Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks

Eric Crawford
Mila, McGill University
Montreal, QC

Joelle Pineau
Facebook AI Research,
Mila, McGill University
Montreal, QC

Abstract

There are many reasons to expect an ability to reason in terms of objects to be a crucial skill for any generally intelligent agent. Indeed, recent machine learning literature is replete with examples of the benefits of object-like representations: generalization, transfer to new tasks, and interpretability, among others. However, in order to reason in terms of objects, agents need a way of discovering and detecting objects in the visual world - a task which we call *unsupervised object detection*. This task has received significantly less attention in the literature than its supervised counterpart, especially in the case of large images containing many objects. In the current work, we develop a neural network architecture that effectively addresses this large-image, many-object setting. In particular, we combine ideas from Attend, Infer, Repeat (AIR), which performs unsupervised object detection but does not scale well, with recent developments in supervised object detection. We replace AIR's core recurrent network with a convolutional (and thus spatially invariant) network, and make use of an object-specification scheme that describes the location of objects with respect to local grid cells rather than the image as a whole. Through a series of experiments, we demonstrate a number of features of our architecture: that, unlike AIR, it is able to discover and detect objects in large, many-object scenes; that it has a significant ability to generalize to images that are larger and contain more objects than images encountered during training; and that it is able to discover and detect objects with enough accuracy to facilitate non-trivial downstream processing.

1 Introduction

The physical world can be naturally sub-divided into discrete objects. Consequently, in the pursuit of constructing ever more intelligent agents, devising methods for reasoning and learning about objects should be regarded as an important sub-goal. Indeed, recent work in machine learning is replete with examples of the benefits of object-like representations (Diuk, Cohen, and Littman 2008; Chang et al. 2016; Kansky et al. 2017; Santoro et al. 2017). For example, learning algorithms that operate on objects can yield significantly improved sample complexity and generalizability (Chang et al. 2016). Moreover, learning about objects can facilitate transfer learning (Kansky et al. 2017); since the world decomposes naturally into objects, a common source of novel tasks will involve familiar objects in new configurations. In such settings, we can expect improved transfer performance from algorithms that learn about objects, as object-level knowledge learned from source tasks is likely to be useful for target tasks. Finally, machine learning models that reason in terms of objects may be more interpretable, since they are forced to use a representation that is well understood by humans (Zambaldi et al. 2018).

For systems that take raw pixels as input, reasoning and learning in terms of objects requires an initial object detection step wherein the pixel-level information is transduced into high-level object representations. Past machine learning methods that reason in terms of objects have obtained objects either using ad hoc, task/environment specific object detection methods (Garnelo, Arulkumaran, and Shanahan 2016; Diuk, Cohen, and Littman 2008), or, in settings where it is possible, using object representations provided directly by the environment (Kansky et al. 2017). Alternatively, if one has access to a large training set of images annotated with ground-truth bounding boxes for the objects therein, one could train any of a large number of sophisticated supervised object detection architectures (Ren et al. 2015; Redmon et al. 2016). However, obtaining such an annotated dataset is expensive in terms of human labor. Thus, in the current work, we are interested in a general-purpose object detection method that can be applied in a wide range of settings without supervision; in particular, without bounding box annotations. We call this task, which involves both discovering which objects are common in a dataset and learning to detect those objects in images, *unsupervised object detection*.

A recent approach called Attend, Infer, Repeat (AIR) (Eslami et al. 2016) has shown promise at unsupervised object detection, but (as we show empirically) has difficulty scaling up to handle large images containing many objects. In the current work we propose an alternative architecture that avoids these scaling issues by taking inspiration from recent work on *supervised* object detection, which has relied heavily on spatially invariant computations, particularly convolutions (LeCun et al. 1998; Long, Shelhamer, and Darrell 2015). A computation is spatially invariant if it applies an identical transformation to many different local image regions. YOLO networks for example use a convolutional neural network to map directly from an input image to a set of detected objects (Redmon et al. 2016).

The object detection task, supervised or not, has a number of features that make spatially invariant computations appropriate. For one, to a large extent object detection can be performed on local region of an image without having information about the remainder of the image. Second, the ability to detect objects in a given sub-region of an image is likely to transfer well to detecting objects in other sub-regions. Approaches that do not make use of spatial invariance, such as training a multi-layer perceptron to map from the image to objects, thus miss out on a great deal of exploitable structure.

In the current work, we propose **Sp**atially Invariant **A**ttend, **I**nfer, **R**epeat (SPAIR), an architecture for unsupervised object detection that is able to handle large, many-object scenes. At the core of SPAIR is a convolutional object detector, similar to that used in YOLO, which maps from images to objects. In order to train this object detector without supervision, we use it as the encoder network of a Variational Autoencoder (VAE) (Kingma and Welling 2013) which is trained to reconstruct the input image, a tactic introduced by AIR. Overall, SPAIR may be regarded as a VAE with a highly structured, object-like latent representation and a spatially invariant convolutional encoder network that effectively exploits the structure of objects in images.

Through a number of experiments, we demonstrate empirically that whereas competing approaches struggle at unsupervised object detection when images contain more than a few objects, our architecture scales up well to as many objects as we tested it with. Moreover, our architecture's ability to exploit spatial invariance allows it to generalize well to images that are larger and/or contain more objects than the images it was trained on. Finally, we show that our network is able to discover and detect objects with enough reliability to facilitate non-trivial downstream tasks.

2 Related Work

One promising approach to unsupervised object detection is known as Attend, Infer, Repeat (AIR) (Eslami et al. 2016). AIR formulates a Variational Autoencoder (VAE) (Kingma and Welling 2013) with a recurrent encoder network, an object-like latent representation and a decoder network that implements an image rendering process. Each processing step, the recurrent encoder takes as input the image and the recurrent hidden state, and outputs the size and position of an object in the scene as well as a new recurrent hidden state. For each image, the recurrent network is allowed to run for multiple time steps; ideally, a separate object is addressed on each time step, and the encoder can track which objects have already been addressed through its hidden state. The network is provided with facilities for halting computation when it judges that all objects have been accounted for. Finally, a decoder network takes the latent object descriptions and renders them into an output image. Weights of both the encoder and decoder are trained to make the rendered image as close as possible to the input image; through this training process, it is expected that the encoder network becomes an effective object detector, all without ground-truth object bounding boxes.

However, one shortcoming of AIR is that it does not scale well to large images with many objects. This is due to two aspects of AIR's formulation. First, the network has to learn to account for a different object on each time step, but when there are many objects, discovering how to do this becomes a difficult exploration problem. Indeed, Eslami et al. (2016) do not test the limits of AIR's scalability, their largest experiments assuming at most 3 objects per image. Second, the computational complexity of a training step (in particular, the image rendering step) scales as $O(mno)$ where (m, n) is the image shape, and o is the number of detected objects. If the object density is assumed fixed, this becomes $O(m^2n^2)$, making it infeasible to train on large images.

Recent work on Neural Expectation Maximization (Greff, van Steenkiste, and Schmidhuber 2017; van Steenkiste et al. 2018) aims to achieve something similar to AIR, but has comparable scaling issues. Related work from computer vision includes (Cao and Fei-Fei 2007; Kim and Torralba 2009; Lee and Grauman 2009; Russell et al. 2006; Karpathy, Miller, and Fei-Fei 2013), as well as work on co-segmentation (Zhu et al. 2016) and unsupervised object recognition (Tuytelaars et al. 2010).

3 Background: Variational Autoencoders

In this section we introduce the Variational Autoencoder (VAE) framework (Kingma and Welling 2013), which provides the theoretical foundation for our approach. We focus specifically on using VAEs to model images, though they can be used for any kind of data.

Assume images are sampled from a generative model as follows. A latent image representation z is sampled from a prior distribution $p(z)$; the image x is then produced according to a likelihood distribution $p(x|z)$. This likelihood distribution is usually unknown so we approximate it using a function $p_\theta(x|z)$ parameterized by a vector θ. The probability of an image is then given by:

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz$$

In general, we may be interested in:

1. Learning model parameters, $\theta^* = \arg\max_\theta \log p_\theta(x)$
2. Inferring a distribution over latent variables for an image, $p_\theta(z|x) = p_\theta(x|z)p(z)/p_\theta(x)$

However, in many settings of interest, computing $p_\theta(x)$ is intractable. This makes directly maximizing $p_\theta(x)$ impossible, and renders computing $p_\theta(z|x)$ intractable as well since it contains $p_\theta(x)$ as a normalizing constant. In such settings, variational methods offer a useful way forward.

Let $q_\phi(z|x)$ be a function, parameterized by a vector ϕ, that maps from an image x to a probability distribution over latent variables z; we will use q_ϕ to approximate $p_\theta(z|x)$. It can be shown that for any θ and ϕ, we have:

$$\log p_\theta(x) = \mathcal{L}(\theta, \phi) + D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$$

where

$$\mathcal{L}(\theta, \phi) := E_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}(q_\phi(z|x) \parallel p(z)) \quad (1)$$

Since the KL divergence is always positive, we have $\log p_\theta(x) \geq \mathcal{L}(\theta, \phi)$, and therefore $\mathcal{L}(\theta, \phi)$ is called the Evidence Lower Bound (ELBO).

Variational methods proceed by maximizing $\mathcal{L}(\theta, \phi)$ with respect to both θ and ϕ. The value θ thus obtained may be used as an approximation of θ^* since it is a maximum of the ELBO. Moreover, since ϕ maximizes $\mathcal{L}(\theta, \phi)$ for our chosen θ, and $p_\theta(x)$ does not change as ϕ changes, it follows that $D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$ is minimized with respect to ϕ. Therefore, $q_\phi(z|x)$ may be taken as an approximation of $p_\theta(z|x)$ and used for performing inference.

Under the VAE framework, $p_\theta(x|z)$ and $q_\phi(z|x)$ are neural networks, and $\mathcal{L}(\theta, \phi)$ is maximized with respect to θ and ϕ by stochastic gradient ascent. While the second term in (1) can usually be computed in closed form, the first term usually cannot and must instead be estimated using samples from $q_\phi(z|x)$. Backpropagating through this sampling process can be achieved using any of a large number of techniques that have been invented in recent years (Schulman et al. 2015).

Several parts of a VAE have counterparts in standard autoencoders. By inspection $\mathcal{L}(\theta, \phi)$ may be seen to be the sum of a term encouraging accurate reconstruction of the input image x (analogous to the reconstruction training objective in standard autoencoders) and a KL divergence regularization term. Meanwhile, $q_\phi(z|x)$, z and $p_\theta(x|z)$ correspond to the encoder, bottle-neck layer, and decoder, respectively, from standard autoencoders. Thus, we generally refer to $q_\phi(z|x)$ as an encoder or encoder network, and to $p_\theta(x|z)$ as a decoder or decoder network.

4 Spatially Invariant Attend, Infer, Repeat

Our model, which we call **Sp**atially Invariant **A**ttend, **I**nfer, **R**epeat (SPAIR), is a VAE with a highly structured, object-like latent representation z, a convolutional, object-detecting encoder network $q_\phi(z|x)$, and a decoder network $p_\theta(x|z)$ that "renders" detected objects into a reconstructed image. We now describe each of these components in detail.

4.1 Object-like Latent Representation

We first outline the representation scheme used by SPAIR for describing objects in its latent layer. Given an image with dimensions $(H_{img}, W_{img}, 3)$, it will be useful to spatially divide the image into an (H, W) grid of cells, where $H = \lceil H_{img}/c_h \rceil$, $W = \lceil W_{img}/c_w \rceil$ and c_h/c_w are fixed integers giving the cell height/width in pixels. We will use a representation that allows a single object per cell (the extension to multiple objects per cell is straightforward).

For a cell with indices (i, j), $i \in \{0, \ldots, H-1\}$, $j \in \{0, \ldots, W-1\}$, the corresponding object is described by the following variables:

$$z^{ij}_{\text{what}} \in \mathbb{R}^A \quad z^{ij}_{\text{depth}} \in \mathbb{R} \quad z^{ij}_{\text{pres}} \in \{0, 1\} \quad z^{ij}_{\text{where}} \in \mathbb{R}^4$$

z^{ij}_{what} is a vector with dimension A that stores appearance information for the object. z^{ij}_{depth} is a real number specifying the relative depth of the object; in the output image, objects with lower depth appear on top of objects with higher depth. z^{ij}_{pres} is a binary variable specifying whether the object exists; objects with $z^{ij}_{\text{pres}} = 0$ do not appear in the output image. z^{ij}_{where} decomposes as $z^{ij}_{\text{where}} = (z^{ij}_y, z^{ij}_x, z^{ij}_h, z^{ij}_w)$. z^{ij}_y and z^{ij}_x parameterize the position of the object according to:

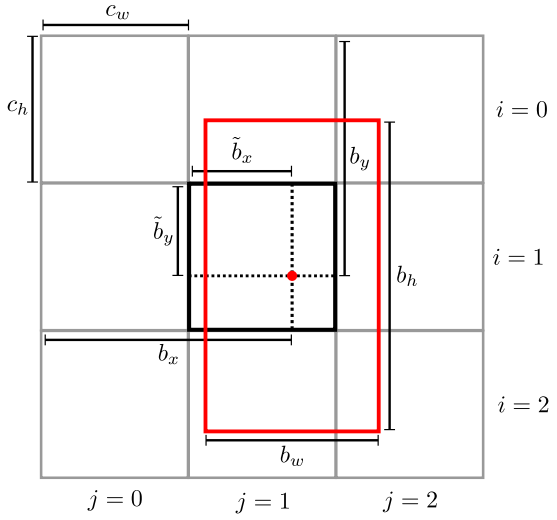

Figure 1: Diagram demonstrating the parameterization of object bounding boxes in SPAIR. We focus on a single cell, highlighted in black, with indices $(i, j) = (1, 1)$. The red box represents the bounding box for an object, with the red dot as its center. Relationships between the quantities in this diagram are given by Equations (2–6). To reduce clutter we have omitted the ij superscripts on the variables.

$$\tilde{b}^{ij}_y = \sigma(z^{ij}_y)\left(\tilde{b}^{max}_y - \tilde{b}^{min}_y\right) + \tilde{b}^{min}_y \tag{2}$$

$$b^{ij}_y = (i + \tilde{b}^{ij}_y)c_h \tag{3}$$

$$\tilde{b}^{ij}_x = \sigma(z^{ij}_x)\left(\tilde{b}^{max}_x - \tilde{b}^{min}_x\right) + \tilde{b}^{min}_x \tag{4}$$

$$b^{ij}_x = (j + \tilde{b}^{ij}_x)c_w \tag{5}$$

where σ denotes the sigmoid function, and b^{min}_* and b^{max}_* are fixed real numbers that impose bounds on the distance between the center of the object and the cell. See Figure 1 for a visual depiction of the relationships between these variables. z^{ij}_h and z^{ij}_w parameterize the size of the object as:

$$b^{ij}_h = \sigma(z^{ij}_h)a_h \qquad b^{ij}_w = \sigma(z^{ij}_w)a_w \tag{6}$$

for fixed real numbers a_h and a_w. Specifying object size with respect to a_h and a_w (as opposed to the the size of the input image) ensures that z^{ij}_h and z^{ij}_w are meaningful regardless of the size of the image the network is applied to[1].

4.2 Prior Distribution on Objects

A crucial component of a VAE is the prior distribution $p(z)$ over the latent variables. For all real-valued variables, we

[1] (a_h, a_w) can be interpreted as the dimensions of an *anchor box* as used in supervised object detection (Ren et al. 2015)

assume independent Normal distributions:

$$z^{ij}_{\text{where}} \sim N(\mu_{\text{where}}, \sigma_{\text{where}})$$
$$z^{ij}_{\text{what}} \sim N(\mu_{\text{what}}, \sigma_{\text{what}})$$
$$z^{ij}_{\text{depth}} \sim N(\mu_{\text{depth}}, \sigma_{\text{depth}})$$

where the Normal parameters are hyperparameters.

For the binary random variables z^{ij}_{pres}, we design a prior that puts pressure on the network to reconstruct the image using as few objects as possible (i.e. as few $z^{ij}_{\text{pres}} = 1$ as possible), similar to the prior used by AIR. This pressure is necessary for the network to extract quality object-like representations; without it, the network is free to set all $z^{ij}_{\text{pres}} = 1$, and use multiple latent objects to explain each object in the image.

We first gather all z^{ij}_{pres} into a binary vector z_{pres} with dimension HW. To generate z_{pres}, we first sample the number of non-zero entries according to a count distribution, and then draw z_{pres} uniformly from all binary vectors with the sampled number of non-zero entries. We choose a count distribution that puts most of the probability mass at low values; in particular, we use a Geometric distribution, truncated and normalized to have support $\{0, 1, \ldots, HW\}$. See Appendix A for the full prior and its derivation.

4.3 Encoder Network

Our goal is to design an encoder network $q_\phi(z|x)$ with spatially invariant properties. To this end, a convolutional neural network $e^{conv}_\phi(x)$ is first used to map from the image x to a feature volume with spatial dimensions (H, W). Next, this volume is processed sequentially cell-by-cell to produce objects (in the format described in Section 4.1) starting from the top-left and proceeding row-by-row toward the bottom-right.

Processing a cell runs as follows. First, a multi-layer perceptron e^{lat}_ϕ produces parameters $(\mu^{ij}_{\text{where}}, \sigma^{ij}_{\text{where}})$, $(\mu^{ij}_{\text{depth}}, \sigma^{ij}_{\text{depth}})$ and β^{ij}_{pres} for distributions over z^{ij}_{where}, z^{ij}_{depth} and z^{ij}_{pres}, respectively. As input, e^{lat}_ϕ accepts the feature vector (output of e^{conv}) at the current cell as well as sampled objects at nearby cells that have already been processed. e^{lat} can thus be thought of as encompassing a set of "lateral" connections which facilitate conditioning between nearby objects. In an ablation study (Section 5.5), we show empirically that these lateral connections are crucial for performance.

Next, values are sampled from the distributions:

$$z^{ij}_{\text{where}} \sim N(\mu^{ij}_{\text{where}}, \sigma^{ij}_{\text{where}})$$
$$z^{ij}_{\text{depth}} \sim N(\mu^{ij}_{\text{depth}}, \sigma^{ij}_{\text{depth}})$$
$$z^{ij}_{\text{pres}} \sim \text{Bernoulli}(\beta^{ij}_{\text{pres}})$$

The sampled value of z^{ij}_{where} is then used along with a spatial transformer T (Jaderberg et al. 2015) to extract a glimpse from the image. This glimpse is processed by an *object encoder network* e^{obj}_ϕ to yield parameters for a distribution over z^{ij}_{what}, which is subsequently sampled:

$$\mu^{ij}_{\text{what}}, \sigma^{ij}_{\text{what}} = e^{obj}_\phi(T(x, z^{ij}_{\text{where}}))$$
$$z^{ij}_{\text{what}} \sim N(\mu^{ij}_{\text{what}}, \sigma^{ij}_{\text{what}})$$

4.4 Decoder Network

The decoder network is responsible for rendering the detected objects back into an image. First, an *object decoder network* d^{obj}_θ processes all z^{ij}_{what} to yield a reconstruction of the appearance of each object:

$$o^{ij}, \alpha^{ij} = d^{obj}_\theta(z^{ij}_{\text{what}})$$

where o^{ij} is a volume with shape $(H_{obj}, W_{obj}, 3)$ (for fixed integers H_{obj}, W_{obj}) giving RGB values for the object, and α^{ij} is volume with shape $(H_{obj}, W_{obj}, 1)$ giving transparency values for the object.

Appearances of all objects are then stitched together to yield a final image, with z^{ij}_{where} used along with a spatial transformer to give the objects the correct size and location. α^{ij} is multiplied by z^{ij}_{pres} to ensure that objects with $z^{ij}_{\text{pres}} = 0$ are not drawn to the image. z^{ij}_{depth} values are used to parameterize a convex combination between the objects that overlap spatially, acting as a differentiable approximation of relative object depth. The output of rendering is an image x_{out}, which parameterizes $p_\theta(x|z)$ as a set of independent, pixel-wise Bernoulli variables. For a full description of this decoding/rendering process, see Appendix C.

Objects are rendered on top of a background; in principle, a separate neural network may be used to learn a background for each image. Throughout this work, we use monochrome backgrounds; either a single color, obtained by taking the statistical mode of a small sampling of training images, or use a multi-layer perceptron to map from each image to a background color.

4.5 Training

SPAIR is trained by gradient ascent on the ELBO (Equation (1)) using a single sample to estimate the expectation. To backpropagate through the sampling process, we make use of the reparameterization trick (Kingma and Welling 2013). For the Normally distributed random variables z^{ij}_{where}, z^{ij}_{what} and z^{ij}_{depth} this is straightforward. The discrete Bernoulli random variables z^{ij}_{pres} are replaced with Concrete variables, continuous relaxations of Bernoullis to which the reparameterization trick can be easily applied (Maddison, Mnih, and Teh 2016; Jang, Gu, and Poole 2016). At validation and test time the Concretes are discretized via rounding.

Equation (1) also requires computation of the KL divergence between $q_\phi(z|x)$ and $p(z)$. This is straightforward for the independent, Normally distributed real-valued random variables, but slightly more difficult for the Bernoulli/Concrete variables; see Appendix B for full details.

4.6 Scaling Properties of SPAIR

Several aspects of SPAIR's design are crucial for scaling. Consider the following simple method for making *any* object detector spatially invariant: have the object detector operate on a small, fixed input size, and when faced with a large image, simply run the detector multiple times on appropriately-sized, potentially overlapping local regions of the image. Pool the objects detected at each location to obtain the set of objects for the image. Assuming the objects are small enough

to fit in a local image region, it should be much easier to learn parameters for this object detector, compared with a non-spatially invariant method that attempts to process the image all at once, since there should be many fewer objects and significantly less complexity in a local image region than the image as a whole. Moreover, each application of the local object detector can be seen a separate training example; we can consequently think of the local object detector as having access to much more training data.

The use of a convolutional network in the encoder, with its weight sharing and spatially local receptive fields (LeCun et al. 1998), is essentially a more computationally efficient (Long, Shelhamer, and Darrell 2015) way of implementing the above object detection scheme, where each neuron in the output layer of the convolutional network is analogous to a single application of the local object detector.

As mentioned in Section 2, one way in which AIR scales poorly is in the computational complexity of the decoder step which stitches together the detected objects into an output image; SPAIR has a similar step, and so does not avoid this issue directly. However, the computational cost of this step is much less problematic for SPAIR because, as shown below in Section 5.2, it can be trained on small random crops of large images, while still processing full images at test time.

5 Experiments

In this section we empirically demonstrate the advantages of SPAIR on a number of different tasks.

5.1 Comparison with AIR

One of the main benefits that we expect to gain from SPAIR's spatial invariance is significantly improved ability to discover and detect objects in many-object scenes. To test this, we trained both AIR and SPAIR on 48×48 images each containing scattered MNIST digits of size 14×14 rendered in white on a black background. The goal is to have the models output accurate bounding boxes for the digits in each image, without ever having access to ground-truth bounding boxes. In order to probe the effect of the number of objects per image on model performance, we used 5 different training conditions; in each condition, the images contain a different number of digits, ranging from 1 to 9. For each training condition, we prepare training, validation and testing datasets, and digit instances are not shared between these (so at test time, the networks are seeing digit instances that they have never seen before). As a performance measure we use an adapted version of the Average Precision[2] between the bounding boxes predicted by the model and the ground-truth bounding boxes, commonly used in the supervised object detection literature (Everingham et al. 2010).

To simplify the comparison with AIR, we fixed the number of steps executed by AIR's recurrent network to the true number of objects in the image, effectively "telling" AIR how many objects are present. A variant of AIR called Difference Attend, Infer, Repeat (DAIR) (Eslami et al. 2016) was also tested and provided with this same information.

[2]In particular, we use AP@IOU=0.1:0.1:0.9, see `http://cocodataset.org/#detection-eval` for details.

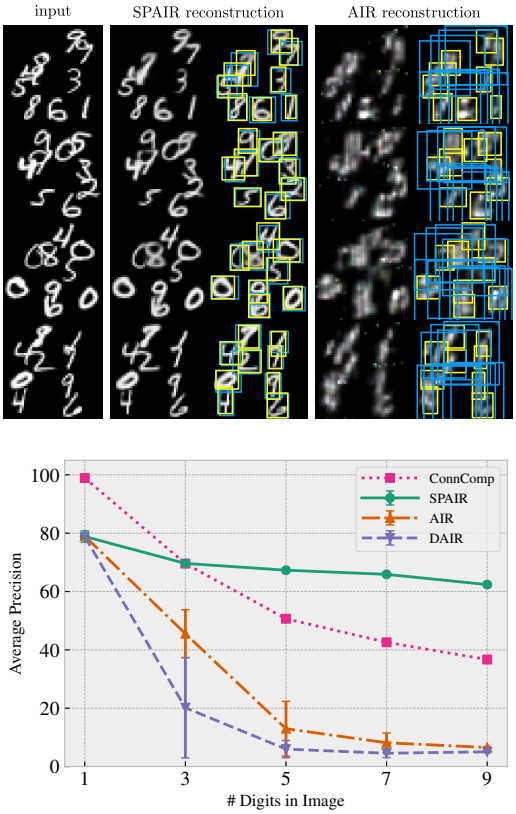

Figure 2: Top: Example input images and reconstructions by SPAIR and AIR. Predicted bounding boxes are shown in blue, ground-truth boxes in yellow. Bottom: Average Precision achieved by different algorithms on a scattered MNIST dataset as the number of digits per image varies.

We also devised a simple baseline method which we call ConnComp which detects objects by finding connected components in an image. For each pixel, ConnComp computes the absolute difference between the color at that pixel and the background color (which is given to it). All pixels for which this difference is greater than a threshold τ (a hyperparameter) are assigned a label of 1, the rest are assigned 0. Each connected cluster of pixels that have label 1 is taken to be an object. Success of this method can be used as a measure of the difficulty of the dataset: it will be successful to the degree that objects do not overlap and are easy to segment.

Results, shown in Figure 2, clearly demonstrate that on this task, SPAIR significantly outperforms all tested algorithms when images contain many objects.

5.2 Generalization

Another hypothesized advantage of SPAIR's spatial invariance is a capacity for generalizing to images that are larger and/or contain different numbers of objects than images encountered during training. Here we test this hypothesis. We created three different training datasets, each consisting of images of size 84×84 containing randomly scattered MNIST digits. In each training condition, the training images con-

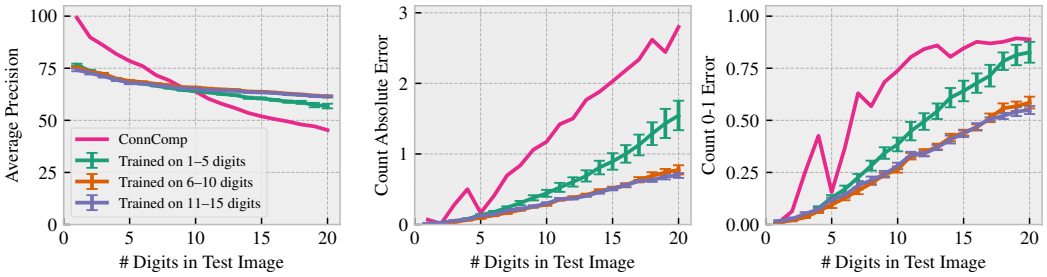

Figure 3: Assessing SPAIR's ability to generalize to images that are both larger and contain more objects than images seen during training. Models were trained on small random crops of large images. The large images contained either 1–5 digits, 6–10 digits or 11–15 digits. Models were then tested on large pre-crop images containing between 1 and 20 digits. A simple baseline algorithm called ConnComp (see Section 5.1) was also tested. Left: Average Precision. The 6–10 and 11–15 lines overlap almost completely. Middle: Absolute difference between the number of objects predicted by the models and the true number of digits. Right: Mean 0-1 error between the number of objects predicted by models and the true number of digits.

tained different numbers of digits: either 1–5, 6–10 or 11–15 digits. The trained models were then tested on images containing between 1 and 20 digits. To demonstrate that SPAIR models can be effectively applied to images of different sizes than what they were trained on, the training set actually consists of random crops, each of size 48×48, of full images, while at test time the network was forced to process the pre-crop 84×84 images. The network never sees full images during training. In addition to AP, we also tracked how well the algorithms performed at guessing the number of objects in the scene.

Results of this experiment, shown in Figure 3, demonstrate that SPAIR models have significant generalization ability. The performance of all SPAIR models degraded gracefully as the number of digits per test image increased, even well above the maximum number of digits seen during training. There is no significant difference between the performance of models trained on the 6–10 digit condition compared with models trained on the 11–15 digit condition. Models trained on the 1–5 digit condition exhibited lower performance when applied to images containing large numbers of digits, presumably because their training experience did not equip them to deal with densely packed digits.

5.3 Downstream Tasks

The main motivation behind the project of extracting objects from scenes is the ability to use those objects in downstream tasks. With this in mind, we perform two experiments confirming that SPAIR networks can be trained to extract objects with enough accuracy to be useful for downstream tasks.

Addition. For the first task, we stay in the domain of scattered MNIST digits, but now rather than asking models to simply locate digits, we ask them to perform an arithmetic operation on the digits, namely addition. Each training example consists of a 48×48 image containing 5 scattered digits, paired with an integer label giving the sum of the digits as a one-hot vector. We tested models over a number of different training set sizes to probe sample efficiency.

For both SPAIR and AIR, we have an initial training stage where the model ignores the labels and maximizes the ELBO (Equation (1)) as usual. In a second training stage, the output of the encoder networks (i.e. the object-like representation) is fed into a downstream classifier, which is trained to minimize cross-entropy classification loss for the addition task. We experimented with two variants of this second training stage. In the first variant (*Fixed*) the weights for the encoder network, obtained during the initial training stage, are kept fixed. In the second (*Unfixed*) all encoder weights are fixed except for the weights of the object encoder network, which are trained to minimize classification cross-entropy loss along with the weights of the downstream classifier.

We also tested several other models on this task. The first model TrueBB is given ground-truth bounding boxes for training images, extracting glimpses from the image using those bounding boxes and feeding those glimpses into a downstream classifier – this is meant to act as an upper-bound on performance, showing what can be achieved with access to perfect object bounding boxes. The second model ConnComp is similar, but obtains bounding boxes using the connected components algorithm described in Section 5.1, rather than using the ground-truth boxes. The final model ConvNet uses a convolutional neural network (with the same architecture as SPAIR's convolutional backbone) to extract features from the image, which are once again passed into a downstream classifier. In all cases, the downstream classifier was an LSTM with 128 hidden units that iterated over objects (for ConvNet, each spatial location in the output volume was treated as a separate object).

Results of this experiment are shown in Figure 4. The two variants of SPAIR significantly outperform all methods other than TrueBB.

The Game of SET. Here we explore using SPAIR as a front-end for learning to play a version of the card game SET (Chaudhuri et al. 2003). Let p be the number of *properties* and v the number of *values* per property, for integers p and v. SET is played using a deck of cards; each card has a value for each of the p properties, and the deck contains a single copy of every unique card (and thus has size v^p). A *set* is a

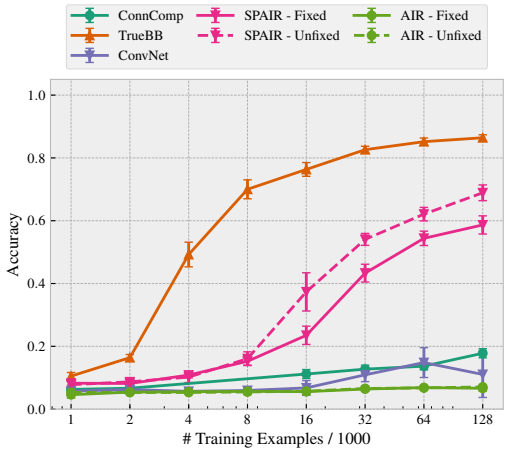

Figure 4: Assessing model performance on a task that requires models to take in images containing 5 digits and output the sum of the digits.

collection of v cards such that for each of the p properties, the cards either all have the same value for the property or all have different values for the property. To play a round of SET, the deck is shuffled and a fixed number of cards n are drawn from the top and displayed face up. The goal is to be the first to identify from the collection of drawn cards a *set* if one exists. The standard game of SET uses $n = 12$, $p = 4$ and $v = 3$, with the four properties being shape, number, color and texture.

Here we consider a simplified version of SET with $n = 7$, $p = 3$, $v = 3$ and shape, number and color as the properties. In this context, we created a dataset designed to train agents to determine, given an image depicting $n = 7$ cards, whether there exists a *set* among those cards. Cards are depicted as colored shapes with number represented by black strokes. Each training example consists of an image of 7 cards and a binary label specifying whether a *set* exists among the cards. To increase perceptual difficulty, we allow the cards to overlap significantly with one another, and use three different background colors. Training, validation and testing datasets all had roughly equal numbers of positive and negative examples, enforced through rejection sampling.

The training dataset contained 128,000 examples, while validation and test datasets contained 500 examples each. We tested SPAIR (with a two-stage training schedule similar to the previous experiment), TrueBB and ConvNet, and used an LSTM with 256 hidden units as the downstream classifier. The results are shown in Table 1. SPAIR discovers and detects cards well enough to allow downstream performance equal to what can be obtained using ground-truth bounding boxes. In contrast, the pure convolutional network, which has no explicit notion of objects, is completely unable to make progress on the task. Example images and qualitative results from SPAIR are shown in Figure (**?**).

	SPAIR	ConvNet	TrueBB
Test Acc.	57.9	50.1	57.7
Std. Dev.	0.019	0	0.012

Table 1: Model performance on the game of SET.

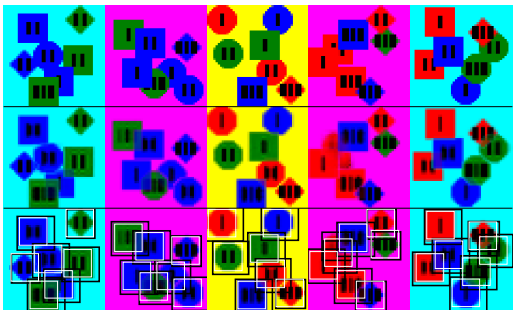

Figure 5: Example images for SET. Top: True image. Middle: SPAIR Reconstruction. Bottom: Reconstruction again, with (ground-truth/predicted) bounding boxes in (white/black).

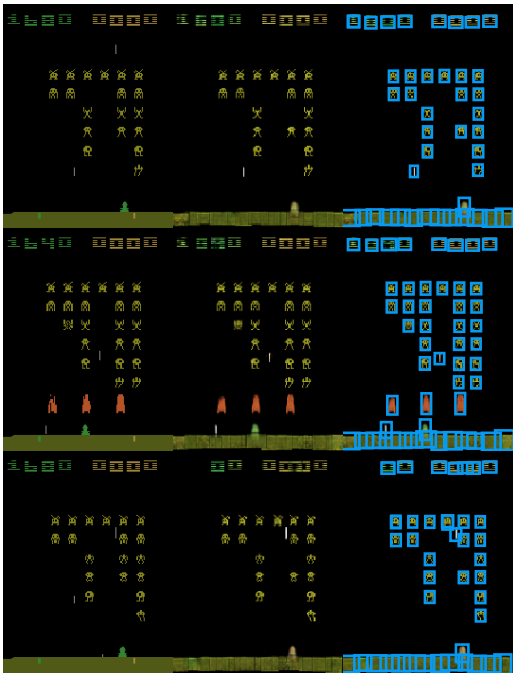

Figure 6: Example images from the Space Invaders Atari game (left), and reconstruction (middle) and object bounding boxes (right) yielded by a trained SPAIR model.

5.4 Space Invaders

To push the scaling capabilities of SPAIR further, we trained it on images from Space Invaders using the Arcade Learning Environment (Bellemare et al. 2013), collected with a random policy. The network was trained on random crops of size 48×48; at test time the network processes full images (size 210×160). Qualitative results are given in Figure 6.

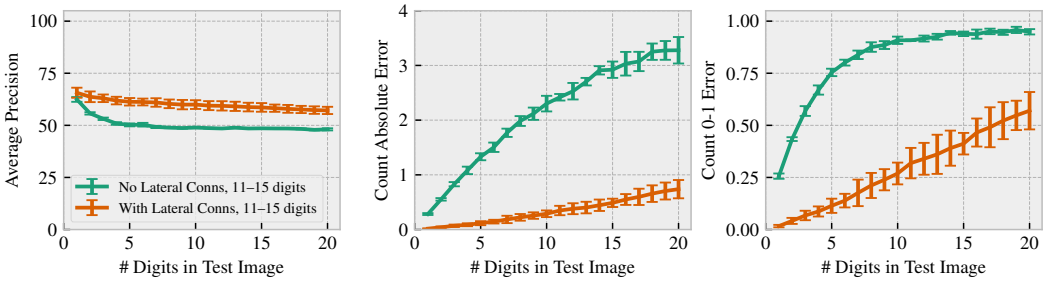

Figure 7: Comparing performance of a SPAIR network with 1-step lateral connections to a network with no lateral connections. The experimental setup is similar to the experiment described in Section 5.2 / Figure 3.

5.5 Ablation Study on the Importance of Lateral Connections

In Section 4.3, we described a set of lateral connections e_ϕ^{lat} in the encoder network which allow the object produced at a cell to depend on objects already sampled at nearby cells. These connections may, for instance, help the network avoid detecting the same object multiple times at different cells (duplicate detections are possible in principle because of overlap between the receptive fields of neighbouring neurons in the output layer of the convolutional network e_ϕ^{conv}). To test that these connections are indeed important, we performed an ablation study. In particular, we repeated the 11–15 digit training condition from Section 5.2 / Figure 3 using one network that uses lateral connections which allow objects to directly depend on objects up to 1 cell away (the default throughout this work), and another network which had no lateral connections. The results, shown in Figure 7, clearly show that the lateral connections are crucial when the network is asked to generalize to larger images with more objects than seen during training. For this experiment we used a slightly different architecture and tuning procedure, and hence these results are not directly comparable to the results in Section 5.2 / Figure 3.

6 Conclusion

In this paper, we introduced a novel architecture for unsupervised object detection which combines features of existing approaches (such as AIR) with the spatial invariance properties of recent supervised object detection architectures such as YOLO. We showed empirically that the addition of this spatial invariance allows for greatly improved scaling; in particular, we showed that our spatially invariant architecture outperforms competing approaches for scenes with many objects, that our approach can generalize to scenes larger and more complex than scenes it was trained on, and that our approach is able to discover and detect objects with enough accuracy to support downstream tasks.

7 Acknowledgements

Funding for this work was provided by NSERC and Samsung. Thanks to Guillaume Rabusseau for helpful preliminary discussions.

References

[Bellemare et al. 2013] Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

[Cao and Fei-Fei 2007] Cao, L., and Fei-Fei, L. 2007. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 1–8. IEEE.

[Chang et al. 2016] Chang, M. B.; Ullman, T.; Torralba, A.; and Tenenbaum, J. B. 2016. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*.

[Chaudhuri et al. 2003] Chaudhuri, K.; Godfrey, B.; Ratajczak, D.; and Wee, H. 2003. On the complexity of the game of set.

[Diuk, Cohen, and Littman 2008] Diuk, C.; Cohen, A.; and Littman, M. L. 2008. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 240–247. ACM.

[Eslami et al. 2016] Eslami, S. A.; Heess, N.; Weber, T.; Tassa, Y.; Szepesvari, D.; Hinton, G. E.; et al. 2016. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, 3225–3233.

[Everingham et al. 2010] Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88(2):303–338.

[Garnelo, Arulkumaran, and Shanahan 2016] Garnelo, M.; Arulkumaran, K.; and Shanahan, M. 2016. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*.

[Greff, van Steenkiste, and Schmidhuber 2017] Greff, K.; van Steenkiste, S.; and Schmidhuber, J. 2017. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, 6691–6701.

[Jaderberg et al. 2015] Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *Advances in neural information processing systems*, 2017–2025.

[Jang, Gu, and Poole 2016] Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[Kansky et al. 2017] Kansky, K.; Silver, T.; Mély, D. A.; Eldawy, M.; Lázaro-Gredilla, M.; Lou, X.; Dorfman, N.; Sidor,

S.; Phoenix, S.; and George, D. 2017. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *International Conference on Machine Learning*, 1809–1818.

[Karpathy, Miller, and Fei-Fei 2013] Karpathy, A.; Miller, S.; and Fei-Fei, L. 2013. Object discovery in 3d scenes via shape analysis. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2088–2095. IEEE.

[Kim and Torralba 2009] Kim, G., and Torralba, A. 2009. Unsupervised detection of regions of interest using iterative link analysis. In *Advances in neural information processing systems*, 961–969.

[Kingma and Welling 2013] Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[LeCun et al. 1998] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

[Lee and Grauman 2009] Lee, Y. J., and Grauman, K. 2009. Shape discovery from unlabeled image collections. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2254–2261. IEEE.

[Long, Shelhamer, and Darrell 2015] Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

[Maddison, Mnih, and Teh 2016] Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

[Redmon et al. 2016] Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

[Ren et al. 2015] Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

[Russell et al. 2006] Russell, B. C.; Freeman, W. T.; Efros, A. A.; Sivic, J.; and Zisserman, A. 2006. Using multiple segmentations to discover objects and their extent in image collections. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, 1605–1614. IEEE.

[Santoro et al. 2017] Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, 4967–4976.

[Schulman et al. 2015] Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 3528–3536.

[Tuytelaars et al. 2010] Tuytelaars, T.; Lampert, C. H.; Blaschko, M. B.; and Buntine, W. 2010. Unsupervised object discovery: A comparison. *International journal of computer vision* 88(2):284–302.

[van Steenkiste et al. 2018] van Steenkiste, S.; Chang, M.; Greff, K.; and Schmidhuber, J. 2018. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*.

[Zambaldi et al. 2018] Zambaldi, V.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.; Lillicrap, T.; Lockhart, E.; et al. 2018. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*.

[Zhu et al. 2016] Zhu, H.; Meng, F.; Cai, J.; and Lu, S. 2016. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation* 34:12–27.