Attend, Infer, Repeat: Fast Scene Understanding with Deep Generative Models

S. M. Ali Eslami aeslami@google.com
Nicolas Heess heess@google.com
Theophane Weber theophane@google.com
Yuval Tassa tassa@google.com
David Szepesvari dsz@google.com
Koray Kavukcuoglu korayk@google.com
Geoffrey E. Hinton geoffhinton@google.com

***Abstract*—We present a framework for efficient inference in structured image models that explicitly reason about objects. We achieve this by performing probabilistic inference using a recurrent neural network that attends to scene elements and processes them one at a time. Crucially, the model itself learns to choose the appropriate number of inference steps, corresponding to the number of entities in the scene. We use this scheme to learn to perform inference in partially specified 2D models (variable-sized variational auto-encoders) and fully specified 3D models (probabilistic renderers). We show that such models learn to identify multiple objects – counting, locating and classifying the elements of a scene – without any supervision, e.g., decomposing images of 3D scenes with various numbers of objects in a single forward pass of a neural network. We further show that the networks produce accurate inferences when compared to supervised counterparts, and that their structure leads to improved generalization.**

The human percept of a visual scene is highly structured. Natural scenes decompose into *objects* that are arranged in space, have visual and physical properties, and are in functional relationships with each other. Artificial systems that interpret images in this way are desirable, as accurate detection of objects and inference of their attributes is thought to be fundamental for many problems of interest. Consider a robot whose task is to clear a table after dinner. To plan a course of action it will need to understand the scene, i.e. to determine which objects are present, what class each object belongs to, and where each one is located in the scene. Furthermore, for that scene representation to be readily usable by the robot, it needs to be disentangled across objects and semantic quantities (e.g., aspect and position).

The notion of using structured models for image understanding has a long history (e.g., 'vision as inverse graphics' [7]), however in practice it has been difficult to define models that are: (a) expressive enough to capture the complexity of natural scenes, and (b) amenable to tractable inference. Meanwhile, advances in deep learning have shown how neural networks can be used to make sophisticated predictions from images using little interpretable structure (e.g., [18]). Here we explore the intersection of structured probabilistic models and deep networks. Prior work on deep generative methods (e.g., VAEs [17]) have been mostly unstructured, therefore despite producing impressive samples and likelihood scores their representations have lacked interpretable meaning. On the other hand, structured generative methods have largely been incompatible with deep learning, and therefore inference has been hard and slow (e.g., via MCMC).

Our proposed framework achieves scene interpretation via learned, amortized inference, and it imposes structure on its representation through appropriate partly- or fully-specified generative models, rather than supervision from labels. The proposed framework crucially allows for reasoning about the complexity of a given scene (the dimensionality of its latent space). We demonstrate that via an Occam's razor type effect, this makes it possible to discover the underlying causes of a dataset of images in an unsupervised manner.

The main contributions of the paper are as follows. First, in Sec. I we formalize a scheme for efficient variational inference in latent spaces of variable dimensionality. The key idea is to treat inference as an *iterative* process, implemented as a recurrent neural network that *attends* to one object at a time, and learns to use an *appropriate number* of inference steps for each image. This approach allows for visual understanding in a way that is scalable with regards to scene complexity, due to its iterative nature, and scalable with regards to model complexity, due to the recurrent implementation of the inference network. The iterative formulation naturally captures the dependencies between latent variables in the posterior, for instance accounting for the fact that parts of the scene have already been explained. This is critical for accurate inference, and hence also for model learning. We call the proposed framework *Attend-Infer-Repeat* (AIR). End-to-end learning is enabled by recent advances in amortized variational inference, e.g., combining gradient based optimization for continuous latent variables with black-box optimization for discrete ones. Second, in Sec. II we show that AIR allows for learning of generative models that decompose multi-object scenes into their underlying causes, e.g., the constituent objects, in an unsupervised manner. We demonstrate these capabilities on image composed of multiple MNIST digits. In Sec. III we demonstrate how our inference framework can be used to perform fast inference for a 3D rendering engine, recovering the counts, identities and poses of objects in scenes containing complex meshes with significant occlusion in a single forward

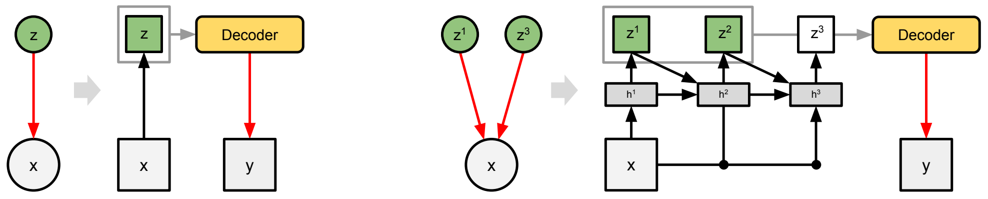


Fig. 1. *Left:* A single random variable z produces the observation x (the image). The relationship between z and x is specified by a model. Inference is the task of computing likely values of z given x. Using an auto-encoding architecture, the model (red arrow) and its inference network (black arrow) can be trained end-to-end via gradient descent. *Right:* For most images of interest, multiple latent variables (e.g., multiple objects) give rise to the image. We propose an iterative, variable-length inference network (black arrows) that attends to one object at a time, and train it jointly with its model. The result is fast, feed-forward, interpretable scene understanding trained without supervision.

pass of a neural network, providing a fast and scalable approach to 'vision as inverse graphics'.

I. Approach

In this paper we take a Bayesian perspective of scene interpretation, namely that of treating this task as inference in a generative model. Thus given an image $\mathbf{x}$ and a model $p^x_\theta(\mathbf{x}|\mathbf{z})p^z_\theta(\mathbf{z})$ parameterized by θ we wish to recover the underlying scene description $\mathbf{z}$ by computing the posterior $p(\mathbf{z}|\mathbf{x}) = p^x_\theta(\mathbf{x}|\mathbf{z})p^z_\theta(\mathbf{z})/p(\mathbf{x})$. In this view, the prior $p^z_\theta(\mathbf{z})$ captures our assumptions about the underlying scene, and the likelihood $p^x_\theta(\mathbf{x}|\mathbf{z})$ is our model of how a scene description is rendered to form an image. Both can take various forms depending on the problem at hand and we will describe particular instances in Sec. II. Together, they define the language that we use to describe a scene.

Many real-world scenes naturally decompose into objects. We therefore make the modeling assumption that the scene description is structured into groups of variables $\mathbf{z}^i$, where each group describes the attributes of one of the objects in the scene, e.g., its type, appearance, and pose. Since the number of objects will vary from scene to scene, we assume models of the following form:

$$p_\theta(\mathbf{x}) = \sum_{n=1}^{N} p_N(n) \int p^z_\theta(\mathbf{z}|n)p^x_\theta(\mathbf{x}|\mathbf{z})\mathrm{d}\mathbf{z}. \quad (1)$$

This can be interpreted as follows. We first sample the number of objects n from a suitable prior (for instance a Binomial distribution) with maximum value N. The latent, variable length, scene descriptor $\mathbf{z} = (\mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^n)$ is then sampled from a scene model $\mathbf{z} \sim p^z_\theta(\cdot|n)$. Finally, we render the image according to $\mathbf{x} \sim p^x_\theta(\cdot|\mathbf{z})$. Since the indexing of objects is arbitrary, $p^z_\theta(\cdot)$ is exchangeable and $p^x_\theta(\mathbf{x}|\cdot)$ is permutation invariant, and therefore the posterior over $\mathbf{z}$ is exchangeable.

The prior and likelihood terms can take different forms. We consider two scenarios: For 2D scenes (Sec. II), each object is characterized in terms of a learned distributed continuous representation for its shape, and a continuous 3-dimensional variable for its pose (position and scale). For 3D scenes (Sec. III) objects are defined in terms of a categorical variable that characterizes their identity, e.g., sphere, cube or cylinder, as well as their positions and rotations. We refer to the two kinds of variables for each object i in both scenarios as $\mathbf{z}^i_{\text{what}}$ and $\mathbf{z}^i_{\text{where}}$ respectively, bearing in mind that their meaning (e.g., position and scale in pixel space vs. position and orientation in 3D space) and their data type (continuous vs. discrete) will vary. We further assume that $\mathbf{z}^i$ are independent under the prior, i.e., $p^z_\theta(\mathbf{z}|n) = \prod_{i=1}^n p^z_\theta(\mathbf{z}^i)$, but non-independent priors, such as a distribution over hierarchical scene graphs (e.g., [38]), can also be accommodated. Furthermore, while the number of objects is bounded as per Eq. 1, it is relatively straightforward to relax this assumption.

A. Inference

Despite their natural appeal, inference for most models in the form of Eq. 1 is intractable. We therefore employ an amortized variational approximation to the true posterior by learning a distribution $q_\phi(\mathbf{z}, n|\mathbf{x})$ parameterized by ϕ that minimizes the divergence $\mathrm{KL}\left[q_\phi(\mathbf{z}, n|\mathbf{x})||p^z_\theta(\mathbf{z}, n|\mathbf{x})\right]$. While amortized variational approximations have recently been used successfully in a variety of works [28, 16, 25] the specific form our model poses two additional difficulties. **Trans-dimensionality:** As a challenging departure from classical latent space models, the size of the the latent space n (i.e., the number of objects) is a random variable itself, which necessitates evaluating $p_N(n|\mathbf{x}) = \int p^z_\theta(\mathbf{z}, n|x)\mathrm{d}\mathbf{z}$, for all $n = 1...N$. **Symmetry:** There are strong symmetries that arise, for instance, from alternative assignments of objects appearing in an image $\mathbf{x}$ to latent variables $\mathbf{z}^i$.

We address these challenges by formulating inference as an iterative process implemented as a recurrent neural network, which infers the attributes of one object at a time. The network is run for N steps and in each step explains one object in the scene, conditioned on the image and on its knowledge of previously explained objects (see Fig. 1).

To simplify sequential reasoning about the number of objects, we parameterize n as a variable length latent vector $\mathbf{z}_{\text{pres}}$ using a unary code: for a given value n, $\mathbf{z}_{\text{pres}}$ is the vector formed of n ones followed by one zero (note that the two representations are equivalent). By the chain rule, the posterior takes the following form:

$$q_\phi(\mathbf{z}, \mathbf{z}_{\text{pres}}|\mathbf{x}) = \prod_{i=1}^{n} q_\phi(\mathbf{z}^i, z^i_{\text{pres}} = 1|\mathbf{x}, \mathbf{z}^{1:i-1}) \times q_\phi(z^{n+1}_{\text{pres}} = 0|\mathbf{z}^{1:n}, x). \quad (2)$$

q_ϕ is implemented as a neural network that, in each step, outputs the parameters of the sampling distributions over the latent variables, e.g., the mean and standard deviation of a Gaussian distribution for continuous variables. z_{pres} can be understood as an interruption variable: at each time step, if the network outputs $z_{\text{pres}} = 1$, it describes at least one more more object and goes on to the next time step, but if it outputs $z_{\text{pres}} = 0$, no more objects are described, and inference terminates for that particular datapoint.

Note that conditioning of $\mathbf{z}^i|\mathbf{x}, \mathbf{z}^{1:i-1}$ is critical to capture dependencies between the latent variables $\mathbf{z}^i$, e.g., to avoid explaining the same object twice. The specifics of the networks that achieve this depend on the particularities of the models and we will describe them in detail in Sec. II.

B. Learning

We can jointly optimize the parameters ϕ of the inference network and (if any) parameters θ of the model by maximizing the lower bound on the marginal likelihood of an image under the model:

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}, n)}{q_\phi(\mathbf{z}, n, |\mathbf{x})}\right] \quad (3)$$

with respect θ and ϕ. $\mathcal{L}$ is called the negative free energy. We provide an outline of how to construct an unbiased estimator of the gradient of Eq. 3 below, for more details see [31].

1) Parameters of the model θ: Computing a Monte Carlo estimate of $\frac{\partial}{\partial\theta}\mathcal{L}$ is relatively straightforward: given a sample from the approximate posterior $(\mathbf{z}, \mathbf{z}_{\text{pres}}) \sim q_\phi(\cdot|\mathbf{x})$ (i.e., when the latent variables have been 'filled in') we can readily compute $\frac{\partial}{\partial\theta} \log p_\theta(\mathbf{x}, \mathbf{z}, n)$ provided p is differentiable in θ. This is effectively a partial M-step in a generalized EM scheme.

2) Parameters of the inference network ϕ: Computing a Monte Carlo estimate of $\frac{\partial}{\partial\phi}\mathcal{L}$ is more involved. As discussed above, the RNN that implements q_ϕ produces the parameters of the sampling distributions for the scene variables $\mathbf{z}$ and presence variables $\mathbf{z}_{\text{pres}}$. For a time step i, denote with ω^i all the parameters of the sampling distributions of variables in $(z^i_{\text{pres}}, \mathbf{z}^i)$. We parameterize the dependence of this distribution on $\mathbf{z}^{1:i-1}$ and $\mathbf{x}$ using a recurrent function $R_\phi(\cdot)$ implemented as a neural network such that $(\omega^i, \mathbf{h}^i) = R_\phi(\mathbf{x}, \mathbf{h}^{i-1})$ with hidden variables $\mathbf{h}$. The full gradient is obtained via chain rule:

$$\frac{\partial\mathcal{L}}{\partial\phi} = \sum_i \frac{\partial\mathcal{L}}{\partial\omega^i}\frac{\partial\omega^i}{\partial\phi}. \quad (4)$$

The gradient $\frac{\partial\omega^i}{\partial\phi}$ is classically computed through the back-propagation rule applied to the recurrent network R. Below we explain how to compute $\partial\mathcal{L}/\partial\omega^i$. We first rewrite our cost function as follows:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi}\left[\ell(\theta, \phi, \mathbf{z}, n)\right], \quad (5)$$

where $\ell(\theta, \phi, \mathbf{z}, n)$ is defined as $\log \frac{p_\theta(\mathbf{x}, \mathbf{z}, n)}{q_\phi(\mathbf{z}, n, |\mathbf{x})}$.Let z^i be an arbitrary element of the vector $(\mathbf{z}^i, z^i_{\text{pres}})$ of type {what, where, pres}. How to proceed depends on whether z^i is continuous or discrete.

a) Continuous variables: Suppose z^i is a continuous variable. We use the path-wise estimator (also known as 're-parameterization trick', e.g., [17, 31]), which allows us to 'back-propagate' through the random variable z^i. For many continuous variables (in fact, without loss of generality), z^i can be sampled as $h(\xi, \omega^i)$, where h is a deterministic transformation function, and ξ a random variable from a fixed noise distribution $p(\xi)$. We then obtain a gradient estimate:

$$\frac{\partial\mathcal{L}}{\partial\omega^i} \approx \frac{\partial\ell(\theta, \phi, \mathbf{z}, n)}{\partial z^i}\frac{\partial h}{\partial\omega^i}. \quad (6)$$

b) Discrete parameters: For discrete scene variables (e.g. z^i_{pres}) we cannot compute the gradient $\partial\mathcal{L}/\partial\omega^i_j$ by back-propagation. Instead we use the likelihood ratio estimator [25, 31]. Given a posterior sample $(\mathbf{z}, n) \sim q_\phi(\cdot|\mathbf{x})$ we can obtain a Monte Carlo estimate of the gradient as follows:

$$\frac{\partial\mathcal{L}}{\partial\omega^i} \approx \frac{\partial \log q(z^i|\omega^i)}{\partial\omega^i}\,\ell(\theta, \phi, \mathbf{z}, n). \quad (7)$$

In the raw form presented here this gradient estimate is likely to have high variance. We reduce its variance using appropriately structured baselines [25] that are functions of the image and the latent variables produced so far.

II. Experiments: Multi-MNIST

We first apply AIR to a dataset of multiple MNIST digits, and show that it can reliably learn to detect and generate the constituent digits from scratch (Sec. II). The structure of the AIR model and networks used in the 2D experiments are best described visually, see Fig. 2.

For the dataset of MNIST digits, we also investigate the behavior of a variant, difference-AIR (DAIR), which employs a slightly different recurrent architecture for the inference network. As opposed to AIR which computes $\mathbf{z}^i$ via $\mathbf{h}^i$ and $\mathbf{x}$, DAIR reconstructs at every time step i a partial reconstruction $\mathbf{x}^i$ of the data $\mathbf{x}$. The partial reconstruction is set as the mean of the distribution $p^x_\theta(\mathbf{x}|\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^{i-1})$. We then create an error canvas $\Delta\mathbf{x}^i = \mathbf{x}^i - \mathbf{x}$. The DAIR inference equation R_ϕ is then specified as $(\omega^i, \mathbf{h}^i) = R_\phi(\Delta\mathbf{x}^i, \mathbf{h}^{i-1})$.

We created a 50×50 dataset of multi-MNIST digits. Each image contains zero, one or two non-overlapping random MNIST digits with equal probability (see Fig. 3a). The desired goal is to train a network that produces sensible explanations for each of the images. We train AIR with $N = 3$ on 60,000 such images from scratch, i.e., without a curriculum or any form of supervision by maximizing $\mathcal{L}$ with respect to the parameters of the inference network and the generative model. Upon completion of training we inspect the model's

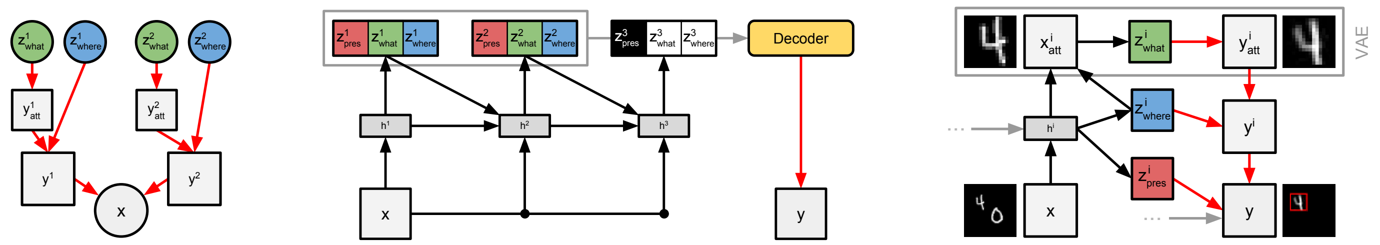


Fig. 2. **AIR in practice:** *Left:* The generative model draws $n \sim \text{Geom}(\rho)$ digits $\{\mathbf{y}^i_{\text{att}}\}$ of size 28×28 (two shown), scales and shifts them according to $\mathbf{z}^i_{\text{where}} \sim \mathcal{N}(0, \Sigma)$ using spatial transformers, and sums the results $\{y^i\}$ to form a 50×50 image. Each digit is obtained by first sampling a latent code $\mathbf{z}^i_{\text{what}}$ from the prior $\mathbf{z}^i_{\text{what}} \sim \mathcal{N}(0, 1)$ and propagating it through the decoder network of a variational autoencoder. The learnable parameters θ of the generative model are the parameters of this decoder network. *Middle:* AIR inference for this model. The inference network produces three sets of variables for each entity at every time-step: a 1-dimensional Bernoulli variable indicating the entity's presence, a C-dimensional distributed vector describing its class or appearance ($\mathbf{z}^i_{\text{what}}$), and a 3-dimensional vector specifying the affine parameters of its position and scale ($\mathbf{z}^i_{\text{where}}$). The recurrent network is chosen to be an LSTM. *Right:* Interaction between the inference and generation networks at every time-step. The inferred pose is used to attend to a part of the image (using a spatial transformer) to produce $\mathbf{x}^i_{\text{att}}$, which is processed to produce the inferred code $\mathbf{z}^i_{\text{code}}$ and the reconstruction of the contents of the attention window $\mathbf{y}^i_{\text{att}}$. The same pose information is used by the generative model to transform $\mathbf{y}^i_{\text{att}}$ to obtain $\mathbf{y}^i$. This contribution is only added to the canvas $\mathbf{y}$ if z^i_{pres} was inferred to be true.

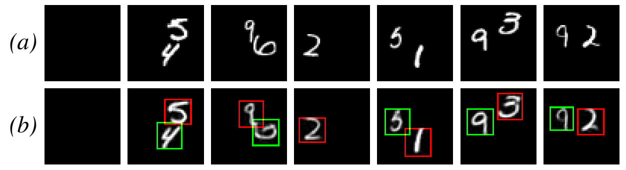


Fig. 3. **Multi-MNIST overview:** (a) Images from the dataset. (b) AIR reconstructions, along with a visualization of the model's attention windows. The 1st, 2nd and 3rd time-steps are displayed using red, green and blue borders respectively. No blue borders are visible as AIR never uses more than two steps on this data.

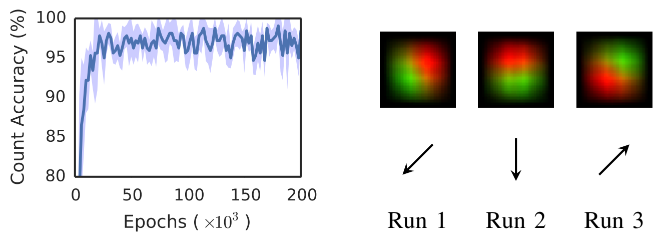


Fig. 4. **Multi-MNIST results:** *Left:* Count accuracy over time. The model detects the counts of digits accurately, despite having never been provided supervision. *Right:* The learned scanning policy for 3 different runs of training (only differing in the random seed). We visualize empirical heatmaps of the attention windows' positions (red, and green for the first and second time-steps respectively). As expected, the policy is random. This suggests that the policy is spatial, as opposed to identity- or size-based.

inferences (see Fig. 3b). We draw the reader's attention to the following observations. First, the model identifies the number of digits correctly, due to the opposing pressures of (a) wanting to explain the scene, and (b) the cost that arises from instantiating an object under the prior. This is indicated by the number of attention windows in each image; we also plot the accuracy of count inference over the course of training (Fig. 4, left). Second, it locates the digits accurately. Third, the recurrent network learns a suitable scanning policy to ensure that different time-steps account for different digits (Fig. 4, right). Note that we did not have to specify any such policy in advance, nor did we have to build in a constraint to prevent

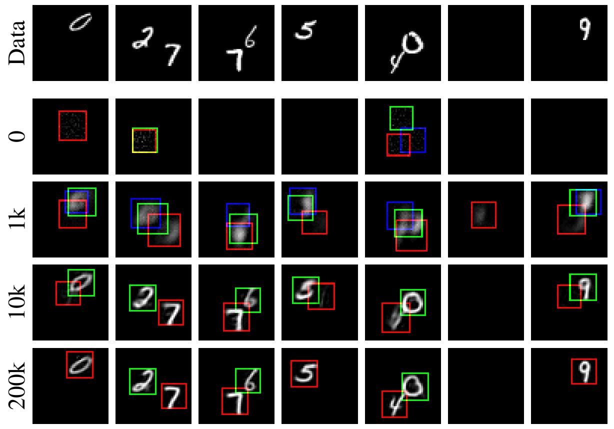
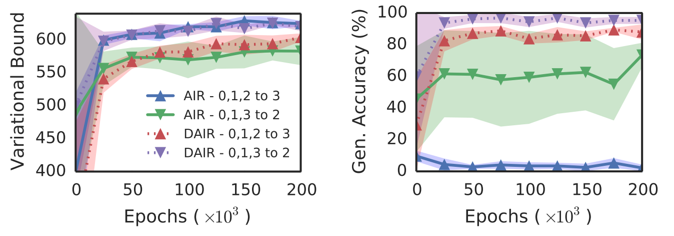


Fig. 5. **Multi-MNIST learning:** *Top:* Images from the dataset. *Bottom:* Reconstructions at different stages of training. A video of this sequence is provided in the supplementary material.

two time-steps from explaining the same part of the image. Finally, that the network learns to not use the second time-step when the image contains only a single digit, and to never use the third time-step (images contain a maximum of two digits). This allows for the inference network to stop upon encountering the first z^i_{pres} equaling 0, leading to potential savings in computation during inference.

It is informative to inspect how the model's inferences evolve over time. In Fig. 5 we show reconstructions of a fixed set of test images at various points during training. The model's reconstructions are at first very poor. It then gradually learns to reconstruct well, however it makes use of all available time-steps. It is only towards the end of training that it learns to use its time-steps more sparingly, leading it to perform correct inference of object counts.

Owing to the structure and nature of the networks used in AIR, inference under a learned model is almost instantaneous in contrast to classical inference techniques e.g., direct optimization or Markov chain Monte Carlo. To demonstrate this and to better understand the learned model, we imple-



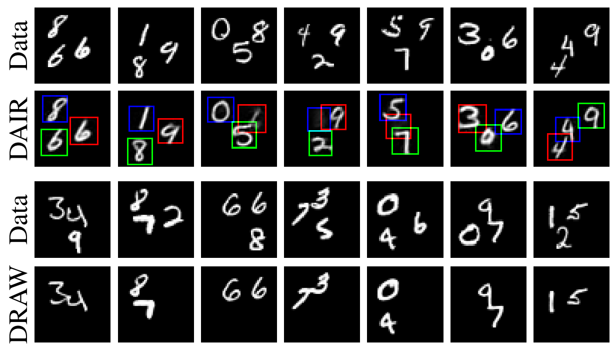


Fig. 6. **Strong generalization:** *Top left:* Variational lower bound over the course of learning. *Top right:* Generalizing / interpolating count accuracy. DAIR out-performs AIR at this task. *Bottom:* Reconstructions of images with 3 digits made by a DAIR model trained on 0, 1 or 2 digits, as well as a comparison with DRAW. AIR sometimes fails to reconstruct the scene despite detecting the presence of 3 digits, e.g., in images 4 and 5. DRAW reconstructions with logit-normal likelihood (found to produce best-looking samples). Interestingly, DRAW learns to ignore precisely one digit in the reconstruction.

ment a graphical user interface for real-time inference and reconstruction. A video showing its use can be found here: https://youtu.be/BQETuyqYBcI.

A. Strong Generalization

Since the model learns the concept of a digit independently of the positions or numbers of times it appears in each image, one would hope that it would be able to generalize, e.g., by demonstrating an understanding of scenes that have structural differences to training scenes. We probe this behavior with the following scenarios: (a) *Extrapolation:* training on images each containing 0, 1 or 2 digits and then testing on images containing 3 digits, and (b) *Interpolation:* training on images containing 0, 1 or 3 digits and testing on images containing 2 digits. The result of this experiment is shown in Fig. 6. An AIR model trained on up to 2 digits is effectively unable to infer the correct count when presented with an image of 3 digits. We believe this to be caused by the LSTM which learns during training never to expect more than 2 digits. AIR's generalization performance is improved somewhat when considering the interpolation task. DAIR by contrast generalizes well in both tasks (and finds interpolation to be slightly easier than extrapolation). A closely related baseline is the Deep Recurrent Attentive Writer (DRAW, [6]), which like AIR, generates data sequentially. However, DRAW has a fixed and large number of steps (40 in our experiments). As a consequence generative steps do not correspond to easily interpretable entities, complex scenes are drawn faster and simpler ones slower. We show DRAW's reconstructions in Fig. 6. Interestingly, DRAW learns to ignore precisely one digit in the image.

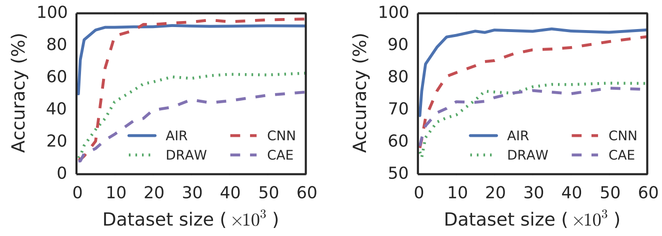


Fig. 7. **Representational power:** AIR achieves high accuracy using only a fraction of the labeled data. *Left:* summing two digits. *Right:* detecting if they appear in increasing order.

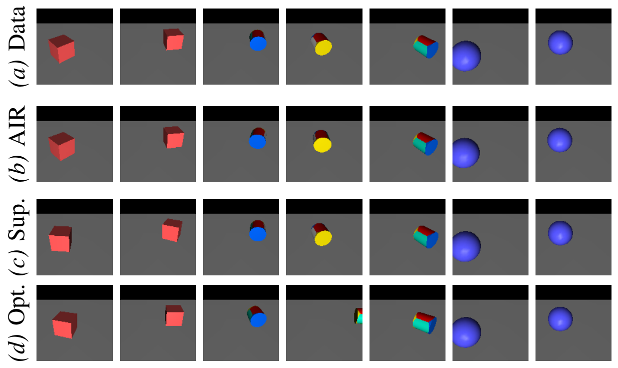


Fig. 8. **3D objects:** The task is to infer the identity and pose of a 3D object. (a) Images from the dataset. (b) Reconstructions produced by re-rendering the inference made by an AIR network trained on the data without supervision. (c) Reconstructions produced by an AIR network trained with ground-truth labels. Note poor performance on cubes due to their symmetry. (d) Reconstructions obtained by performing direct gradient descent on the scene representation to minimize reconstruction error. This approach is less stable and much more susceptible to local minima.

B. Representational Power

A second motivation for the use of structured generative models is that their inferences about the structure of a scene provides useful representations for downstream tasks. We examine this ability by first training an AIR model on 0, 1 or 2 digits and then produce inferences for a separate collection of images that contains precisely 2 digits. We split this data into training and test and consider two tasks: (a) predicting the sum of the two digits (as was done in 1), and (b) determining if the digits appear in an ascending order. We compare with a CNN trained from the raw pixels, as well as interpretations produced by a convolutional autoencoder (CAE) and DRAW (Fig. 7). We optimize each model's hyper-parameters (e.g. depth and size) for maximal performance. AIR achieves high accuracy even when data is scarce, indicating the power of its disentangled, structured representation. In contrast, DRAW and CAE representations lead to poor performance, hinting that in spite of having good reconstruction errors (and excellent data likelihood for DRAW), they lead to representations which are hard to interpret and re-use for auxiliary tasks.

III. 3D Scenes

The previous experiments demonstrate learning of inference *and* generative networks in models where we impose structure in the form of a variable-sized representation and spatial attention mechanisms. We now consider an additional way of imparting knowledge to the system: we specify the generative model via a 3D renderer, i.e., we completely specify how any scene representation is transformed to produce the pixels in an image. Therefore the task is to learn to infer the counts, identities and poses of several objects, given different images containing these objects and an implementation of a 3D renderer from which we can draw new samples. This formulation of computer vision is often called 'vision as inverse graphics' (see e.g [7, 22, 14]).

The primary challenge in this view of computer vision is that of inference. While it is relatively easy to specify high-quality generative models in the form of probabilistic renderers, performing posterior inference is either extremely computationally expensive or prone to getting stuck in local minima (e.g., via optimization or Markov chain Monte Carlo). Therefore it would be highly desirable amortize this cost over training in the form of an inference network. In addition, probabilistic renderers (and in particular 3D renderers) typically are not capable of providing gradients with respect to their inputs, and 3D scene representations often involve discrete variables, e.g., mesh identities. We address these challenges by using finite-differencing to obtain a gradient through the renderer, using the score function estimator to get gradients with respect to discrete variables, and using an AIR inference architecture to handle correlated posteriors and variable-length representations. The experiments in this section were performed using the rendering capabilities of the MuJoCo physics simulator [34]. Differentiation of MuJoCo's graphics engine was performed using forward finite-differencing (with a constant $\epsilon = 10^{-4}$) with respect to the scene configuration.

A. Simple objects

We demonstrate the capabilities of this approach by first considering scene consisting of only one of three objects: a red cube, a blue sphere, and a textured cylinder (see Fig. 8a). The objects are randomly moved in the plane of the table and rotated along the axis orthogonal to it (i.e. 3 degrees of freedom per object).

Since the scenes only consist of single objects, the task is only to infer the identity (cube, sphere, cylinder) and pose (position and rotation) of the object present in the image. We train a single-step ($N = 1$) AIR inference network for this task. The network is only provided with unlabeled images and is trained to maximize the likelihood of those images under the model specified by the renderer. The quality of the scene representations produced by the learned inference network can be visually inspected in Fig. 8b. The network accurately and reliably infers the identity and pose of the object present in the scene. In contrast, an identical network trained in a supervised fashion to predict the ground-truth identity and pose values of the training data (wherein we maximize the probability of the correct labels, in a similar style to [19]) has much more difficulty in accurately determining the cube's orientation (Fig. 8c). The supervised loss forces the network to predict the exact angle of rotation. However this is not identifiable from the image due to the rotational symmetries of some of the objects, which leads to conditional probabilities that are multi-modal and difficult to represent using standard network architectures. We also compare with direct optimization of the likelihood from scratch for every test image (Fig. 8d), and observe that this method is slower, less stable and more susceptible to local minima. So not only does amortization reduce the cost of inference, but it also overcomes the pitfalls of independent gradient optimization.

B. Table-top scenes

We then consider a more complex setup, where we infer the counts, identities and positions of a variable number of crockery items in a table-top scene (Figs. 9 and 10). This would for instance be of critical importance to a robot which is in the process of interacting with the objects and the table. The goal is to learn to achieve this task with as little supervision as possible, and indeed we observe that with AIR it is possible to do so with no supervision other than a specification of the renderer.

1) Experimental setup: We used scenes with a box object for the table, and nine objects (meshes) for the crockery items. The cup, pan, and plate were each replicated three times to allow for arbitrary three-objects scenes. Each object had three degrees of freedom (position in the table plane and rotation). Random scenes with up to $N = 3$ objects were created by randomly sampling position, rotation angle, object presence, and object type three times.

We experimented with two versions of the scene: one with a fixed camera, and one version where the camera could be moved in an orbit around the table (i.e. one degree of freedom). For those experiments, the camera position was chosen randomly and the image was rendered from the random camera position. Camera movement was restricted to $\pm$ 40 degrees from the central position. In this experiment the model has to learn to infer the camera position in addition to the objects on the table. The montage in Fig. 9 shows a ground truth scene (with camera) and the inferred identities and positions of the objects as well as the inferred position of the camera. We show several examples of random scenes with variable camera and the associated inferences in Fig. 10. For the most part the network infers all scene parameters reliably.

All scene images at 128 $\times$ 128 pixels. We down-sampled scene images to 32 $\times$ 32 pixels for input to the network.

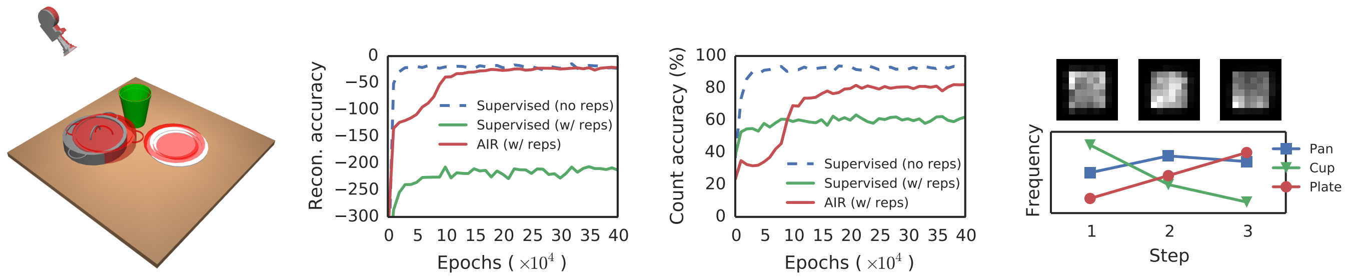


Fig. 9. **3D scenes details:** *Left:* Ground-truth object and camera positions with inferred positions overlayed in red (note that inferred cup is closely aligned with ground-truth, thus not clearly visible). We demonstrate fast inference of all relevant scene elements using the AIR framework. *Middle:* AIR achieves significantly lower reconstruction error than a naive supervised implementation, and achieves much higher count inference accuracy. *Right:* Heatmap of locations on the table in which objects are detected at each time-step (top). The learned policy appears to be more dependent on identity (bottom).

2) Model: We trained a network to perform inference in the following fixed generative model:

$$p(\mathbf{x}, z_{\text{pres}}^{1:N}, \mathbf{z}_{\text{where}}^{1:N}, \mathbf{z}_{\text{what}}^{1:N}) = \tag{8}$$
$$p(\mathbf{x}|z_{\text{pres}}^{1:N}, \mathbf{z}_{\text{where}}^{1:N}, \mathbf{z}_{\text{what}}^{1:N}) \prod_{i=1}^{N} p(z_{\text{pres}}^{i}) p(\mathbf{z}_{\text{what}}^{i}) p(\mathbf{z}_{\text{where}}^{i}),$$

where z_{pres}^i is the visibility indicator: $z_{\text{pres}}^i \sim \text{Bernoulli}(\alpha)$ for object i; $\mathbf{z}_{\text{where}} \in \mathbb{R}^3$ indicates position and rotation angle: $\mathbf{z}_{\text{where}}^i \sim \mathcal{N}(0, \Sigma_{\text{where}})$; and $\mathbf{z}_{\text{what}}^i$ is a three-valued discrete variable indicating the object type (mesh / geom type): $\mathbf{z}_{\text{what}}^i \sim \text{Discrete}(\beta)$.

The marginal distribution over scenes under this model is the same as the marginal distribution under a model of form described in Section I where $p(n) = \text{Binomial}(N, \alpha)$ and $n = \sum_{i=1}^{N} z_i$.

For the variable camera scenes the model included an additional random variable $z_{\text{cam}} \in \mathbb{R}$ where $z_{\text{cam}} \sim \mathcal{N}(0, \sigma_{\text{cam}}^2)$.

To evaluate the likelihood term $p(\mathbf{x}|\mathbf{z})$ we (1) render the scene description using the MuJoCo rendering engine to produce a high-resolution image $\mathbf{y}$; (2) blur the resulting image $\mathbf{y}$ as well as $\mathbf{x}$ using a fixed-with blur kernel; (3) compute $\mathcal{N}(\mathbf{x}|\mathbf{y}, \mathbf{I}\sigma_x^2)$. The AIR inference network for our experiments is a standard recurrent network (no LSTM) that is run for a fixed number of steps ($N = 1$ or $N = 3$). In each step the network computes:

$$(\omega_{\text{pres}}^i, \omega_{\text{what}}^i, \omega_{\text{where}}^i, \mathbf{h}^i) = R(\mathbf{x}, z_{\text{pres}}^{i-1}, \mathbf{z}_{\text{what}}^{i-1}, \mathbf{z}_{\text{where}}^{i-1}, \mathbf{h}^{i-1}),$$

where the ω^i represent the parameters of the sampling distributions for the random variables: Bernoulli for z_{pres}; Discrete for $\mathbf{z}_{\text{what}}$; and Gaussian for $\mathbf{z}_{\text{where}}$. For the experiments with random camera angle we use a separate network that computes $\omega_{\text{cam}} = F(\mathbf{x})$ and we provide the sampled camera angle as additional input to R at each time step.

3) Results: We show reconstructions of AIR's inferences on synthetic data, as well as real images of a table with varying numbers of plates, in Fig. 9 and Fig. 10. AIR's inferences of counts, identities and positions are accurate for the most part. For transfer to real scenes we perform random color and size pertubations to rendered objects during training, however we note that robust transfer remains a challenging problem in general. We provide a quantitative comparison of AIR's inference robustness and accuracy with that of a fully supervised network in Fig. 9. We consider two scenarios: one where each object type only appears exactly once, and one where objects can repeat in the scene. A naive supervised setup struggles greatly with object repetitions or when an arbitrary ordering of the objects is imposed by the labels, however training is more straightforward when there are no repetitions. AIR achieves competitive reconstruction and count accuracy despite the added difficulty of object repetitions.

IV. Related Work

Deep neural networks have had great success in learning to predict various quantities from images, e.g., object classes [18], camera positions [15] and actions [27]. These methods work best when large labeled datasets are available for training.

At the other end of the spectrum, e.g., in 'vision as inverse graphics', only a generative model is specified in advance and prediction is treated as an inference problem, which is then solved using MCMC or message passing at test-time. These models range from highly specified [24, 23], to partially specified [38, 29, 8, 3, 32, 33], to largely unspecified [9, 30, 4]. Inference is very challenging and almost always the bottleneck in model design.

Various researchers [10, 35, 19, 14, 36] exploit data-driven predictions to empower the 'vision as inverse graphics' paradigm. For instance, PICTURE [19] uses a deep network to distill the results of slow MCMC, speeding up predictions at test-time.

Variational auto-encoders [28, 16] and their discrete counterparts [25] made the important contribution of showing how the gradient computations for learning of amortized inference and generative models could be interleaved, allowing both to be learned simultaneously in an end-to-end fashion (see also [31]). Works like that of [11, 20] aim to learn disentangled representations in an auto-encoding framework using special network structures and / or careful training schemes.

It is also worth noting that attention mechanisms in neural networks have been studied in discriminative and generative settings, e.g. in [26, 1, 13, 6].

AIR draws upon, extends and links these ideas. Similar to our work is also [12], however they assume a fixed number

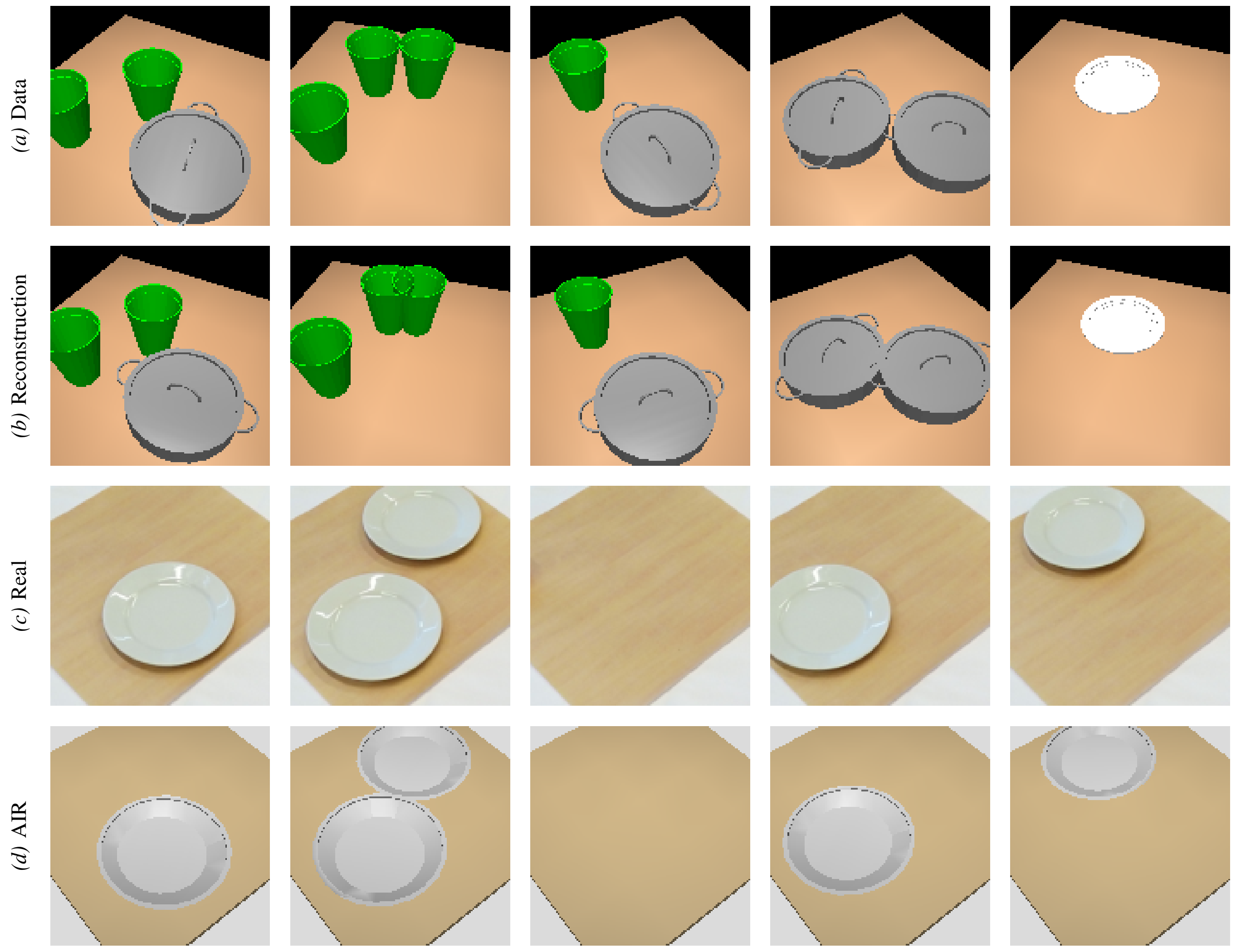


Fig. 10. **3D scenes with variable camera:** AIR results for inferring the camera angle of the scene, as well as the counts, identities and poses of multiple objects in a 3D table-top scene similar to the results presented in Section III in the main text but with the additional complication of an unknown camera angle. (a) Images from the dataset. (b) Reconstruction of the scene description inferred by our AIR network. Note that due to the down-sampling of the images that were used as input to the inference network and the blurring in the likelihood computation accurate estimation of the rotation angle is essentially impossible.

of objects. By its nature AIR is also related to the following problems: counting [21, 37], trans-dimensionality [5], sparsity [2] and gradient estimation through renderers [22]. It is the combination of these elements that unlocks the full capabilities of the proposed approach.

V. Discussion

We presented several principled models that not only learn to count, locate, classify and reconstruct the elements of a scene, but do so in a fraction of a second at test-time. The main ingredients are (a) building in meaning using appropriately structured models, (b) amortized inference that is attentive, iterative and variable-length, and (c) end-to-end learning. Learning is most successful when the variance of the gradients is low and the likelihood is well suited to the data. It will be of interest to examine the scaling of variance with the number of objects and more sophisticated likelihoods (e.g., occlusion). It is straightforward to extend the framework to semi- or fully-supervised settings. Furthermore, the framework admits a plug-and-play approach where existing state-of-the-art detectors, classifiers and renderers are used as sub-components of an AIR inference network. We plan to investigate these lines of research in future work.

References

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention. In *International Conference on Learning Representations*, 2015.

[2] Samy Bengio, Fernando Pereira, Yoram Singer, and Dennis Strelow. Group Sparse Coding. In *Advances in Neural Information Processing Systems 22*. 2009.

[3] S. M. Ali Eslami and Christopher K. I. Williams. A Generative Model for Parts-based Object Segmentation. In *Advances in Neural Information Processing Systems 25*, 2014.

[4] S. M. Ali Eslami, Nicolas Heess, and John Winn. The Shape Boltzmann Machine: a Strong Model of Object Shape. In *Computer Vision and Pattern Recognition*, 2012.

[5] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[6] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. DRAW: A Recurrent Neural Network For Image Generation. In *International Conference on Machine Learning*, 2015.

[7] Ulf Grenander. *Pattern Synthesis: Lectures in Pattern Theory*. Applied Mathematical Sciences. 1976.

[8] Nicolas Heess, Nicolas Le Roux, and John M. Winn. Weakly Supervised Learning of Foreground-Background Segmentation Using Masked RBMs. In *International Conference on Artificial Neural Networks*, 2011.

[9] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8), 2002.

[10] Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Randford M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214), 1995.

[11] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming Auto-encoders. In *International Conference on Artificial Neural Networks*, 2011.

[12] Jonathan Huang and Kevin Murphy. Efficient inference in occlusion-aware generative models of images. *CoRR*, abs/1511.06362, 2015.

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. 2015.

[14] Varun Jampani, Sebastian Nowozin, Matthew Loper, and Peter V. Gehler. The Informed Sampler: A Discriminative Approach to Bayesian Inference in Generative Computer Vision Models. In *Special Issue on Generative Models in Computer Vision and Medical Imaging*, volume 136, 2015.

[15] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *International Conference on Computer Vision*, 2015.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, 2012.

[19] Tejas D. Kulkarni, Pushmeet Kohli, Joshua B. Tenenbaum, and Vikash K. Mansinghka. Picture: A probabilistic programming language for scene perception. In *Computer Vision and Pattern Recognition*, 2015.

[20] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep Convolutional Inverse Graphics Network. In *Advances in Neural Information Processing Systems 28*. 2015.

[21] Victor Lempitsky and Andrew Zisserman. Learning To Count Objects in Images. In *Advances in Neural Information Processing Systems 23*. 2010.

[22] Matthew M. Loper and Michael J. Black. OpenDR: An Approximate Differentiable Renderer. In *European Conference on Computer Vision*, volume 8695, 2014.

[23] Vikash Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. Approximate Bayesian Image Interpretation using Generative Probabilistic Graphics Programs. In *Advances in Neural Information Processing Systems 26*. 2013.

[24] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *International Joint Conference on Artificial Intelligence*, pages 1352–1359, 2005.

[25] Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. In *International Conference on Machine Learning*, 2014.

[26] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems 27*, 2014.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

[28] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Tony Jebara and Eric P. Xing, editors, *International Conference on Machine Learning*, 2014.

[29] Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John M. Winn. Learning a generative model of images by factoring appearance and shape. In *International Conference on Artificial Neural Networks*, 2011.

[30] Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann Machines. In *International Conference on Artificial*

Intelligence and Statistics, volume 5, 2009.

[31] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs. In *Advances in Neural Information Processing Systems 28*. 2015.

[32] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Tensor Analyzers. In *International Conference on Machine Learning*, 2013.

[33] Yichuan Tang, Nitish Srivastava, and Ruslan Salakhutdinov. Learning Generative Models With Visual Attention. In *Advances in Neural Information Processing Systems 27*, 2014.

[34] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012. URL www.mujoco.org.

[35] Zhuowen Tu and Song-Chun Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *Transactions on Pattern Analysis and Machine Intelligence*, (5), 2002.

[36] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*. 2015.

[37] Jianming Zhang, Shuga Ma, Mehrnoosh Sameki, Stan Sclaroff, Margrit Betke, Zhe Lin, Xiaohui Shen, Brian Price, and Radomír Mĕch. Salient Object Subitizing. In *Conference on Computer Vision and Pattern Recognition*, 2015.

[38] Song-Chun Zhu and David Mumford. A Stochastic Grammar of Images. *Foundations and Trends in Computer Graphics and Vision*, 2(4), 2006.