

Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects

Adam R. Kosiorek^{*,§,†}Hyunjik Kim[†]Ingmar Posner[§]Yee Whye Teh[†]

[§] **Applied Artificial Intelligence Lab**
Oxford Robotics Institute
University of Oxford

[†] **Department of Statistics**
University of Oxford

Abstract

We present Sequential Attend, Infer, Repeat (SQAIR), an interpretable deep generative model for videos of moving objects. It can reliably discover and track objects throughout the sequence of frames, and can also generate future frames conditioning on the current frame, thereby simulating expected motion of objects. This is achieved by explicitly encoding object presence, locations and appearances in the latent variables of the model. SQAIR retains all strengths of its predecessor, Attend, Infer, Repeat (AIR, Eslami et al., 2016), including learning in an unsupervised manner, and addresses its shortcomings. We use a moving multi-MNIST dataset to show limitations of AIR in detecting overlapping or partially occluded objects, and show how SQAIR overcomes them by leveraging temporal consistency of objects. Finally, we also apply SQAIR to real-world pedestrian CCTV data, where it learns to reliably detect, track and generate walking pedestrians with no supervision.

1 Introduction

The ability to identify objects in their environments and to understand relations between them is a cornerstone of human intelligence (Kemp and Tenenbaum, 2008). Arguably, in doing so we rely on a notion of spatial and temporal consistency which gives rise to an expectation that objects do not appear out of thin air, nor do they spontaneously vanish, and that they can be described by properties such as location, appearance and some dynamic behaviour that explains their evolution over time. We argue that this notion of consistency can be seen as an *inductive bias* that improves the efficiency of our learning. Equally, we posit that introducing such a bias towards spatio-temporal consistency into our models should greatly reduce the amount of supervision required for learning.

One way of achieving such inductive biases is through model structure. While recent successes in deep learning demonstrate that progress is possible without explicitly imbuing models with interpretable structure (LeCun, Bengio, et al., 2015), recent works show that introducing such structure into deep models can indeed lead to favourable inductive biases improving performance e.g. in convolutional networks (LeCun, Boser, et al., 1989) or in tasks requiring relational reasoning (Santoro et al., 2017). Structure can also make neural networks useful in new contexts by significantly improving generalization, data efficiency (Jacobsen et al., 2016) or extending their capabilities to unstructured inputs (Graves et al., 2016).

Attend, Infer, Repeat (AIR), introduced by Eslami et al., 2016, is a notable example of such a structured probabilistic model that relies on deep learning and admits efficient amortized inference. Trained without any supervision, AIR is able to decompose a visual scene into its constituent components and to generate a (learned) number of latent variables that explicitly encode the location and appearance of each object. While this approach is inspiring, its focus on modelling individual (and thereby inherently static) scenes leads to a number of limitations. For example, it often merges two objects that are close together into one since no temporal context is available to distinguish between them.

^{*}Corresponding author: adamk@robots.ox.ac.uk

Similarly, we demonstrate that AIR struggles to identify partially occluded objects, e.g. when they extend beyond the boundaries of the scene frame (see Figure 7 in Section 4.1).

Our contribution is to mitigate the shortcomings of AIR by introducing a sequential version that models sequences of frames, enabling it to discover and track objects over time as well as to generate convincing extrapolations of frames into the future. We achieve this by leveraging temporal information to learn a richer, more capable generative model. Specifically, we extend AIR into a spatio-temporal state-space model and train it on unlabelled image sequences of dynamic objects. We show that the resulting model, which we name Sequential AIR (SQAIR), retains the strengths of the original AIR formulation while outperforming it on moving MNIST digits.

The rest of this work is organised as follows. In Section 2, we describe the generative model and inference of AIR. In Section 3, we discuss its limitations and how it can be improved, thereby introducing Sequential Attend, Infer, Repeat (SQAIR), our extension of AIR to image sequences. In Section 4, we demonstrate the model on a dataset of multiple moving MNIST digits (Section 4.1) and compare it against AIR trained on each frame and Variational Recurrent Neural Network (VRNN) of Chung et al., 2015 with convolutional architectures, and show the superior performance of SQAIR in terms of log marginal likelihood and interpretability of latent variables. We also investigate the utility of inferred latent variables of SQAIR in downstream tasks. In Section 4.2 we apply SQAIR on real-world pedestrian CCTV data, where SQAIR learns to reliably detect, track and generate walking pedestrians without any supervision. Code for the implementation on the MNIST dataset² and the results video³ are available online.

2 Attend, Infer, Repeat (AIR)

AIR, introduced by Eslami et al., 2016, is a structured variational auto-encoder (VAE) capable of decomposing a static scene \mathbf{x} into its constituent objects, where each object is represented as a separate triplet of continuous latent variables $\mathbf{z} = \{\mathbf{z}^{\text{what},i}, \mathbf{z}^{\text{where},i}, z^{\text{pres},i}\}_{i=1}^n$, $n \in \mathbb{N}$ being the (random) number of objects in the scene. Each triplet of latent variables explicitly encodes position, appearance and presence of the respective object, and the model is able to infer the number of objects present in the scene. Hence it is able to count, locate and describe objects in the scene, all learnt in an unsupervised manner, made possible by the inductive bias introduced by the model structure.

Generative Model The generative model of AIR is defined as follows

$$\begin{aligned} p_\theta(n) &= \text{Geom}(n \mid \theta), & p_\theta(\mathbf{z}^w \mid n) &= \prod_{i=1}^n p_\theta(\mathbf{z}^{w,i}) = \prod_{i=1}^n \mathcal{N}(\mathbf{z}^{w,i} \mid \mathbf{0}, \mathbf{I}), \\ p_\theta(\mathbf{x} \mid \mathbf{z}) &= \mathcal{N}(\mathbf{x} \mid \mathbf{y}_t, \sigma_x^2 \mathbf{I}), & \text{with } \mathbf{y}_t &= \sum_{i=1}^n h_\theta^{\text{dec}}(\mathbf{z}^{\text{what},i}, \mathbf{z}^{\text{where},i}), \end{aligned} \quad (1)$$

where $\mathbf{z}^{w,i} := (\mathbf{z}^{\text{what},i}, \mathbf{z}^{\text{where},i})$, $z^{\text{pres},i} = 1$ for $i = 1 \dots n$ and h_θ^{dec} is the object decoder with parameters θ . It is composed of a *glimpse decoder* $f_\theta^{\text{dec}} : \mathbf{g}_t^i \mapsto \mathbf{y}_t^i$, which constructs an image patch and a spatial transformer (ST, Jaderberg et al., 2015), which scales and shifts it according to $\mathbf{z}^{\text{where},i}$; see Figure 1 for details.

Inference Eslami et al., 2016 use a sequential inference algorithm, where latent variables are inferred one at a time; see Figure 2. The number of inference steps n is given by $z^{\text{pres},1:n+1}$, a random vector of n ones followed by a zero. The \mathbf{z}^i are sampled sequentially from

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = q_\phi(z^{\text{pres},n+1} = 0 \mid \mathbf{z}^{w,1:n}, \mathbf{x}) \prod_{i=1}^n q_\phi(\mathbf{z}^{w,i}, z^{\text{pres},i} = 1 \mid \mathbf{z}^{1:i-1}, \mathbf{x}), \quad (2)$$

where q_ϕ is implemented as a neural network with parameters ϕ . To implement explaining away, e.g. to avoid encoding the same object twice, it is vital to capture the dependency of $\mathbf{z}^{w,i}$ and $z^{\text{pres},i}$ on $\mathbf{z}^{1:i-1}$ and \mathbf{x} . This is done using a recurrent neural network (RNN) R_ϕ with hidden state \mathbf{h}^i , namely: $\omega^i, \mathbf{h}^i = R_\phi(\mathbf{x}, \mathbf{z}^{i-1}, \mathbf{h}^{i-1})$. The outputs ω^i , which are computed iteratively and depend on the previous latent variables (cf. Algorithm 3), parametrise $q_\phi(\mathbf{z}^{w,i}, z^{\text{pres},i} \mid \mathbf{z}^{1:i-1}, \mathbf{x})$. For simplicity the latter is assumed to factorise such that $q_\phi(\mathbf{z}^w, \mathbf{z}^{\text{pres}} \mid \mathbf{z}^{1:i-1}, \mathbf{x}) = q_\phi(z^{\text{pres},n+1} = 0 \mid \omega^{n+1}) \prod_{i=1}^n q_\phi(\mathbf{z}^{w,i} \mid \omega^i) q_\phi(z^{\text{pres},i} = 1 \mid \omega^i)$.

²code: github.com/akosiorek/sqair

³video: youtu.be/-IUNQgSLE0c

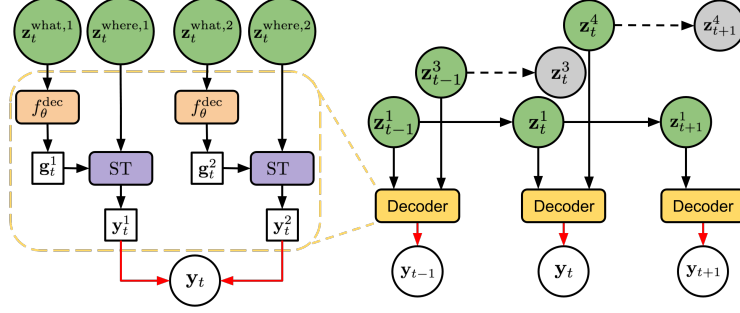


Figure 1: *Left*: Generation in AIR. The image mean y_t is generated by first using the *glimpse decoder* f_θ^{dec} to map the *what* variables into glimpses g_t , transforming them with the *spatial transformer* ST according to the *where* variables and summing up the results. *Right*: Generation in SQAIR. When new objects enter the frame, new latent variables (here, z_t^2) are sampled from the *discovery* prior. The temporal evolution of already present objects is governed by the *propagation* prior, which can choose to forget some variables (here, z_t^3 and z_{t+1}^4) when the object moves out of the frame. The image generation process, which mimics the left-hand side of the figure, is abstracted in the *decoder* block.

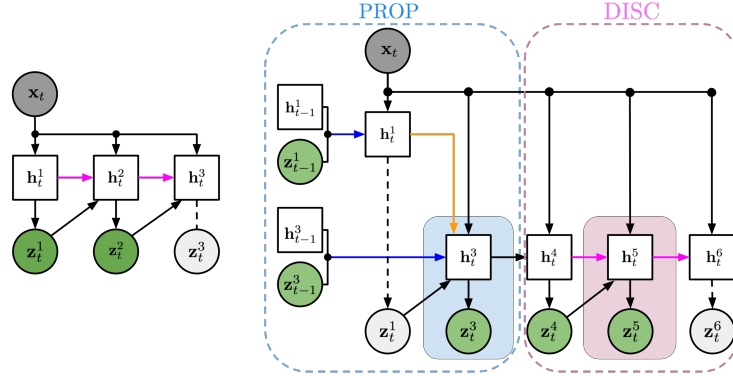


Figure 2: *Left*: Inference in AIR. The *pink RNN* attends to the image sequentially and produces one latent variable z_t^i at a time. Here, it decides that two latent variables are enough to explain the image and z_t^3 is not generated. *Right*: Inference in SQAIR starts with the *Propagation* (PROP) phase. PROP iterates over latent variables from the previous time-step $t - 1$ and updates them based on the new observation x_t . The *blue RNN* runs forward in time to update the hidden state of each object, to model its change in appearance and location throughout time. The *orange RNN* runs across all current objects and models the relations between different objects. Here, when attending to z_{t-1}^1 , it decides that the corresponding object has disappeared from the frame and *forgets* it. Next, the *Discovery* (DISC) phase detects new objects as in AIR, but in SQAIR it is also conditioned on the results of PROP, to prevent rediscovering objects. See Figure 3 for details of the colored RNNs.

3 Sequential Attend-Infer-Repeat

While capable of decomposing a scene into objects, AIR only describes single images. Should we want a similar decomposition of an image sequence, it would be desirable to do so in a temporally consistent manner. For example, we might want to detect objects of the scene as well as infer dynamics and track identities of any persistent objects. Thus, we introduce Sequential Attend, Infer, Repeat (SQAIR), whereby AIR is augmented with a state-space model (SSM) to achieve temporal consistency in the generated images of the sequence. The resulting probabilistic model is composed of two parts: Discovery (DISC), which is responsible for detecting (or introducing, in the case of the generation) new objects at every time-step (essentially equivalent to AIR), and Propagation (PROP), responsible for updating (or forgetting) latent variables from the previous time-step given the new observation (image), effectively implementing the temporal SSM. We now formally introduce SQAIR by first describing its generative model and then the inference network.

Generative Model The model assumes that at every-time step, objects are first propagated from the previous time-step (PROP). Then, new objects are introduced (DISC). Let $t \in \mathbb{N}$ be the current time-step. Let \mathcal{P}_t be the set of objects propagated from the previous time-step and let \mathcal{D}_t be the set of objects discovered at the current time-step, and let $\mathcal{O}_t = \mathcal{P}_t \cup \mathcal{D}_t$ be the set of all objects present at time-step t . Consequently, at every time step, the model retains a set of latent variables $\mathbf{z}_t^{\mathcal{P}_t} = \{z_t^i\}_{i \in \mathcal{P}_t}$, and

generates a set of new latent variables $\mathbf{z}_t^{\mathcal{D}_t} = \{\mathbf{z}_t^i\}_{i \in \mathcal{D}_t}$. Together they form $\mathbf{z}_t := [\mathbf{z}_t^{\mathcal{P}_t}, \mathbf{z}_t^{\mathcal{D}_t}]$, where the representation of the i^{th} object $\mathbf{z}_t^i := [\mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, z_t^{\text{pres},i}]$ is composed of three components (as in AIR): $\mathbf{z}_t^{\text{what},i}$ and $\mathbf{z}_t^{\text{where},i}$ are real vector-valued variables representing appearance and location of the object, respectively. $z_t^{\text{pres},i}$ is a binary variable representing whether the object is present at the given time-step or not.

At the first time-step ($t = 1$) there are no objects to propagate, so we sample D_1 , the number of objects at $t = 1$, from the discovery prior $p^D(D_1)$. Then for each object $i \in \mathcal{D}_t$, we sample latent variables $\mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}$ from $p^D(\mathbf{z}_1^i | D_1)$. At time $t = 2$, the *propagation* step models which objects from $t = 1$ are propagated to $t = 2$, and which objects disappear from the frame, using the binary random variable $(z_t^{\text{pres},i})_{i \in \mathcal{P}_t}$. The *discovery* step at $t = 2$ models new objects that enter the frame, with a similar procedure to $t = 1$: sample D_2 (which depends on $\mathbf{z}_2^{\mathcal{P}_2}$) then sample $(\mathbf{z}_2^{\text{what},i}, \mathbf{z}_2^{\text{where},i})_{i \in \mathcal{D}_2}$. This procedure of propagation and discovery recurs for $t = 2, \dots, T$. Once the \mathbf{z}_t have been formed, we may generate images \mathbf{x}_t using the exact same generative distribution $p_\theta(\mathbf{x}_t | \mathbf{z}_t)$ as in AIR (cf. Equation (1), Fig. 1, and Algorithm 1). In full, the generative model is:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, D_{1:T}) = p^D(D_1, \mathbf{z}_1^{\mathcal{D}_1}) \prod_{t=2}^T p^D(D_t, \mathbf{z}_t^{\mathcal{D}_t} | \mathbf{z}_t^{\mathcal{P}_t}) p^P(\mathbf{z}_t^{\mathcal{P}_t} | \mathbf{z}_{t-1}) p_\theta(\mathbf{x}_t | \mathbf{z}_t), \quad (3)$$

The *discovery prior* $p^D(D_t, \mathbf{z}_t^{\mathcal{D}_t} | \mathbf{z}_t^{\mathcal{P}_t})$ samples latent variables for new objects that enter the frame. The *propagation prior* $p^P(\mathbf{z}_t^{\mathcal{P}_t} | \mathbf{z}_{t-1})$ samples latent variables for objects that persist in the frame and removes latents of objects that disappear from the frame, thereby modelling dynamics and appearance changes. Both priors are learned during training. The exact forms of the priors are given in Appendix B.

Inference Similarly to AIR, inference in SQAIR can capture the number of objects and the representation describing the location and appearance of each object that is necessary to explain every image in a sequence. As with generation, inference is divided into PROP and DISC. During PROP, the inference network achieves two tasks. Firstly, the latent variables from the previous time step are used to infer the current ones, modelling the change in location and appearance of the corresponding objects, thereby attaining temporal consistency. This is implemented by the *temporal* RNN \mathbf{R}_ϕ^T , with hidden states \mathbf{h}_t^T (recurs in t). Crucially, it does not access the current image directly, but uses the output of the *relation* RNN (cf. Santoro et al., 2017). The relation RNN takes relations between objects into account, thereby implementing the *explaining away* phenomenon; it is essential for capturing any interactions between objects as well as occlusion (or overlap, if one object is occluded by another). See Figure 7 for an example. These two RNNs together decide whether to retain or to forget objects that have been propagated from the previous time step. During DISC, the network infers further latent variables that are needed to describe any new objects that have entered the frame. All latent variables remaining after PROP and DISC are passed on to the next time step.

See Figures 2 and 3 for the inference network structure. The full variational posterior is defined as

$$q_\phi(D_{1:T}, \mathbf{z}_{1:T} | \mathbf{x}_{1:T}) = \prod_{t=1}^T q_\phi^D(D_t, \mathbf{z}_t^{\mathcal{D}_t} | \mathbf{x}_t, \mathbf{z}_t^{\mathcal{P}_t}) \prod_{i \in \mathcal{O}_{t-1}} q_\phi^P(\mathbf{z}_t^i | \mathbf{z}_{t-1}^i, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}). \quad (4)$$

Discovery, described by q_ϕ^D , is very similar to the full posterior of AIR, cf. Equation (2). The only difference is the conditioning on $\mathbf{z}_t^{\mathcal{P}_t}$, which allows for a different number of discovered objects at each time-step and also for objects explained by PROP not to be explained again. The second term, or q_ϕ^P , describes propagation. The detailed structures of q_ϕ^D and q_ϕ^P are shown in Figure 3, while all the pertinent algorithms and equations can be found in Appendices A and C, respectively.

Learning We train SQAIR as an importance-weighted auto-encoder (IWAE) of Burda et al., 2016. Specifically, we maximise the importance-weighted evidence lower-bound $\mathcal{L}_{\text{IWAE}}$, namely

$$\mathcal{L}_{\text{IWAE}} = \mathbb{E}_{\mathbf{x}_{1:T} \sim p_{\text{data}}(\mathbf{x}_{1:T})} \left[\mathbb{E}_q \left[\log \frac{1}{K} \sum_{k=1}^K \frac{p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T} | \mathbf{x}_{1:T})} \right] \right]. \quad (5)$$

To optimise the above, we use RMSPROP, $K = 5$ and batch size of 32. We use the VIMCO gradient estimator of Mnih and Rezende, 2016 to backpropagate through the discrete latent variables z^{pres} ,

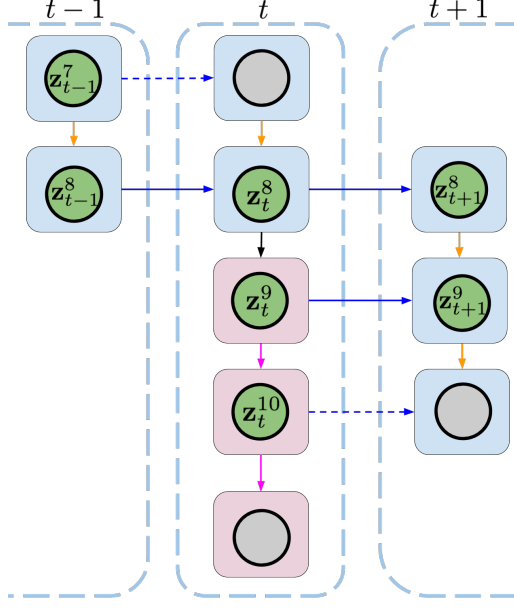
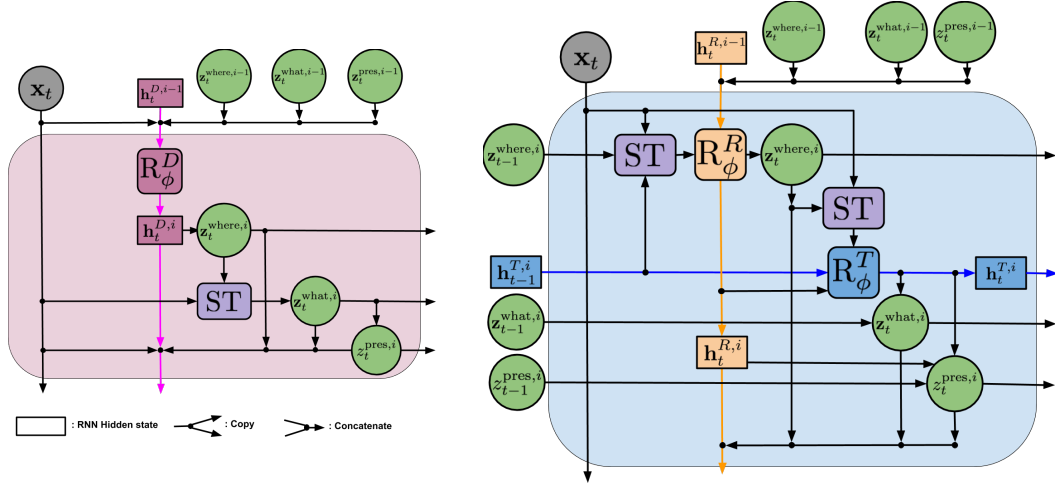


Figure 3: *Left*: Interaction between PROP and DISC in SQAIR. Firstly, objects are propagated to time t , and object $i = 7$ is dropped. Secondly, DISC tries to discover new objects. Here, it manages to find two objects: $i = 9$ and $i = 10$. The process recurs for all remaining time-steps. **Blue arrows** update the temporal hidden state, **orange ones** infer relations between objects, **pink ones** correspond to discovery. *Bottom*: Information flow in a single discovery block (*left*) and propagation block (*right*). In DISC we first predict *where* and extract a glimpse. We then predict *what* and *presence*. PROP starts with extracting a glimpse at a candidate location and updating *where*. Then it follows a procedure similar to DISC, but takes the respective latent variables from the previous time-step into account. It is approximately two times more computationally expensive than DISC. For details, see Algorithms 2 and 3 in Appendix A.



and use reparameterisation for the continuous ones (Kingma and Welling, 2013). We also tried to use NVIL of Mnih and Gregor, 2014 as in the original work on AIR, but found it very sensitive to hyper-parameters, fragile and generally under-performing.

4 Experiments

We evaluate SQAIR on two datasets. Firstly, we perform an extensive evaluation on moving MNIST digits, where we show that it can learn to reliably detect, track and generate moving digits (Section 4.1). Moreover, we show that SQAIR can simulate moving objects into the future — an outcome it has not been trained for. We also study the utility of learned representations for a downstream task. Secondly, we apply SQAIR to real-world pedestrian CCTV data from static cameras (*DukeMTMC*, Ristani et al., 2016), where we perform background subtraction as pre-processing. In this experiment, we show that SQAIR learns to detect, track, predict and generate walking pedestrians without human supervision.

4.1 Moving multi-MNIST

The dataset consists of sequences of length 10 of multiple moving MNIST digits. All images are of size 50×50 and there are zero, one or two digits in every frame (with equal probability). Sequences are generated such that no objects overlap in the first frame, and all objects are present through the sequence; the digits can move out of the frame, but always come back. See Appendix F for an experiment on a harder version of this dataset. There are 60,000 training and 10,000 testing sequences created from the respective MNIST datasets. We train two variants of SQAIR: the MLP-SQAIR uses

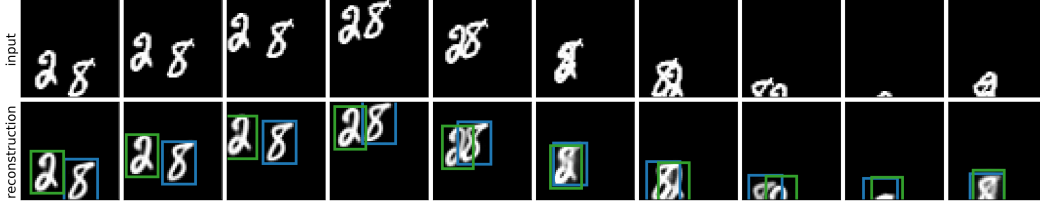


Figure 4: Input images (top) and SQAIR reconstructions with marked glimpse locations (bottom). For more examples, see Figure 13 in Appendix H.

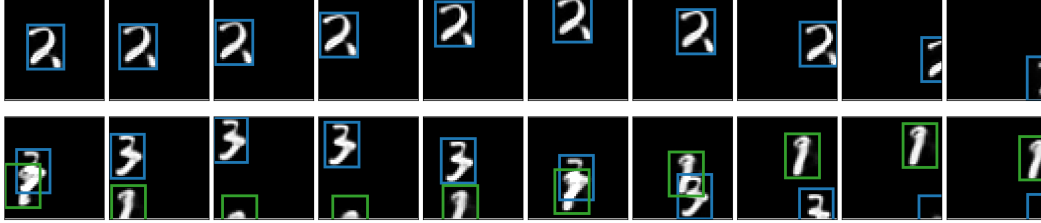


Figure 5: Samples from SQAIR. Both motion and appearance are consistent through time, thanks to the propagation part of the model. For more examples, see Figure 15 in Appendix H.

only fully-connected networks, while the CONV-SQAIR replaces the networks used to encode images and glimpses with convolutional ones; it also uses a subpixel-convolution network as the glimpse decoder (Shi et al., 2016). See Appendix D for details of the model architectures and the training procedure.

We use AIR and VRNN (Chung et al., 2015) as baselines for comparison. VRNN can be thought of as a sequential VAE with an RNN as its deterministic backbone. Being similar to a VAE, its latent variables are not structured, nor easily interpretable. For a fair comparison, we control the latent dimensionality of VRNN and the number of learnable parameters. We provide implementation details in Appendix D.3.

The quantitative analysis consists of comparing all models in terms of the marginal log-likelihood $\log p_\theta(\mathbf{x}_{1:T})$ evaluated as the $\mathcal{L}_{\text{IWAE}}$ bound with $K = 1000$ particles, reconstruction quality evaluated as a single-sample approximation of $\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}_{1:T} | \mathbf{z}_{1:T})]$ and the KL-divergence between the approximate posterior and the prior (Table 1). Additionally, we measure the accuracy of the number of objects modelled by SQAIR and AIR. SQAIR achieves superior performance across a range of metrics — its convolutional variant outperforms both AIR and the corresponding VRNN in terms of model evidence and reconstruction performance. The KL divergence for SQAIR is almost twice as low as for VRNN and by a yet larger factor for AIR. We can interpret KL values as an indicator of the ability to compress, and we can treat SQAIR/AIR type of scheme as a version of run-length encoding. While VRNN has to use information to explicitly describe every part of the image, even if some parts are empty, SQAIR can explicitly allocate content information (\mathbf{z}^{what}) to specific parts of the image (indicated by $\mathbf{z}^{\text{where}}$). AIR exhibits the highest values of KL, but this is due to encoding every frame of the sequence independently — its prior cannot take *what* and *where* at the previous time-step into account, hence higher KL. The fifth column of Table 1 details the object counting accuracy, that is indicative of the quality of the approximate posterior. It is measured as the sum of z_t^{pres} for a given frame against the true number of objects in that frame. As there is no z_t^{pres} for VRNN no score is provided. Perhaps surprisingly, this metric is much higher for SQAIR than for AIR. This is because AIR mistakenly infers overlapping objects as a single object. Since SQAIR can incorporate temporal

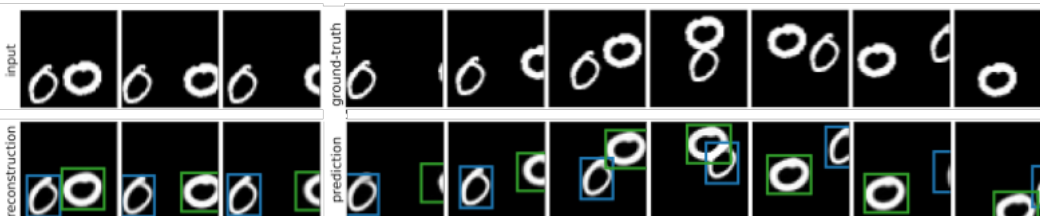


Figure 6: The first three frames are input to SQAIR, which generated the rest conditional on the first frames.

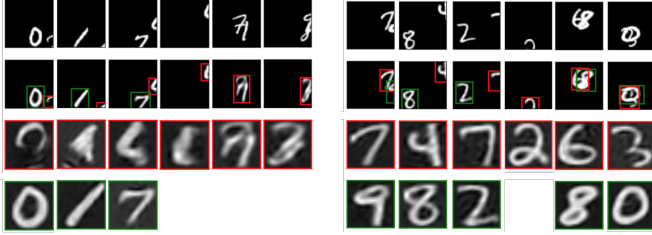


Figure 7: Inputs, reconstructions with marked glimpse locations and reconstructed glimpses for AIR (left) and SQAIR (right). SQAIR can model partially visible and heavily overlapping objects by aggregating temporal information.

	$\log p_\theta(\mathbf{x}_{1:T})$	$\log p_\theta(\mathbf{x}_{1:T} \mathbf{z}_{1:T})$	$\text{KL}(q_\phi p_\theta)$	Counting	Addition
CONV-SQAIR	6784.8	6923.8	134.6	0.9974	0.9990
MLP-SQAIR	6617.6	6786.5	164.5	0.9986	0.9998
MLP-AIR	6443.6	6830.6	352.6	0.9058	0.8644
CONV-VRNN	6561.9	6737.8	270.2	n/a	0.8536
MLP-VRNN	5959.3	6108.7	218.3	n/a	0.8059

Table 1: SQAIR achieves higher performance than the baselines across a range of metrics. The third column refers to the Kullback-Leibler (KL) divergence between the approximate posterior and the prior. Counting refers to accuracy of the inferred number of objects present in the scene, while addition stands for the accuracy of a supervised digit addition experiment, where a classifier is trained on the learned latent representations of each frame.

information, it does not exhibit this failure mode (*cf.* Figure 7). Next, we gauge the utility of the learnt representations by using them to determine the sum of the digits present in the image (Table 1, column six). To do so, we train a 19-way classifier (mapping from any combination of up to two digits in the range $[0, 9]$ to their sum) on the extracted representations and use the summed labels of digits present in the frame as the target. Appendix D contains details of the experiment. SQAIR significantly outperforms AIR and both variants of VRNN on this task. VRNN under-performs due to the inability of disentangling overlapping objects, while both VRNN and AIR suffer from low temporal consistency of learned representations, see Appendix H. Finally, we evaluate SQAIR qualitatively by analyzing reconstructions and samples produced by the model against reconstructions and samples from VRNN. We observe that samples and reconstructions from SQAIR are of better quality and, unlike VRNN, preserve motion and appearance consistently through time. See Appendix H for direct comparison and additional examples. Furthermore, we examine conditional generation, where we look at samples from the generative model of SQAIR conditioned on three images from a real sequence (see Figure 6). We see that the model can preserve appearance over time, and that the simulated objects follow similar trajectories, which hints at good learning of the motion model (see Appendix H for more examples). Figure 7 shows reconstructions and corresponding glimpses of AIR and SQAIR. Unlike SQAIR, AIR is unable to recognize objects from partial observations, nor can it distinguish strongly overlapping objects (it treats them as a single object; columns five and six in the figure). We analyze failure cases of SQAIR in Appendix G.

4.2 Generative Modelling of Walking Pedestrians

To evaluate the model in a more challenging, real-world setting, we turn to data from static CCTV cameras of the *DukeMTMC* dataset (Ristani et al., 2016). As part of pre-processing, we use standard background subtraction algorithms (Itseez, 2015). In this experiment, we use 3150 training and 350 validation sequences of length 5. For details of model architectures, training and data pre-processing, see Appendix E. We evaluate the model qualitatively by examining reconstructions, conditional samples (conditioned on the first four frames) and samples from the prior (Figure 8 and Appendix I). We see that the model learns to reliably detect and track walking pedestrians, even when they are close to each other.

There are some spurious detections and re-detections of the same objects, which is mostly caused by imperfections of the background subtraction pipeline — backgrounds are often noisy and there are sudden appearance changes when a part of a person is treated as background in the pre-processing pipeline. The object counting accuracy in this experiment is 0.5712 on the validation dataset, and we noticed that it does increase with the size of the training set. We also had to use early stopping to prevent overfitting, and the model was trained for only 315k iterations ($> 1\text{M}$ for MNIST experiments). Hence, we conjecture that accuracy and marginal likelihood can be further improved by using a bigger dataset.



Figure 8: Inputs on the top, reconstructions in the second row, samples in the third row; rows four and five contain inputs and conditional generation: the first four frames in the last row are reconstructions, while the remaining ones are predicted by sampling from the prior. There is no ground-truth, since we used sequences of length five of training and validation.

5 Related Work

Object Tracking There have been many approaches to modelling objects in images and videos. Object detection and tracking are typically learned in a supervised manner, where object bounding boxes and often additional labels are part of the training data. Single-object tracking commonly use Siamese networks, which can be seen as an RNN unrolled over two time-steps (Valmadre et al., 2017). Recently, Kosiorrek et al., 2017 used an RNN with an attention mechanism in the HART model to predict bounding boxes for single objects, while robustly modelling their motion and appearance. Multi-object tracking is typically attained by detecting objects and performing data association on bounding-boxes (Bewley et al., 2016). Schuster et al., 2017 used an end-to-end supervised approach that detects objects and performs data association. In the unsupervised setting, where the training data consists of only images or videos, the dominant approach is to distill the inductive bias of spatial consistency into a discriminative model. Cho et al., 2015 detect single objects and their parts in images, and Kwak et al., 2015; Xiao and Jae Lee, 2016 incorporate temporal consistency to better track single objects. SQAIR is unsupervised and hence it does not rely on bounding boxes nor additional labels for training, while being able to learn arbitrary motion and appearance models similarly to HART (Kosiorrek et al., 2017). At the same time, is inherently multi-object and performs data association implicitly (*cf.* Appendix A). Unlike the other unsupervised approaches, temporal consistency is baked into the model structure of SQAIR and further enforced by lower KL divergence when an object is tracked.

Video Prediction Many works on video prediction learn a deterministic model conditioned on the current frame to predict the future ones (Ranzato et al., 2014; Srivastava et al., 2015). Since these models do not model uncertainty in the prediction, they can suffer from the multiple futures problem — since perfect prediction is impossible, the model produces blurry predictions which are a mean of possible outcomes. This is addressed in stochastic latent variable models trained using variational inference to generate multiple plausible videos given a sequence of images (Babaeizadeh et al., 2017; Denton and Fergus, 2018). Unlike SQAIR, these approaches do not model objects or their positions explicitly, thus the representations they learn are of limited interpretability.

Learning Decomposed Representations of Images and Videos Learning decomposed representations of object appearance and position lies at the heart of our model. This problem can be also seen as perceptual grouping, which involves modelling pixels as spatial mixtures of entities. Greff, Rasmus, et al., 2016 and Greff, Steenkiste, et al., 2017 learn to decompose images into separate entities by iterative refinement of spatial clusters using either learned updates or the Expectation Maximization algorithm; Ilin et al., 2017 and Steenkiste et al., 2018 extend these approaches to videos, achieving very similar results to SQAIR. Perhaps the most similar work to ours is the concurrently developed model of Hsieh et al., 2018. The above approaches rely on iterative inference procedures,

but do not exhibit the object-counting behaviour of SQAIR. For this reason, their computational complexities are proportional to the predefined maximum number of objects, while SQAIR can be more computationally efficient by adapting to the number of objects currently present in an image.

Another interesting line of work is the GAN-based unsupervised video generation that decomposes motion and content (Tulyakov et al., 2018; Denton and Birodkar, 2017). These methods learn interpretable features of content and motion, but deal only with single objects and do not explicitly model their locations. Nonetheless, adversarial approaches to learning structured probabilistic models of objects offer a plausible alternative direction of research.

Bayesian Nonparametric Models To the best of our knowledge, Neiswanger and Wood, 2012 is the only known approach that models pixels belonging to a variable number of objects in a video together with their locations in the generative sense. This work uses a Bayesian nonparametric (BNP) model, which relies on mixtures of Dirichlet processes to cluster pixels belonging to an object. However, the choice of the model necessitates complex inference algorithms involving Gibbs sampling and Sequential Monte Carlo, to the extent that any sensible approximation of the marginal likelihood is infeasible. It also uses a fixed likelihood function, while ours is learnable.

The object appearance-persistence-disappearance model in SQAIR is reminiscent of the Markov Indian buffet process (MIBP) of Gael et al., 2009, another BNP model. MIBP was used as a model for blind source separation, where multiple sources contribute toward an audio signal, and can appear, persist, disappear and reappear independently. The prior in SQAIR is similar, but the crucial differences are that SQAIR combines the BNP prior with flexible neural network models for the dynamics and likelihood, as well as variational learning via amortized inference. The interface between deep learning and BNP, and graphical models in general, remains a fertile area of research.

6 Discussion

In this paper we proposed SQAIR, a probabilistic model that extends AIR to image sequences, and thereby achieves temporally consistent reconstructions and samples. In doing so, we enhanced AIR’s capability of disentangling overlapping objects and identifying partially observed objects.

This work continues the thread of Greff, Steenkiste, et al., 2017, Steenkiste et al., 2018 and, together with Hsieh et al., 2018, presents unsupervised object detection & tracking with learnable likelihoods by the means of generative modelling of objects. In particular, our work is the first one to explicitly model object presence, appearance and location through time. Being a generative model, SQAIR can be used for conditional generation, where it can extrapolate sequences into the future. As such, it would be interesting to use it in a reinforcement learning setting in conjunction with Imagination-Augmented Agents (Weber et al., 2017) or more generally as a world model (Ha and Schmidhuber, 2018), especially for settings with simple backgrounds, e. g., games like Montezuma’s Revenge or Pacman.

The framework offers various avenues of further research; SQAIR leads to interpretable representations, but the interpretability of *what* variables can be further enhanced by using alternative objectives that disentangle factors of variation in the objects (Kim and Mnih, 2018). Moreover, in its current state, SQAIR can work only with simple backgrounds and static cameras. In future work, we would like to address this shortcoming, as well as speed up the sequential inference process whose complexity is linear in the number of objects. The generative model, which currently assumes additive image composition, can be further improved by e. g., autoregressive modelling (Oord et al., 2016). It can lead to higher fidelity of the model and improved handling of occluded objects. Finally, the SQAIR model is very complex, and it would be useful to perform a series of ablation studies to further investigate the roles of different components.

Acknowledgements

We would like to thank Ali Eslami for his help in implementing AIR, Alex Bewley and Martin Engelcke for discussions and valuable insights and anonymous reviewers for their constructive feedback. Additionally, we acknowledge that HK and YWT’s research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071.

References

- Babaeizadeh, M., C. Finn, D. Erhan, R. H. Campbell, and S. Levine (2017). “Stochastic Variational Video Prediction”. In: *CoRR*. arXiv: 1710.11252.
- Bewley, A., Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft (2016). “Simple online and realtime tracking”. In: *ICIP*.
- Burda, Y., R. Grosse, and R. Salakhutdinov (2016). “Importance Weighted Autoencoders”. In: *ICLR*. arXiv: 1509.00519.
- Cho, M., S. Kwak, C. Schmid, and J. Ponce (2015). “Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals”. In: *CoRR*. arXiv: 1501.06170.
- Chung, J., K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio (2015). “A Recurrent Latent Variable Model for Sequential Data”. In: *NIPS*. arXiv: 1506.02216.
- Clevert, D.-A., T. Unterthiner, and S. Hochreiter (2015). “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *CoRR*. arXiv: 1511.07289.
- Denton, E. and V. Birodkar (2017). “Unsupervised learning of disentangled representations from video”. In: *NIPS*.
- Denton, E. and R. Fergus (2018). “Stochastic Video Generation with a Learned Prior”. In: *ICML*.
- Eslami, S. M. A., N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K. Kavukcuoglu, and G. E. Hinton (2016). “Attend, Infer, Repeat: Fast Scene Understanding with Generative Models”. In: *NIPS*. arXiv: 1603.08575.
- Gael, J. V., Y. W. Teh, and Z. Ghahramani (2009). “The Infinite Factorial Hidden Markov Model”. In: *NIPS*.
- Graves, A., G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis (2016). “Hybrid computing using a neural network with dynamic external memory”. In: *Nature* 538.7626.
- Greff, K., A. Rasmus, M. Berglund, T. H. Hao, H. Valpola, and J. Schmidhuber (2016). “Tagger: Deep Unsupervised Perceptual Grouping”. In: *NIPS*.
- Greff, K., S. van Steenkiste, and J. Schmidhuber (2017). “Neural Expectation Maximization”. In: *NIPS*.
- Gulrajani, I., K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville (2016). “Pixelvae: A latent variable model for natural images”. In: *CoRR*. arXiv: 1611.05013.
- Ha, D. and J. Schmidhuber (2018). “World Models”. In: *CoRR*. arXiv: 1603.10122.
- Hsieh, J.-T., B. Liu, D.-A. Huang, L. Fei-Fei, and J. C. Niebles (2018). “Learning to Decompose and Disentangle Representations for Video Prediction”. In: *NIPS*.
- Ilin, A., I. Prémont-Schwarz, T. H. Hao, A. Rasmus, R. Boney, and H. Valpola (2017). “Recurrent Ladder Networks”. In: *NIPS*.
- Itseez (2015). *Open Source Computer Vision Library*. <https://github.com/itseez/opencv>.
- Jacobsen, J.-H., J. Van Gemert, Z. Lou, and A. W. M. Smeulders (2016). “Structured Receptive Fields in CNNs”. In: *CVPR*.
- Jaderberg, M., K. Simonyan, A. Zisserman, and K. Kavukcuoglu (2015). “Spatial Transformer Networks”. In: *NIPS*. DOI: 10.1038/nbt.3343. arXiv: 1506.02025v1.
- Kemp, C. and J. B. Tenenbaum (2008). “The discovery of structural form”. In: *Proceedings of the National Academy of Sciences* 105.31.
- Kim, H. and A. Mnih (2018). “Disentangling by factorising”. In: *ICML*. arXiv: 1802.05983.
- Kingma, D. P. and J. Ba (2015). “Adam: A Method for Stochastic Optimization”. In: *ICLR*. arXiv: 1412.6980.
- Kingma, D. P. and M. Welling (2013). “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114*.
- Kosiorek, A. R., A. Bewley, and I. Posner (2017). “Hierarchical Attentive Recurrent Tracking”. In: *NIPS*. arXiv: 1706.09262.
- Kwak, S., M. Cho, I. Laptev, J. Ponce, and C. Schmid (2015). “Unsupervised object discovery and tracking in video collections”. In: *ICCV*. IEEE.
- LeCun, Y., Y. Bengio, and G. Hinton (2015). “Deep learning”. In: *Nature* 521.7553.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4.
- Maddison, C. J., J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh (2017). “Filtering Variational Objectives”. In: *Advances in Neural Information Processing Systems*.

- Mnih, A. and K. Gregor (2014). “Neural Variational Inference and Learning in Belief Networks”. In: *ICML*. arXiv: 1402.0030v2.
- Mnih, A. and D. J. Rezende (2016). “Variational inference for Monte Carlo objectives”. In: *ICML*. arXiv: 1602.06725.
- Neiswanger, W. and F. Wood (2012). “Unsupervised Detection and Tracking of Arbitrary Objects with Dependent Dirichlet Process Mixtures”. In: *CoRR*. arXiv: 1210.3288.
- Oord, A. van den, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu (2016). “Conditional Image Generation with PixelCNN Decoders”. In: *NIPS*. arXiv: 1606.05328.
- Ranzato, M., A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra (2014). “Video (language) modeling: a baseline for generative models of natural videos”. In: *CoRR*. arXiv: 1412.6604.
- Ristani, E., F. Solera, R. Zou, R. Cucchiara, and C. Tomasi (2016). “Performance measures and a data set for multi-target, multi-camera tracking”. In: *ECCV*. Springer.
- Santoro, A., D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap (2017). “A simple neural network module for relational reasoning”. In: *NIPS*. arXiv: 1706.01427.
- Schulter, S., P. Vernaza, W. Choi, and M. K. Chandraker (2017). “Deep Network Flow for Multi-object Tracking”. In: *CVPR*.
- Shi, W., J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang (2016). “Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network”. In: *CVPR*.
- Srivastava, N., E. Mansimov, and R. Salakhudinov (2015). “Unsupervised learning of video representations using lstms”. In: *ICML*.
- Steenkiste, S. van, M. Chang, K. Greff, and J. Schmidhuber (2018). “Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions”. In: *ICLR*.
- Tieleman, T. and G. Hinton (2012). *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning.
- Tulyakov, S., M.-Y. Liu, X. Yang, and J. Kautz (2018). “Mocogan: Decomposing motion and content for video generation”. In: *CVPR*.
- Valmadre, J., L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr (2017). “End-to-end representation learning for Correlation Filter based tracking”. In: *CVPR*. arXiv: 1704.06036.
- Weber, T., S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. (2017). “Imagination-augmented agents for deep reinforcement learning”. In: *NIPS*.
- Xiao, F. and Y. Jae Lee (2016). “Track and segment: An iterative unsupervised approach for video object proposals”. In: *CVPR*.
- Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhudinov, and A. J. Smola (2017). “Deep Sets”. In: *NIPS*.

A Algorithms

Image generation, described by Algorithm 1, is exactly the same for SQAIR and AIR. Algorithms 2 and 3 describe inference in SQAIR. Note that DISC is equivalent to AIR if no latent variables are present in the inputs.

If a function has multiple inputs and if not stated otherwise, all the inputs are concatenated and linearly projected into some fixed-dimensional space, e. g., Lines 9 and 15 in Algorithm 2. Spatial Transformer (ST, e. g., Line 7 in Algorithm 2) has no learnable parameters: it samples a uniform grid of points from an image \mathbf{x} , where the grid is transformed according to parameters $\mathbf{z}^{\text{where}}$. \mathbf{f}_ϕ^1 is implemented as a perceptron with a single hidden layer. Statistics of q^P and q^D are a result of applying a two-layer multilayer perceptron (MLP) to their respective conditioning sets. Different distributions q do not share parameters of their MLPs. The *glimpse encoder* $\mathbf{h}_\phi^{\text{glimpse}}$ (Lines 8 and 12 in Algorithm 2 and Line 12 in Algorithm 3; they share parameters) and the *image encoder* $\mathbf{h}_\phi^{\text{enc}}$ (Line 3 in Algorithm 3) are implemented as two-layer MLPs or convolutional neural networks (CNNs), depending on the experiment (see Appendices D and E for details).

One of the important details of PROP is the proposal glimpse extracted in lines Lines 6 and 7 of Algorithm 2. It has a dual purpose. Firstly, it acts as an information bottleneck in PROP, limiting the flow of information from the current observation \mathbf{x}_t to the updated latent variables \mathbf{z}_t . Secondly, even though the information is limited, it can still provide a high-resolution view of the object corresponding to the currently updated latent variable, *given* that the location of the proposal glimpse correctly predicts motion of this object. Initially, our implementation used encoding of the raw observation ($\mathbf{h}_\phi^{\text{enc}}(\mathbf{x}_t)$, similarly to Line 3 in Algorithm 3) as an input to the relation-RNN (Line 9 in Algorithm 2). We have also experimented with other bottlenecks: (1) low resolution image as an input to the image encoder and (2) a low-dimensional projection of the image encoding before the relation-RNN. Both approaches have led to *ID swaps*, where the order of explaining objects were sometimes swapped for different frames of the sequence (see Figure 10 in Appendix G for an example). Using encoded proposal glimpse extracted from a predicted location has solved this issue.

To condition DISC on propagated latent variables (Line 4 in Algorithm 3), we encode the latter by using a two-layer MLP similarly to Zaheer et al., 2017,

$$\mathbf{l}_t = \sum_{i \in \mathcal{P}_t} \text{MLP} \left(\mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i} \right). \quad (6)$$

Note that other encoding schemes are possible, though we have experimented only with this one.

Algorithm 1: Image Generation

Input : $\mathbf{z}_t^{\text{what}}, \mathbf{z}_t^{\text{where}}$ - latent variables from the current time-step.

- 1 $\mathcal{O}_t = \text{indices}(\mathbf{z}_t^{\text{what}})$ // Indices of all present latent variables.
- 2 $\mathbf{y}_t^0 = \mathbf{0}$
- 3 **for** $i \in \mathcal{O}_t$ **do**
- 4 $\mathbf{y}_t^{\text{att},i} = \mathbf{f}_\theta^{\text{dec}}(\mathbf{z}_t^{\text{what},i})$ // Decode the glimpse.
- 5 $\mathbf{y}_t^i = \mathbf{y}_t^{i-1} + \text{ST}^{-1}(\mathbf{y}_t^{\text{att},i}, \mathbf{z}_t^{\text{where},i})$
- 6 $\hat{\mathbf{x}}_t \sim \mathcal{N}(\mathbf{x} \mid \mathbf{y}_t, \sigma_x^2 \mathbf{I})$

Output : $\hat{\mathbf{x}}$

Algorithm 2: Inference for Propagation

Input : x_t - image at the current time-step,
 $z_{t-1}^{\text{what}}, z_{t-1}^{\text{where}}, z_{t-1}^{\text{pres}}$ - latent variables from the previous time-step
 h_{t-1}^T - hidden states from the previous time-step.

- 1 $h_t^{R,0}, z_t^{\text{what},0}, z_t^{\text{where},0} = \text{initialize}()$
- 2 $j = 0$ // Index of the object processed in the last iteration.
- 3 **for** $i \in \mathcal{O}_{t-1}$ **do**
- 4 **if** $z_{t-1}^{\text{pres},i} == 0$ **then**
- 5 **continue**
- 6 $\hat{z}_t^{\text{where},i} = f_\phi^1(z_{t-1}^{\text{where},i}, h_t^{T,i})$ // Proposal location.
- 7 $\hat{g}_t^i = \text{ST}(x_t, \hat{z}_t^{\text{where},i})$ // Extract a glimpse from a proposal location.
- 8 $\hat{e}_t^i = h_\phi^{\text{glimpse}}(\hat{g}_t^i)$ // Encode the proposal glimpse.
- 9 $w_t^{R,i}, h_t^{R,i} = R_\phi^R(\hat{e}_t^i, z_{t-1}^{\text{what},i}, z_{t-1}^{\text{where},i}, h_{t-1}^{T,i}, h_t^{R,j}, z_t^{\text{what},j}, z_t^{\text{where},j})$ // Relational state, see Equation (14).
- 10 $z_t^{\text{where},i} \sim q_\phi^P(z_t^{\text{where}} | z_{t-1}^{\text{where},k}, w_t^{R,i})$
- 11 $g_t^i = \text{ST}(x_t, z_t^{\text{where},i})$ // Extract the final glimpse.
- 12 $e_t^i = h_\phi^{\text{glimpse}}(g_t^i)$ // Encode the final glimpse.
- 13 $w_t^{T,i}, h_t^{T,i} = R_\phi^T(e_t^i, z_t^{\text{where},i}, h_{t-1}^{T,i}, h_t^{R,i})$ // Temporal state, see Equation (15).
- 14 $z_t^{\text{what},i} \sim q_\phi^P(z_t^{\text{what}} | e_t^i, z_{t-1}^{\text{what},i}, w_t^{R,i}, w_t^{T,i})$
- 15 $z_t^{\text{pres},i} \sim q_\phi^P(z_t^{\text{pres}} | z_{t-1}^{\text{pres},i}, z_t^{\text{what},i}, z_t^{\text{where},i}, w_t^{R,i}, w_t^{T,i})$ // Equation (13).
- 16 $j = i$

Output : $z_t^{\text{what},\mathcal{P}_t}, z_t^{\text{where},\mathcal{P}_t}, z_t^{\text{pres},\mathcal{P}_t}$

Algorithm 3: Inference for Discovery

Input : x_t - image at the current time-step,
 $z_t^{\mathcal{P}_t}$ - propagated latent variables for the current time-step,
 N - maximum number of inference steps for discovery.

- 1 $h_t^{D,0}, z_t^{\text{what},0}, z_t^{\text{where},0} = \text{initialize}()$
- 2 $j = \max_index(z_t^{\mathcal{P}_t})$ // Maximum index among the propagated latent variables.
- 3 $e_t = h_\phi^{\text{enc}}(x_t)$ // Encode the image.
- 4 $l_t = h_\phi^{\text{enc}}(z_t^{\text{what}}, z_t^{\text{where}}, z_t^{\text{pres}})$ // Encode latent variables.
- 5 **for** $i \in [j+1, \dots, j+N]$ **do**
- 6 $w_t^{D,i}, h_t^{D,i} = R_\phi^D(e_t, l_t, z_t^{\text{what},i-1}, z_t^{\text{where},i-1}, h_t^{D,i-1})$
- 7 $z_t^{\text{pres},i} \sim q_\phi^D(z_t^{\text{pres}} | w_t^{D,i})$
- 8 **if** $z_t^{\text{pres},i} = 0$ **then**
- 9 **break**
- 10 $z_t^{\text{where},i} \sim q_\phi^D(z_t^{\text{where}} | w_t^{D,i})$
- 11 $g_t^i = \text{ST}(x_t, z_t^{\text{where},i})$
- 12 $e_t^i = h_\phi^{\text{glimpse}}(g_t^i)$ // Encode the glimpse.
- 13 $z_t^{\text{what},i} \sim q_\phi^D(z_t^{\text{what}} | e_t^i)$

Output : $z_t^{\text{what},\mathcal{D}_t}, z_t^{\text{where},\mathcal{D}_t}, z_t^{\text{pres},\mathcal{D}_t}$

B Details for the Generative Model of SQAIR

In implementation, we upper bound the number of objects at any given time by N . In detail, the discovery prior is given by

$$p^D(D_t, \mathbf{z}_t^D | \mathbf{z}_t^P) = p^D(D_t | P_t) \prod_{i \in \mathcal{D}_t} p^D(\mathbf{z}_t^{\text{what},i}) p^D(\mathbf{z}_t^{\text{where},i}) \delta_1(z_t^{\text{pres},i}), \quad (7)$$

$$p^D(D_t | P_t) = \text{Categorical}(D_t; N - P_t, p_\theta(P_t)), \quad (8)$$

where $\delta_x(\cdot)$ is the delta function at x , $\text{Categorical}(k; K, p)$ implies $k \in \{0, 1, \dots, K\}$ with probabilities p_0, p_1, \dots, p_K and $p^D(\mathbf{z}_t^{\text{what},i}), p^D(\mathbf{z}_t^{\text{where},i})$ are fixed isotropic Gaussians. The propagation prior is given by

$$p^P(\mathbf{z}_t^P | \mathbf{z}_{t-1}) = \prod_{i \in \mathcal{P}_t} p^P(\mathbf{z}_t^{\text{pres},i} | \mathbf{z}_{t-1}^{\text{pres},i}, \mathbf{h}_{t-1}) p^P(\mathbf{z}_t^{\text{what},i} | \mathbf{h}_{t-1}) p^P(\mathbf{z}_t^{\text{where},i} | \mathbf{h}_{t-1}), \quad (9)$$

$$p^P(\mathbf{z}_t^{\text{pres},i} | \mathbf{z}_{t-1}^{\text{pres},i}, \mathbf{h}_{t-1}) = \text{Bernoulli}(z_t^{\text{pres},i}; f_\theta(\mathbf{h}_{t-1})) \delta_1(z_{t-1}^{\text{pres},i}), \quad (10)$$

with f_θ a scalar-valued function with range $[0, 1]$ and $p^P(\mathbf{z}_t^{\text{what},i} | \mathbf{h}_{t-1}), p^P(\mathbf{z}_t^{\text{where},i} | \mathbf{h}_{t-1})$ both factorised Gaussians parameterised by some function of \mathbf{h}_{t-1} .

C Details for the Inference of SQAIR

The propagation inference network q_ϕ^P is given as below,

$$q_\phi^P(\mathbf{z}_t^P | \mathbf{x}_t, \mathbf{z}_{t-1}, \mathbf{h}_t^{T,P_t}) = \prod_{i \in \mathcal{O}_{t-1}} q_\phi^P(\mathbf{z}_t^i | \mathbf{x}_t, \mathbf{z}_{t-1}^i, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}), \quad (11)$$

with $\mathbf{h}_t^{R,i}$ the hidden state of the relation RNN (see Equation (14)). Its role is to capture information from the observation \mathbf{x}_t as well as to model dependencies between different objects. The propagation posterior for a single object can be expanded as follows,

$$\begin{aligned} q_\phi^P(\mathbf{z}_t^i | \mathbf{x}_t, \mathbf{z}_{t-1}^i, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}) = & \\ & q_\phi^P(\mathbf{z}_t^{\text{where},i} | \mathbf{z}_{t-1}^{\text{what},i}, \mathbf{z}_{t-1}^{\text{where},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}) \\ & q_\phi^P(\mathbf{z}_t^{\text{what},i} | \mathbf{x}_t, \mathbf{z}_t^{\text{where},i}, \mathbf{z}_{t-1}^{\text{what},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}) \\ & q_\phi^P(z_t^{\text{pres},i} | \mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, z_{t-1}^{\text{pres},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}). \end{aligned} \quad (12)$$

In the second line, we condition the object location $\mathbf{z}_t^{\text{where},i}$ on its previous appearance and location as well as its dynamics and relation with other objects. In the third line, current appearance $\mathbf{z}_t^{\text{what},i}$ is conditioned on the new location. Both $\mathbf{z}_t^{\text{where},i}$ and $\mathbf{z}_t^{\text{what},i}$ are modelled as factorised Gaussians. Finally, presence depends on the new appearance and location as well as the presence of the same object at the previous time-step. More specifically,

$$\begin{aligned} q_\phi^P(z_t^{\text{pres},i} | \mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, z_{t-1}^{\text{pres},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}) \\ = \text{Bernoulli}(z_t^{\text{pres},i} | f_\phi(\mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i})) \delta_1(z_{t-1}^{\text{pres},i}), \end{aligned} \quad (13)$$

where the second term is the delta distribution centered on the presence of this object at the previous time-step. If it was not there, it cannot be propagated. Let $j \in \{0, \dots, i-1\}$ be the index of the most recent present object before object i . Hidden states are updated as follows,

$$\mathbf{h}_t^{R,i} = \mathbf{R}_\phi^R(\mathbf{x}_t, \mathbf{z}_{t-1}^{\text{what},i}, \mathbf{z}_{t-1}^{\text{where},i}, \mathbf{h}_{t-1}^{T,i}, \mathbf{h}_{t-1}^{R,i-1}, \mathbf{z}_t^{\text{what},j}, \mathbf{z}_t^{\text{where},j}), \quad (14)$$

$$\mathbf{h}_t^{T,i} = \mathbf{R}_\phi^T(\mathbf{x}_t, \mathbf{z}_t^{\text{where},i}, \mathbf{h}_{t-1}^{T,i}, \mathbf{h}_{t-1}^{R,i}), \quad (15)$$

where \mathbf{R}_ϕ^T and \mathbf{R}_ϕ^R are temporal and propagation RNNs, respectively. Note that in Eq. (14) the RNN does not have direct access to the image \mathbf{x}_t , but rather accesses it by extracting an attention glimpse at a proposal location, predicted from $\mathbf{h}_{t-1}^{T,i}$ and $\mathbf{z}_{t-1}^{\text{where},i}$. This might seem like a minor detail, but in practice structuring computation this way prevents ID swaps from occurring, cf. Appendix G. For computational details, please see Algorithms 2 and 3 in Appendix A.

D Details of the moving-MNIST Experiments

D.1 SQAIR and AIR Training Details

All models are trained by maximising the evidence lower bound (ELBO) \mathcal{L}_{IWAE} (Equation (5)) with the RMSPROP optimizer (Tieleman and Hinton, 2012) with momentum equal to 0.9. We use the learning rate of 10^{-5} and decrease it to $\frac{1}{3} \cdot 10^{-5}$ after 400k and to 10^{-6} after 1000k training iterations. Models are trained for the maximum of $2 \cdot 10^6$ training iterations; we apply early stopping in case of overfitting. SQAIR models are trained with a curriculum of sequences of increasing length: we start with three time-steps, and increase by one time-step every 10^5 training steps until reaching the maximum length of 10. When training AIR, we treated all time-steps of a sequence as independent, and we trained it on all data (sequences of length ten, split into ten independent sequences of length one).

D.2 SQAIR and AIR Model Architectures

All models use glimpse size of 20×20 and exponential linear unit (ELU) (Clevert et al., 2015) non-linearities for all layers except RNNs and output layers. MLP-SQAIR uses fully-connected layers for all networks. In both variants of SQAIR, the \mathbf{R}_ϕ^D and \mathbf{R}_ϕ^R RNNs are the vanilla RNNs. The propagation prior RNN and the temporal RNN \mathbf{R}_ϕ^T use gated recurrent unit (GRU). AIR follows the same architecture as MLP-SQAIR. All fully-connected layers and RNNs in MLP-SQAIR and AIR have 256 units; they have 2.9M and 1.7M trainable parameters, respectively.

CONV-SQAIR differs from the MLP version in that it uses CNNs for the glimpse and image encoders and a subpixel-CNN (Shi et al., 2016) for the glimpse decoder. All fully connected layers and RNNs have 128 units. The encoders share the CNN, which is followed by a single fully-connected layer (different for each encoder). The CNN has four convolutional layers with $[16, 32, 32, 64]$ features maps and strides of $[2, 2, 1, 1]$. The glimpse decoder is composed of two fully-connected layers with $[256, 800]$ hidden units, whose outputs are reshaped into 32 features maps of size 5×5 , followed by a subpixel-CNN with three layers of $[32, 64, 64]$ feature maps and strides of $[1, 2, 2]$. All filters are of size 3×3 . CONV-SQAIR has 2.6M trainable parameters.

We have experimented with different sizes of fully-connected layers and RNNs; we kept the size of all layers the same and altered it in increments of 32 units. Values greater than 256 for MLP-SQAIR and 128 for CONV-SQAIR resulted in overfitting. Models with as few as 32 units per layer (< 0.9 M trainable parameters for MLP-SQAIR) displayed the same qualitative behaviour as reported models, but showed lower quantitative performance.

The output likelihood used in both SQAIR and AIR is Gaussian with a fixed standard deviation set to 0.3, as used by Eslami et al., 2016. We tried using a learnable scalar standard deviation, but decided not to report it due to unstable behaviour in the early stages of training. Typically, standard deviation would converge to a low value early in training, which leads to high penalties for reconstruction mistakes. In this regime, it is beneficial for the model to perform no inference steps (z^{pres} is always equal to zero), and the model never learns. Fixing standard deviation for the first 10k iterations and then learning it solves this issue, but it introduces unnecessary complexity into the training procedure.

D.3 VRNN Implementation and Training Details

Our VRNN implementation is based on the implementation⁴ of Filtering Variational Objectives (FIVO) by Maddison et al., 2017. We use an LSTM with hidden size J for the deterministic backbone of the VRNN. At time t , the LSTM receives $\psi^x(\mathbf{x}_{t-1})$ and $\psi^z(\mathbf{z}_{t-1})$ as input and outputs o_t , where ψ^x is a data feature extractor and ψ^z is a latent feature extractor. The output is mapped to the mean and standard deviation of the Gaussian prior $p_\theta(\mathbf{z}_t | \mathbf{x}_{t-1})$ by an MLP. The likelihood $p_\theta(\mathbf{x}_t | \mathbf{z}_t, \mathbf{x}_{t-1})$ is a Gaussian, with mean given by $\psi^{\text{dec}}(\psi^z(\mathbf{z}_t), o_t)$ and standard deviation fixed to be 0.3 as for SQAIR and AIR. The inference network $q_\phi(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_t)$ is a Gaussian with mean and standard deviation given by the output of separate MLPs with inputs $[o_t, \psi^x(\mathbf{x}_t)]$.

All aforementioned MLPs use the same number of hidden units H and the same number of hidden layers L . The CONV-VRNN uses a CNN for ψ^x and a transposed CNN for ψ^{dec} . The MLP-VRNN uses an MLP with H' hidden units and L' hidden layers for both. ELU were used throughout as activations. The latent dimensionality was fixed to 165, which is the upper bound of the number of

⁴<https://github.com/tensorflow/models/tree/master/research/fivo>

Table 2: Number of trainable parameters for the reported models.

	CONV-SQAIR	MLP-SQAIR	MLP-AIR	CONV-VRNN	MLP-VRNN
number of parameters	2.6M	2.9M	1.7M	2.6M	2.1M

latent dimensions that can be used per time-step in SQAIR or AIR. Training was done by optimising the FIVO bound, which is known to be tighter than the IWAE bound for sequential latent variable models (Maddison et al., 2017). We also verified that this was the case with our models on the moving-MNIST data. We train with the RMSPROP optimizer with a learning rate of 10^{-5} , momentum equal to 0.9, and training until convergence of test FIVO bound.

For each of MLP-VRNN and CONV-VRNN, we experimented with three architectures: small/medium/large. We used $H=H'=J=128/256/512$ and $L=L'=2/3/4$ for MLP-VRNN, giving number of parameters of 1.2M/2.1M/9.8M. For CONV-VRNN, the number of features maps we used was $[32, 32, 64, 64]$, $[32, 32, 32, 64, 64, 64]$ and $[32, 32, 32, 64, 64, 64, 64, 64]$, with strides of $[2, 2, 2, 2]$, $[1, 2, 1, 2, 1, 2]$ and $[1, 2, 1, 2, 1, 2, 1, 1]$, all with 3×3 filters, $H=J=128/256/512$ and $L=1$, giving number of parameters of 0.8M/2.6M/6.1M. The largest convolutional encoder architecture is very similar to that in Gulrajani et al., 2016 applied to MNIST.

We have chosen the medium-sized models for comparison with SQAIR due to overfitting encountered in larger models.

D.4 Addition Experiment

We perform the addition experiment by feeding latent representations extracted from the considered models into a 19-way classifier, as there are 19 possible outputs (addition of two digits between 0 and 9). The classifier is implemented as an MLP with two hidden layers with 256 ELU units each and a softmax output. For AIR and SQAIR, we use concatenated z^{what} variables multiplied by the corresponding z^{pres} variables, while for VRNN we use the whole 165-dimensional latent vector. We train the model over 10^7 training iterations with the ADAM optimizer (Kingma and Ba, 2015) with default parameters (in tensorflow).

E Details of the *DukeMTMC* Experiments

We take videos from cameras one, two, five, six and eight from the *DukeMTMC* dataset (Ristani et al., 2016). As pre-processing, we invert colors and subtract backgrounds using standard OpenCV tools (Itseez, 2015), downsample to the resolution of 240×175 , convert to gray-scale and randomly crop fragments of size 64×64 . Finally, we generate 3500 sequences of length five such that the maximum number of objects present in any single frame is three and we split them into training and validation sets with the ratio of 9 : 1.

We use the same training procedure as for the MNIST experiments. The only exception is the learning curriculum, which goes from three to five time-steps, since this is the maximum length of the sequences.

The reported model is similar to CONV-SQAIR. We set the glimpse size to 28×12 to account for the expected aspect ratio of pedestrians. Glimpse and image encoders share a CNN with $[16, 32, 64, 64]$ feature maps and strides of $[2, 2, 2, 1]$ followed by a fully-connected layer (different for each encoder). The glimpse decoder is implemented as a two-layer fully-connected network with 128 and 1344 units, whose outputs are reshaped into 64 feature maps of size 7×3 , followed by a subpixel-CNN with two layers of $[64, 64]$ feature maps and strides of $[2, 2]$. All remaining fully-connected layers in the model have 128 units. The total number of trainable parameters is 3.5M.

F Harder multi-MNIST Experiment

We created a version of the multi-MNIST dataset, where objects can appear or disappear at an arbitrary point in time. It differs from the dataset described in Section 4.1, where all digits are present throughout the sequence. All other dataset parameters are the same as in Section 4.1. Figure 9 shows an example sequence and MLP-SQAIR reconstructions with marked glimpse locations. The model has no trouble detecting new digits in the middle of the sequence and rediscovering a digit that was previously present.

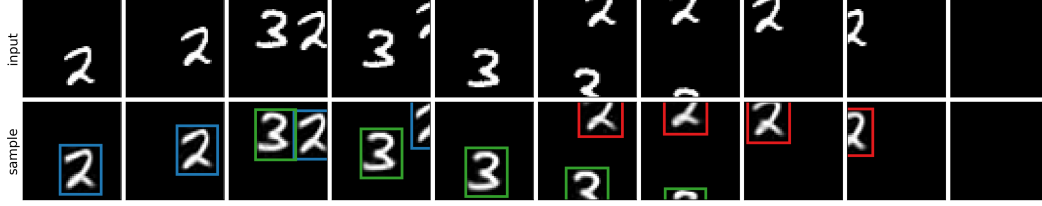


Figure 9: SQAIR trained on a harder version of moving-textscmnist. Input images (top) and SQAIR reconstructions with marked glimpse locations (bottom)

G Failure cases of SQAIR

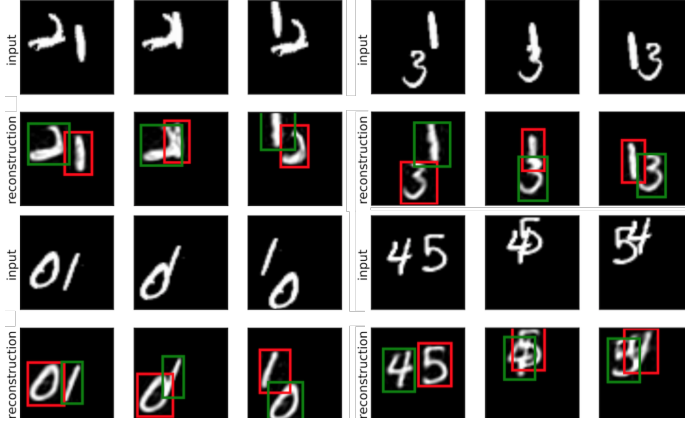


Figure 10: Examples of ID swaps in a version of SQAIR *without* proposal glimpse extraction in PROP (see Appendix A for details). Bounding box colours correspond to object index (or its identity). When PROP is allowed the same access to the image as DISC, then it often prefers to ignore latent variables, which leads to swapped inference order.

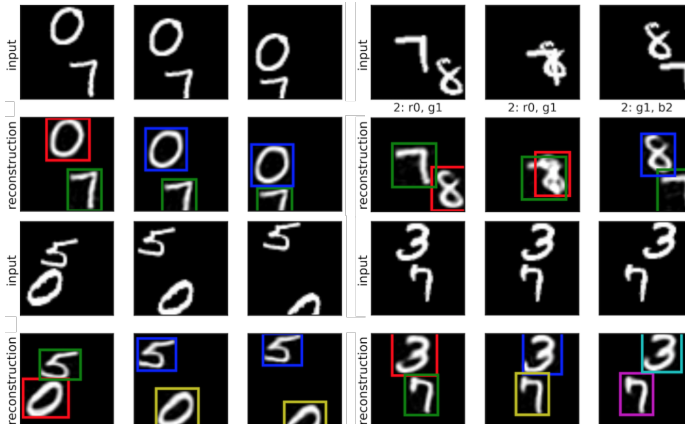


Figure 11: Examples of re-detections in MLP-SQAIR. Bounding box colours correspond to object identity, assigned to it upon discovery. In some training runs, SQAIR converges to a solution, where objects are re-detected in the second frame, and PROP starts tracking only in the third frame (left). Occasionally, an object can be re-detected after it has severely overlapped with another one (top right). Sometimes the model decides to use only DISC and repeatedly discovers all objects (bottom right). These failure mode seem to be mutually exclusive – they come from different training runs.



Figure 12: Two failed reconstructions of SQAIR. *Left*: SQAIR re-detects objects in the second time-step. Instead of 5 and 2, however, it reconstructs them as 6 and 7. Interestingly, reconstructions are consistent through the rest of the sequence. *Right*: At the second time-step, overlapping 6 and 8 are explained as 6 and a small 0. The model realizes its mistake in the third time-step, re-detects both digits and reconstructs them properly.

H Reconstruction and Samples from the Moving-MNIST Dataset

H.1 Reconstructions



Figure 13: Sequences of input (first row) and SQAIR reconstructions with marked glimpse locations. Reconstructions are all temporally consistent.

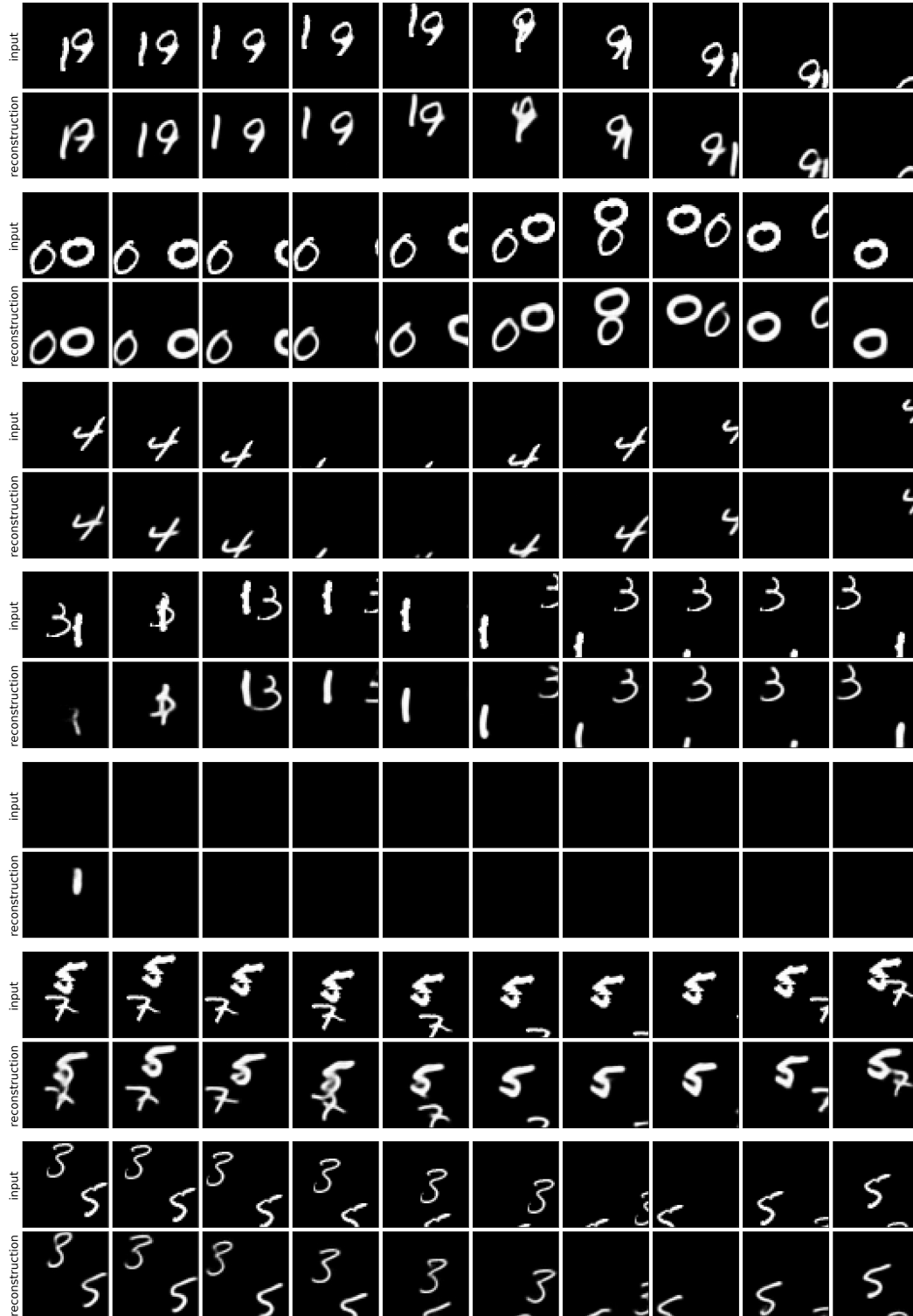


Figure 14: Sequences of input (first row) and CONV-VRNN reconstructions. They are not temporally consistent. The reconstruction at time $t = 1$ is typically of lower quality and often different than the rest of the sequence.

H.2 Samples

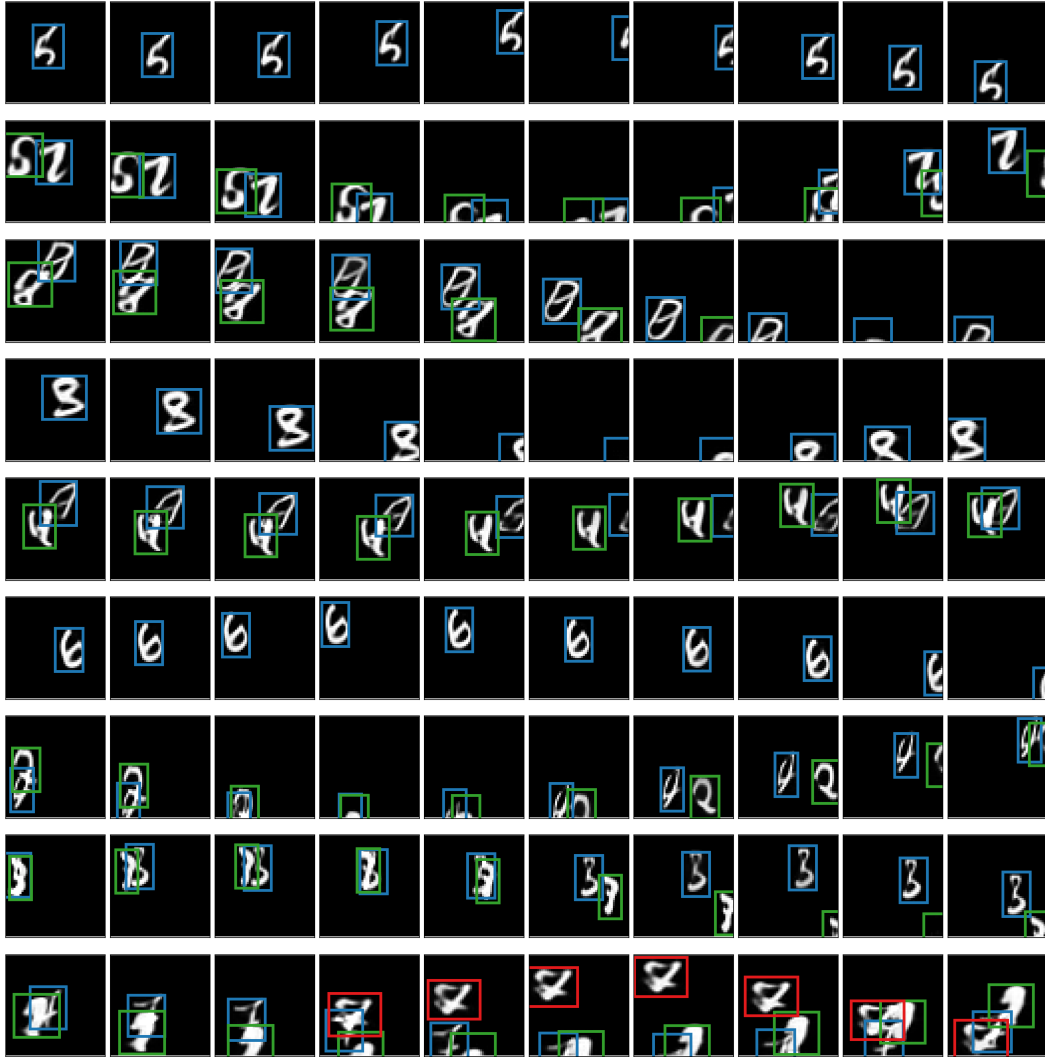


Figure 15: Samples from SQAIR. Both motion and appearance are temporally consistent. In the last sample, the model introduces the third object despite the fact that it has seen only up to two objects in training.

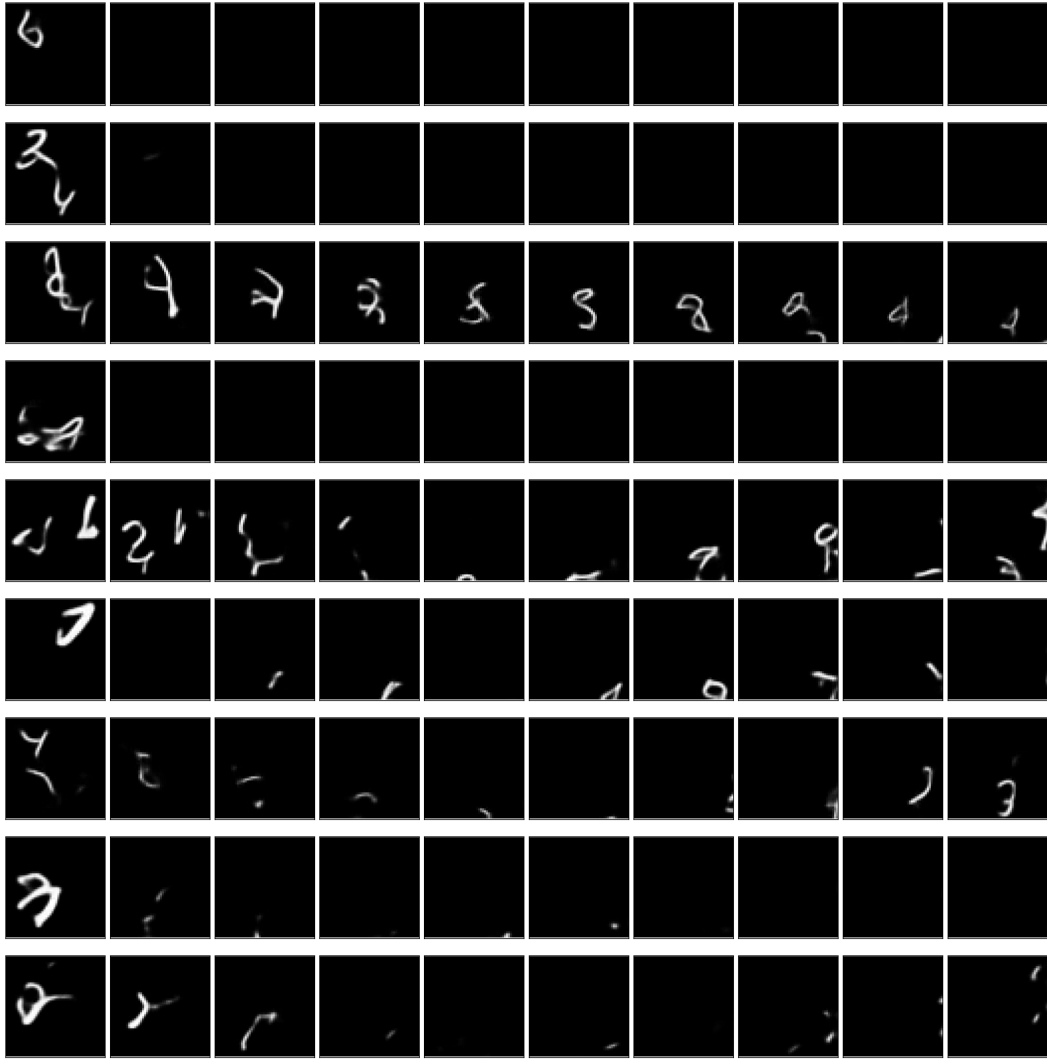


Figure 16: Samples from CONV-VRNN. They show lack of temporal consistency. Objects in the generated frames change between consecutive time-steps and they do not resamble digits from the training set.

H.3 Conditional Generation

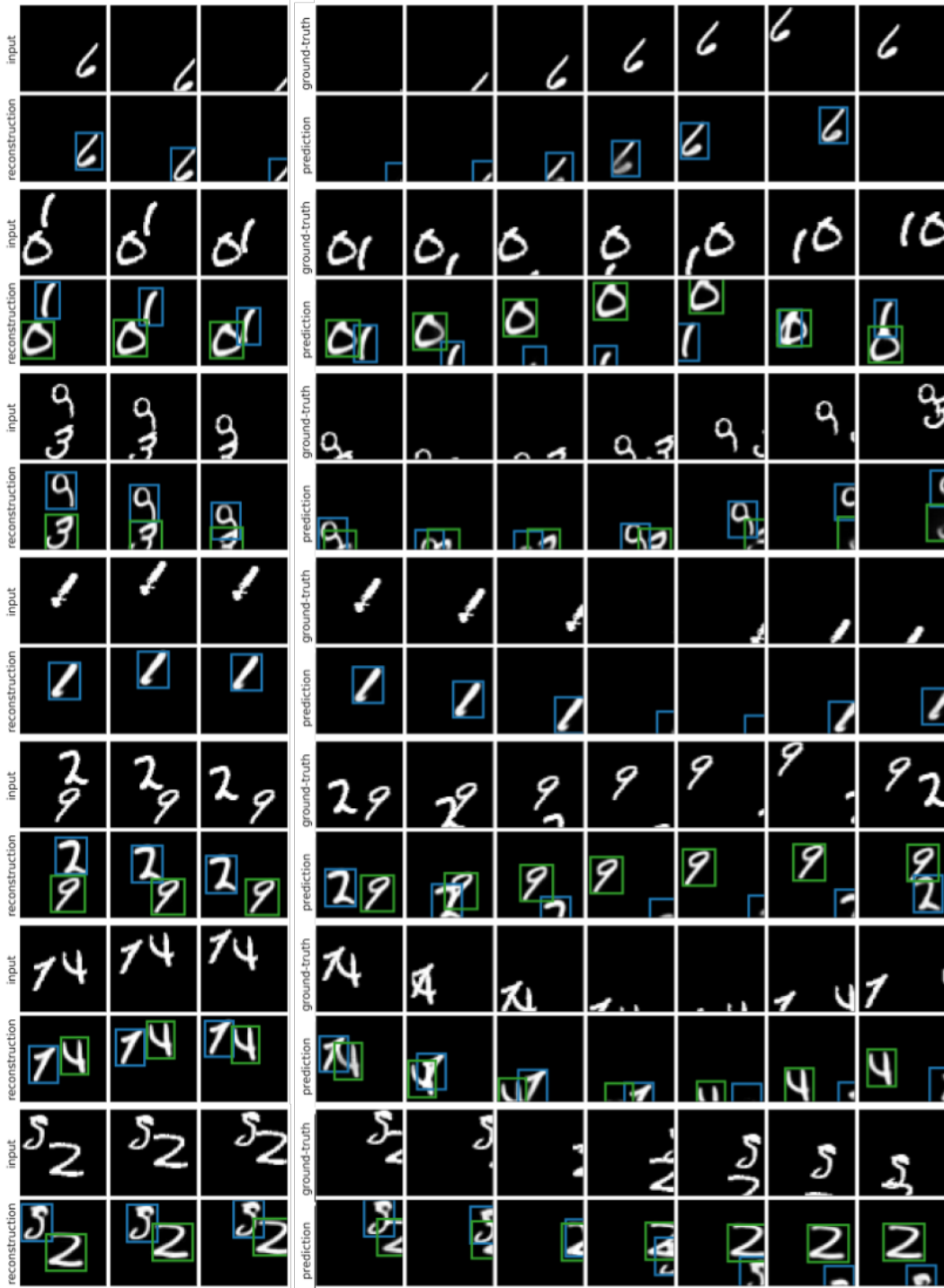


Figure 17: Conditional generation from SQAIR, which sees only the first three frames in every case. Top is the input sequence (and the remaining ground-truth), while bottom is reconstruction (first three time-steps) and then generation.

I Reconstruction and Samples from the DukeMTMC Dataset

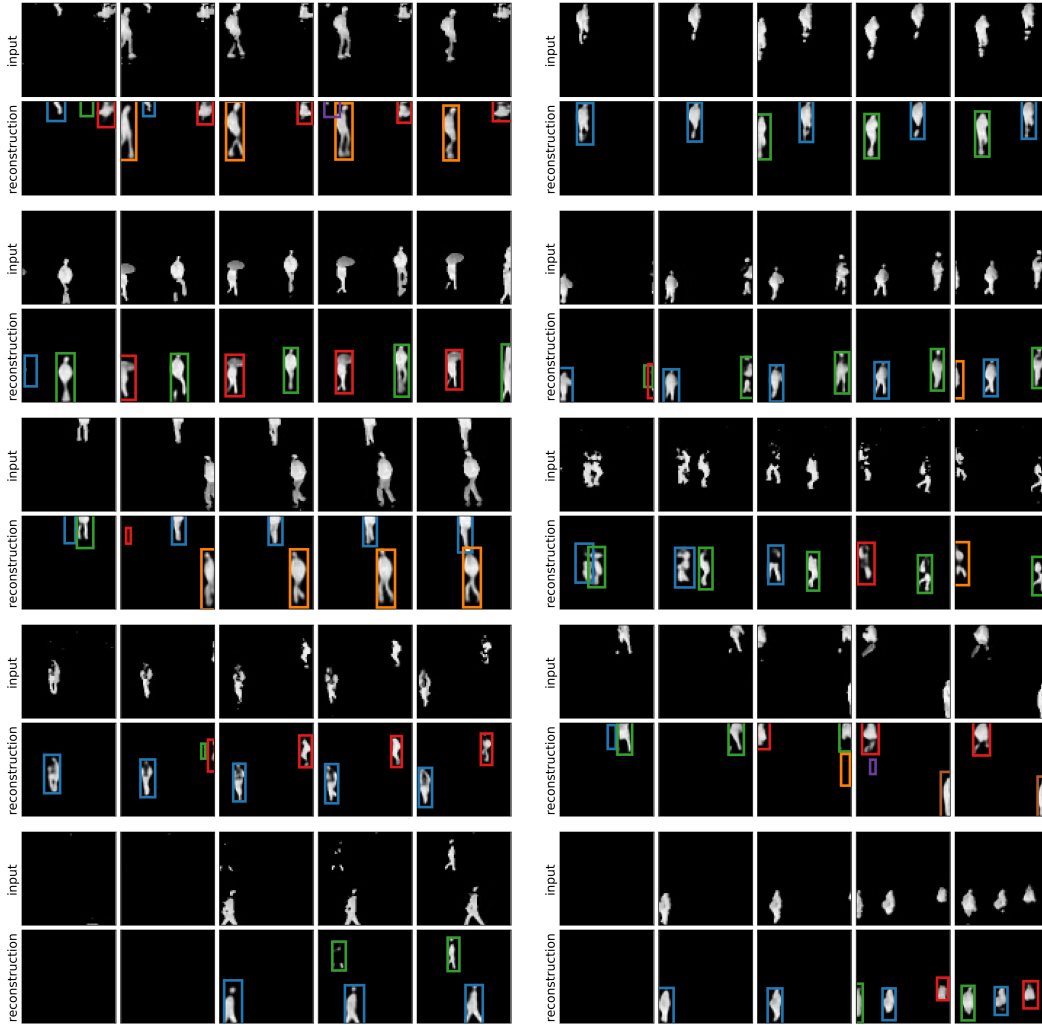


Figure 18: Sequences of input (first row) and SQAIR reconstructions with marked glimpse locations. While not perfect (spurious detections, missed objects), they are temporally consistent and similar in appearance to the inputs.

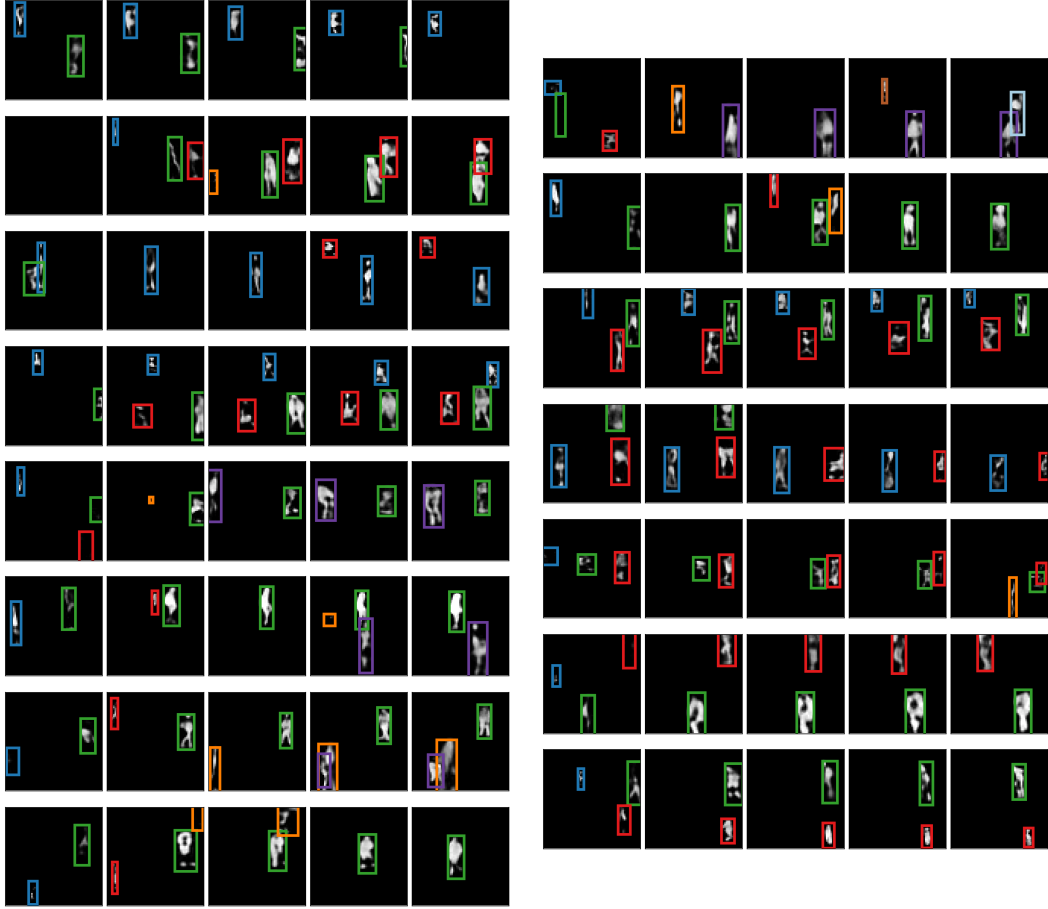


Figure 19: Samples with marked glimpse locations from SQAIR trained on the DukeMTMC dataset. Both appearance and motion is spatially consistent. Generated objects are similar in appearance to pedestrians in the training data. Samples are noisy, but so is the dataset.

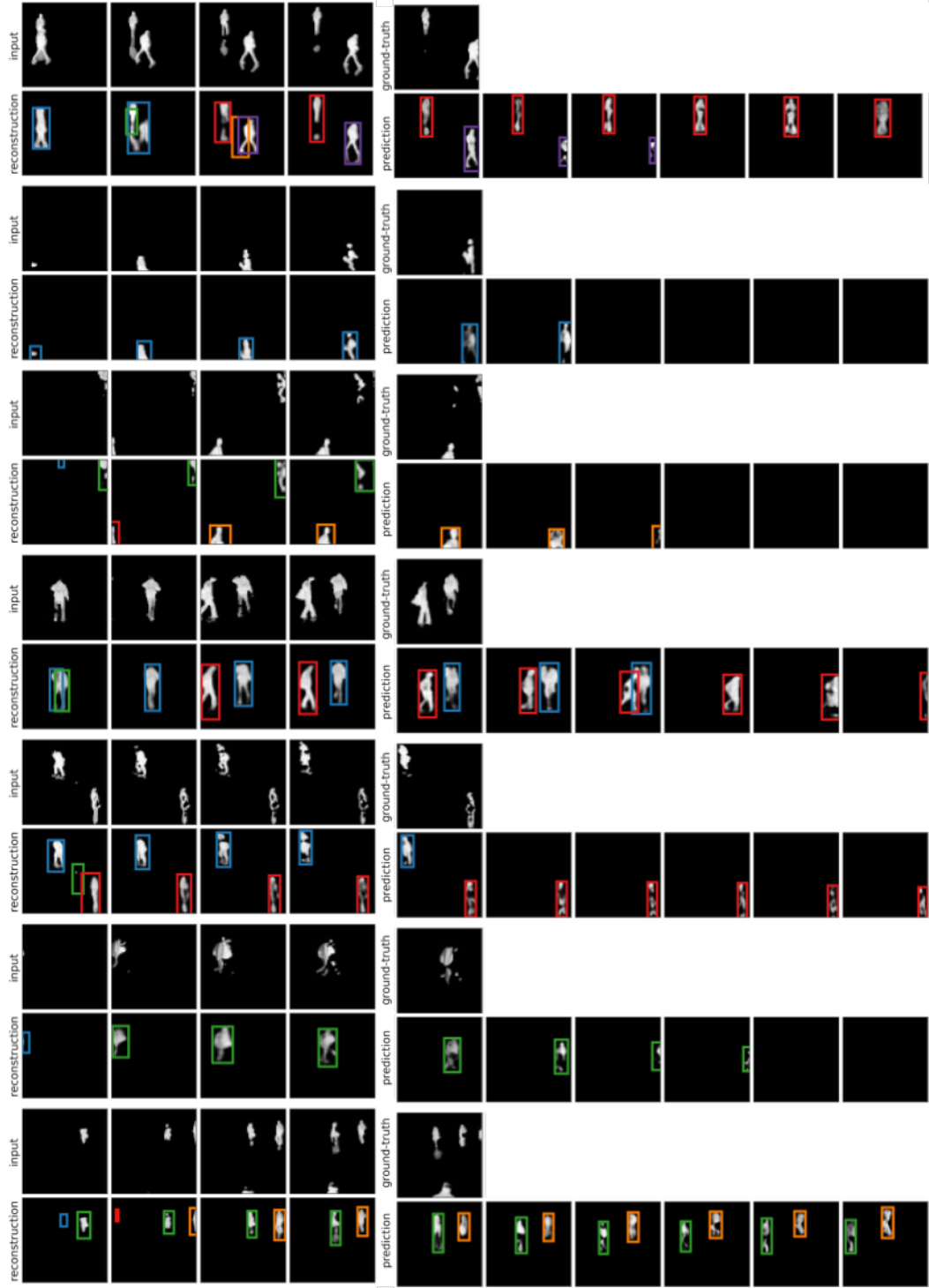


Figure 20: Conditional generation from SQAIR, which sees only the first four frames in every case. Top is the input sequence (and the remaining ground-truth), while bottom is reconstruction (first four time-steps) and then generation.