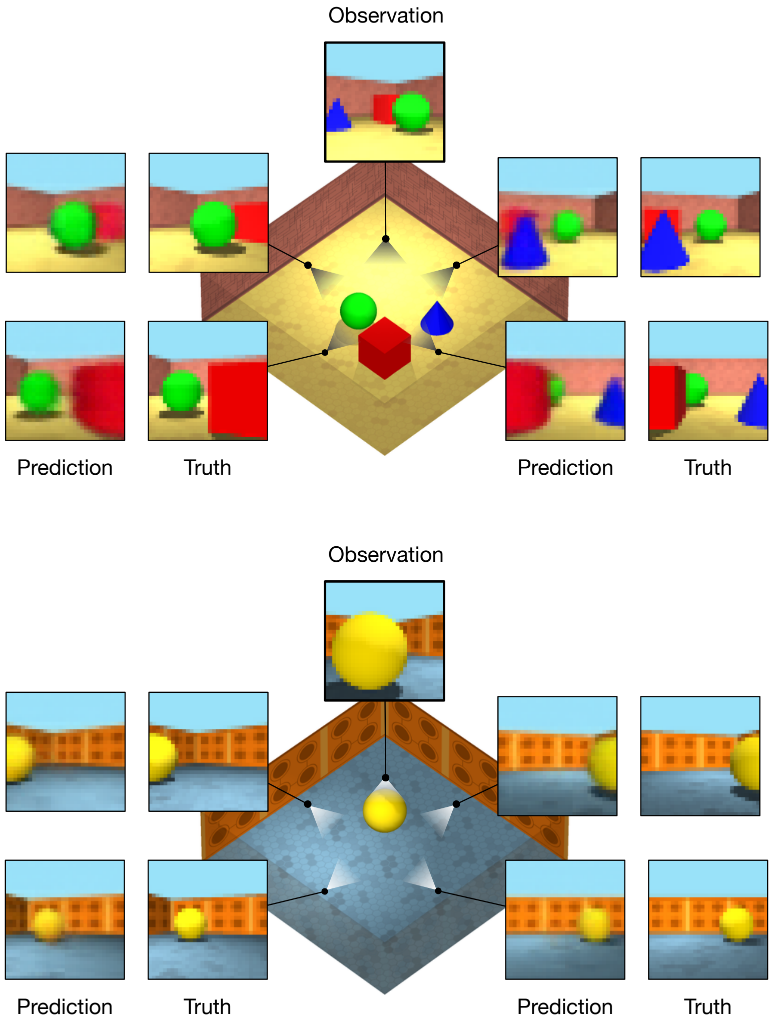


Figure S7: **Neural scene representation and rendering**. Given a single observation of a test scene, the representation network produces a neural description of the scene. The generator is capable of predicting accurate images from arbitrary query viewpoints. This implies that the scene description captures the identities, counts, positions, colours of the objects, as well as the position of the light, and the colours of the walls and floor.

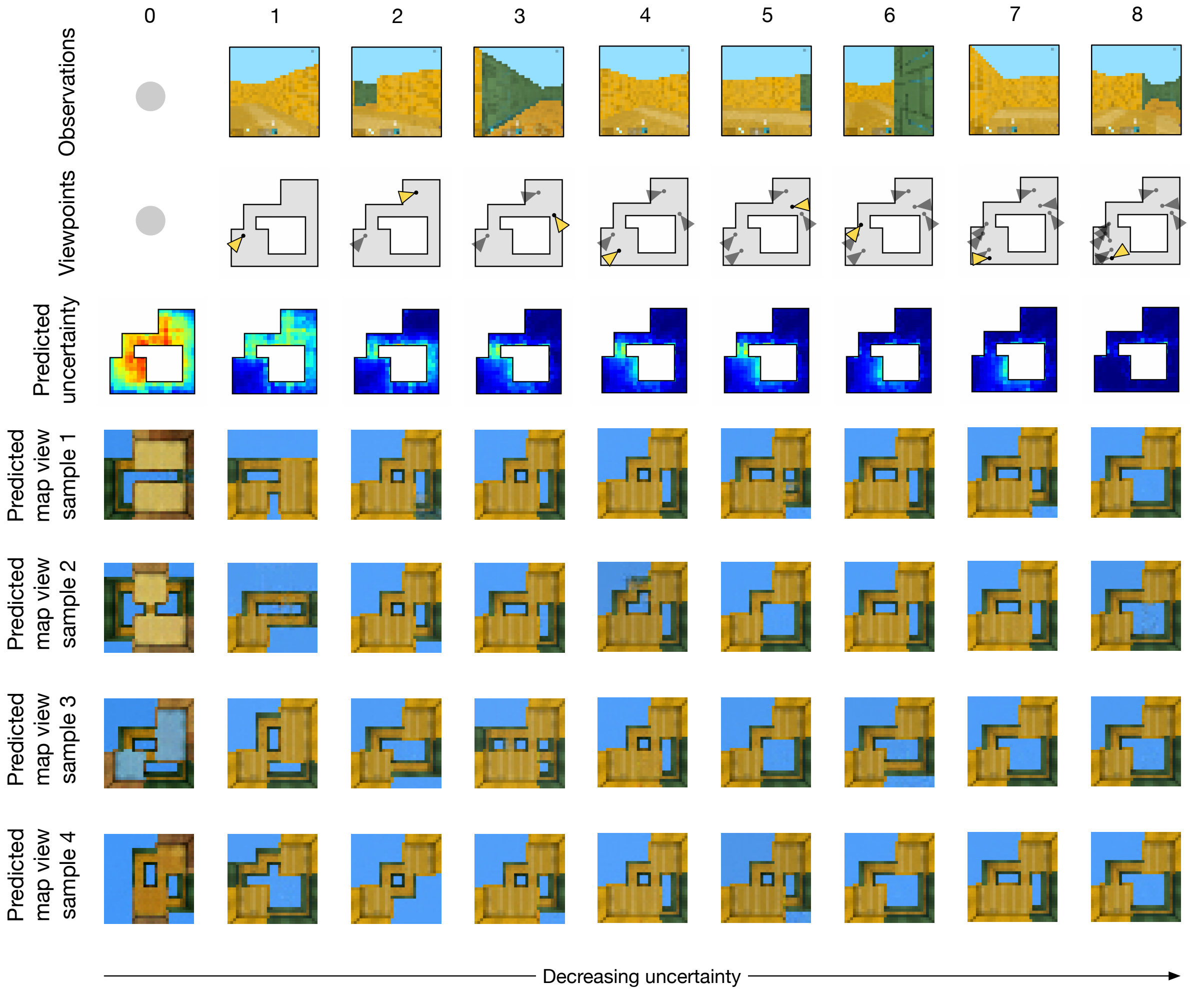

0
1
2
3
4
5
6
7
8
Observations
Viewpoints
Predicted uncertainty
Predicted map view sample 1
Predicted map view sample 2
Predicted map view sample 3
Predicted map view sample 4
Decreasing uncertainty

Figure S8: **Partial observability and uncertainty**. In the kth column, we condition GQN on observations 1 to k, and show GQN's predicted uncertainty, as well as four of GQN's sampled predictions of the top-down view of the maze. Predicted uncertainty is measured by computing the model's predicted information gain at each location, averaged over 3 different heading directions. This measures how uncertain the model itself thinks it is at every location, and for instance can be used for exploration. The model's predicted uncertainty decreases as more observations are made, which is also evident in the reduction of variability in its top-down samples. With only a handful of first-person observations, the model is capable of predicting the top-down view with high accuracy, indicating successful integration of egocentric observations. Errors often correspond to the precise points at which corridors connect with rooms. See supplementary video for further results.

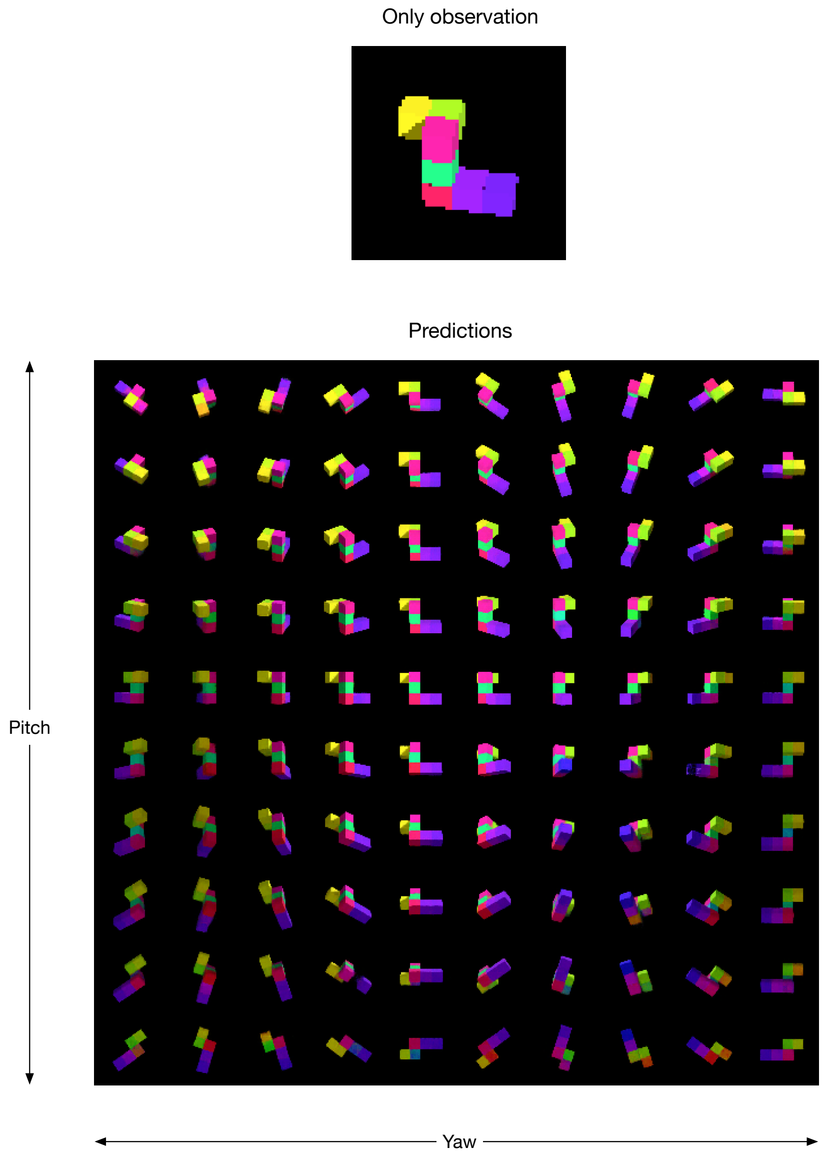


Figure S9: **Shepard-Metzler environment**. Given a single observation of a test object, the representation network produces a neural description of it. The generator is capable of predicting accurate images from arbitrary query viewpoints. This implies that the scene description accurately captures the positions and colours of multiple parts that make up the object. The model's predictions are consistent with occlusion, lighting and shading, and are typically indistinguishable from ground-truth.

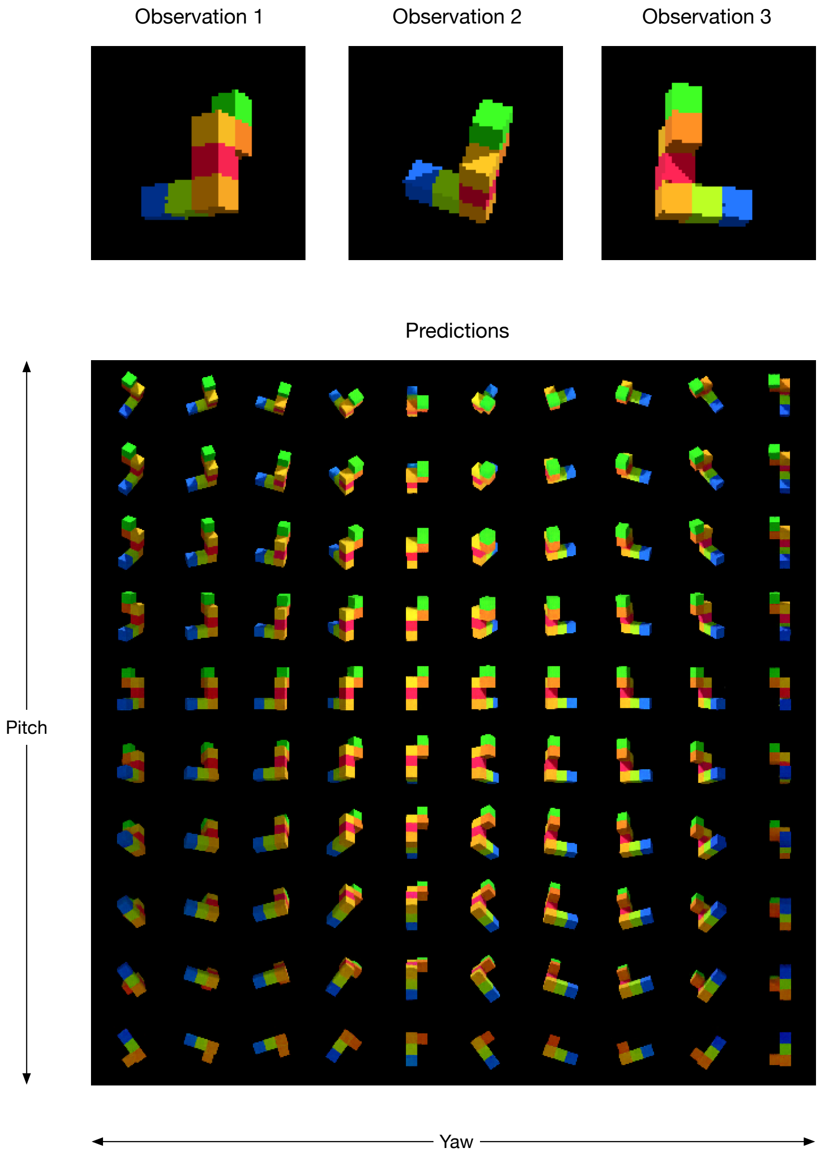


Figure S10: **Shepard-Metzler environment**. Given 3 observations of a test object, the representation network produces a neural description of it. The generator is capable of predicting accurate images from arbitrary query viewpoints. This implies that the scene description accurately captures the positions and colours of multiple parts that make up the object. The model's predictions are consistent with occlusion, lighting and shading, and are typically indistinguishable from ground-truth.

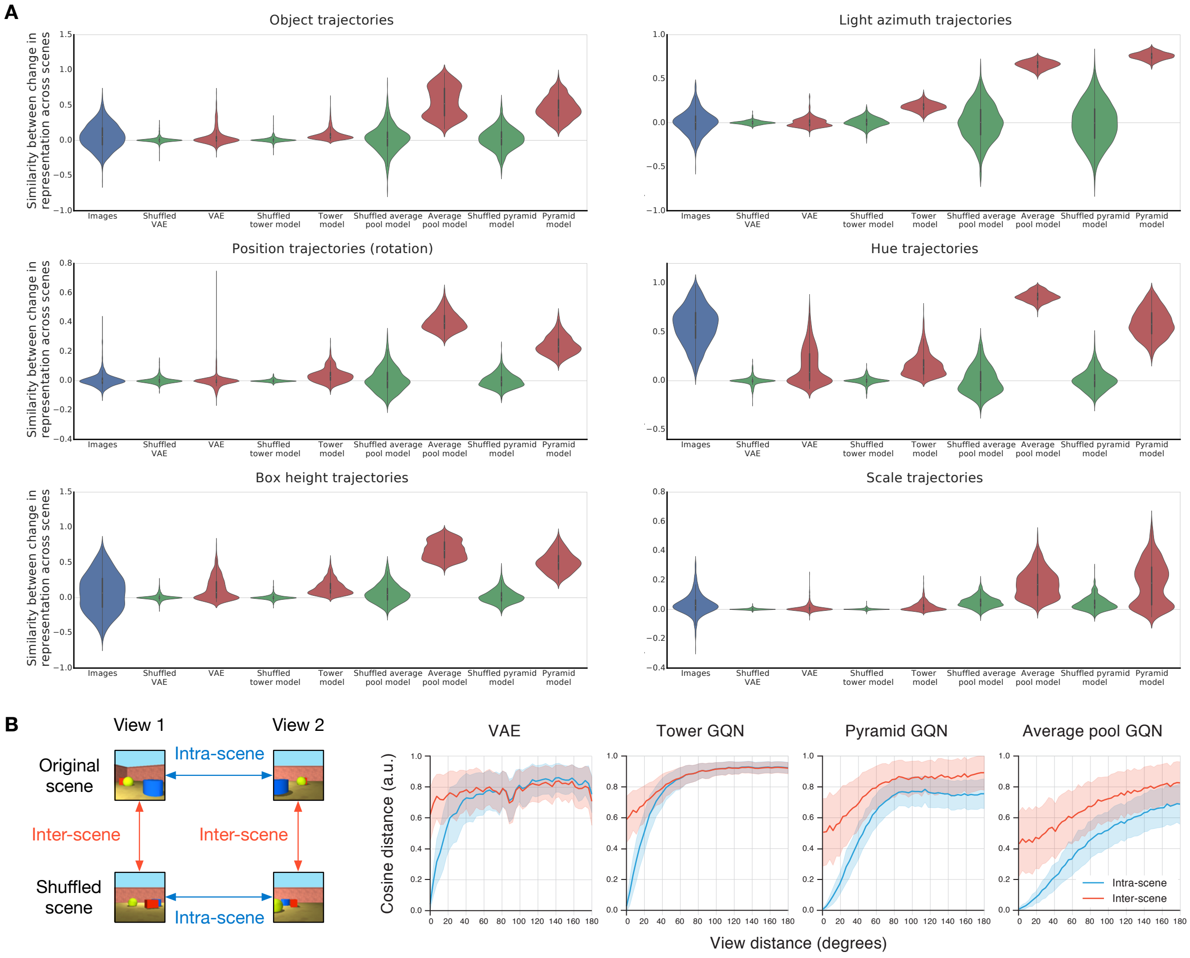


Figure S11: **Analysis**. **(A)** To test whether the GQN learns a factorised representation, we investigate whether changing a single scene property (e.g., object colour) whilst keeping others fixed (e.g., object shape and position), leads to similar changes in the scene representation (Section 5.3). Consistently, the 'pool' and 'pyramid' representation network result in factorised representations, while the VAE and 'tower' representation network result in joint representations of object properties (e.g., object shape impacts the way object colour is represented). **(B)** Quantitative view dependence analysis demonstrates that for the VAE and 'tower' GQN representation network, changes in view larger than 40 degrees and changes in scene have similar impacts on the resulting representation. In contrast, for the 'pool' representation network, and, to a lesser extent, the 'pyramid' representation network, changes in scene are consistently more impactful than changes in view. See Section 5.2 for details.

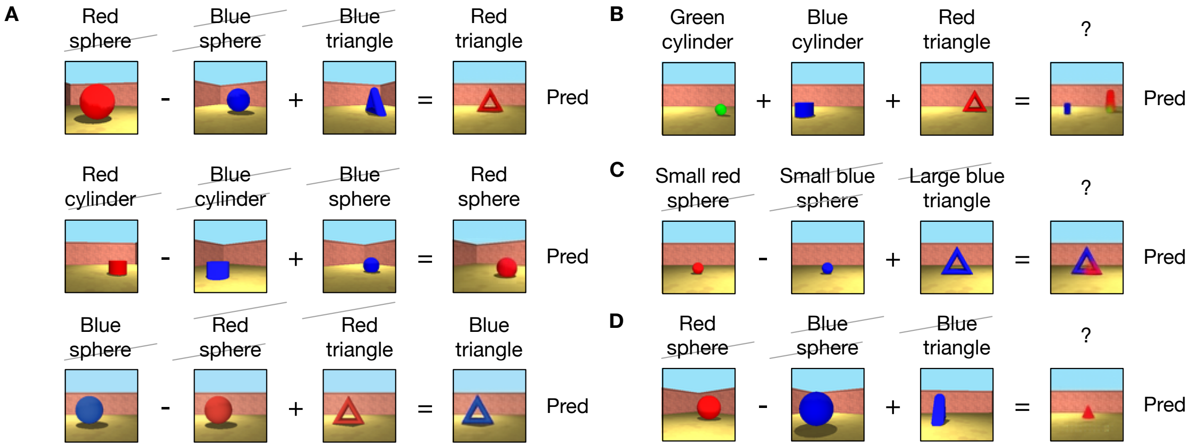


Figure S12: **Scene algebra**. **(A)** Additional scene algebra successes. For objects in the same position, scene algebra operates successfully when inputs are conditioned on different sets of views. **(B)-(D)**, Examples of scene algebra failures: **(B)** for the addition of multiple objects, **(C)** for objects with different sizes, and **(D)** across different views and object positions.

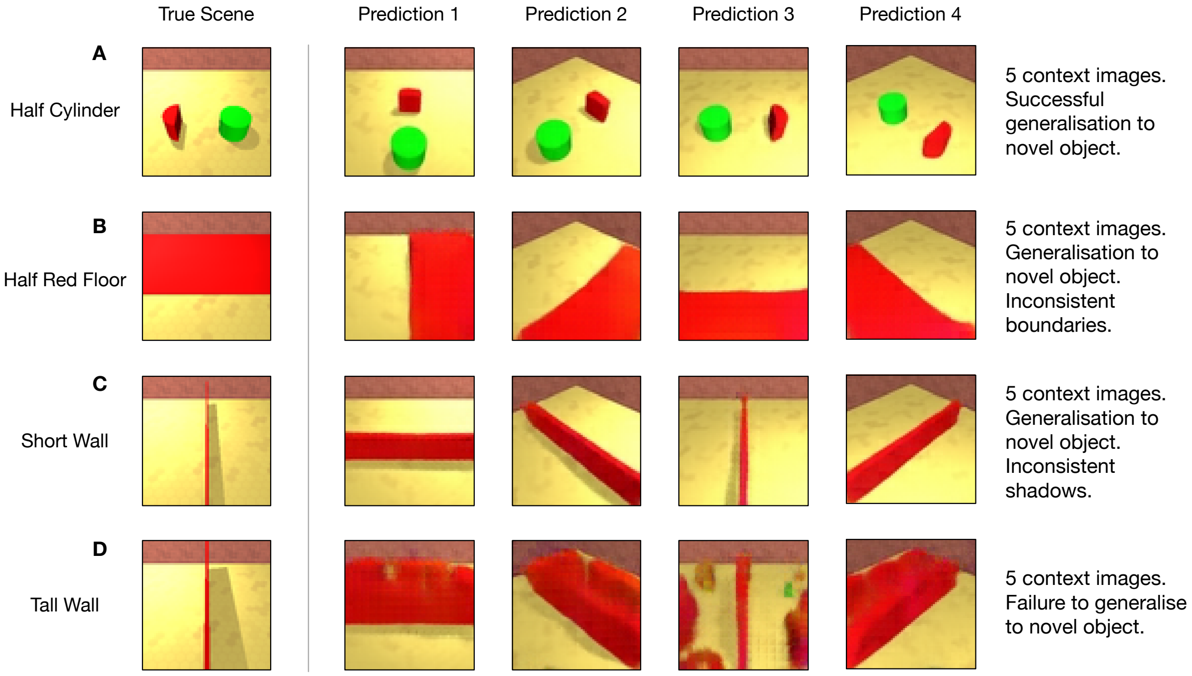
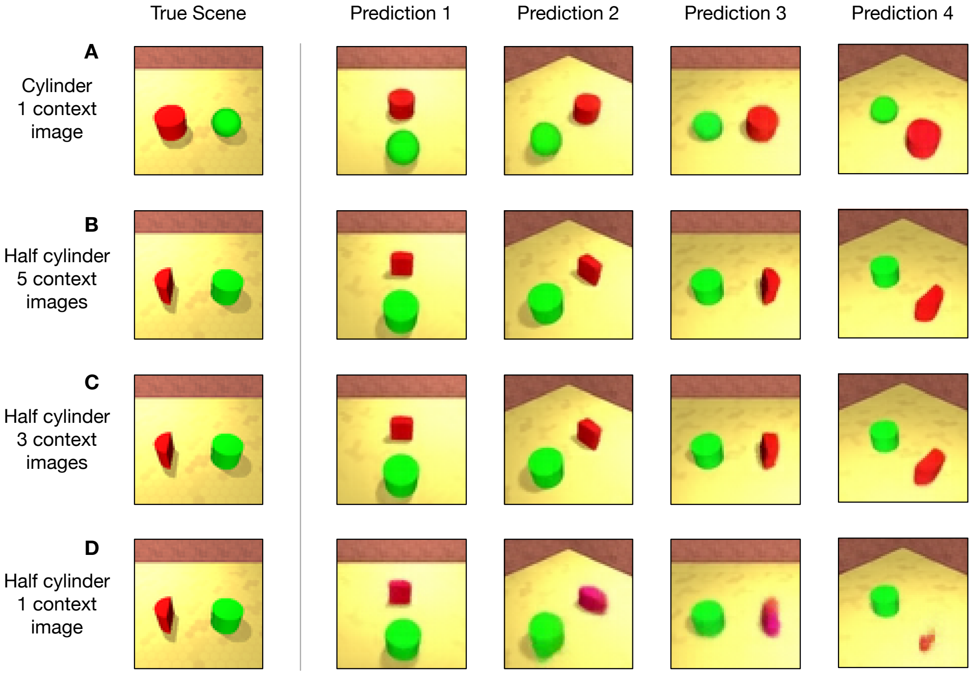

True Scene
Prediction 1
Prediction 2
Prediction 3
Prediction 4
A
Half Cylinder
5 context images. Successful generalisation to novel object.
B
Half Red Floor
5 context images. Generalisation to novel object. Inconsistent boundaries.
C
Short Wall
5 context images. Generalisation to novel object. Inconsistent shadows.
D
Tall Wall
5 context images. Failure to generalise to novel object.

Figure S13: **Out-of-distribution generalisation**. We train a GQN on objects of varying sizes, colours and shapes as before; here, however, we test its performance on a variety of strongly out-of-distribution scenes. **(A)** The GQN has never seen a half-cylinder during training, yet is capable of representing and rendering scenes containing this object successfully. **(B)** When presented with a half-red floor (never seen during training), the model is mostly capable of re-rendering the scene. Small inconsistencies can be seen at the boundary of the red part of the floor. **(C)** The model successfully represents and renders a short wall with a similar height to objects observed during training. Note mostly accurate rendering of the wall's shadows. **(D)** When the wall is substantially taller than any object observed during training, the model fails to represent and/or render the scene, possibly due to confusion about the wall's depth. Interestingly, samples often contain two offset short walls, which when viewed from the proper angle, may appear to combine as one taller wall.



For known objects, renders at new viewpoints are accurate, even for only 1 context observation.

Unknown object (half cylinder) but renders at new viewpoints are accurate.

Unknown object but renders at new viewpoints mostly accurate.

Unknown object and only 1 context image. Renders at viewpoints far from context image are inconsistent.

Figure S14: **Relationship between generalisation and number of context observations**. We observe that the GQN's ability to generalise to out-of-distribution scenes is affected by the number of context images it is allowed to use to compute the scene representation. **(A)** With only a single observation, the model successfully renders a familiar object from new viewpoints. **(B)** With 5 context observations, the model successfully renders an out-of-distribution object (half-cylinder) from new viewpoints. As the number of context images is reduced (C-D), the model's renders become progressively less consistent. The renders are most accurate from viewpoints that are closest to the context observations' viewpoints.

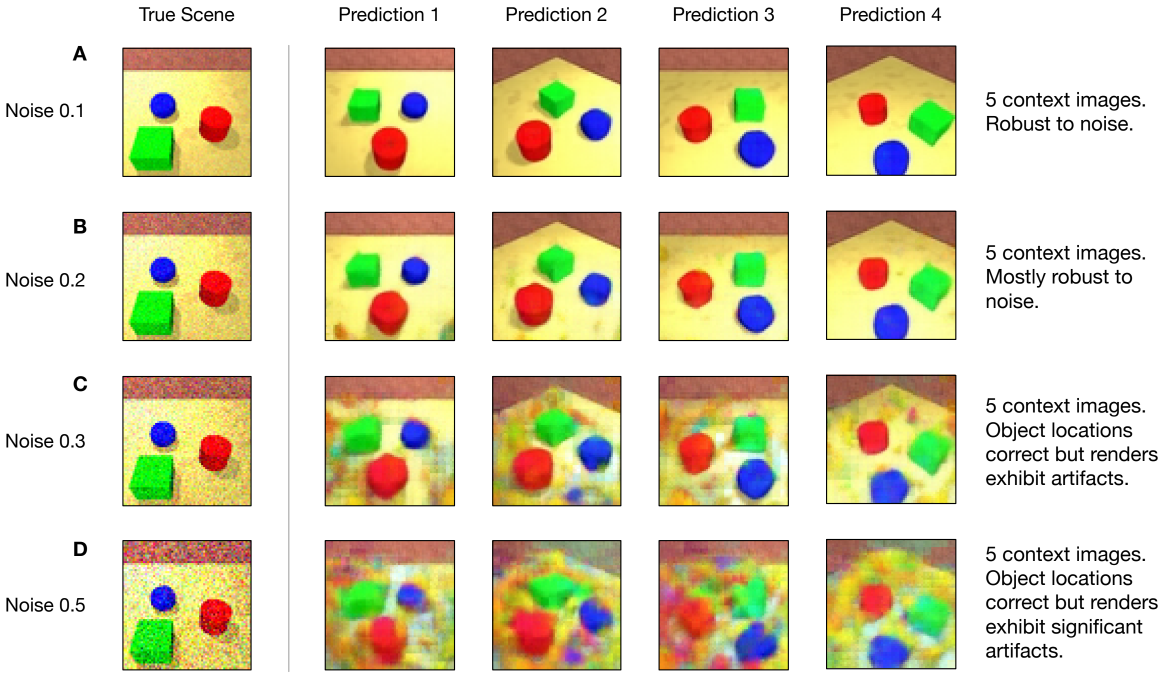

True Scene
Prediction 1
Prediction 2
Prediction 3
Prediction 4
A
Noise 0.1
5 context images. Robust to noise.
B
Noise 0.2
5 context images. Mostly robust to noise.
C
Noise 0.3
5 context images. Object locations correct but renders exhibit artifacts.
D
Noise 0.5
5 context images. Object locations correct but renders exhibit significant artifacts.

Figure S15: **Noise sensitivity**. We train a GQN on noiseless images as before, but test its performance when conditioned on context observation with increasing noise. Gaussian observation noise with zero mean and standard deviations of 0.1, 0.2, 0.3, and 0.5 (A-D, respectively). The model's renders become progressively less consistent as the standard deviation of the noise increases.

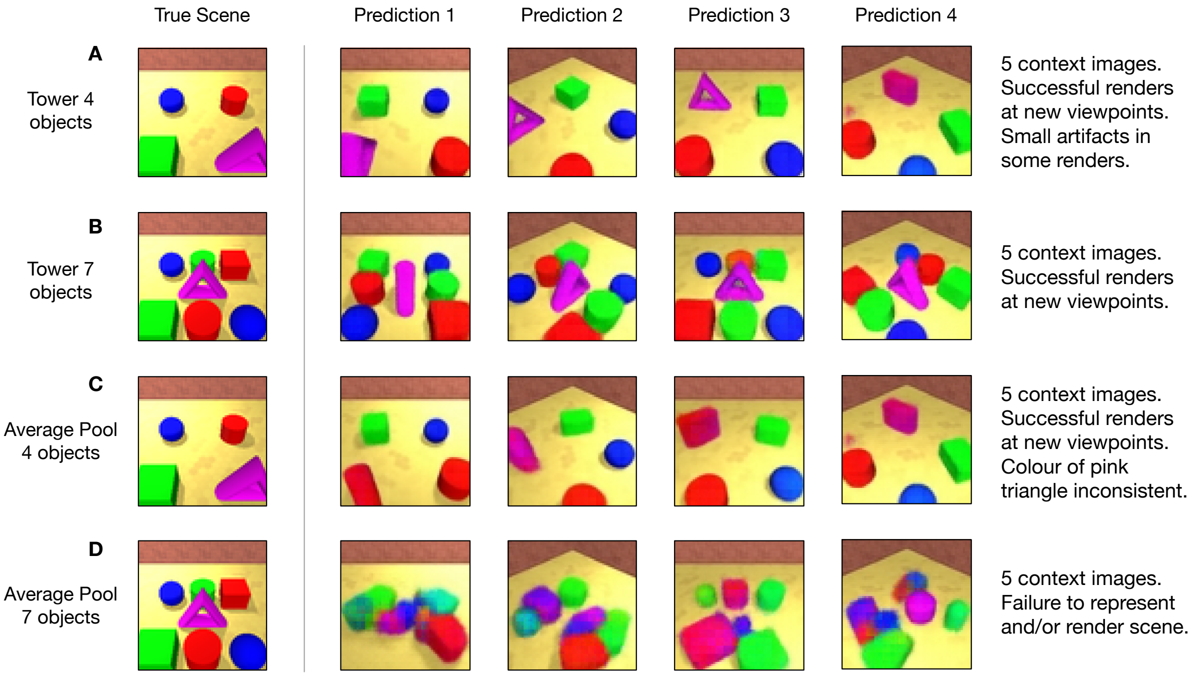

True Scene
Prediction 1
Prediction 2
Prediction 3
Prediction 4
A
Tower 4 objects
5 context images. Successful renders at new viewpoints. Small artifacts in some renders.
B
Tower 7 objects
5 context images. Successful renders at new viewpoints.
C
Average Pool 4 objects
5 context images. Successful renders at new viewpoints. Colour of pink triangle inconsistent.
D
Average Pool 7 objects
5 context images. Failure to represent and/or render scene.

Figure S16: **Generalisation to scenes with more objects than trained.** We train GQNs on up to 3 objects as before; here, however, we test their performance on a number of strongly out-of-distribution scenes with 4 or 7 objects each. The tower architecture (see Fig. S1) is capable of generalising to 4 **(A)** and 7 **(B)** objects. The average pool architecture is mostly accurate on **(C)** 4 objects, however performance degrades with **(D)** 7 objects. The tower architecture's superior performance is due to the spatial nature of its scene representation. By contrast, the average pool architecture's non-spatial representation appears to bind object properties in a manner which is dependent on the number of objects in the scene, resulting in poor extrapolation to scenes with more objects than trained.

Algorithm S1: GQN training loop.

Data: Choose dataset D from Room, Jaco, Labyrinth or Shepard-Metzler

Input: Initial parameters θ and ϕ. Optimizer parameters μ_i, μ_f, n, S_{max}, σ_i, σ_f, β_1 and β_2.

Output: Learned parameters θ and ϕ

```
def SampleBatch(B, M, K):
    /* Sample number of views */
    $M \sim \text{Uniform}(0, K)$
    /* Initialize data batch */
    $D = \{\}$
    for $b \leftarrow 0$ in $(B - 1)$:
        /* Sample scene index */
        $i \sim \text{Uniform}(0, N - 1)$
        for $k \leftarrow 0$ in $(M - 1)$:
            /* Sample view */
            $(\mathbf{x}_i^k, \mathbf{v}_i^k) \sim \text{scene } i$
            $D \leftarrow D + \{(\mathbf{x}_i^k, \mathbf{v}_i^k)\}$
        /* Sample query view */
        $(\mathbf{x}_i^q, \mathbf{v}_i^q) \sim \text{scene } i$
        $D \leftarrow D + \{(\mathbf{x}_i^q, \mathbf{v}_i^q)\}$
/* Training Iterations */
for $t \leftarrow 0$ in $(S_{max} - 1)$:
    $D \leftarrow$ SampleBatch($B$, $M$, $K$)
    ELBO $\leftarrow$ EstimateELBO($D$, $\sigma_t$) (Algorithm S2)
    /* Compute empirical ELBO gradients */
    $\nabla_\theta$ELBO, $\nabla_\phi$ELBO $\leftarrow$ Backprop(ELBO).
    /* Update parameters */
    $\theta, \phi \leftarrow$ Optimizer($\nabla_\theta$ELBO, $\nabla_\phi$ELBO, $\mu_t$)
    /* Update optimizer state */
    $\mu_t \leftarrow \max\left(\mu_f + (\mu_i - \mu_f)\left(1 - \frac{t}{n}\right), \mu_f\right)$
    /* Pixel-variance annealing */
    $\sigma_t \leftarrow \max\left(\sigma_f + (\sigma_i - \sigma_f)\left(1 - \frac{t}{n}\right), \sigma_f\right)$
```

Algorithm S2: Generating a sample from the approximate variational GQN posterior and estimating the ELBO.

```
Input: Observed views $\{(\mathbf{x}^k, \mathbf{v}^k)\}$, query camera: $\mathbf{v}^q$, target image: $\mathbf{x}^q$, pixel-variance: $\sigma_t$
Output: Sample from the posterior $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r})$, empirical estimate of the ELBO

def EstimateELBO($\{(\mathbf{x}^k, \mathbf{v}^k)\}, (\mathbf{v}^q, \mathbf{x}^q), \sigma_t$):
    Output: Empirical estimate of the ELBO

    /* Scene encoder */
    $\mathbf{r} \leftarrow 0$
    for $k \leftarrow 0$ in $(M-1)$:
        $\hat{\mathbf{v}}^k \leftarrow (\mathbf{w}^k, \cos(\mathbf{y}^k), \sin(\mathbf{y}^k), \cos(\mathbf{p}^k), \sin(\mathbf{p}^k))$
        $\mathbf{r}^k \leftarrow \psi\left(\mathbf{x}^k, \hat{\mathbf{v}}^k\right)$
        $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{r}^k$
    /* Generator initial state */
    $(\mathbf{c}_0^g, \mathbf{h}_0^g, \mathbf{u}_0) \leftarrow (\mathbf{0}, \mathbf{0}, \mathbf{0})$
    /* Inference initial state */
    $(\mathbf{c}_0^e, \mathbf{h}_0^e) \leftarrow (\mathbf{0}, \mathbf{0})$
    ELBO $\leftarrow 0$
    for $l \leftarrow 0$ in $(L-1)$:
        /* Prior factor */
        $\pi_{\theta_l}(\cdot|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}) \leftarrow \mathcal{N}\left(\cdot\middle|\eta_\theta^\pi(\mathbf{h}_l^g)\right)$
        /* Inference state update */
        $\left(\mathbf{c}_{l+1}^e, \mathbf{h}_{l+1}^e\right) \leftarrow C_\phi^e(\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{c}_l^e, \mathbf{h}_l^e, \mathbf{h}_l^g, \mathbf{u}_l)$
        /* Posterior factor */
        $q_{\phi_l}(\cdot|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}) \leftarrow \mathcal{N}\left(\cdot\middle|\eta_\theta^e(\mathbf{h}_l^e)\right)$
        /* Posterior sample */
        $\mathbf{z}_l \sim q_{\phi_l}(\cdot|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l})$
        /* Generator state update */
        $\left(\mathbf{c}_{l+1}^g, \mathbf{h}_{l+1}^g, \mathbf{u}_{l+1}\right) \leftarrow C_\theta^g(\mathbf{v}^q, \mathbf{r}, \mathbf{c}_l^g, \mathbf{h}_l^g, \mathbf{u}_l)$
        /* ELBO KL contribution update */
        ELBO $\leftarrow$ ELBO $-$ KL $[q_{\phi_l}(\cdot|\mathbf{x}^q, \mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}) || \pi_{\theta_l}(\cdot|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l})]$

    /* ELBO likelihood contribution update */
    ELBO $\leftarrow$ ELBO $+ \log \mathcal{N}\left(\mathbf{x}^q\middle|\mu = \eta_\theta^g(\mathbf{u}_L), \sigma = \sigma_t\right)$
```

Algorithm S3: Generating a prediction from GQN.

```
def Generate($\{(\mathbf{x}^k, \mathbf{v}^k)\}, \mathbf{v}^q$):
    Output: Generated image sample $\hat{\mathbf{x}}^q$
    /* Scene encoder */
    $\mathbf{r} \leftarrow 0$
    for $k \leftarrow 0$ in $(M-1)$:
        $\hat{\mathbf{v}}^k \leftarrow (\mathbf{w}^k, \cos(\mathbf{y}^k), \sin(\mathbf{y}^k), \cos(\mathbf{p}^k), \sin(\mathbf{p}^k))$
        $\mathbf{r}^k \leftarrow \psi\left(\mathbf{x}^k, \hat{\mathbf{v}}^k\right)$
        $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{r}^k$
    /* Initial state */
    $(\mathbf{c}_0^g, \mathbf{h}_0^g, \mathbf{u}_0) \leftarrow (\mathbf{0}, \mathbf{0}, \mathbf{0})$
    for $l \leftarrow 0$ in $(L-1)$:
        /* Prior factor */
        $\pi_{\theta_l}(\cdot|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l}) \leftarrow \mathcal{N}\left(\cdot\middle|\eta_\theta^\pi(\mathbf{h}_l^g)\right)$
        /* Prior sample */
        $\mathbf{z}_l \sim \pi_{\theta_l}(\cdot|\mathbf{v}^q, \mathbf{r}, \mathbf{z}_{<l})$
        /* State update */
        $\left(\mathbf{c}_{l+1}^g, \mathbf{h}_{l+1}^g, \mathbf{u}_{l+1}\right) \leftarrow C_\theta^g(\mathbf{v}^q, \mathbf{r}, \mathbf{c}_l^g, \mathbf{h}_l^g, \mathbf{u}_l, \mathbf{z}_l)$
    /* Image sample */
    $\hat{\mathbf{x}}^q \sim \mathcal{N}\left(\mathbf{x}^q\middle|\mu = \eta_\theta^g(\mathbf{u}_L), \sigma = \sigma_t\right)$
```

Name	**Description**	**Values**
μ_s	Learning rate at training step s with annealing $\mu_s = \max\left(\mu_f + (\mu_i - \mu_f)\left(1 - \frac{s}{n}\right), \mu_f\right)$	$\mu_i = 5 \times 10^{-4}$ $\mu_f = 5 \times 10^{-5}$ $n = 1.6 \times 10^6$
γ_s	Learning rate as used by the Adam algorithm $\gamma_s = \mu_s \frac{\sqrt{1-\beta_2^s}}{1-\beta_1^s}$	$\beta_1 = 0.9$ $\beta_2 = 0.999$
ϵ	Adam regularisation parameter	$\epsilon = 10^{-8}$
σ_s	Pixel standard-deviation with annealing $\sigma_s = \max\left(\sigma_f + (\sigma_i - \sigma_f)\left(1 - \frac{s}{n}\right), \sigma_f\right)$	$\sigma_i = 2.0$ $\sigma_f = 0.7$ $n = 2 \times 10^5$
L	Number of generative layers	12
B	Number of scenes over which each weight update is computed	36
S_{max}	Maximum number of training steps	2×10^6

Table S1: **List of hyper-parameters.** The values of all hyper-parameters were selected by performing informal search. We did not perform a systematic grid search owing to the high computational cost.

References

45. T. T. S. Jaakkola, M. M. I. Jordan, *Statistics and computing* **10**, 25 (1999), *Bayesian parameter estimation via variational methods*, vol. 10 (Springer, 1999).

46. D. P. Kingma, J. L. Ba, *ICLR* (2015), pp. 1–15, *Adam: a method for stochastic optimization* (2015).

47. J. Schmidhuber, *Trans. autonomous mental dev.* **2**, 230 (2010), *Formal theory of creativity, fun, and intrinsic motivation*, vol. 2 (IEEE, 2010).

48. D. J. C. MacKay, *Neural comput.* **4**, 590 (1992), *Information-based objective functions for active data selection*, vol. 4 (MIT Press, 1992).

49. E. Todorov, T. Erez, Y. Tassa, *IROS* (2012), pp. 5026–5033, *MuJoCo: a physics engine for model-based control* (2012).

50. R. N. Shepard, J. Metzler, *Science* **171**, 701 (1971), *Mental rotation of three-dimensional objects*, vol. 171 (American Association for the Advancement of Science, 1971).

51. C. Beattie, *et al.*, *arXiv:1612.03801* (2016), *DeepMind Lab* (2016).

52. V. Mnih, *et al.*, *ICML* (2016), pp. 1928–1937, *Asynchronous methods for deep reinforcement learning* (2016).