Learning Object-Centric Representations

Adam Roman Kosiorek

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2019

Abstract

Whenever an agent interacts with its environment, it has to take into account and interact with any objects present in this environment. And yet, the majority of machine learning solutions either treat objects only implicitly or employ highly-engineered solutions that account for objects through object detection algorithms. In this thesis, we explore supervised and unsupervised methods for learning object-centric representations from vision. We focus on end-to-end learning, where information about objects can be extracted directly from images, and where every object can be separately described by a single vector-valued variable. Specifically, we present three novel methods:

- HART and MOHART, which track single- and multiple-objects in video, respectively, by using RNNs with a hierarchy of differentiable attention mechanisms. These algorithms learn to anticipate future appearance changes and movement of tracking objects, thereby learning representations that describe every tracked object separately.
- SQAIR, a VAE-based generative model of moving objects, which explicitly models disappearance and appearance of new objects in the scene. It models every object with a separate latent variable, and disentangles appearance, position and scale of each object. Posterior inference in this model allows for unsupervised object detection and tracking.
- SCAE, an unsupervised autoencoder with in-built knowledge of two-dimensional geometry and object-part decomposition, which is based on capsule networks. It learns to discover parts present in an image, and group those parts into objects. Each object is modelled by a separate *object capsule*, whose activation probability is highly correlated with the object class, therefore allowing for state-of-the-art unsupervised image classification.

Contents

1 Introduction

Our natural environments are filled with objects, and we interact with objects all the time: when driving a car, we need to think about other cars and pedestrians; opening a door often requires using a key. Objects are inherently of interest to us, and what follows is that they are also of interest to computer vision and machine learning communities. While there exists a vast body of literature on object detection and tracking (see e. g., Ciaparrone et al. [21] and L. Liu et al. [106] for respective reviews), these two problems are typically solved using enormous amounts of human-labelled data. The resulting models allow locating objects in images and tracking them in videos but offer no further information about object behaviour, its intent, or its relationship with the environment and other objects. It is possibly for this reason that the majority of machine learning models for other tasks reported in the literature do not explicitly rely on object detection or tracking. Instead, these models treat objects only implicitly and gain knowledge about objects as a by-product of solving the main task, which can be playing Atari games (V. Mnih et al. [118]), visual question answering (Malinowski et al. [111]), image captioning (K. Xu et al. [184]), or many others. Meanwhile, it has been verified that having disentangled representations[1]

[1]That is, representations in which different parts of the representation vector encode different factors of variation present in the dataset.

of the scene can lead to faster learning and better performance (Steenkiste et al. [157]). Similarly, having explicit representations of different objects in the scene—a kind of disentanglement—can lead to faster learning and more accurate predictive models (Veerapaneni et al. [172]). Explicit object representations can also be used with relational reasoning modules, and therefore improve results on a wide variety of tasks ranging from comparing objects (Santoro et al. [146]) to learning to use tools (Baker et al. [7]). Interestingly, we also know that humans (and other mammals) can learn about objects without any supervision (Lambert et al. [97]), contrary to current approaches to detection and tracking. Given the above findings, it is natural to ask whether it is possible to build a machine learning system that learns about objects from vision inputs and without human supervision, and ideally one that can provide more information about objects than just their location. Trying to answer this question is the main subject of this thesis.

The very related question is that of "what is an object?". While we have no clear answer in sight, the works presented in this thesis can be seen as different attempts at finding an empirical definition of what an object might be. We revisit this question at the end of this thesis, in Section 7.3, where we try to delve deeper into the nature of objects in images and videos.

In order to answer this question, we begin by reviewing the existing literature in Chapter 2. Since the subject of learning object-centric representations overlaps with (1) relational reasoning, (2) object detection and tracking, (3) representation learning and disentanglement, and (4) generative modelling, we devote separate sections to each of these areas. Moreover, each of the following chapters, that is Chapters 3 to 6 contain additional related work sections that investigate literature that is most relevant to each of the respective chapters.

Specifically, we start the overview of related works in Chapter 2 by reviewing relational reasoning algorithms, where we discuss pros and cons of different methods and show that these methods rely on having access to some description of entities of interest, e. g., ground-truth object state. If that description is not available explicitly, it is

often provided by a heuristic which can lead to sub-optimal results and increased computational cost. We then change our focus to object detection and tracking algorithms, which could, in principle, be used to provide entity descriptions for relational reasoning. In practice, this is not the case, however—even though these approaches can provide the object location and sometimes their class, they typically do not estimate object state[2] and therefore cannot provide any additional information about the detected or tracked objects. This could still be better than using heuristic approaches but is far from using the ground-truth object state. Next, we survey representation learning and disentangled representations literature. The goal of representation learning is to learn to encode an input into a low-dimensional variable that contains relevant information about that input. If this representation is disentangled, then using such a variable in downstream tasks can be much easier and more interpretable. Under some circumstances, different objects could be described by different parts of that variable, which would constitute object-centric representations. We show, however, that the community is interested in disentangling properties describing visual scenes in general, without particular focus on objects. Finally, we turn to generative models and show that the majority of approaches do not focus on representation learning, but there are some that do, and they sometimes use object-centric representations.

Having talked about the relevant literature, we go back to our original question. We begin by simplifying it and asking what does it take to learn object representations end-to-end, but with using supervision? Chapter 3 describes HART—a biologically-inspired single-object tracking algorithm, which uses two-stream processing and a hierarchy of attention mechanisms to single out the tracked object from its surroundings. More specifically, it employs a recurrent neural network (RNN) to predict how the position and appearance of the object are going to change in the next time-step. This knowledge enables extracting a small *attention glimpse* that is likely to contain

[2]State can have a very broad meaning. What we mean here is a sufficient statistic of the object's future behaviour conditioned on the environment, or an internal Markovian state of that object, if such a state exists.

the object, as well as extracting the relevant features from this glimpse using a top-down attention mechanism. To solve this task, the state of the RNN has to contain information about the object motion, its appearance, and its interactions with the environment and meets the requirements of an object-centric representation.

While tracking single objects might be enough in some scenarios, it is often desirable to track multiple objects at the same time. In Chapter 4 we extend HART to multiple object tracking. To do so, we run multiple trackers in parallel and use a self-attention mechanism to facilitate communication between the trackers. This approach achieves better tracking performance in the presence of global motion (e. g., when the camera moves), when objects occlude each other or when they affect each others' trajectory—for example, to avoid collisions.

In Chapter 5, we go back to the original question and build a system capable of generative modelling of multiple moving objects, which is learnt without any human supervision. Using this system, we can detect and track objects as well as predict their future trajectories. We apply it to videos from static CCTV cameras, where it allows to detect and track pedestrians without supervision.

Finally, in Chapter 6 we build a deterministic generative model of images that has in-built knowledge of 2D geometry and can compose objects from their parts, and images from objects. This embedded knowledge allows it to learn what parts belong to which objects, and what are their typical geometric configurations. In some cases, this is sufficient for learning to classify objects without any supervision.

Chapter 7 discusses the implications of our work and possible future research directions, as well as inherent limitations that approaches of this kind face. It is worth noting that Chapters 3 to 6 are self-contained and based on the following publications:

1. A. R. Kosiorek et al. "Hierarchical Attentive Recurrent Tracking". In: *Advances in Neural Information Processing Systems*. 2017. arXiv: 1706.09262. URL: http://arxiv.org/abs/1706.09262,

2. F. B. Fuchs et al. "End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning". In: *CoRR*. 2019. arXiv: 1907.12887. URL: https://arxiv.org/abs/1907.12887,

3. A. R. Kosiorek et al. "Sequential Attend, Infer, Repeat: Generative modelling of moving objects". In: *Advances in Neural Information Processing Systems*. 2018. arXiv: 1806.01794. URL: https://arxiv.org/abs/1806.01794,

4. A. R. Kosiorek et al. "Stacked Capsule Autoencoders". In: *Advances in Neural Information Processing Systems*. 2019. arXiv: 1906.06818. URL: https://arxiv.org/abs/1906.06818.

1.1 Works Omitted from the Thesis and Developed in the Course of This DPhil

1. T. Rainforth et al. "Tighter Variational Bounds are Not Necessarily Better". In: *International Conference on Machine Learning*. 2018. URL: https://arxiv.org/abs/1802.04537

2. T. A. Le et al. "Revisiting Reweighted Wake-Sleep". In: *Conference on Uncertainty in Artificial Intelligence*. 2019. arXiv: 1805.10469

3. J. Lee et al. "Set Transformer". In: *International Conference on Machine Learning*. 2019. arXiv: 1810.00825. URL: http://proceedings.mlr.press/v97/lee19d.html

4. M. Engelcke et al. "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *International Conference on Learning Representations*. 2019. URL: https://arxiv.org/abs/1907.13052

5. F. B. Fuchs et al. "Scrutinizing and De-Biasing Intuitive Physics with Neural Stethoscopes." In: *British Machine Vision Conference*. 2019. URL: https://arxiv.org/abs/1806.05502

6. J. Xu et al. "MetaFun: Meta-Learning with Iterative Functional Updates". In: *CoRR*. 2019. arXiv: 1912.02738. URL: https://arxiv.org/abs/1912.02738

7. N. Dhir et al. "Bayesian Delay Embeddings for Dynamical Systems". In: *NIPS Timeseries Workshop*. 2017. URL: https://pdfs.semanticscholar.org/78e4/cf6912236acc1035519595c34e35a5ee4381.pdf

2
Background

Learning object-centric representations lies at the intersection of several areas of machine learning research. Concretely, these are (1) relational reasoning, because we can discover objects by analysing relations between different elements of a visual scene; but also having explicit representations of objects can facilitate relational reasoning, (2) object detection and tracking, as these two areas of research focus on locating objects in images and videos, respectively, (3) representation learning, since our task is to learn representations, albeit ones that describe objects and not entire scenes, and finally (4) compositional generative modelling, because object-centric representations can be useful for this task, but also because this area of research is among the most-promising candidates for learning object-centric representations in the absence of labels. We now describe these four areas in their dedicated sections, starting with relational reasoning. The purpose here is to provide overviews (but not exhaustive surveys) of those fields.

2.1 Relational Reasoning

We define relational reasoning as making inferences about a group of entities such as physical objects with certain properties (e. g., shape, mass) that exist in a shared

context. Importantly, we can make inferences about relations between any objects in a group, or we can use the knowledge of such relations to answer other (externally-posed) questions about the entities[1]. This definition closely follows that of Battaglia et al. [8], who also present an overview of this field of research. A typical example of relational reasoning is that of inferring which object in a scene is the biggest, or counting all objects made of metal (Santoro et al. [146]). In a more complicated example, agents can learn the relations between objects in the environment and use some objects to indirectly interact with other objects, which constitutes tool use (Baker et al. [7]).

Reasoning about relations requires comparing different entities to each other. Santoro et al. [146] introduced the relation network, which embeds every pair of present entities separately, sums all pair embeddings, to finally arrive at a single embedding that describes all the entities. This approach resembles the DeepSet approach for encoding sets (Zaheer et al. [188]), with the difference that it considers all pairs of entities explicitly. Recently introduced self-attention (Vaswani et al. [171]) can also be used to compare entities to each other and facilitates e. g., clustering, see Lee et al. [101], which is a clear improvement on the DeepSet method. We use self-attention in Chapter 4 to consider relations between different objects and improve multi-object tracking. More generally, relation networks and self-attention can be seen as graph neural networks (GNNs) operating on a fully-connected graph of entities. Both approaches scale quadratically in the number of entities and are therefore not suited for large numbers of entities. If the existence and/or type of relations between entities are known, one can build a special-purpose GNN (Schlichtkrull et al. [148]) that is cheaper to evaluate and potentially more expressive. Alternatively, one could construct a nearest-neighbour graph (Ramachandran et al. [129]) or introduce inducing points (Lee et al. [101]), which results in linear computation complexity.

Relational reasoning can be useful for inferring which parts of the scene constitute objects, but in order to reason about relations, one has to know what entities are

[1]Note that such answer can depend on the shared context, which can affect how objects relate to each other.

present in the scene. Typically, this data is not available, and one has to resort to heuristic solutions. Santoro et al. [146] use a convolutional neural network (CNN) to embed an image into a feature map and assumes that each location in this feature map can correspond to an object. They then consider pairwise relations between every pair of locations. Baker et al. [7] and Battaglia et al. [9] assume access to the ground-truth state of different entities, and Yi et al. [186] use MASK R-CNN (He et al. [58]) for segmenting out different objects before feeding them into a relational reasoning module. More generally, one can use any object detection, segmentation, or region proposal[2] algorithm to identify *interesting* parts of the scene and supply information about their whereabouts and appearance to a relational reasoning module. In the context of timeseries, it is useful to know the temporal characteristics of objects, which can be provided by object tracking algorithms. Therefore, we now discuss object detection and tracking methods and their suitability for relational reasoning applications.

2.2 Object Detection and Tracking

Object detection and tracking have been of long-standing interest to the computer vision community; both problems are typically addressed in a supervised manner, where object bounding boxes and/or segmentation masks and often additional labels are provided as a part of the training data.

Object detection In object detection, the user is interested in knowing how many objects there are in an image, where they are and often what they are. Historically, this problem has been typically addressed by matching patterns within a sliding window. For example, the seminal work of Viola and Jones [174] uses a cascade of simple filters found with boosting (Schapire [147]) that are matched against small image patches at every possible image location to find faces. This is computationally expensive, and recent state-of-the-art approaches use CNNs to predict locations in which objects might exist (Redmon et al. [135]). Some algorithms employ additional post-processing, where

[2]Regions of interest need not correspond to objects.

they evaluate every object proposal separately, possibly predicting some additional attributes like object class in RCNN (Girshick et al. [44] and Ren et al. [136]) or the corresponding segmentation mask in MASK-RCNN (He et al. [58]).

Object detection can be useful for relational reasoning because it can provide segmentation masks or bounding boxes and often additional features such as class labels. Moreover, one could extract hidden representations of the CNN and use it as an input to a relational reasoning module. There is no guarantee that these features would contain any useful information for this task; however since the decision of whether an object exists at any particular location is made independently of other locations. Additionally, the majority of object detection algorithms are supervised, which constraints their use to modalities for which large labelled datasets exist. The last two properties sharply contrast with unsupervised object detection methods, e. g., Attend, Infer, Repeat (AIR) of Eslami et al. [38], which needs no labels to train, but which also generates latent variables that are highly informative of discovered objects. AIR has been recently scaled to hundreds of objects and applied to Atari games (Crawford and Pineau [26] and Jiang et al. [76]) and enabled to infer the posterior distribution over object appearance exactly with sum-product networks (Stelzner et al. [158]). We provide an extension of AIR to image sequences in Chapter 5. Also applicable to relational reasoning are compositional generative models, which in effect perform image segmentation without supervision. These are further discussed in Section 2.3.

Object tracking The task of object tracking is to estimate the location of an object of interest in a video frame and to maintain that estimate over time as the video progresses. Importantly, unlike in object detection, here the user is interested in how the location of the tracked object changes, i. e., it is important to know if an object at a given timestep was present in the video before and where it was. Object tracking is typically initialised with an object bounding box (either ground-truth or from a detector). It can be divided into two broad categories of long- or short-term single-object tracking and multi-object tracking. In short-term tracking, the algorithm

is not provided with further detections. In contrast, in long-term tracking, it typically has access to external detections and can restart in case of a failure.

Single-object tracking commonly uses Siamese networks (Held et al. [59] and Valmadre et al. [169]), first introduced in Bromley et al. [14] and Schmidhuber and Prelinger [150]. These approaches typically work by comparing a source patch containing the object of interest to patches at several locations in the target image. The highest-scoring patch in the target image is deemed to match the source patch, and its centre is taken as the new position of the tracked object. In order to handle scale differences, the source patch is compared to the target image at several different scales. Instead of comparing it to the whole target image, one typically constrains the region of interest to an area around the last known location of the object. Sometimes simple motion models, e. g., Kalman filter (Kálmán and Bucy [78] and Swerling [162]), are used to reduce the size of the search region further.

These types of algorithms can provide relational reasoning modules with a sequence of locations, but no other information is available. Similarly to object detection algorithms, one could use intermediate representation from one of the layers of a Siamese net as input for relational reasoning. However, since these methods perform only appearance-based matching of source and target features, there is no guarantee for these features to contain any useful information e. g., about appearance changes or motion of the object. Moreover, since these methods do not aggregate information through time, they are unable to inform about object intent or its motion. This is in contrast to end-to-end tracking approaches, where models do estimate object state and learn to anticipate position and appearance changes. More specifically, Siamese networks can be seen as an RNN unrolled over two timesteps. In Chapter 3, we describe HART—a model that uses an RNN with an attention mechanism to predict bounding boxes for single objects, while robustly modelling their motion and appearance. HART is based on RATM (Kahoú et al. [77]), but decouples attention and bounding-box prediction and features an additional attention mechanism. At the time, HART was the best-performing RNN-based tracker, outperfomed only by the concurrently developed RE3 of Gordon et al. [46], which is also based on an RNN, but is not end-to-end due to

the use of hard attention for cropping. HART was extended to handle RGBD inputs in Rasouli Danesh et al. [132]. It was also augmented with a meta-learning update to better remember the tracked object in B. Li et al. [104]. We also extended it to multi-object tracking in Chapter 4, which is discussed further below.

Multi-object tracking is typically attained by detecting objects and performing data association on bounding-boxes in order to link them into coherent trajectories (Bae and Yoon [6], Keuper et al. [83], Milan et al. [114], and L. Zhang et al. [189]), which differs considerably from the single-object tracking paradigm. Many approaches additionally use motion models and appearance to improve data association, see e. g., Bewley et al. [12]. While recent approaches replace some steps of this method with learned elements (Bae and Yoon [6], Keuper et al. [83], Nam and Han [119], Ning et al. [122], Schulter et al. [153], and Xiang et al. [181]), not a single purly-vision-based end-to-end multi-object tracking method exists, with the exception of unsupervised video-modelling models described in Section 2.3 and including SQAIR of Chapter 5.

2.3 Generative Modelling and Representation Learning

Representation learning and generative modelling are inherently tied, as representations we care about can be often obtained by posterior inference for an appropriate generative model. Since representation learning is the main topic of this thesis, we now briefly review these two areas.

Formally, the goal of representation learning is to learn a concise representation $\mathbf{z} \in \mathbb{Z}$, typically $\mathbb{Z} \subseteq \mathbb{R}^d$, which summarizes information contained in some high-dimensional input $\mathbf{x} \in \mathbb{X}$. The usual requirements for the learned representations are to be low-dimensional and easy to use in downstream tasks: e. g., it should be possible to classify images by using a linear classifier on $\mathbf{z}$. This is different from transfer learning, which can use similar methods to representation learning, but where the goal is to pre-train a feature extractor without supervision and fine-tune it on a target task using supervised data (Devlin et al. [31] and He et al. [57]).

While there exist a plethora of representation learning methods, an exhaustive review of available approaches is beyond the scope of this work, and we are going to discuss some recent methods focused on (i) generative modelling, and especially its (ii) disentangled and (iii) compositional flavours, as these are best suited for object-centric representation learning. We note that there is a vast body of work on contrastive representation learning (He et al. [57], Hjelm et al. [65], Oord et al. [124], Schmidhuber [149], and Tschannen et al. [166]), but it focuses on learning global representations. The recent work from Kipf et al. [89] is a notable exception, and offers the first method to learn object-centric representations in this framework.

Generative modelling In parametric[3] generative modelling, the goal is to approximate the data distribution $p_{\text{data}}(\mathbf{x})$ with a parametric distribution $p_\theta(\mathbf{x})$ with parameters θ. Recently, the most popular approaches have revolved around flow-based and autoregressive methods as well as the variational auto-encoder (VAE) and generative adversarial network (GAN) frameworks.

Flow-based models can provide an exact estimate of the data likelihood, but do not use latent variables (Dinh et al. [34], Kingma and Dhariwal [86], and Rezende and Mohamed [137]). Since all data modelling is done by bijective transformations of some simple random variable, flow-based models cannot reduce the original data dimensionality, which is in contrast with our requirements. Autoregressive models need not use latent variables to model data well (Oord et al. [123] and Uria et al. [168]), and even if combined with latent variables, they often choose not to use them, therefore not learning useful representations (Gulrajani et al. [52]). It follows that only VAEs and GANs are viable candidates for representation learning. However, vanilla GANs approximate the data distribution only implicitly by learning to transform a simple random variable appropriately, and posterior inference in these models is complicated and can be inefficient (Bojanowski et al. [13]). Therefore, they do not provide any useful features (Goodfellow et al. [45] and Radford et al. [127]). The bidirectional GAN (BIGAN) equips a traditional GAN with an encoder that approximates

[3]Nonparametric modelling is also possible, but is out of scope of this work.

the posterior distribution over encodings and can therefore learn good representations (Donahue et al. [35] and Donahue and Simonyan [36]). Training of these models is poorly understood, however, as is the case with other GAN-like models. We note that this method is interesting and should be investigated in future, but is out of scope of this work. Finally, VAEs are well understood and allow for joint training of approximate posterior inference and generative models.

VAEs were first introduced in Kingma and Welling [88] and Rezende et al. [138] and model the data distribution by introducing a latent variable as $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) \, d\mathbf{z}$. Learning in a VAE is accomplished by maximizing the evidence lower bound (ELBO), where $q_\phi(\mathbf{z} \mid \mathbf{x})$ approximates the true posterior distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right]. \tag{2.1}$$

The approximate posterior q can be used as an encoder that learns to extract representations from data. While the majority of recent works on VAEs focuses on improving density estimation as opposed to learning representations, e. g., Kingma et al. [87], Maaløe et al. [107], and Razavi et al. [134], there are two streams of work, that of disentangled representation learning and compositional generative modelling, that are of interest to this thesis.

Disentangled Representation Learning The majority of representation learning literature is concerned with learning *global* representations. In this setting, the input $\mathbf{x}$ is described by a single vector-valued $\mathbf{z}$. Since we would like the learned representations to be easily readable (or usable), e. g., by using linear predictors, it is desirable to have representation with a compositional structure. That is, the learned representation should be composed of a number of parts, with different parts being independent of each other and describing different factors of variations in the data. Such representations are called *disentangled*, though there exists no universally-agreed upon definition of this property. The concept was introduced in Higgins et al. [61], with the majority of approaches focused on VAE-style models with an additional

objective encouraging disentanglement (Higgins et al. [61] and Kim and A. Mnih [84]) or using specifically-structured decoders (Watters et al. [177]).

Learning object-centric representations means learning representations that separately describe each object in a scene, and as such, is a subset of disentangled representation learning. However, in disentangled representation literature, there is no focus on separating the representation into specific parts corresponding to different objects; different objects and their properties are seen simply as different factors of variation in the data.

Recent studies show that learning disentangled representations can lead to more sample-efficient learning and better performance on downstream tasks (Steenkiste et al. [157]). This finding is also confirmed for object-centric representations (Veerapaneni et al. [172]), which improve sample-efficiency in reinforcement learning (RL) tasks. We now turn to compositional generative modelling, which tries to model data as a composition of effects caused by different latent variables.

Compositional Generative Modelling In compositional generative models, we are interested in cases where the data can be explained as a composition of effects caused by different latent variables. In vision, that is in images and videos, this can often happen when different latent variables correspond to different objects present in the scene, hence leading to learning object-centric representations.

Attend, Infer, Repeat (AIR) of Eslami et al. [38] was one of the first compositional generative models. It learns a sequential inference procedure, which allows decomposing an input image into its constituent objects. Compositionality in this model is achieved by creating an image by adding several local components, where every component occupies a rectangular, and usually small, part of the image canvas. While it works well on images with simple backgrounds and non-overlapping objects, this model has trouble dealing with cluttered or noisy backgrounds and therefore does not work on real-world images. AIR was extended to handle many more objects in Crawford and Pineau [26] and to deal with simple backgrounds in Stelzner et al. [158].

We noticed that learning in AIR can be unstable in the sense that different training runs converge to vastly different solutions, and addressed this problem by introducing Sequential Attend, Infer, Repeat (SQAIR) in Chapter 5, which extends AIR to image sequences. Once SQAIR decomposes an image into objects, it prefers to keep that decomposition over subsequent frames, therefore introducing the notion of spatial consistency. SQAIR was further extended to handle tens of objects in Crawford and Pineau [25] and Jiang et al. [76] and to handle fairly complicated non-constant backgrounds in Jiang et al. [76].

An alternative line of work explores modelling images with spatial Gaussian mixture models, first introduced in Greff et al. [49]. In this work, as well as in its extension by Ilin et al. [68], the inference is deterministic and performed by a neural network. Greff et al. [50] and Steenkiste et al. [156] extend this approach by using the expectation maximization (EM) algorithm for inference and then substitute parts of EM with learned components to further improve performance. Burgess et al. [16] introduced MONET, which is a similar model implemented as a VAE, but where inference over some latent variables is deterministic. Finally, Greff et al. [48] introduce IODINE, which improves upon MONET's inference mechanism by applying the iterative amortised variational inference framework (Marino et al. [112]), where gradient updates are also replaced with learned components. IODINE is much more computationally expensive, however. Recently, we introduced GENESIS (Engelcke et al. [37]), which is a VAE, similarily to MONET, but with a slightly modified graphical model. GENESIS is more computationally efficient than MONET and performs on par or better than IODINE.

3

Hierarchical Attentive Recurrent Tracking

Class-agnostic object tracking is particularly difficult in cluttered environments as target specific discriminative models cannot be learned *a priori*. Inspired by how the human visual cortex employs spatial attention and separate "where" and "what" processing pathways to actively suppress irrelevant visual features, this work develops a hierarchical attentive recurrent model for single object tracking in videos. The first layer of attention discards the majority of background by selecting a region containing the object of interest, while the subsequent layers tune in on visual features *particular* to the tracked object. This framework is fully differentiable and can be trained in a purely data driven fashion by gradient methods. To improve training convergence, we augment the loss function with terms for auxiliary tasks relevant for tracking. Evaluation of the proposed model is performed on two datasets: pedestrian tracking on the KTH activity recognition dataset and the more difficult KITTI object tracking dataset.

3.1 Introduction

In computer vision, designing an algorithm for model-free tracking of anonymous objects is challenging, since no target-specific information can be gathered *a priori* and yet the algorithm has to handle target appearance changes, varying lighting conditions and occlusion. To make it even more difficult, the tracked object often constitutes but a small fraction of the visual field. The remaining parts may contain *distractors*, which are visually salient objects resembling the target but hold no relevant information. Despite this fact, recent models often process the whole image, which exposes them to noise and increases the associated computational cost or they use heuristic methods to decrease the size of search regions. This in contrast to human visual perception, which does not process the visual field in its entirety, but rather acknowledges it briefly and focuses on processing small fractions thereof, which we dub *visual attention*.

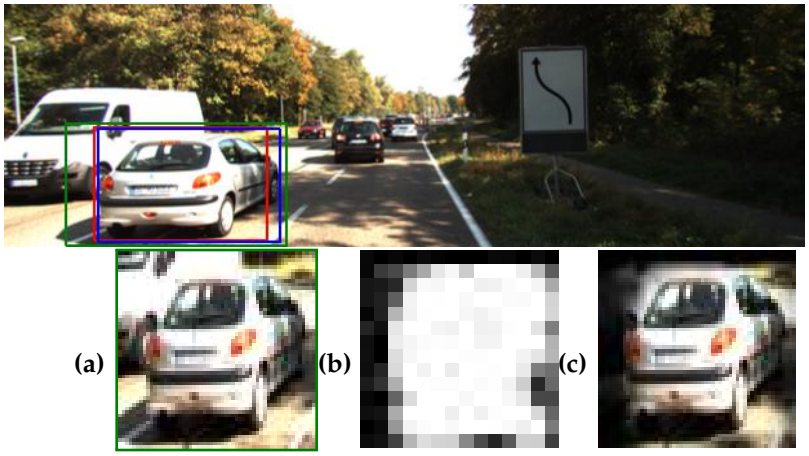


Figure 3.1: KITTI image with the ground-truth and predicted bounding boxes and an attention glimpse. The lower row corresponds to the hierarchical attention of our model: 1^{st} layer extracts an attention glimpse (a), the 2^{nd} layer uses appearance attention to build a location map (b). The 3^{rd} layer uses the location map to suppress distractors, visualised in (c).

Attention mechanisms have recently been explored in machine learning in a wide variety of contexts Jaderberg et al. [73] and Vinyals et al. [173], often providing new capabilities to machine learning algorithms Eslami et al. [38], Graves et al. [47], and Gregor et al. [51]. While they improve efficiency V. Mnih et al. [117] and performance

on state-of-the-art machine learning benchmarks Vinyals et al. [173], their architecture is much simpler than that of the mechanisms found in the human visual cortex Dayan and Abbott [27]. Attention has also been long studied by neuroscientists Kastner and Ungerleider [80], who believe that it is crucial for visual perception and cognition Cheung et al. [18], since it is inherently tied to the architecture of the visual cortex and can affect the information flow inside it. Whenever more than one visual stimulus is present in the receptive field of a neuron, all the stimuli compete for computational resources due to the limited processing capacity. Visual attention can lead to suppression of distractors by reducing the size of the receptive field of a neuron and by increasing sensitivity at a given location in the visual field (*spatial attention*). It can also amplify activity in different parts of the cortex, which are specialised in processing different types of features, leading to response enhancement with respect to those features (*appearance attention*). The functional separation of the visual cortex is most apparent in two distinct processing pathways. After leaving the eye, the sensory inputs enter the primary visual cortex (known as *V1*) and then split into the *dorsal stream*, responsible for estimating spatial relationships (*where*), and the *ventral stream*, which targets appearance-based features (*what*).

Inspired by the general architecture of the human visual cortex and the role of attention mechanisms, this work presents a biologically-inspired recurrent model for single object tracking in videos (*cf.* Section 3.3). Tracking algorithms typically use simple motion models and heuristics to decrease the size of the search region. It is interesting to see whether neuroscientific insights can aid our computational efforts, thereby improving the efficiency and performance of single object tracking. It is worth noting that visual attention can be induced by the stimulus itself (due to, e. g., high contrast) in a *bottom-up* fashion or by back-projections from other brain regions and working memory as *top-down* influence. The proposed approach exploits this property to create a feedback loop that steers the *three* layers of visual attention mechanisms in our hierarchical attentive recurrent tracking (*HART*) framework, see Figure 3.1. The first stage immediately discards spatially irrelevant input, while later

stages focus on producing target-specific filters to emphasise visual features *particular* to the object of interest.

The resulting framework is end-to-end trainable and we resort to maximum likelihood estimation (MLE) for parameter learning. This follows from our interest in estimating the distribution over object locations in a sequence of images, given the initial location from whence our tracking commenced. Formally, given a sequence of images $\mathbf{x}_{1:T} \in \mathbb{R}^{H \times W \times C}$, where the superscript denotes height, width and the number of channels of the image, respectively, and an initial location for the tracked object given by a bounding box $\mathbf{b}_1 \in \mathbb{R}^4$, the conditional probability distribution factorises as

$$p(\mathbf{b}_{2:T} \mid \mathbf{x}_{1:T}, \mathbf{b}_1) = \int p(\mathbf{h}_1 \mid \mathbf{x}_1, \mathbf{b}_1) \prod_{t=2}^{T} \int p(\mathbf{b}_t \mid \mathbf{h}_t) p(\mathbf{h}_t \mid \mathbf{x}_t, \mathbf{b}_{t-1}, \mathbf{h}_{t-1}) \, d\mathbf{h}_t \, d\mathbf{h}_1, \quad (3.1)$$

where we assume that motion of an object can be described by a Markovian state $\mathbf{h}_t$. Our bounding box estimates are given by $\widehat{\mathbf{b}}_{2:T}$, found by the MLE of the model parameters. In sum, our contributions are threefold: Firstly, a hierarchy of attention mechanisms that leads to suppressing distractors and computational efficiency is introduced. Secondly, a biologically plausible combination of attention mechanisms and recurrent neural networks is presented for object tracking. Finally, our attention-based tracker is demonstrated using real-world sequences in challenging scenarios where previous recurrent attentive trackers have failed. Next we briefly review related work (Section 3.2) before describing how information flows through the components of our hierarchical attention in Section 3.3. Section 3.4 details the losses applied to guide the attention. Section 3.5 presents experiments on KTH and KITTI datasets with comparison to related attention-based trackers. Section 3.6 discusses the results and intriguing properties of our framework and Section 3.7 concludes the work. Code and results are available online[1].

[1] `https://github.com/akosiorek/hart`

3.2 Related Work

A number of recent studies have demonstrated that visual content can be captured through a sequence of spatial glimpses or foveation Gregor et al. [51] and V. Mnih et al. [117]. Such a paradigm has the intriguing property that the computational complexity is proportional to the number of steps as opposed to the image size. Furthermore, the fovea centralis in the retina of primates is structured with maximum visual acuity in the centre and decaying resolution towards the periphery, Cheung et al. [18] show that if spatial attention is capable of zooming, a regular grid sampling is sufficient. Jaderberg et al. [73] introduced the spatial transformer network (STN) which provides a fully differentiable means of transforming feature maps, conditioned on the input itself. Eslami et al. [38] use the STN as a form of attention in combination with a recurrent neural network (RNN) to sequentially locate and identify objects in an image. Moreover, Eslami et al. [38] use a latent variable to estimate the presence of additional objects, allowing the RNN to adapt the number of time-steps based on the input. Our spatial attention mechanism is based on the two dimensional Gaussian grid filters of Kahoú et al. [77] which is both fully differentiable and more biologically plausible than the STN.

Whilst focusing on a specific location has its merits, focusing on particular appearance features might be as important. A policy with feedback connections can learn to adjust filters of a convolutional neural network (CNN), thereby adapting them to features present in the current image and improving accuracy Stollenga et al. [159]. De Brabandere et al. [28] introduced dynamic filter network (DFN), where filters for a CNN are computed on-the-fly conditioned on input features, which can reduce model size without performance loss. Karl et al. [79] showed that an input-dependent state transitions can be helpful for learning latent Markovian state-space system. While not the focus of this work, we follow this concept in estimating the expected appearance of the tracked object.

In the context of single object tracking, both attention mechanisms and RNNs appear to be perfectly suited, yet their success has mostly been limited to simple

monochromatic sequences with plain backgrounds Kahoú et al. [77]. Cheung [17] applied STNs Jaderberg et al. [73] as attention mechanisms for real-world object tracking, but failed due to exploding gradients potentially arising from the difficulty of the data. Ning et al. [121] achieved competitive performance by using features from an object detector as inputs to a long-short memory network (LSTM), but requires processing of the whole image at each time-step.

Two recent state-of-the-art trackers employ convolutional Siamese networks which can be seen as an RNN unrolled over two time-steps Held et al. [59] and Valmadre et al. [169]. Both methods explicitly process small search areas around the previous target position to produce a bounding box offset Held et al. [59] or a correlation response map with the maximum corresponding to the target position Valmadre et al. [169]. We acknowledge the recent work[2] of Gordon et al. [46] which employ an RNN based model and use explicit cropping and warping as a form of non-differentiable spatial attention. The work presented in this paper is closest to Kahoú et al. [77] where we share a similar spatial attention mechanism which is guided through an RNN to effectively learn a motion model that spans multiple time-steps. The next section describes our additional attention mechanisms in relation to their biological counterparts.

3.3 Hierarchical Attention

Inspired by the architecture of the human visual cortex, we structure our system around working memory responsible for storing the motion pattern and an appearance description of the tracked object. If both quantities were known, it would be possible to compute the expected location of the object at the next time step. Given a new frame, however, it is not immediately apparent which visual features correspond to the appearance description. If we were to pass them on to an RNN, it would have to implicitly solve a data association problem. As it is non-trivial, we prefer to model it explicitly by outsourcing the computation to a separate processing stream

[2]Gordon et al. [46] only became available at the time of submitting this paper.

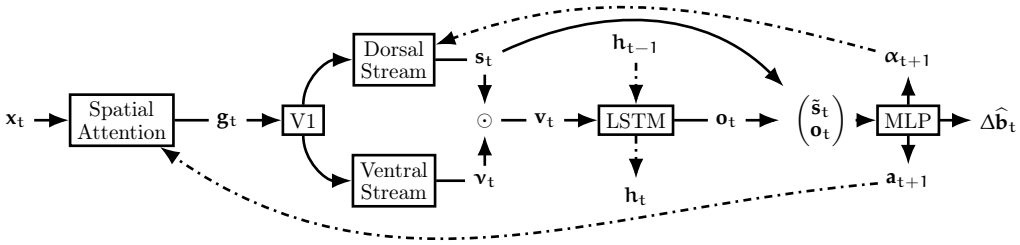


Figure 3.2: Hierarchical Attentive Recurrent Tracking. Spatial attention extracts a glimpse $\mathbf{g}_t$ from the input image $\mathbf{x}_t$. V1 and the ventral stream extract appearance-based features $\boldsymbol{\nu}_t$ while the dorsal stream computes a foreground/background segmentation $\mathbf{s}_t$ of the attention glimpse. Masked features $\mathbf{v}_t$ contribute to the working memory $\mathbf{h}_t$. The LSTM output $\mathbf{o}_t$ is then used to compute attention $\mathbf{a}_{t+1}$, appearance $\boldsymbol{\alpha}_{t+1}$ and a bounding box correction $\Delta\widehat{\mathbf{b}}_t$. Dashed lines correspond to temporal connections, while solid lines describe information flow within one time-step.

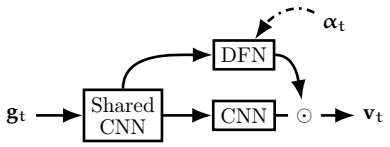


Figure 3.3: Architecture of the appearance attention. V1 is implemented as a CNN shared among the dorsal stream (DFN) and the ventral stream (CNN). The $\odot$ symbol represents the Hadamard product and implements masking of visual features by the foreground/background segmentation.

conditioned on the expected appearance. This results in a location-map, making it possible to neglect features inconsistent with our memory of the tracked object. We now proceed with describing the information flow in our model.

Given attention parameters $\mathbf{a}_t$, the *spatial attention* module extracts a glimpse $\mathbf{g}_t$ from the input image $\mathbf{x}_t$. We then apply *appearance attention*, parametrised by appearance $\boldsymbol{\alpha}_t$ and comprised of V1 and dorsal and ventral streams, to obtain object-specific features $\mathbf{v}_t$, which are used to update the hidden state $\mathbf{h}_t$ of an LSTM. The LSTM's output is then decoded to predict both spatial and appearance attention parameters for the next time-step along with a bounding box correction $\Delta\widehat{\mathbf{b}}_t$ for the current time-step. Spatial attention is driven by top-down signal $\mathbf{a}_t$, while appearance attention depends on top-down $\boldsymbol{\alpha}_t$ as well as bottom-up (contents of the glimpse $\mathbf{g}_t$) signals. Bottom-up signals have local influence and depend on stimulus salience at a given location, while top-down signals incorporate global context into local processing. This attention hierarchy, further enhanced by recurrent connections, mimics that of the human visual

cortex Kastner and Ungerleider [80]. We now describe the individual components of the system.

Spatial Attention Our spatial attention mechanism is similar to the one used by Kahoú et al. [77]. Given an input image $\mathbf{x}_t \in \mathbb{R}^{H \times W}$, it creates two matrices $\mathbf{A}_t^x \in \mathbb{R}^{w \times W}$ and $\mathbf{A}_t^y \in \mathbb{R}^{h \times H}$, respectively. Each matrix contains one Gaussian per row; the width and positions of the Gaussians determine which parts of the image are extracted as the attention glimpse. Formally, the glimpse $\mathbf{g}_t \in \mathbb{R}^{h \times w}$ is defined as

$$\mathbf{g}_t = \mathbf{A}_t^y \mathbf{x}_t \left(\mathbf{A}_t^x\right)^\mathsf{T}. \tag{3.2}$$

Attention is described by centres μ of the Gaussians, their variances σ^2 and strides γ between centers of Gaussians of consecutive rows of the matrix, one for each axis. In contrast to the work by Kahoú et al. [77], only centres and strides are estimated from the hidden state of the LSTM, while the variance depends solely on the stride. This prevents excessive aliasing during training caused when predicting a small variance (compared to strides) leading to smoother convergence. The relationship between variance and stride is approximated using linear regression with polynomial basis functions (up to 4^{th} order) before training the whole system. The glimpse size we use depends on the experiment.

Appearance Attention This stage transforms the attention glimpse $\mathbf{g}_t$ into a fixed-dimensional vector $\mathbf{v}_t$ comprising appearance and spatial information about the tracked object. Its architecture depends on the experiment. In general, however, we implement $V1 : \mathbb{R}^{h \times w} \to \mathbb{R}^{h_v \times w_v \times c_v}$ as a number of convolutional and max-pooling layers. They are shared among later processing stages, which corresponds to the primary visual cortex in humans Dayan and Abbott [27]. Processing then splits into ventral and dorsal streams. The ventral stream is implemented as a CNN, and handles visual features and outputs feature maps $\boldsymbol{\nu}_t$. The dorsal stream, implemented as a DFN, is responsible for handling spatial relationships. Let $\text{MLP}\cdot$ denote a multi-layered perceptron. The dorsal stream uses appearance $\boldsymbol{\alpha}_t$ to dynamically compute

convolutional filters $\psi_t^{a\times b\times c\times d}$, where the superscript denotes the size of the filters and the number of input and output feature maps, as

$$\Psi_t = \left\{\psi_t^{a_i\times b_i\times c_i\times d_i}\right\}_{i=1}^{K} = \text{MLP}\,\alpha_t. \tag{3.3}$$

The filters with corresponding nonlinearities form K convolutional layers applied to the output of V1. Finally, a convolutional layer with a 1×1 kernel and a sigmoid non-linearity is applied to transform the output into a spatial Bernoulli distribution $\mathbf{s}_t$. Each value in $\mathbf{s}_t$ represents the probability of the tracked object occupying the corresponding location.

The location map of the dorsal stream is combined with appearance-based features extracted by the ventral stream, to imitate the distractor-suppressing behaviour of the human brain. It also prevents drift and allows occlusion handling, since object appearance is not overwritten in the hidden state when input does not contain features particular to the tracked object. Outputs of both streams are combined as[3]

$$\mathbf{v}_t = \text{MLP}\,\text{vec}(\nu_t \odot \mathbf{s}_t), \tag{3.4}$$

with $\odot$ being the Hadamard product.

State Estimation Our approach relies on being able to predict future object appearance and location, and therefore it heavily depends on state estimation. We use an LSTM, which can learn to trade-off spatio-temporal and appearance information in a data-driven fashion. It acts like a working memory, enabling the system to be robust to occlusions and oscillating object appearance e. g., when an object rotates and comes back to the original orientation.

$$\mathbf{o}_t, \mathbf{h}_t = \text{LSTM}(\mathbf{v}_t, \mathbf{h}_{t-1}), \tag{3.5}$$

$$\alpha_{t+1}, \Delta\mathbf{a}_{t+1}, \Delta\widehat{\mathbf{b}}_t = \text{MLP}\,\mathbf{o}_t, \text{vec}(\mathbf{s}_t), \tag{3.6}$$

$$\mathbf{a}_{t+1} = \mathbf{a}_t + \tanh(\mathbf{c})\Delta\mathbf{a}_{t+1}, \tag{3.7}$$

$$\widehat{\mathbf{b}}_t = \mathbf{a}_t + \Delta\widehat{\mathbf{b}}_t \tag{3.8}$$

[3] $\text{vec} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ is the vectorisation operator, which stacks columns of a matrix into a column vector.

Equations (3.5) to (3.8) detail the state updates. Spatial attention at time t is formed as a cumulative sum of attention updates from times $t = 1$ to $t = T$, where $\mathbf{c}$ is a learnable parameter initialised to a small value to constrain the size of the updates early in training. Since the spatial-attention mechanism is trained to predict where the object is going to go (Section 3.4), the bounding box $\widehat{\mathbf{b}}_t$ is estimated relative to attention at time t.

3.4 Loss

We train our system by minimising a loss function comprised of: a tracking loss term, a set of terms for auxiliary tasks and regularisation terms. Auxiliary tasks are essential for real-world data, since convergence does not occur without them. They also speed up learning and lead to better performance for simpler datasets. Unlike the auxiliary tasks used by Jaderberg et al. [72], ours are relevant for our main objective — object tracking. In order to limit the number of hyperparameters, we automatically learn loss weighting. The loss $\mathcal{L}(\cdot)$ is given by

$$\mathcal{L}_{\text{HART}}(\mathcal{D}, \theta) = \lambda_t \mathcal{L}_t(\mathcal{D}, \theta) + \lambda_s \mathcal{L}_s(\mathcal{D}, \theta) + \lambda_a \mathcal{L}_a(\mathcal{D}, \theta) + R(\boldsymbol{\lambda}) + \beta R(\mathcal{D}, \theta), \tag{3.9}$$

with dataset $\mathcal{D} = \left\{(\mathbf{x}_{1:T}, \mathbf{b}_{1:T})^i\right\}_{i=1}^M$, network parameters θ, regularisation terms $R(\cdot)$, adaptive weights $\boldsymbol{\lambda} = \{\lambda_t, \lambda_s, \lambda_d\}$ and a regularisation weight β. We now present and justify components of our loss, where expectations $\mathbb{E}[\cdot]$ are evaluated as an empirical mean over a minibatch of samples $\left\{\mathbf{x}_{1:T}^i, \mathbf{b}_{1:T}^i\right\}_{i=1}^M$, where M is the batch size.

Tracking To achieve the main tracking objective (localising the object in the current frame), we base the first loss term on Intersection-over-Union (IoU) of the predicted bounding box w. r. t. the ground truth, where the IoU of two bounding boxes is defined as $\text{IoU}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cap \mathbf{b}}{\mathbf{a} \cup \mathbf{b}} = \frac{\text{area of overlap}}{\text{area of union}}$. The IoU is invariant to object and image scale, making it a suitable proxy for measuring the quality of localisation. Even though it (or an exponential thereof) does not correspond to any probability distribution (as it

cannot be normalised), it is often used for evaluation Kristan et al. [93]. We follow the work by Yu et al. [187] and express the loss term as the negative log of IoU:

$$\mathcal{L}_t(\mathcal{D}, \theta) = \mathbb{E}_{p(\widehat{\mathbf{b}}_{1:T}|\mathbf{x}_{1:T}, \mathbf{b}_1)}\Big[-\log \mathrm{IoU}(\widehat{\mathbf{b}}_t, \mathbf{b}_t)\Big], \tag{3.10}$$

with IoU clipped for numerical stability.

Spatial Attention Spatial attention singles out the tracked object from the image. To estimate its parameters, the system has to predict the object's motion. In case of an error, especially when the attention glimpse does not contain the tracked object, it is difficult to recover. As the probability of such an event increases with decreasing size of the glimpse, we employ two loss terms. The first one constrains the predicted attention to cover the bounding box, while the second one prevents it from becoming too large, where the logarithmic arguments are appropriately clipped to avoid numerical instabilities:

$$\mathcal{L}_s(\mathcal{D}, \theta) = \mathbb{E}_{p(\mathbf{a}_{1:T}|\mathbf{x}_{1:T}, \mathbf{b}_1)}\left[-\log\left(\frac{\mathbf{a}_t \cap \mathbf{b}_t}{\mathrm{area}(\mathbf{b}_t)}\right) - \log(1 - \mathrm{IoU}(\mathbf{a}_t, \mathbf{x}_t))\right]. \tag{3.11}$$

Appearance Attention The purpose of appearance attention is to suppress distractors while keeping full view of the tracked object e. g., focus on a *particular* pedestrian moving within a group. To guide this behaviour, we put a loss on appearance attention that encourages picking out only the tracked object. Let $\tau(\mathbf{a}_t, \mathbf{b}_t) : \mathbb{R}^4 \times \mathbb{R}^4 \to \{0, 1\}^{h_v \times w_v}$ be a target function. Given the bounding box $\mathbf{b}$ and attention $\mathbf{a}$, it outputs a binary mask of the same size as the output of V1. The mask corresponds to the the glimpse $\mathbf{g}$, with the value equal to one at every location where the bounding box overlaps with the glimpse and equal to zero otherwise. If we take $H(p, q) = -\sum_z p(z) \log q(z)$ to be the cross-entropy, the loss reads

$$\mathcal{L}_a(\mathcal{D}, \theta) = \mathbb{E}_{p(\mathbf{a}_{1:T}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{b}_1)}[H(\tau(\mathbf{a}_t, \mathbf{b}_t), \mathbf{s}_t)]. \tag{3.12}$$

Regularisation We apply the L2 regularisation to the model parameters θ and to the expected value of dynamic parameters $\boldsymbol{\psi}_t(\boldsymbol{\alpha}_t)$ as

$$R(\mathcal{D}, \theta) = \frac{1}{2}\|\theta\|_2^2 + \frac{1}{2}\Big\|\mathbb{E}_{p(\boldsymbol{\alpha}_{1:T}|\mathbf{x}_{1:T}, \mathbf{b}_1)}[\boldsymbol{\Psi}_t \mid \boldsymbol{\alpha}_t]\Big\|_2^2. \tag{3.13}$$

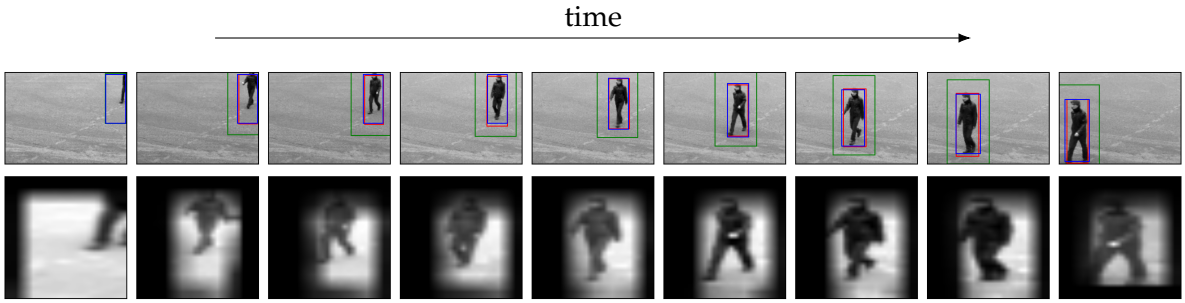


Figure 3.4: Tracking results on KTH dataset Schuldt et al. [152]. Starting with the first initialisation frame where all three boxes overlap exactly, time flows from left to right showing every 16^{th} frame of the sequence captured at 25fps. The colour coding follows from Figure 3.1. The second row shows attention glimpses multiplied with appearance attention.

Adaptive Loss Weights To avoid hyper-parameter tuning, we follow the work by Kendall et al. [82] and learn the loss weighting λ. After initialising the weights with a vector of ones, we add the following regularisation term to the loss function: $R(\lambda) = -\sum_i \log(\lambda_i^{-1})$.

3.5 Experiments

3.5.1 KTH Pedestrian Tracking

Kahoú et al. [77] performed a pedestrian tracking experiment on the KTH activity recognition dataset Schuldt et al. [152] as a real-world case-study. We replicate this experiment for comparison. We use code provided by the authors for data preparation and we also use their pre-trained feature extractor. Unlike them, we did not need to upscale ground-truth bounding boxes by a factor of 1.5 and then downscale them again for evaluation. We follow the authors and set the glimpse size $(h, w) = (28, 28)$. We replicate the training procedure exactly, with the exception of using the RMSProp optimiser Tieleman and G. Hinton [163] with learning rate of 3.33×10^{-5} and momentum set to 0.9 instead of the stochastic gradient descent with momentum. The original work reported an IoU of 55.03% on average, on test data, while the presented work achieves an average IoU score of 77.11%, reducing the relative error by almost a factor of two. Figure 3.4 presents qualitative results.

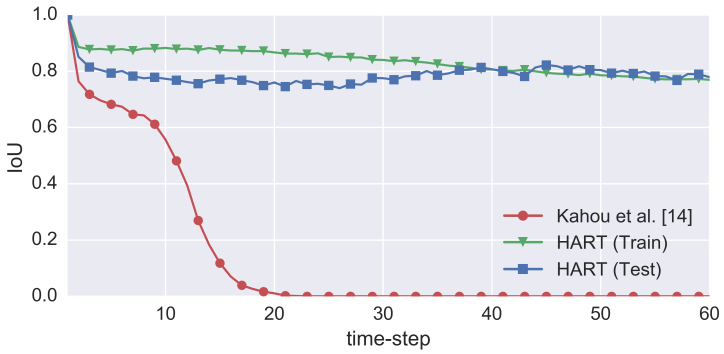


Figure 3.5: IoU curves on KITTI over 60 timesteps. HART (train) presents evaluation on the train set (we do not overfit).

Method	Avg. IoU
Kahoú et al. [77]	0.14
Spatial Att	0.60
App Att	0.78
HART	**0.81**

Table 3.1: Average IoU on KITTI over 60 time-steps.

3.5.2 Scaling to Real-World Data: KITTI

Since we demonstrated that pedestrian tracking is feasible using the proposed architecture, we proceed to evaluate our model in a more challenging multi-class scenario on the KITTI dataset Geiger et al. [43]. It consists of 21 high resolution video sequences with multiple instances of the same class posing as potential distractors. We split all sequences into 80/20 sequences for train and test sets, respectively. As images in this dataset are much more varied, we implement V1 as the first three convolutional layers of a modified AlexNet A. Krizhevsky et al. [1]. The original AlexNet takes inputs of size 227×227 and downsizes them to 14×14 after *conv3* layer. Since too low resolution would result in low tracking performance, and we did not want to upsample the extracted glimpse, we decided to replace the initial stride of four with one and to skip one of the max-pooling operations to conserve spatial dimensions. This way, our feature map has the size of $14 \times 14 \times 384$ with the input glimpse of size $(h, w) = (56, 56)$. We apply dropout with probability 0.25 at the end of V1. The ventral stream is comprised of a single convolutional layer with a 1×1 kernel and five output feature maps. The dorsal stream has two dynamic filter layers with kernels of size 1×1 and 3×3, respectively and five feature maps each. We used 100 hidden units in the RNN with orthogonal initialisation and Zoneout Krueger et al. [94] with probability set to 0.05. The system was trained via curriculum learning Bengio et al. [11], by starting with sequences of length five and increasing sequence

length every 13 epochs, with epoch length decreasing with increasing sequence length. We used the same optimisation settings, with the exception of the learning rate, which we set to 3.33×10^{-6}.

Table 3.1 and Figure 3.5 contain results of different variants of our model and of the RATM tracker by Kahoú et al. [77] related works. *Spatial Att* does not use appearance attention, nor loss on attention parameters. *App Att* does not apply any loss on appearance attention, while *HART* uses all described modules; it is also our biggest model with 1.8 million parameters. Qualitative results in the form of a video with bounding boxes and attention are available online [4]. We implemented the RATM tracker of Kahoú et al. [77] and trained with the same hyperparameters as our framework, since both are closely related. It failed to learn even with the initial curriculum of five time-steps, as RATM cannot integrate the frame $\mathbf{x}_t$ into the estimate of $\mathbf{b}_t$ (it predicts location at the next time-step). Furthermore, it uses feature-space distance between ground-truth and predicted attention glimpses as the error measure, which is insufficient on a dataset with rich backgrounds. It did better when we initialised its feature extractor with weights of our trained model but, despite passing a few stags of the curriculum, it achieved very poor final performance.

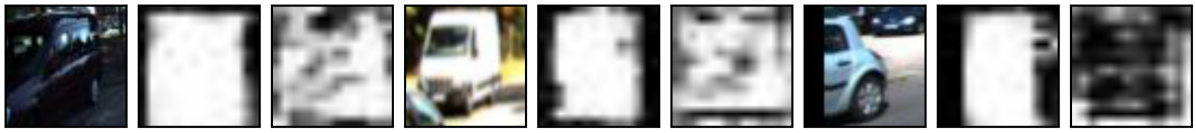
3.6 Discussion

The experiments in the previous section show that it is possible to track real-world objects with a recurrent attentive tracker. While similar to the tracker by Kahoú et al. [77], our approach uses additional building blocks, specifically: (1) bounding-box regression loss, (2) loss on spatial attention, (3) appearance attention with an additional loss term, and (4) combines all of these in a unified approach. We now discuss properties of these modules.

Spatial Attention Loss prevents Vanishing Gradients Our early experiments suggest that using only the tracking loss causes an instance of the vanishing gradient problem. Early in training, the system is not able to estimate object's motion correctly,

[4] `https://youtu.be/Vvkjm0FRGSs`

(a) The model with appearance attention loss (top) learns to focus on the tracked object, which prevents an ID swap when a pedestrian is occluded by another one (bottom).



(b) Three examples of glimpses and locations maps for a model with and without appearance loss (left to right). Attention loss forces the appearance attention to pick out only the tracked object, thereby suppressing distractors.

Figure 3.6: Glimpses and corresponding location maps for models trained with and without appearance loss. The appearance loss encourages the model to learn foreground/background segmentation of the input glimpse.

leading to cases where the extracted glimpse does not contain the tracked object or contains only a part thereof. In such cases, the supervisory signal is only weakly correlated with the model's input, which prevents learning. Even when the object is contained within the glimpse, the gradient path from the loss function is rather long, since any teaching signal has to pass to the previous timestep through the feature extractor stage. Penalising attention parameters directly seems to solve this issue.

Is Appearance Attention Loss Necessary? Given enough data and sufficiently high model capacity, appearance attention should be able to filter out irrelevant input features before updating the working memory. In general, however, this behaviour can be achieved faster if the model is constrained to do so by using an appropriate loss. Figure 3.6 shows examples of glimpses and corresponding location maps for a model with and without loss on the appearance attention. In Fig. 3.6a the model with loss on appearance attention is able to track a pedestrian even after it was occluded by another human. Figure 3.6b shows that, when not penalised, location map might not be very object-specific and can miss the object entirely (right-most figure). By using the appearance attention loss, we not only improve results but also make the

model more interpretable.

Spatial Attention Bias is Always Positive To condition the system on the object's appearance and make it independent of the starting location, we translate the initial bounding box to attention parameters, to which we add a learnable bias, and create the hidden state of LSTM from corresponding visual features. In our experiments, this bias always converged to positive values favouring attention glimpse slightly larger than the object bounding box. It suggests that, while discarding irrelevant features is desirable for object tracking, the system as a whole learns to trade off attention responsibility between the spatial and appearance based attention modules.

3.7 Conclusion

Inspired by the cascaded attention mechanisms found in the human visual cortex, this work presented a neural attentive recurrent tracking architecture suited for the task of object tracking. Beyond the biological inspiration, the proposed approach has a desirable computational cost and increased interpretability due to location maps, which select features essential for tracking. Furthermore, by introducing a set of auxiliary losses we are able to scale to challenging real world data, outperforming predecessor attempts and approaching state-of-the-art performance. Future research will look into extending the proposed approach to multi-object tracking, as unlike many single object tracking, the recurrent nature of the proposed tracker offers the ability to attend each object in turn.

Acknowledgements

We would like to thank Oiwi Parker Jones and Martin Engelcke for discussions and valuable insights and Neil Dhir for his help with editing the paper. Additionally, we would like to acknowledge the support of the UK's Engineering and Physical Sciences Research Council (EPSRC) through the Programme Grant EP/M019918/1 and the Doctoral Training Award (DTA). The donation from Nvidia of the Titan Xp GPU used in this work is also gratefully acknowledged.

Appendix

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

Title of Paper	Hierarchical Attentive Recurrent Tracking
Publication Status	Published
Publication Details	A. R. Kosiorek, A. Bewley, and I. Posner. "Hierarchical Attentive Recurrent Tracking". In: *Advances in Neural Information Processing Systems*. 2017. arXiv: 1706.09262. url: http://arxiv.org/abs/1706.09262.

Student Confirmation

Student Name:	Adam R. Kosiorek		
Contribution to the Paper	• Notice that the tracking problem can be addressed with RNNS. • Identify the need for attention mechanisms to improve computational efficiency and credit assignment problems. • Invent a novel second-level attention mechanism, which performs foreground-background segmentation, and a method for training based only on object bounding boxes • Implement the algorithm and validate it experimentally. • Write the paper.		
Signature		Date	06/01/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Ingmar Posner		
I confirm that the candidate made a substantial contribution to the publication.		
Signature		Date

This completed form should be included in the thesis, at the end of the relevant chapter.

4

End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning

Relational reasoning—the ability to model interactions and relations between objects—is valuable for robust multi-object tracking and pivotal for trajectory prediction. In this paper, we propose MOHART, a class-agnostic, end-to-end multi-object tracking and trajectory prediction algorithm, which explicitly accounts for permutation invariance in its relational reasoning. We explore a number of permutation invariant architectures and show that multi-headed self-attention outperforms the provided baselines and better accounts for complex physical interactions in a challenging toy experiment. We show on three real-world tracking datasets that adding relational reasoning capabilities in this way increases the tracking and trajectory prediction performance, particularly in the presence of ego-motion, occlusions, crowded scenes, and faulty sensor inputs. To the best of our knowledge, MOHART is the first fully end-to-end multi-object tracking from vision approach applied to real-world data reported in the literature.

4.1 Introduction

Real-world environments can be rich and contain countless types of interacting objects. Intelligent autonomous agents need to understand both the objects and interactions between them if they are to operate in those environments. This motivates the need for *class-agnostic* algorithms for tracking multiple objects—a capability that is not supported by the popular tracking-by-detection paradigm. In tracking-by-detection, objects are detected in each frame independently, e. g., by a pre-trained deep CNN such as YOLO (Redmon et al. [135]), and then linked across frames. Algorithms from this family can achieve high accuracy, provided sufficient labelled data to train the object detector, and given that all encountered objects can be associated with known classes, but fail when faced with objects from previously unseen categories.

Hierarchical attentive recurrent tracking (HART) is a recently-proposed, alternative method for single-object tracking (SOT), which can track arbitrary objects indicated by the user (see Chapter 3). This is done by providing an initial bounding-box, which may be placed over any part of the image, regardless of whether it contains an object or what class the object is. HART efficiently processes just the relevant part of an image using spatial attention; it also integrates object detection, feature extraction, and motion modelling into one network, which is trained fully end-to-end. Contrary

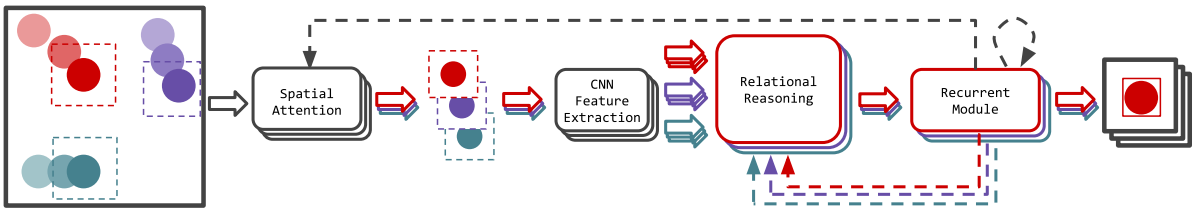


Figure 4.1.1: Multi-object hierarchical attentive recurrent tracking (MOHART). A glimpse is extracted for each object using a (fully differentiable) spatial attention mechanism. These glimpses are further processed with a CNN and fed into a relational reasoning module. A recurrent module which iterates over time steps allows for capturing of complex motion patterns. It also outputs spatial attention parameters and a feature vector per object for the relational reasoning module. Dashed lines indicate temporal connections (from time step t to $t+1$). The entire pipeline operates in parallel for the different objects, only the relational reasoning module allows for exchange of information between tracking states of each object. MOHART is an extension of HART (a single-object tracker), which features the same pipeline without the relational reasoning module.

to tracking-by-detection, where only one video frame is typically processed at any given time to generate bounding box proposals, end-to-end learning in HART allows for discovering complex visual and spatio-temporal patterns in videos, which is conducive to inferring what an object is and how it moves.

In the original formulation, HART is limited to the single-object modality—as are other existing end-to-end trackers (Gordon et al. [46], Kahoú et al. [77], and Rasouli Danesh et al. [132]). In this work, we present MOHART, a class-agnostic tracker with complex relational reasoning capabilities provided by a multi-headed self-attention module (Lee et al. [101] and Vaswani et al. [171]). MOHART infers the latent state of every tracked object in parallel, and uses self-attention to inform per-object states about other tracked objects. This helps to avoid performance loss under self-occlusions of tracked objects or strong camera motion. Moreover, since the model is trained end-to-end, it is able to learn how to manage faulty or missing sensor inputs. See Fig. 4.1.1 for a high-level illustration of MOHART.

In order to track objects, MOHART estimates their states, which can be naturally used to predict future trajectories over short temporal horizons, which is especially useful for planning in the context of autonomous agents. MOHART can be trained simultaneously for object tracking and trajectory prediction at the same time, thereby increasing statistical efficiency of learning. In contrast to prior art, where trajectory prediction and object tracking are usually addressed as separate problems with unrelated solutions, our work show trajectory prediction and object tracking are best addressed jointly.

Section 4.2 describes prior art in tracking-by-detection, end-to-end tracking and predestrian trajectory prediction. In Section 4.3, we describe our approach, which uses a permutation-invariant self-attention module to enable tracking multiple objects end-to-end with relational reasoning. Section 4.4 contrasts our approach with multi-object trackers which do not explicitly enforce permutation invariance but have the capacity to learn it, simpler permutation-invariant architectures, as well as multiple single-object trackers running in parallel. We show that multi-headed self-attention

significantly outperforms other approaches. Finally, in Section 4.5, we apply MOHART to real world datasets and show that permutation-invariant relational reasoning leads to consistent performance improvement compared to HART both in tracking and trajectory prediction.

4.2 Related Work

Tracking-by-Detection Vision-based tracking approaches typically follow a tracking-by-detection paradigm: objects are first detected in each frame independently, and then a tracking algorithm links the detections from different frames to propose a coherent trajectory (Bae and Yoon [6], Keuper et al. [83], Milan et al. [114], and L. Zhang et al. [189]). Motion models and appearance are often used to improve the association between detected bounding-boxes in a postprocessing step. Tracking-by-detection algorithms currently provide the state-of-the-art in multi-object tracking on common benchmark suites, and we fully acknowledge that MOHART is not competitive at this stage in scenarios where high-quality detections are available for each frame. MOHART can in principle be equipped with the ability to use bounding boxes provided by an object detector, but this is beyond the scope of this project.

End-to-End Tracking A newly established and much less explored stream of work approaches tracking in an end-to-end fashion. A key difficulty here is that extracting an image crop (according to bounding-boxes provided by a detector), is non-differentiable and results in high-variance gradient estimators. Kahoú et al. [77] propose an end-to-end tracker with soft spatial-attention using a 2D grid of Gaussians instead of a hard bounding-box. HART draws inspiration from this idea, employs an additional attention mechanism, and shows promising performance on the real-world KITTI dataset (Chapter 3). HART forms the foundation of this work. It has also been extended to incorporate depth information from RGBD cameras (Rasouli Danesh et al. [132]). Gordon et al. [46] propose an approach in which the crop corresponds to the scaled up previous bounding-box. This simplifies the approach, but does not allow the model to learn where to look— i. e., no gradient is backpropagated through crop

coordinates. To the best of our knowledge, there are no successful implementations of any such end-to-end approaches for multi-object tracking beyond SQAIR (Kosiorek et al. [91]), which works only on datasets with static backgrounds. On real-world data, the only end-to-end approaches correspond to applying multiple single-object trackers in parallel—a method which does not leverage the potential of scene context or inter-object interactions.

Pedestrian trajectory prediction Predicting pedestrian trajectories has a long history in computer vision and robotics. Initial research modelled social forces using hand-crafted features (Lerner et al. [103], Pellegrini et al. [126], Trautman and Krause [165], and Yamaguchi et al. [185]) or MDP-based motion transition models (Rudenko et al. [142]), while more recent approaches learn from context information, e. g., positions of other pedestrians or landmarks in the environment. Social-LSTM (Alahi et al. [3]) employs a long short-term memory (LSTM) to predict pedestrian trajectories and uses max-pooling to model global social context. Attention mechanisms have been employed to query the most relevant information, such as neighbouring pedestrians, in a learnable fashion (Fernando et al. [39], Sadeghian et al. [145], and Su et al. [160]). Apart from relational learning, context (Varshneya and Srinivasaraghavan [170]), periodical time information (Sun et al. [161]), and constant motion priors (Schöller et al. [151]) have proven effective in predicting long-term trajectories.

Our work stands apart from this prior art by not relying on ground truth tracklets. It addresses the more challenging task of working directly with visual input, performing tracking, modelling interactions, and, depending on the application scenario, simultaneously predicting future motions. As such, it can also be compared to Visual Interaction Networks (VIN) (Watters et al. [178]), which use a CNN to encode three consecutive frames into state vectors—one per object—and feed these into a RNN, which has an Interaction Network (Battaglia et al. [9]) at its core. More recently, Relational Neural Expectation Maximization (R-NEM) has been proposed as an unsupervised approach which combines scene segmentation and relational reasoning (**Steenkiste2018**). Both VINs and R-NEM make accurate predictions in

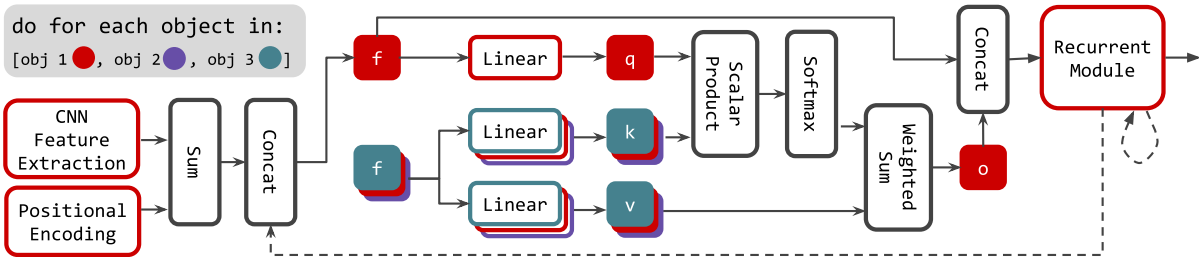


Figure 4.3.1: The relational reasoning module in MOHART based on multi-headed self-attention. Here, we show the computation of the interaction of the red object with all other objects. Object representations $f_{t,m}$ are computed using visual features, positional encoding and the hidden state from the recurrent module. These are linearly projected onto keys (k), queries (q), and values (v) to compute a weighted sum of interactions between objects, yielding an interaction vector $o_{t,m}$. Subscripts t, m are dropped from all variables for clarity of presentation, so is the splitting into multiple heads.

physical scenarios, but, to the best of our knowledge, have not been applied to real world data.

4.3 Recurrent Multi-Object Tracking with Self-Attention

This section describes the model architecture in Fig. 4.1.1. We start by describing the hierarchical attentive recurrent tracking (HART) algorithm of Chapter 3, and then follow with an extension of HART to tracking multiple objects, where multiple instances of HART communicate with each other using multi-headed attention to facilitate relational reasoning. We also explain how this method can be extended to trajectory prediction instead of just tracking.

4.3.1 Hierarchical Attentive Recurrent Tracking (hart)

HART is an attention-based recurrent algorithm, which can efficiently track single objects in a video. It uses a spatial attention mechanism to extract a *glimpse* $\mathbf{g}_t$, which corresponds to a small crop of the image $\mathbf{x}_t$ at time-step t, containing the object of interest. This allows it to dispense with the processing of the whole image and can significantly decrease the amount of computation required. HART uses a CNN to convert the glimpse $\mathbf{g}_t$ into features $\mathbf{f}_t$, which then update the hidden state $\mathbf{h}_t$ of a LSTM core. The hidden state is used to estimate the current bounding-box $\mathbf{b}_t$,

spatial attention parameters for the next time-step $\mathbf{a}_{t+1}$, as well as object appearance. Importantly, the recurrent core can learn to predict complicated motion conditioned on the past history of the tracked object, which leads to relatively small attention glimpses—contrary to CNN-based approaches (Held et al. [59] and Valmadre et al. [169]), HART does not need to analyse large regions-of-interest to search for tracked objects. In the original paper, HART processes the glimpse with an additional ventral and dorsal stream on top of the feature extractor. Early experiments have shown that this does not improve performance on the MOTChallenge dataset, presumably due to the oftentimes small objects and overall small amount of training data. Further details are provided in Section 4.B.

The algorithm is initialised with a bounding-box[1] $\mathbf{b}_1$ for the first time-step, and operates on a sequence of raw images $\mathbf{x}_{1:T}$. For time-steps $t \geqslant 2$, it recursively outputs bounding-box estimates for the current time-step and predicted attention parameters for the next time-step. The performance is measured as intersection-over-union (IoU) averaged over all time steps in which an object is present, excluding the first time step.

Although HART can track arbitrary objects, it is limited to tracking one object at a time. While it can be deployed on several objects in parallel, different HART instances have no means of communication. This results in performance loss, as it is more difficult to identify occlusions, ego-motion and object interactions. Below, we propose an extension of HART which remedies these shortcomings.

4.3.2 Multi-Object Hierarchical Attentive Recurrent Tracking (mohart)

Multi-object support in HART requires the following modifications. Firstly, in order to handle a dynamically changing number of objects, we apply HART to multiple objects in parallel, where all parameters between HART instances are shared. We refer to each HART instance as a *tracker*. Secondly, we introduce a presence variable $p_{t,m}$ for object m. It is used to mark whether an object should interact with other objects, as

[1] We can use either a ground-truth bounding-box or one provided by an external detector; the only requirement is that it contains the object of interest.

well as to mask the loss function (described in Chapter 3) for the given object when it is not present. In this setup, parallel trackers cannot exchange information and are conceptually still single-object trackers, which we use as a baseline, referred to as HART (despite it being an extension of the original algorithm). Finally, to enable communication between trackers, we augment HART with an additional step between feature extraction and the LSTM.

For each object, a glimpse is extracted and processed by a CNN (see Fig. 4.1.1). Furthermore, spatial attention parameters are linearly projected on a vector of the same size and added to this representation, acting as a positional encoding. This is then concatenated with the hidden state of the recurrent module of the respective object (see Fig. 4.3.1). Let $\mathbf{f}_{t,m}$ denote the resulting feature vector corresponding to the m^{th} object, and let $\mathbf{f}_{t,1:M}$ be the set of such features for all objects. Since different objects can interact with each other, it is necessary to use a method that can inform each object about the effects of their interactions with other objects. Moreover, since features extracted from different objects comprise a set, this method should be permutation-equivariant, i. e., the results should not depend on the order in which object features are processed. Therefore, we use the multi-head self-attention block (SAB, Lee et al. [101]), which is able to account for higher-order interactions between set elements when computing their representations. Intuitively, in our case, SAB allows any of the trackers to query other trackers about attributes of their respective objects, e. g., distance between objects, their direction of movement, or their relation to the camera. This is implemented as follows,

$$Q = W_q \mathbf{f}_{1:M} + b_q\,, \qquad K = W_k \mathbf{f}_{1:M} + b_k\,, \qquad V = W_v \mathbf{f}_{1:M} + b_v\,, \tag{4.1}$$

$$O_i = \text{softmax}\left(Q_i K_i^T\right) V_i\,, \qquad i = 1, \ldots, H\,, \tag{4.2}$$

$$o_{1:M} = O = \text{concat}(O_i, \ldots, O_H)\,, \tag{4.3}$$

where o_m is the output of the relational reasoning module for object m. Time-step subscripts are dropped to decrease clutter. In Eq. 4.1, each of the extracted features $\mathbf{f}_{t,m}$ is linearly projected into a triplet of key $\mathbf{k}_{t,m}$, query $\mathbf{q}_{t,m}$ and value $\mathbf{v}_{t,m}$ vectors. Together, they comprise K, Q and V matrices with M rows and d_q, d_k, d_k columns,

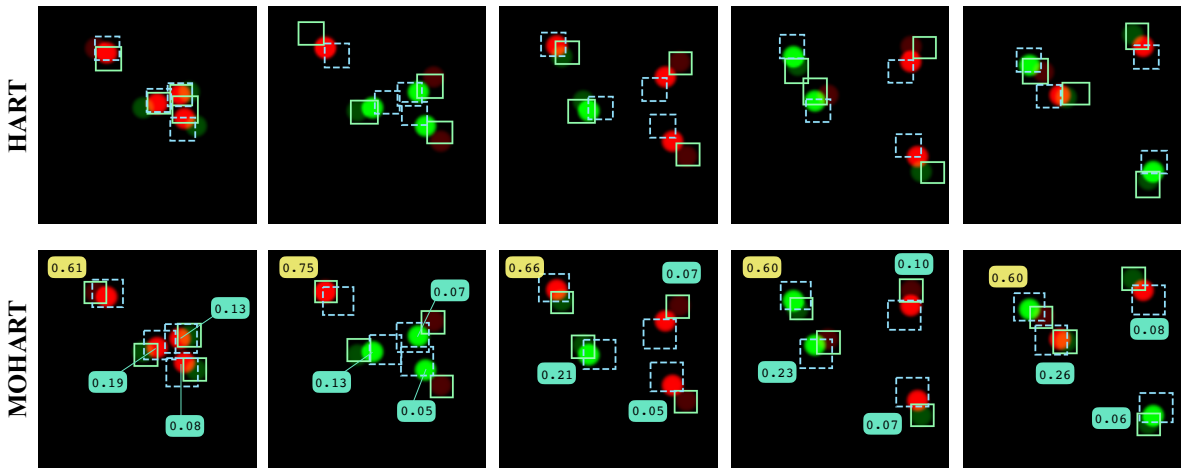


Figure 4.3.2: A scenario constructed to be impossible to solve without relational reasoning. Circles of the same colour repel each other, circles of different colour attract each other. Crucially, each circle is randomly assigned its identity in each time step. Hence, the algorithm can not infer the forces exerted on one object without knowledge of the state of the other objects in the current time step. The forces in this scenario scale with $1/\sqrt{r}$ and the algorithm was trained to predict one time step into the future. HART (top) is indeed unable to predict the future location of the objects accurately. The achieved average IoU is 47%, which is only slightly higher than predicting the objects to have the same position in the next time step as in the current one (34%). Using the relational reasoning module, MOHART (bottom) is able to make meaningful predictions (76% IoU). The numbers in the bottom row indicate the self-attention weights from the perspective of the top left tracker (yellow number box). Interestingly, the attention scores have a strong correlation with the interaction strength (which scales with distance) without receiving supervision.

respectively. K, Q and V are then split up into multiple heads $H \in \mathbb{N}_+$, which allows to query different attributes by comparing and aggregating different projection of features. Multiplying $Q_i K_i^T$ in Eq. 4.2 allows to compare every query vector $\mathbf{q}_{t,m,i}$ to all key vectors $\mathbf{k}_{t,1:M,i}$, where the value of the corresponding dot-products represents the degree of similarity. Similarities are then normalised via a softmax operation and used to aggregate values V. Finally, outputs of different attention heads are concatenated in Eq. 4.3. SAB produces M output vectors, one for each input, which are then concatenated with corresponding inputs and fed into separate LSTMs for further processing, as in HART—see Fig. 4.1.1.

MOHART is trained fully end-to-end, contrary to other tracking approaches. It maintains a hidden state, which can contain information about the object's motion. One benefit is that in order to predict future trajectories, one can simply feed black

frames into the model. Our experiments show that the model learns to fall back on the motion model captured by the LSTM in this case.

4.3.3 Multi-Object Baselines

Multilayer perceptron (mlp) In this version, the representations of all objects are concatenated and fed into a fully connected layer followed by ELU activations. The output is then again concatenated to the unaltered feature vector of each object. This concatenated version is then fed to the recurrent module of HART. This way of exchanging information allows for universal function approximation (in the limit of infinite layer sizes) but does not impose permutation invariance.

DeepSets Here, the learned representations of the different objects are summed up instead of concatenated and then divided by total number of objects. This is closely related to DeepSets (Zaheer et al. [188]) and allows for universal function approximation of all permutation invariant functions (Wagstaff et al. [175]).

Max-Pooling Similar to DeepSets, but using max-pooling as the permutation invariant operation. This way of exchanging information is used, e.g., by Alahi et al. [3] who predict future pedestrian trajectories from ground truth tracklets in coordinate space.

4.4 Validation on Simulated Data

We test and compare the relational reasoning capabilities of the proposed algorithms on a toy domain. The domain is a 2D squared box which contains circular objects with approximated elastic collisions (energy and momentum conservation) between objects and with walls. To investigate how the model understands motion patterns and interactions between objects, we train it to predict future object locations in contrast to traditional tracking.

In the first experiment, each circle exerts repulsive forces on each other, where the force scales with $1/r$, r being the distance between them. Predicting the future

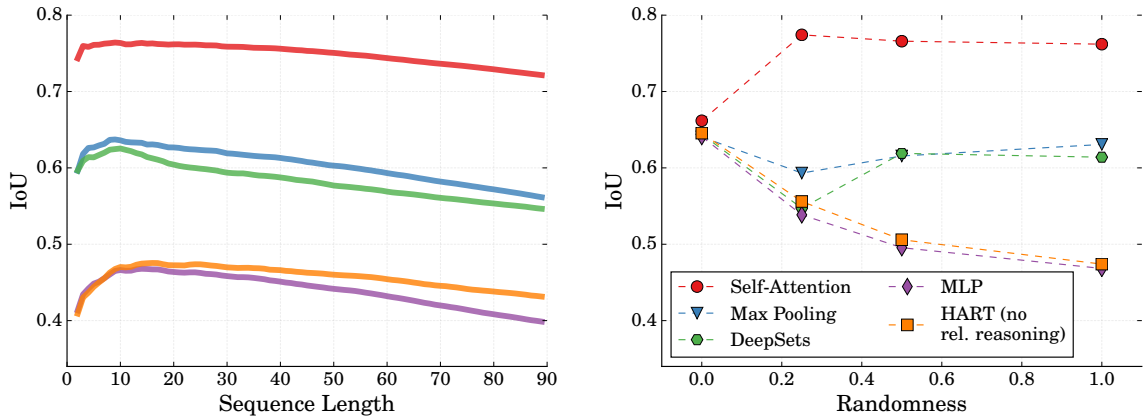


Figure 4.4.1: Left: average IoU over sequence length for different implementations of relational reasoning on the toy domain shown in Fig. 4.3.2 (randomness = 1.0). Right: performance depending on how often agents are re-assigned identities randomly (sequence length 15). The higher the randomness, the less static the force field is and the more vital relational reasoning is. For randomness = 0.0, identities still have to be reassigned in some cases in order to prevent deadlocks, this leads to a performance loss for all models, which explains lower performance of self-attention for randomness = 0.0.

location just using the previous motion of one object (i.e. without relational reasoning) accurately is therefore challenging. We show that HART as an end-to-end single-object tracker is able to capture complex motion patterns and leverage these to make accurate predictions (see Section 4.C). This indicates that HART is able to draw conclusions about the (deterministic, but not static) force field.

In the second experiment, we introduce randomness, rendering the scenario not solvable for a single object tracker as it requires knowledge about the state of the other objects and relational reasoning (see Fig. 4.3.2). In each time step, we assign a colour-coded identity to the objects. Objects of the same identity repel each other, object of different identities attract each other (the objects can be thought of as electrons and protons). The qualitative results in Fig. 4.3.2 show that MOHART, using self-attention for relational reasoning, is able to capture these interactions with high accuracy. Figure 4.4.1 (left) shows a quantitative comparison of augmenting HART with different relational reasoning modules when identities are re-assigned in every timestep (randomness = 1.0). Exchanging information between trackers of different objects in the latent space with an MLP leads to slightly worse performance than the HART baseline, while simple max-pooling performs significantly better (ΔIoU $\sim$ 17%). This can be explained through the permutation invariance of the problem: the list

of latent representation of the different objects has no meaningful order and the output of the model should therefore be invariant to the ordering of the objects. The MLP is in itself not permutation invariant and therefore prone to overfit to the (meaningless) order of the objects in the training data. Max-pooling, however, is permutation invariant and can in theory, despite its simplicity, be used to approximate any permutation invariant function given a sufficiently large latent space (Wagstaff et al. [175]). Max-pooling is often used to exchange information between different tracklets, e.g., in the trajectory prediction domain (Alahi et al. [3] and Gupta et al. [53]). However, self-attention, allowing for learned querying and encoding of information, solves the relational reasoning task significantly more accurately. In Fig. 4.4.1 (right), the frequency with which object identities are reassigned randomly is varied. The results show that, in a deterministic environment, tracking does not necessarily profit from relational reasoning - even in the presence of long-range interactions. The less random, the more static the force field is and a static force field can be inferred from a small number of observations (see Fig. 4.C.1). This does not mean that all stochastic environments profit from relational reasoning. What these experiments indicate is that tracking can not be expected to profit from relational reasoning by default in any environment, but instead in environments which feature (potentially non-deterministic) dynamics and predictable interactions.

4.5 Relational Reasoning in Real-World Tracking

Having established that MOHART is capable of performing complex relational reasoning, we now test the algorithm on three real world datasets and analyse the effects of relational reasoning on performance depending on dataset and task. We find consistent improvements of MOHART compared to HART throughout. Relational reasoning yields particularly high gains for scenes with ego-motion, crowded scenes, and simulated faulty sensor inputs.

Figure 4.5.1: Camera blackout experiment on a street scene from the MOTChallenge dataset with strong ego-motion. Solid boxes are MOHART predictions (for $t \geqslant 2$), faded bounding boxes indicate object locations in the first frame. As the model is trained end-to-end, MOHART learns to fall back onto its internal motion model if no new observations are available (black frames). As soon as new observations come in, the model 'snaps' back onto the tracked objects.

4.5.1 Experimental Details

We investigate three qualitatively different datasets: the MOTChallenge dataset (Milan et al. [113]), the UA-DETRAC dataset (Wen et al. [180]), and the Stanford Drone dataset (Robicquet et al. [140]). To increase scene dynamics and make the tracking/prediction problems more challenging, we sub-sample some of the high framerate scenes with a stride of two, resulting in scenes with 7-15 frames per second. Training and architecture details are given in Sections 4.A and 4.B. We conduct experiments in three different modes:

Tracking. The model is initialised with the ground truth bounding boxes for a set of objects in the first frame. It then consecutively sees the following frames and predicts the bounding boxes. The sequence length is 30 time steps and the performance is measured as intersection over union (IoU) averaged over the entire sequence excluding the first frame. This algorithm is either applied to the entire dataset or subsets of it to

study the influence of certain properties of the data.

Camera Blackout. This simulates unsteady or faulty sensor inputs. The setup is the same as in *Tracking*, but sub-sequences of the input are replaced with black images. The algorithm is expected to recognise that no new information is available and that it should resort to its internal motion model.

Prediction. Testing MOHART's ability to capture motion patterns, only the first two frames are shown to the model followed by three black frames. IoU is measured seperately for each time step.

Table 4.5.1: Tracking performance on the MOTChallenge dataset measured in IoU.

	Entire Dataset	Only Ego-Motion	No Ego-Motion	Crowded Scenes	Camera Blackout
MOHART	**68.5%**	**66.9%**	**64.7%**	**69.1%**	**63.6%**
HART	66.6%	64.0%	62.9%	66.9%	60.6%
Δ	1.9%	2.9%	1.8%	2.2%	3.0%

Table 4.5.2: UA-DETRAC Dataset

	All	Crowded Scenes	Camera Blackout
MOHART	68.1%	**69.5%**	**64.2%**
HART	**68.4%**	68.6%	53.8%
Δ	-0.3%	0.9%	0.4%

Table 4.5.3: Stanford Drone Dataset

	All	Camera Blackout	CamBlack Bikes
	57.3%	**53.3%**	**53.3%**
	56.1%	52.6%	50.7%
	1.2%	0.7%	2.6%

4.5.2 Results and Analysis

On the MOTChallenge dataset, HART achieves 66.6% intersection over union (see Table 4.5.1), which in itself is impressive given the small amount of training data of only 5225 training frames and no pre-training. MOHART achieves 68.5% (both numbers are averaged over 5 runs, independent samples t-test resulted in $p < 0.0001$). The performance gain increases when only considering ego-motion data. This is readily explained: movements of objects in the image space due to ego-motion are correlated and can therefore be better understood when combining information from movements of multiple objects, i.e. performing relational reasoning. In another

ablation, we filtered for only crowded scenes by requesting five objects to be present for, on average, 90% of the frames in a sub-sequence. For the MOT-Challenge dataset, this only leads to a minor increase of the performance gain of MOHART indicating that the dataset exhibits a sufficient density of objects to learn interactions. The biggest benefit from relational reasoning can be observed in the *camera blackout* experiments (setup explained in Section 4.5.1). Both HART and MOHART learn to rely on their internal motion models when confronted with black frames and propagate the bounding boxes according to the previous movement of the objects. It is unsurprising that this scenario profits particularly from relational reasoning. Qualitative tracking and *camera blackout* results are shown in Fig. 4.5.1 and Section 4.E.

Tracking performance on the UA-DETRAC dataset only profits from relational reasoning when filtering for crowded scenes (see Table 4.5.2). The fact that the performance of MOHART is slightly worse on the vanilla dataset ($\Delta = -0.3\%$) can be explained with more overfitting. As there is no exchange between trackers for each object, each object constitutes an independent training sample.

The Stanford drone dataset (see Table 4.5.3) is different to the other two—it is filmed from a birds-eye view. The scenes are more crowded and each object covers a small number of pixels, rendering it a difficult problem for tracking. The dataset was designed for trajectory prediction—a setup where an algorithm is typically provided with ground-truth tracklets in coordinate space and potentially an image as context information. The task is then to extrapolate these tracklets into the future. The tracking performance profits from relational reasoning more than on the UA-DETRAC dataset but less than on the MOTChallenge dataset. The performance gain on the *camera blackout* experiments are particularly strong when only considering cyclists.

In the *prediction* experiments (see Section 4.D), MOHART consistently outperforms HART. On both datasets, the model outperforms a baseline which uses momentum to linearly extrapolate the bounding boxes from the first two frames. This shows that even from just two frames, the model learns to capture motion models which are more complex than what could be observed from just the bounding boxes (i.e.

momentum), suggesting that it uses visual information (HART & MOHART) as well as relational reasoning (MOHART).

4.6 Conclusion

With MOHART, we introduce an end-to-end multi-object tracker that is capable of capturing complex interactions and leveraging these for precise predictions as experiments both on toy and real world data show. However, the experiments also show that the benefit of relational reasoning strongly depends on the nature of the data. The toy experiments showed that in an entirely deterministic world relational reasoning was much less important than in a stochastic environment. Amongst the real-world dataset, the highest performance gains from relational reasoning were achieved on the MOTChallenge dataset, which features crowded scenes, ego-motion and occlusions.

Acknowledgements

We thank Stefan Saftescu for his contributions, particularly for integrating the Stanford Drone Dataset, and Adam Golinski as well as Stefan Saftescu for proof-reading. This research was funded by the EPSRC AIMS Centre for Doctoral Training at Oxford University and an EPSRC Programme Grant (EP/M019918/1). We acknowledge use of Hartree Centre resources in this work. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. The Centre aims to develop and demonstrate next generation software, optimised to take advantage of the move towards exa-scale computing.

Appendix

4.A Experimental Details

The MOTChallenge and the UA-DETRAC dataset discussed in this section are intended to be used as a benchmark suite for multi-object-tracking in a tracking-by-detection paradigm. Therefore, ground truth bounding boxes are only available for the training datasets. The user is encouraged to upload their model which performs tracking in a data association paradigm leveraging the provided bounding box proposals from an external object detector. As we are interested in a different analysis (IoU given inital bounding boxes), we divide the training data further into training and test sequences. To make up for the smaller training data, we extend the MOTChallenge 2017 dataset with three sequences from the 2015 dataset (ETH-Sunnyday, PETS09-S2L1, ETH-Bahnhof). We use the first 70% of the frames of each of the ten sequences for training and the rest for testing. Sequences with high frame rates (30Hz) are sub-sampled with a stride of two. For the UA-DETRAC dataset, we split the 60 available sequences into 44 training sequences and 16 test sequences. For the considerably larger Stanford Drone dataset we took three videos of the scene *deathCircle* for training and the remaining two videos from the same scene for testing. The videos of the drone dataset were also sub-sampled with a stride of two to increase scene dynamics.

4.B Architecture Details

The architecture details were chosen to optimise HART performance on the MOTChallenge dataset. They deviate from the original HART implementation (Chapter 3) as follows: A presence variable predicts whether an object is in the scene and

successfully tracked. This is trained with a binary cross entropy loss. The maximum number of objects to be tracked simultaneously was set to 5 for the UA-DETRAC and MOTChallenge dataset. For the more crowded Stanford drone dataset, this number was set to 10. The feature extractor is a three layer convolutional network with a kernel size of 5, a stride of 2 in the first and last layer, 32 channels in the first two layers, 64 channels in the last layer, ELU activations, and skip connections. This converts the initial $32 \times 32 \times 3$ glimpse into a $7 \times 7 \times 64$ feature representation. This is followed by a fully connected layer with a 128 dimensional output and an elu activation. The spatial attention parameters are linearly projected onto 128 dimensions and added to this feature representation serving as a positional encoding. The LSTM has a hidden state size of 128. The self-attention unit in MOHART comprises linear projects the inputs to dimensionality 128 for each keys, queries and values. For the real-world experiments, in addition to the extracted features from the glimpse, the hidden states from the previous LSTM state are also fed as an input by concatinating them with the features. In all cases, the output of the attention module is concatenated to the input features of the respective object.

As an optimizer, we used RMSProp with momentum set to 0.9 and learning rate $5 * 10^{-6}$. For the MOTChallenge dataset and the UA-DETRAC dataset, the models were trained for 100,000 iterations of batch size 10 and the reported IoU is exponentially smoothed over iterations to achieve lower variance. For the Stanford Drone dataset, the batch size was increased to 32, reducing time to convergence and hence model training to 50,000 iterations.

4.C Deterministic Toy Domain

In our first experiment in the toy domain (Figure 4.C.1), four circles each exert repulsive forces on each other, where the force scales with $1/r$, r being their distance. HART is applied four times in parallel and is trained to predict the location of each circle three time steps into the future. The different forces from different objects lead to a non-trivial force field at each time step. Predicting the future location just using

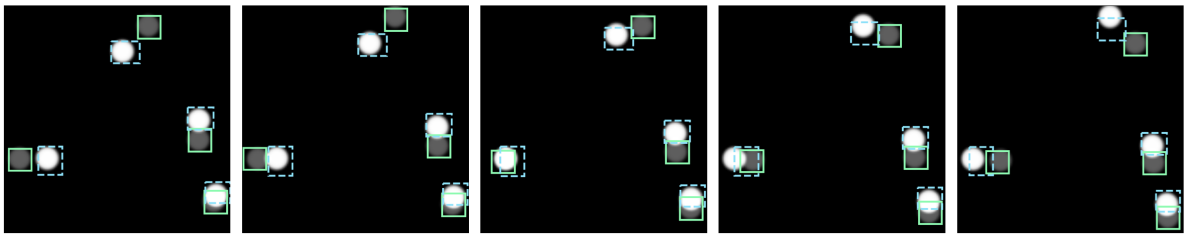
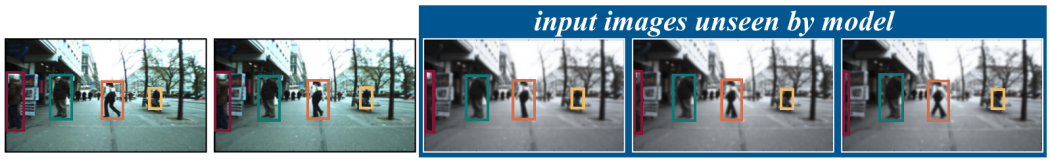


Figure 4.C.1: HART single object tracking applied four times in parallel and trained to predict the location of each circle three time steps into the future. Dashed lines indicate spatial attention, solid lines are predicted bounding boxes, faded circles show ground truth location at $T + 3$. Each circle exerts repulsive forces on each other, where the force scales with $1/r$, r being their distance.

the previous motion of one object (Figure 4.C.1 shows that each spatial attention box covers only the current object) accurately is therefore challenging. Surprisingly, the single object tracker solves this task with an average of 95% IoU over sequences of 15 time steps. This shows the efficacy of end-to-end tracking to capture complex motion patterns and use them to predict future locations. This, of course, could also be used to generate robust bounding boxes for a tracking task.

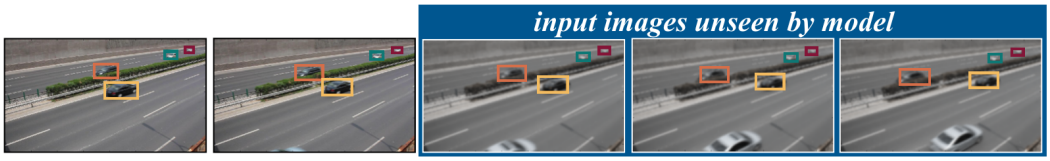
4.D Prediction Experiments

In the results from the *prediction* experiments (see Figure 4.D.1) MOHART consistently outperforms HART. On both datasets, the model outperforms a baseline which uses momentum to linearly extrapolate the bounding boxes from the first two frames. This shows that even from just two frames, the model learns to capture motion models which are more complex than what could be observed from just the bounding boxes (i.e. momentum), suggesting that it uses visual information (HART & MOHART) as well as relational reasoning (MOHART). The strong performance gain of MOHART compared to HART on the UA-DETRAC dataset, despite the small differences for tracking on this dataset, can be explained as follows: this dataset features little interactions but strong correlations in motion. Hence when only having access to the first two frames, MOHART profits from estimating the velocities of multiple cars simultaneously.



MOHART	–	**86.4%**	**80.0%**	**74.0%**	**68.4%**
HART	–	85.3%	79.2%	73.1%	67.4%
Momentum	–	–	78.2%	70.9%	63.9%

(a) Prediction results on the MOTChallenge dataset Milan et al. [113].



MOHART	–	**87.8%**	**81.3%**	**75.5%**	**70.0%**
HART	–	86.6%	79.6%	72.6%	66.1%
Momentum	–	-	80.5%	73.6%	67.2%

(b) Prediction results on the UA-DETRAC dataset (crowded scenes only) Wen et al. [180].

Figure 4.D.1: Peeking into the future. Only the first two frames are shown to the tracking algorithm followed by three black frames. MOHART learns to fall back on its internal motion model when no observation (i.e. only a black frame) is available. The reported IoU scores show the performance for the respective frames 0, 1, 2, and 3 time steps into the future.

4.E Qualitative Tracking Results

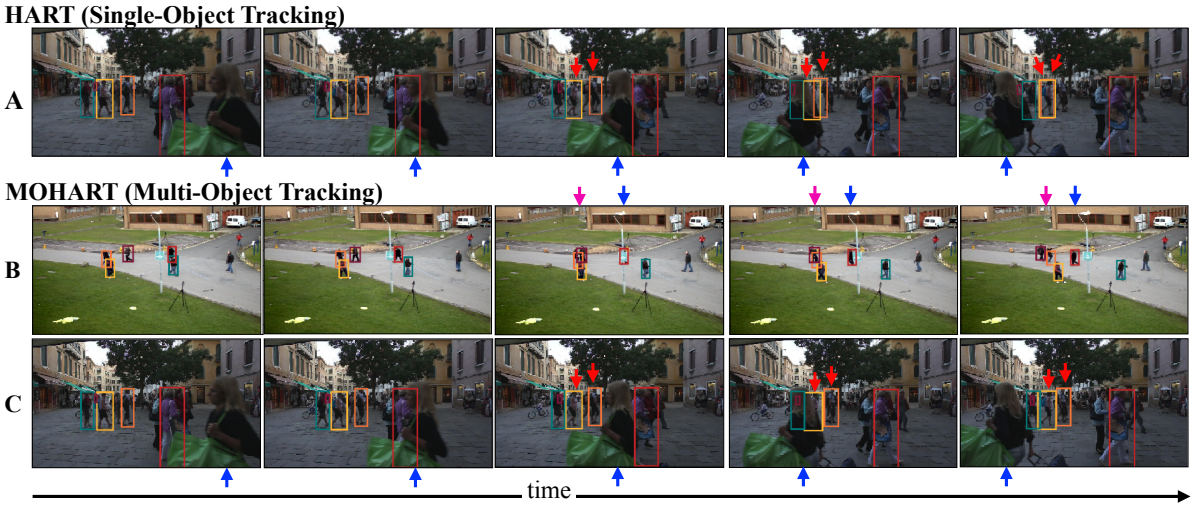


Figure 4.E.1: Tracking examples of both HART and MOHART. Coloured boxes are bounding boxes predicted by the model, arrows point at challenging aspects of the scenes. (A) & (C): Each person being tracked is temporarily occluded by a woman walking across the scene (blue arrows). MOHART, which includes a relational reasoning module, handles this more robustly (compare red arrows).

In Section 4.5, we tested MOHART on three different real world data sets and in a

number of different setups. Figure 4.E.1 shows qualitative results both for HART and MOHART on the MOTChallenge dataset.

Furthermore, we conducted a set of camera blackout experiments to test MOHART's capability of dealing with faulty sensor inputs. While traditional pipeline methods require careful consideration of different types of corner cases to properly handle erroneous sensor inputs, MOHART is able to capture these automatically, especially when confronted with similar issues in the training scenarios. To simulate this, we replace subsequences of the images with black frames. Figure 4.E.2 and Figure 4.5.1 show two such examples from the test data together with the model's prediction. MOHART learns not to update its internal model when confronted with black frames and instead uses the LSTM to propagate the bounding boxes. When proper sensor input is available again, the model uses this to make a rapid adjustment to its predicted location and 'snap' back onto the object. This works remarkably well in both the presence of occlusion (Figure 4.E.2) and ego-motion (Figure 4.5.1). Tables 4.5.1 to 4.5.3 show that the benefit of relational reasoning is particularly high in these scenarios

specifically. These experiments can also be seen as a proof of concept of MOHART's capabalities of predicting future trajectories—and how this profits from relational reasoning.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

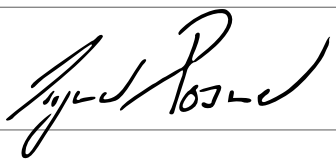
Title of Paper	End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning
Publication Status	Submitted for Publication
Publication Details	F. B. Fuchs, A. R. Kosiorek, L. Sun, O. P. Jones, and I. Posner. "End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning". In: *CoRR*. 2019. arXiv: 1907.12887. url: https://arxiv.org/abs/1907.12887.

Student Confirmation

Student Name:	Adam R. Kosiorek		
Contribution to the Paper	• Propose using self-attention for exchanging information between independent trackers. • Help in designing and implementing the algorithm. • Help in designing and interpreting the experiments as well as writing the paper.		
Signature		Date	06/01/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Ingmar Posner		
I confirm that the candidate made a substantial contribution to the publication.		
Signature		Date

This completed form should be included in the thesis, at the end of the relevant chapter.

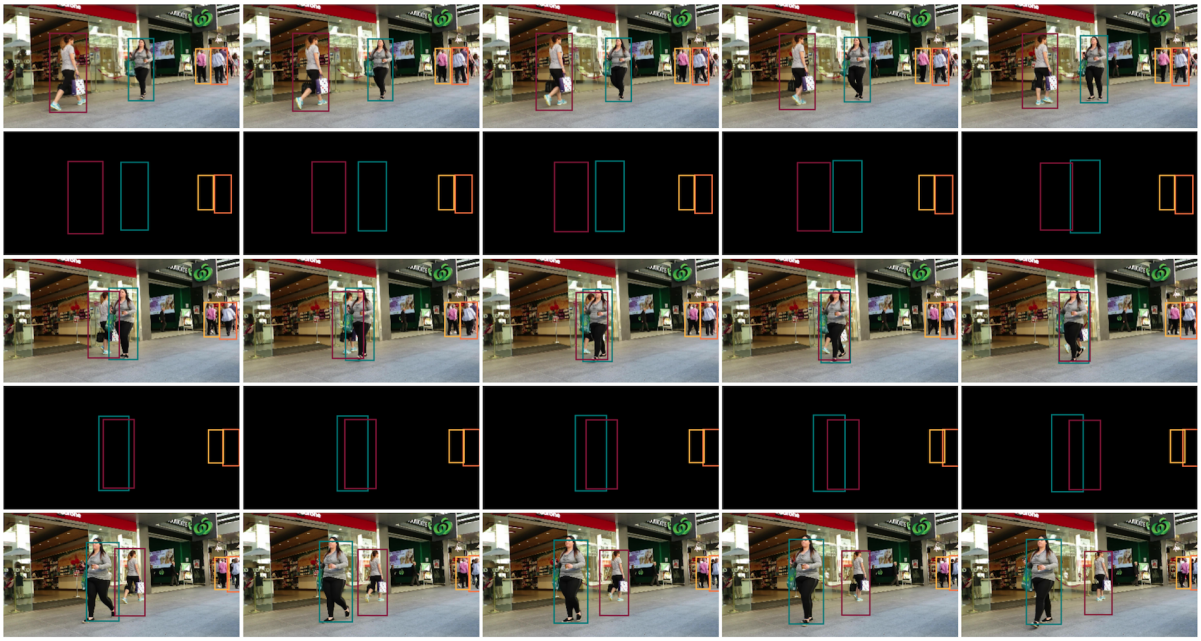


Figure 4.E.2: Camera blackout experiment on a pedestrian street scene from the MOTChallenge dataset without ego-motion. Subsequent frames are displayed going from top left to bottom right. Shown are the inputs to the model (some of them being black frames, i.e. arrays of zeroes) and bounding boxes predicted by MOHART (coloured boxes). This scene is particularly challenging as occlusion and missing sensor input coincide (fourth row).

5

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

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. [38]), 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.

5.1 Introduction

The ability to identify objects in their environments and to understand relations between them is a cornerstone of human intelligence (Kemp and Tenenbaum [81]). 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 et al. [99]), 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 et al. [100]) or in tasks requiring relational reasoning (Santoro et al. [146]). Structure can also make neural networks useful in new contexts by significantly improving generalization, data efficiency (Jacobsen et al. [71]) or extending their capabilities to unstructured inputs (Graves et al. [47]).

AIR, introduced by Eslami et al. [38], 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. 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 5.4.4 in Section 5.4.1).

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 5.2, we describe the generative model and inference of AIR. In Section 5.3, we discuss its limitations and how it can be improved, thereby introducing SQAIR, our extension of AIR to image sequences. In Section 5.4, we demonstrate the model on a dataset of multiple moving MNIST digits (Section 5.4.1) and compare it against AIR trained on each frame and Variational Recurrent Neural Network (VRNN) of Chung et al. [20] 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 5.4.2 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 dataset[1] and the results video[2] are available online.

5.2 Attend, Infer, Repeat (AIR)

AIR, introduced by Eslami et al. [38], is a structured 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

[1] code: github.com/akosiorek/sqair
[2] video: youtu.be/-IUNQgSLEoc

(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

$$p_\theta(n) = \text{Geom}(n \mid \theta), \qquad p_\theta(\mathbf{z}^w \mid n) = \prod_{i=1}^{n} p_\theta\left(\mathbf{z}^{w,i}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{z}^{w,i} | \mathbf{0}, \mathbf{I}\right),$$
$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mathbf{x} \mid \mathbf{y}_t, \sigma_x^2 \mathbf{I}\right), \qquad \text{with } \mathbf{y}_t = \sum_{i=1}^{n} h_\theta^{\text{dec}}(\mathbf{z}^{\text{what},i}, \mathbf{z}^{\text{where},i}), \tag{5.1}$$

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. [73]), which scales and shifts it according to $\mathbf{z}^{\text{where}}$; see Figure 5.2.1 for details.

Inference Eslami et al. [38] use a sequential inference algorithm, where latent variables are inferred one at a time; see Figure 5.3.1. 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

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = q_\phi\left(z^{\text{pres},n+1} = 0 \mid \mathbf{z}^{w,1:n}, \mathbf{x}\right) \prod_{i=1}^{n} q_\phi\left(\mathbf{z}^{w,i}, z^{\text{pres},i} = 1 \mid \mathbf{z}^{1:i-1}, \mathbf{x}\right), \tag{5.2}$$

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 RNN R_ϕ with hidden state $\mathbf{h}^i$, namely: $\boldsymbol{\omega}^i, \mathbf{h}^i = R_\phi(\mathbf{x}, \mathbf{z}^{i-1}, \mathbf{h}^{i-1})$. The outputs $\boldsymbol{\omega}^i$, which are computed iteratively and depend on the previous latent variables (*cf.* Algorithm 3), parametrise $q_\phi\left(\mathbf{z}^{w,i}, \mathbf{z}^{\text{pres},i} \mid \mathbf{z}^{1:i-1}, \mathbf{x}\right)$. For simplicity the latter is assumed to factorise such that $q_\phi\left(\mathbf{z}^w, \mathbf{z}^{\text{pres}} \mid \mathbf{z}^{1:i-1}, \mathbf{x}\right) = q_\phi\left(z^{\text{pres},n+1} = 0 \mid \boldsymbol{\omega}^{n+1}\right) \prod_{i=1}^{n} q_\phi\left(\mathbf{z}^{w,i} \mid \boldsymbol{\omega}^i\right) q_\phi\left(z^{\text{pres},i} = 1 \mid \boldsymbol{\omega}^i\right)$.

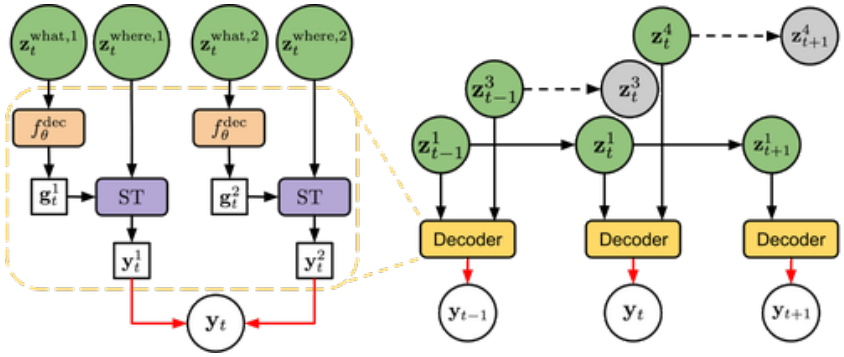


Figure 5.2.1: *Left*: Generation in AIR. The image mean $\mathbf{y}_t$ is generated by first using the *glimpse decoder* f_θ^{dec} to map the *what* variables into glimpses $\mathbf{g}_t$, transforming them with the *spatial transformer* ST according to the *where* variables and summing up the results. *Right*: Generation in SQAIR.

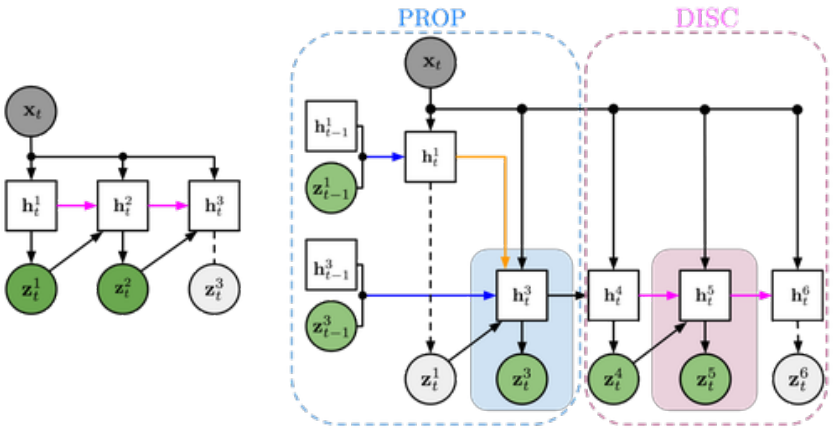


Figure 5.3.1: *Left*: Inference in AIR. The pink RNN attends to the image sequentially and produces one latent variable $\mathbf{z}_t^i$ at a time. Here, it decides that two latent variables are enough to explain the image and $\mathbf{z}_t^3$ is not generated. *Right*: Inference in SQAIR starts with the Propagation (PROP) phase. PROP iterates over latent variables from the previous time-step $t-1$ and updates them based on the new observation $\mathbf{x}_t$. 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 $\mathbf{z}_{t-1}^1$, 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 5.3.2 for details of the colored RNNs.

5.3 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 $\mathbf{z}_t^{\mathcal{P}_t} = \{\mathbf{z}_t^i\}_{i \in \mathcal{P}_t}$, 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 $\mathbf{z}_t \coloneqq [\mathbf{z}_t^{\mathcal{P}_t}, \mathbf{z}_t^{\mathcal{D}_t}]$, where the representation of the i^{th} object $\mathbf{z}_t^i \coloneqq [\mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, z_t^{\text{pres},i}]$ 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 $p^D\left(z_1^i \mid D_1\right)$. 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 $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t)$ as in AIR (*cf.* Equation (5.1), Fig. 5.2.1, and Algorithm 1). In full, the generative model is:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, D_{1:T}) = p^D(D_1, \mathbf{z}_1^{\mathcal{D}_1}) \prod_{t=2}^{T} p^D(D_t, \mathbf{z}_t^{\mathcal{D}_t} | \mathbf{z}_t^{\mathcal{P}_t}) p^P(\mathbf{z}_t^{\mathcal{P}_t} | \mathbf{z}_{t-1}) p_\theta(\mathbf{x}_t | \mathbf{z}_t), \tag{5.3}$$

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 Section 5.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 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. [146]). 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 5.4.4 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

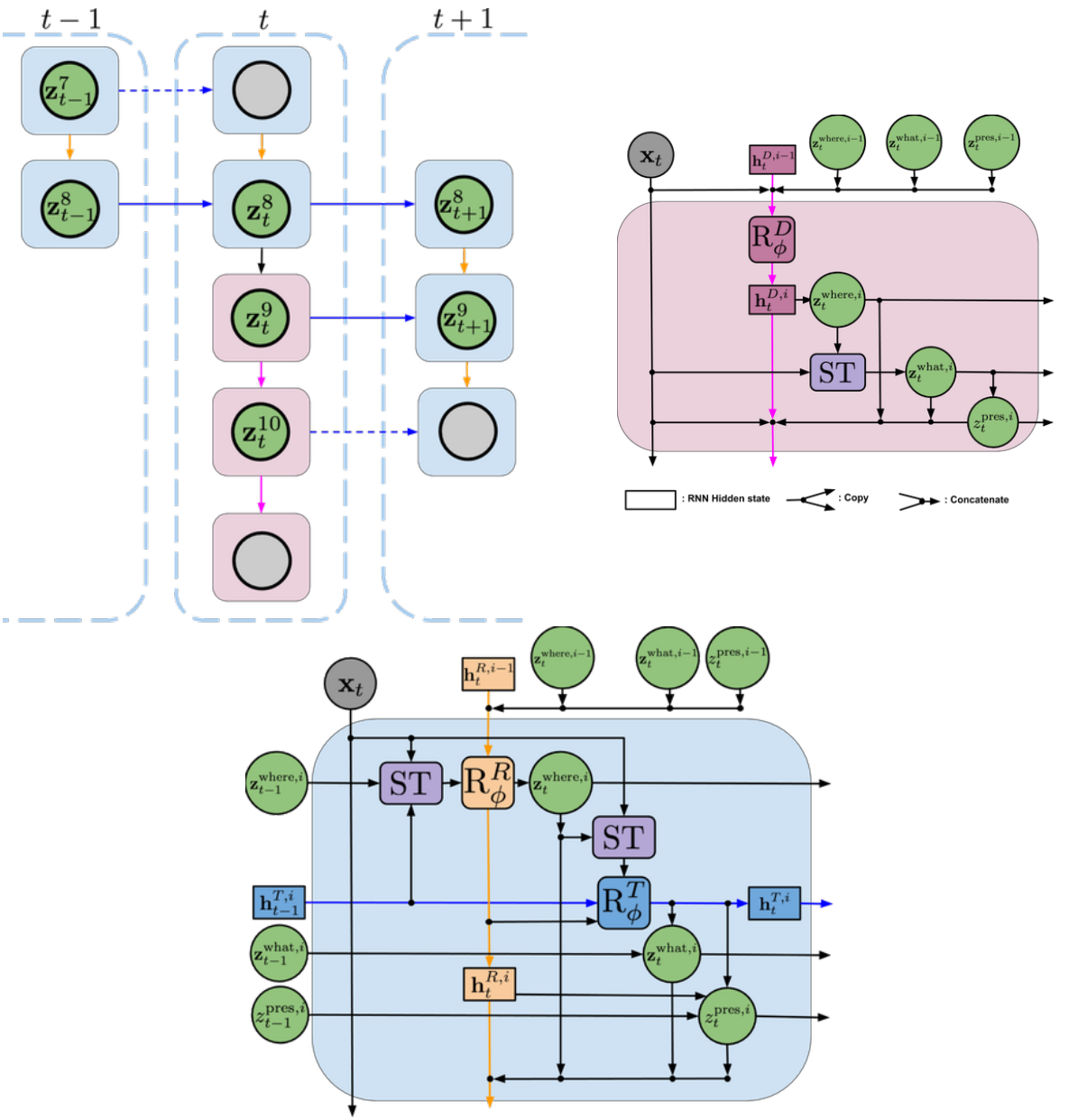


Figure 5.3.2: *Left*: Interaction between PROP and DISC in SQAIR. Firstly, objects are propagated to time t, and object $\mathrm{i} = 7$ is dropped. Secondly, DISC tries to discover new objects. Here, it manages to find two objects: $\mathrm{i} = 9$ and $\mathrm{i} = 10$. The process recurs for all remaining 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 Section 5.A.

entered the frame. All latent variables remaining after PROP and DISC are passed on to the next time step.

See Figures 5.3.1 and 5.3.2 for the inference network structure . The full variational posterior is defined as

$$q_\phi(D_{1:t}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi^D\left(D_t, \mathbf{z}_t^{\mathcal{D}_t} \mid \mathbf{x}_t, \mathbf{z}_t^{\mathcal{P}_t}\right) \prod_{i \in \mathcal{O}_{t-1}} q_\phi^P\left(\mathbf{z}_t^i \mid \mathbf{z}_{t-1}^i, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}\right). \quad (5.4)$$

Discovery, described by q_ϕ^D, is very similar to the full posterior of AIR, *cf.* Equation (5.2). 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 5.3.2, while all the pertinent algorithms and equations can be found in Sections 5.A and 5.C, respectively.

Learning We train SQAIR as an importance-weighted auto-encoder (IWAE) of Burda et al. [15]. Specifically, we maximise the importance-weighted evidence lower-bound $\mathcal{L}_{\text{IWAE}}$, namely

$$\mathcal{L}_{\text{IWAE}} = \mathbb{E}_{\mathbf{x}_{1:T} \sim p_{\text{data}}(\mathbf{x}_{1:T})}\left[\mathbb{E}_q\left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})}\right]\right]. \quad (5.5)$$

To optimise the above, we use RMSPROP, $K = 5$ and batch size of 32. We use the VIMCO gradient estimator of A. Mnih and Rezende [116] to backpropagate through the discrete latent variables z^{pres}, and use reparameterisation for the continuous ones (Kingma and Welling [88]). We also tried to use NVIL of A. Mnih and Gregor [115] as in the original work on AIR, but found it very sensitive to hyper-parameters, fragile and generally under-performing.

5.4 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 5.4.1). Moreover, we show that SQAIR can simulate

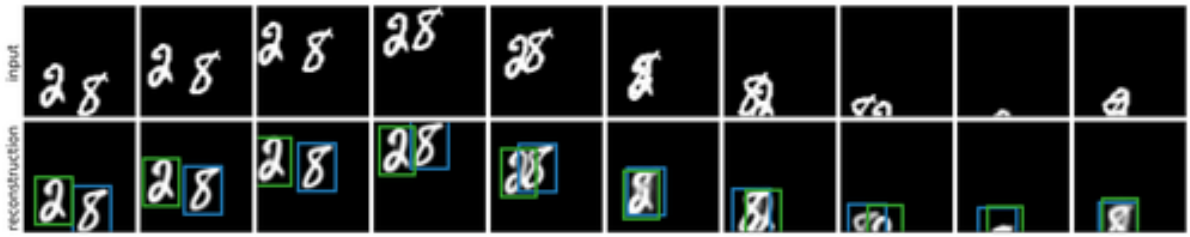


Figure 5.4.1: Input images (top) and SQAIR reconstructions with marked glimpse locations (bottom). For more examples, see Figure 5.H.1 in Section 5.H.

Figure 5.4.2: Samples from SQAIR. Both motion and appearance are consistent through time, thanks to the propagation part of the model. For more examples, see Figure 5.H.3 in Section 5.H.

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. [139]), 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.

Figure 5.4.3: The first three frames are input to SQAIR, which generated the rest conditional on the first frames.

Figure 5.4.4: 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.

5.4.1 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 Section 5.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 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. [154]). See Section 5.D for details of the model architectures and the training procedure.

We use AIR and VRNN (Chung et al. [20]) 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 Section 5.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 $\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T})]$ and the KL-divergence between the approximate posterior and the prior (Table 5.4.1).

	$\log p_\theta(\mathbf{x}_{1:T})$	$\log p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T})$	$\mathrm{KL}(q_\phi \,\|\, p_\theta)$	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 5.4.1: 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.

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 ($\mathbf{z}^{\text{what}}$) 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 5.4.1 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^{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 information, it does not exhibit this failure mode (*cf.* Figure 5.4.4). Next, we gauge the utility of the learnt representations by using them to determine the sum of the digits present in the image (Table 5.4.1, 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. Section 5.D contains details of the experiment. SQAIR significantly outperforms AIR and both variants of VRNN on this tasks. 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 Section 5.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 Section 5.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 5.4.3). 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 Section 5.H for more examples). Figure 5.4.4 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 Section 5.G.

5.4.2 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. [139]). As part of pre-precessing, we use standard background subtraction algorithms (Itseez [69]). 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 Section 5.E. We evaluate the model qualitatively by examining reconstructions, conditional samples (conditioned on the first four frames) and samples from the prior (Figure 5.4.5 and Section 5.I). We see that the model learns to reliably detect and track walking pedestrians, even when they are close to each other.

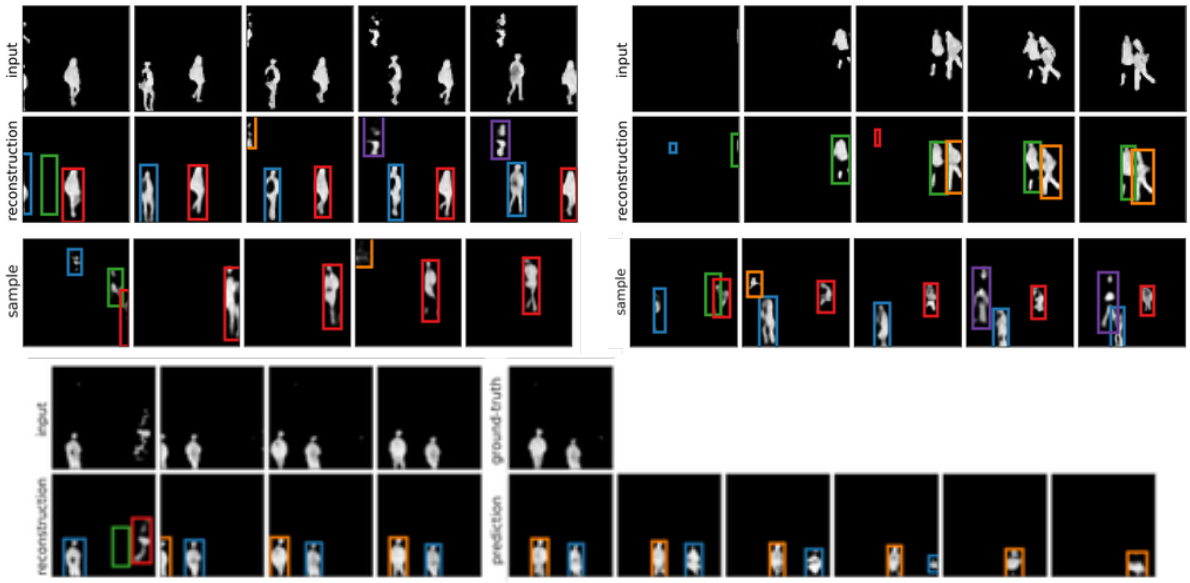


Figure 5.4.5: Inputs on the top, reconstructions in the second row, samples in the third row; rows four and five contain inputs and conditional generation: the first four frames in the last row are reconstructions, while the remaining ones are predicted by sampling from the prior. There is no ground-truth, since we used sequences of length five of training and validation.

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 ($>$ 1M for MNIST experiments). Hence, we conjecture that accuracy and marginal likelihood can be further improved by using a bigger dataset.

5.5 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. [169]). Recently, Kosiorek et al. [90] 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. [12]). Schulter et al. [153] 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. [19] detect single objects and their parts in images, and Kwak et al. [96] and Xiao and Jae Lee [182] 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 (Kosiorek et al. [90]). At the same time, is inherently multi-object and performs data association implicitly (*cf*. Section 5.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. [130] and Srivastava et al. [155]). 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. [5] and Denton and Fergus [30]). 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 et al. [49] and Greff et al. [50] 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. [68] and **Steenkiste2018** 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. [66]. 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 (Denton and Birodkar [29] and Tulyakov et al. [167]). 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 [120] 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. [42], 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.

5.6 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 et al. [50], **Steenkiste2018** and, together with Hsieh et al. [66], 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. [179]) or more generally as a world model (Ha and Schmidhuber [55]), 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 A. Mnih [84]). 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. [124]). 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.

Appendix

5.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 $\mathbf{z}^{\text{where}}$. f^1_{ffi} 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 MLP to their respective conditioning sets. Different distributions q do not share parameters of their MLPs. The *glimpse encoder* $h^{\text{glimpse}}_{\text{ffi}}$ (Lines 8 and 12 in Algorithm 2 and Line 12 in Algorithm 3; they share parameters) and the *image encoder* $h^{\text{enc}}_{\text{ffi}}$ (Line 3 in Algorithm 3) are implemented as two-layer MLPs or CNNs, depending on the experiment (see Sections 5.D and 5.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 ($h^{enc}_{ffi}(\mathbf{x}_t)$, 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 5.G.1 in Section 5.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. [188],

$$\mathbf{l}_t = \sum_{i \in \mathcal{P}_t} \text{MLP}\left(\mathbf{z}_t^{what,i}, \mathbf{z}_t^{where,i}\right). \tag{5.6}$$

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

Algorithm 1: Image Generation

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

$\mathcal{O}_t = \text{indices}\left(\mathbf{z}_t^{what}\right)$ `// Indices of all present latent variables.`

$\mathbf{y}_t^0 = \mathbf{0}$

for $i \in \mathcal{O}_t$ **do**

  $\mathbf{y}_t^{att,i} = f_{"}^{dec}\left(\mathbf{z}_t^{what,i}\right)$ `// Decode the glimpse.`

  $\mathbf{y}_t^i = \mathbf{y}_t^{i-1} + \text{ST}^{-1}\left(\mathbf{y}_t^{att,i}, \mathbf{z}_t^{where,i}\right)$

$\hat{\mathbf{x}}_t \sim \mathcal{N}\left(\mathbf{x} \mid \mathbf{y}_n, \sigma_x^2 \mathbf{I}\right)$

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

Algorithm 2: Inference for Propagation

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

$\mathbf{h}^{R,0}_t, \mathbf{z}^{\text{what},0}_t, \mathbf{z}^{\text{where},0}_t = \text{initialize}()$
$j = 0$ `// Index of the object processed in the last iteration.`
for $i \in \mathcal{O}_{t-1}$ **do**
  if $z^{\text{pres},i}_{t-1} == 0$ **then**
    continue
  $\hat{\mathbf{z}}^{\text{where},i}_t = f^1_{\text{ffi}}\left(\mathbf{z}^{\text{where},i}_{t-1}, \mathbf{h}^{T,i}_t\right)$ `// Proposal location.`
  $\hat{\mathbf{g}}^i_t = \text{ST}\left(\mathbf{x}_t, \hat{\mathbf{z}}^{\text{where},i}_t\right)$ `// Extract a glimpse from a proposal location.`
  $\hat{\mathbf{e}}^i_t = h^{\text{glimpse}}_{\text{ffi}}\left(\hat{\mathbf{g}}^i_t\right)$ `// Encode the proposal glimpse.`
  $\mathbf{w}^{R,i}_t, \mathbf{h}^{R,i}_t = R^R_\phi\left(\hat{\mathbf{e}}^i_t, \mathbf{z}^{\text{what},i}_{t-1}, \mathbf{z}^{\text{where},i}_{t-1}, \mathbf{h}^{T,i}_{t-1}, \mathbf{h}^{R,j}_t, \mathbf{z}^{\text{what},j}_t, \mathbf{z}^{\text{where},j}_t\right)$ `// Relational state, see Equation (5.14).`
  $\mathbf{z}^{\text{where},i}_t \sim q^P_\phi\left(z^{\text{where}} \mid z^{\text{where},k}_{t-1}, \mathbf{w}^{R,i}_t\right)$
  $\mathbf{g}^i_t = \text{ST}\left(\mathbf{x}_t, \mathbf{z}^{\text{where},i}_t\right)$ `// Extract the final glimpse.`
  $\mathbf{e}^i_t = h^{\text{glimpse}}_{\text{ffi}}\left(\mathbf{g}^i_t\right)$ `// Encode the final glimpse.`
  $\mathbf{w}^{T,i}_t, \mathbf{h}^{T,i}_t = R^T_\phi\left(\mathbf{e}^i_t, \mathbf{z}^{\text{where},i}_t, \mathbf{h}^{T,i}_{t-1}, \mathbf{h}^{R,i}_t\right)$ `// Temporal state, see Equation (5.15).`
  $\mathbf{z}^{\text{what},i}_t \sim q^P_\phi\left(\mathbf{z}^{\text{what}} \mid \mathbf{e}^i_t, \mathbf{z}^{\text{what},i}_{t-1}, \mathbf{w}^{R,i}_t, \mathbf{w}^{T,i}_t\right)$
  $z^{\text{pres},i}_t \sim q^P_\phi\left(z^{\text{pres}} \mid z^{\text{pres},i}_{t-1}, \mathbf{z}^{\text{what},i}_t, \mathbf{z}^{\text{where},i}_t, \mathbf{w}^{R,i}_t, \mathbf{w}^{T,i}_t\right)$ `// Equation (5.13).`
  $j = i$

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

5.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

$$p^D\left(D_t, \mathbf{z}^{\mathcal{D}_t}_t \mid \mathbf{z}^{\mathcal{P}_t}_t\right) = p^D(D_t \mid P_t) \prod_{i \in \mathcal{D}_t} p^D(\mathbf{z}^{\text{what},i}_t) p^D(\mathbf{z}^{\text{where},i}_t) \delta_1(z^{\text{pres},i}_t), \tag{5.7}$$

$$p^D(D_t \mid P_t) = \text{Categorical}\left(D_t; N - P_t, p_\theta(P_t)\right), \tag{5.8}$$

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

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.

$h_t^{D,0}, z_t^{what,0}, z_t^{where,0} = \text{initialize}()$
$j = \text{max\_index}\left(z_t^{\mathcal{P}_t}\right)$ `// Maximum index among the propagated latent variables.`
$e_t = h_{ffi}^{enc}(x_t)$ `// Encode the image.`
$l_t = h_{ffi}^{enc}(z_t^{what}, z_t^{where}, z_t^{pres})$ `// Encode latent variables.`
for $i \in [j+1, \ldots, j+N]$ **do**
  $w_t^{D,i}, h_t^{D,i} = R_\phi^D\left(e_t, l_t, z_t^{what,i-1}, z_t^{where,i-1}, h_t^{D,i-1}\right)$
  $z_t^{pres,i} \sim q_\phi^D\left(z^{pres} \mid w_t^{D,i}\right)$
  if $z^{pres,i} = 0$ **then**
    break
  $z_t^{where,i} \sim q_\phi^D\left(z^{where} \mid w_t^{D,i}\right)$
  $g_t^i = \text{ST}\left(x_t, z_t^{where,i}\right)$
  $e_t^i = h_{ffi}^{glimpse}\left(g_t^i\right)$ `// Encode the glimpse.`
  $z_t^{what,i} \sim q_\phi^D\left(z_t^{what} \mid e_t^i\right)$

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

The propagation prior is given by

$$p^P\left(z_t^{\mathcal{P}_t} \mid z_{t-1}\right) = \prod_{i \in \mathcal{P}_t} p^P\left(z_t^{pres,i} \mid z_{t-1}^{pres,i}, h_{t-1}\right) p^P\left(z_t^{what,i} \mid h_{t-1}\right) p^P\left(z_t^{where,i} \mid h_{t-1}\right), \tag{5.9}$$

$$p^P\left(z_t^{pres,i} \mid z_{t-1}^{pres,i}, h_{t-1}\right) = \text{Bernoulli}(z_t^{pres,i}; f_\theta(h_{t-1}))\delta_1(z_{t-1}^{pres,i}), \tag{5.10}$$

with f_θ a scalar-valued function with range $[0, 1]$ and $p^P(z_t^{what,i}|h_{t-1})$, $p^P(z_t^{where,i}|h_{t-1})$ both factorised Gaussians parameterised by some function of h_{t-1}.

5.C Details for the Inference of SQAIR

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

$$q_\phi^P\left(z_t^{\mathcal{P}_t} \mid x_t, z_{t-1}, h_t^{T,\mathcal{P}_t}\right) = \prod_{i \in \mathcal{O}_{t-1}} q_\phi^P\left(z_t^i \mid x_t, z_{t-1}^i, h_t^{T,i}, h_t^{R,i}\right), \tag{5.11}$$

with $\mathbf{h}_t^{R,i}$ the hidden state of the relation RNN (see Equation (5.14)). 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,

$$\begin{aligned} &q_\phi^P\left(\mathbf{z}_t^i \mid \mathbf{x}_t, \mathbf{z}_{t-1}^i, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}\right) = \\ &\quad q_\phi^P\left(\mathbf{z}_t^{\text{where},i} \mid \mathbf{z}_{t-1}^{\text{what},i}, \mathbf{z}_{t-1}^{\text{where},i}, \mathbf{h}_{t-1}^{T,i}, \mathbf{h}_t^{R,i}\right) \\ &\quad q_\phi^P\left(\mathbf{z}_t^{\text{what},i} \mid \mathbf{x}_t, \mathbf{z}_t^{\text{where},i}, \mathbf{z}_{t-1}^{\text{what},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}\right) \\ &\quad q_\phi^P\left(z_t^{\text{pres},i} \mid \mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, z_{t-1}^{\text{pres},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}\right). \end{aligned} \tag{5.12}$$

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,

$$\begin{aligned} &q_\phi^P\left(z_t^{\text{pres},i} \mid \mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, z_{t-1}^{\text{pres},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}\right) \\ &\quad = \text{Bernoulli}\left(z_t^{\text{pres},i} \mid f_\phi\left(\mathbf{z}_t^{\text{what},i}, \mathbf{z}_t^{\text{where},i}, \mathbf{h}_t^{T,i}, \mathbf{h}_t^{R,i}\right)\right) \delta_1(z_{t-1}^{\text{pres},i}), \end{aligned} \tag{5.13}$$

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 $j \in \{0, \ldots, i-1\}$ be the index of the most recent present object before object i. Hidden states are updated as follows,

$$\mathbf{h}_t^{R,i} = R_\phi^R\left(\mathbf{x}_t, \mathbf{z}_{t-1}^{\text{what},i}, \mathbf{z}_{t-1}^{\text{where},i}, \mathbf{h}_{t-1}^{T,i}, \mathbf{h}_t^{R,i-1}, \mathbf{z}_t^{\text{what},j}, \mathbf{z}_t^{\text{where},j}\right), \tag{5.14}$$

$$\mathbf{h}_t^{T,i} = R_\phi^T\left(\mathbf{x}_t, \mathbf{z}_t^{\text{where},i}, \mathbf{h}_{t-1}^{T,i}, \mathbf{h}_t^{R,i}\right), \tag{5.15}$$

where R_ϕ^T and R_ϕ^R are temporal and propagation RNNs, respectively. Note that in Eq. (5.14) 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.* Section 5.G. For computational details, please see Algorithms 2 and 3 in Section 5.A.

5.D Details of the moving-mnist Experiments

5.D.1 Sqair and air Training Details

All models are trained by maximising the ELBO $\mathcal{L}_{IWAE}$ (Equation (5.5)) with the RMSPROP optimizer (Tieleman and G. Hinton [163]) 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).

5.D.2 Sqair and air Model Architectures

All models use glimpse size of 20×20 and exponential linear unit (ELU) (Clevert et al. [22]) 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 R^D_ϕ and R^R_ϕ RNNs are the vanilla RNNs. The propagation prior RNN and the temporal RNN R^T_ϕ 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. [154]) 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 (< 0.9M 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. [38]. We tried using a learnable scalar standard deviation, but decided not to report it due to unsable 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.

5.D.3 Vrnn Implementation and Training Details

Our VRNN implementation is based on the implementation[3] of Filtering Variational Objectives (FIVO) by Maddison et al. [109]. 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 $p_\theta(\mathbf{z}_t \mid \mathbf{x}_{t-1})$ by an MLP. The likelihood $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{x}_{t-1})$ 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 $q_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$ is a Gaussian with

[3] https://github.com/tensorflow/models/tree/master/research/fivo

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

Table 5.D.1: Number of trainable parameters for the reported models.

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 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. [109]). 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 $[32, 32, 32, 64, 64, 64, 64, 64, 64]$, with strides of $[2, 2, 2, 2]$, $[1, 2, 1, 2, 1, 2]$ and $[1, 2, 1, 2, 1, 2, 1, 1, 1]$, 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. [52] applied to MNIST.

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

5.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 $\mathbf{z}^{\text{what}}$ 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 [85]) with default parameters (in tensorflow).

5.E Details of the *DukeMTMC* Experiments

We take videos from cameras one, two, five, six and eight from the *DukeMTMC* dataset (Ristani et al. [139]). As pre-processing, we invert colors and subtract backgrounds using standard OpenCV tools (Itseez [69]), 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.

Figure 5.F.1: SQAIR trained on a harder version of moving-MNIST. Input images (top) and SQAIR reconstructions with marked glimpse locations (bottom)

5.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 5.4.1, where all digits are present throughout the sequence. All other dataset parameters are the same as in Section 5.4.1. Figure 5.F.1 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.

5.G Failure cases of sqair

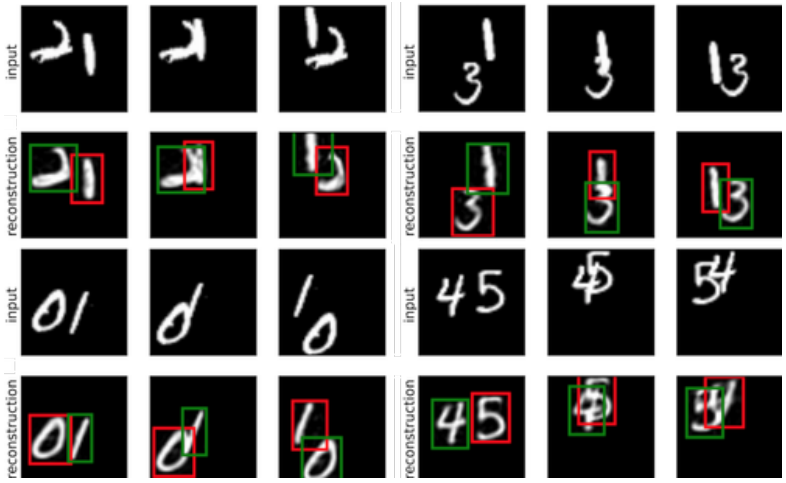


Figure 5.G.1: Examples of ID swaps in a version of SQAIR *without* proposal glimpse extraction in PROP (see Section 5.A for details). Bounding box colours correspond to object index (or its identity). When PROP is allowed the same access to the image as DISC, then it often prefers to ignore latent variables, which leads to swapped inference order.

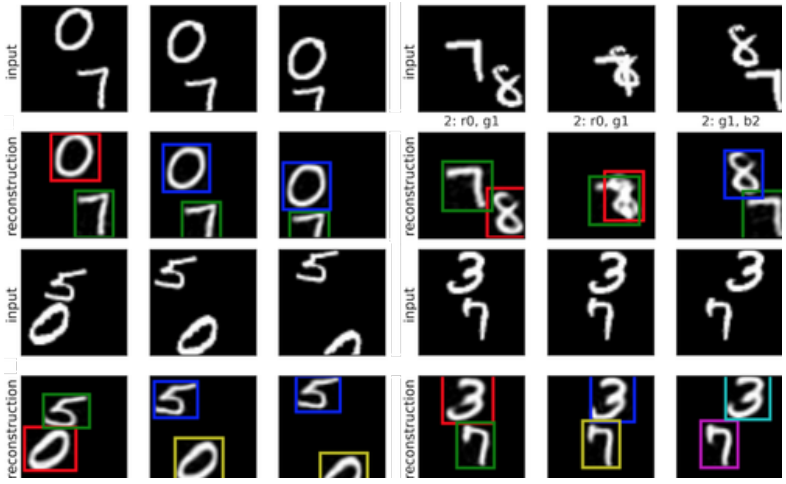


Figure 5.G.2: Examples of re-detections in MLP-SQAIR. Bounding box colours correspond to object identity, assigned to it upon discovery. In some training runs, SQAIR converges to a solution, where objects are re-detected in the second frame, and PROP starts tracking only in the third frame (left). Occasionally, 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 5.G.3: 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.

5.H Reconstruction and Samples from the Moving-MNIST Dataset

5.H.1 Reconstructions

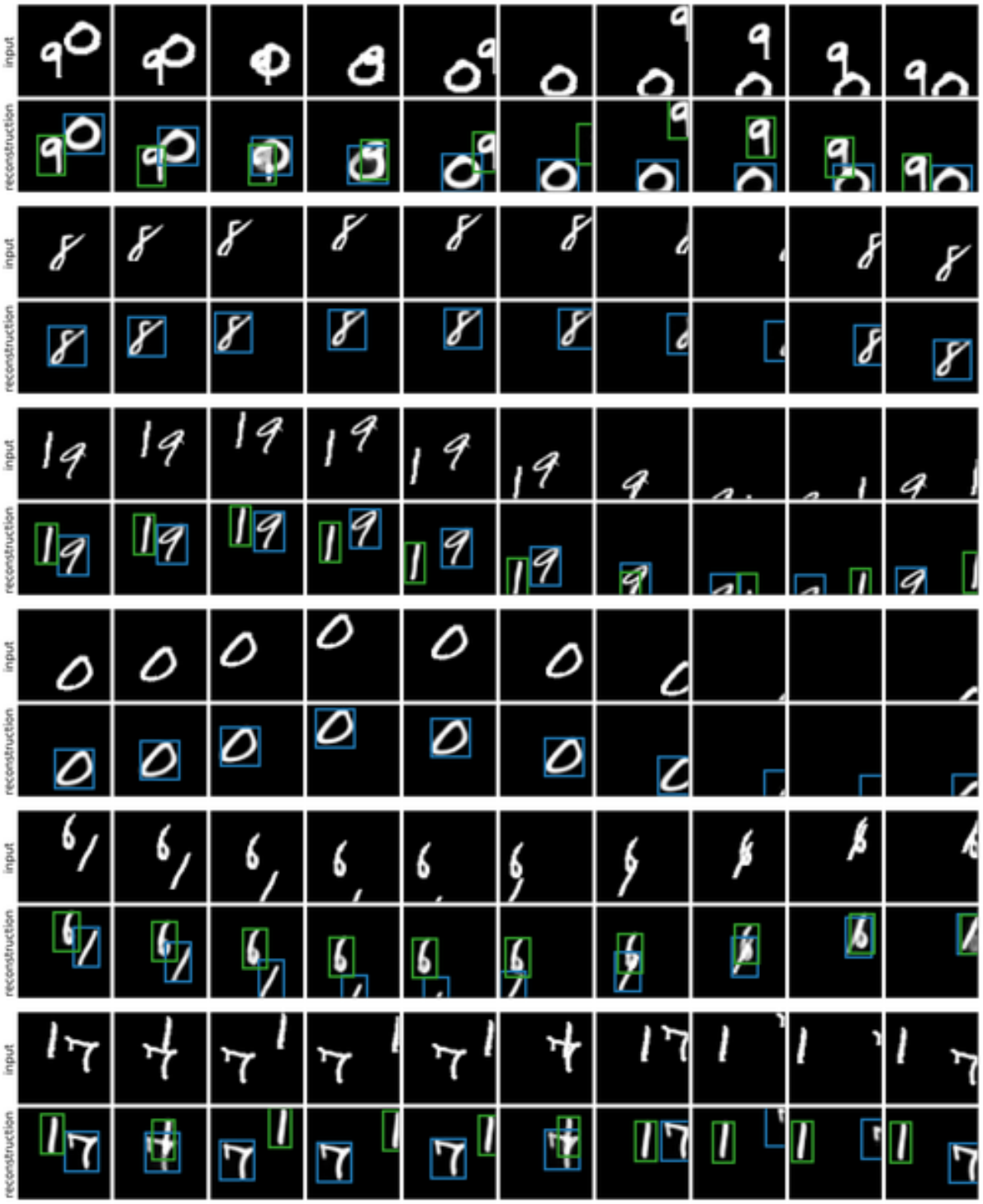


Figure 5.H.1: Sequences of input (first row) and SQAIR reconstructions with marked glimpse locations. Reconstructions are all temporally consistent.

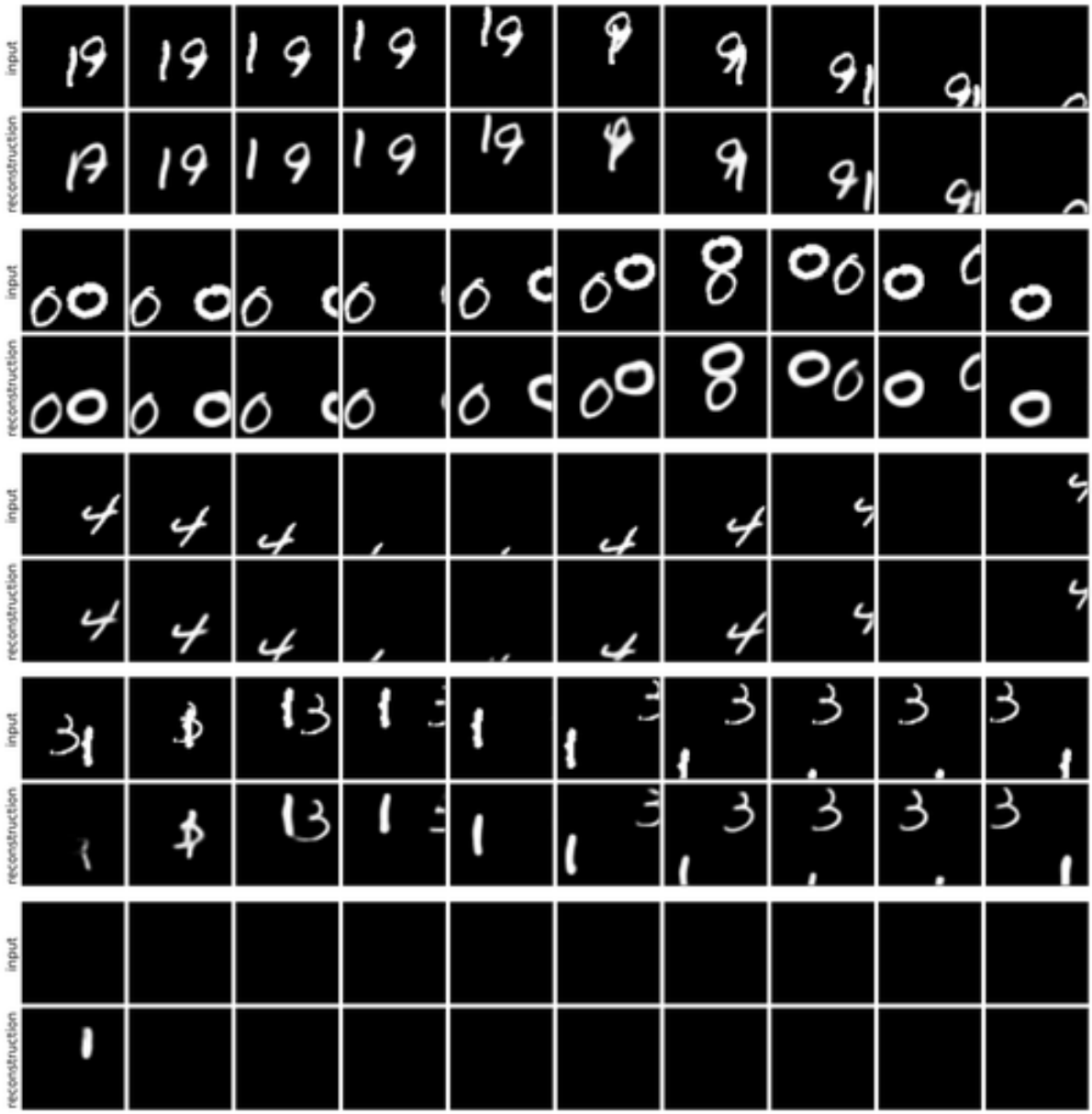


Figure 5.H.2: Sequences of input (first row) and CONV-VRNN reconstructions. They are not temporally consistent. The reconstruction at time $t = 1$ is typically of lower quality and often different than the rest of the sequence.

5.H.2 Samples

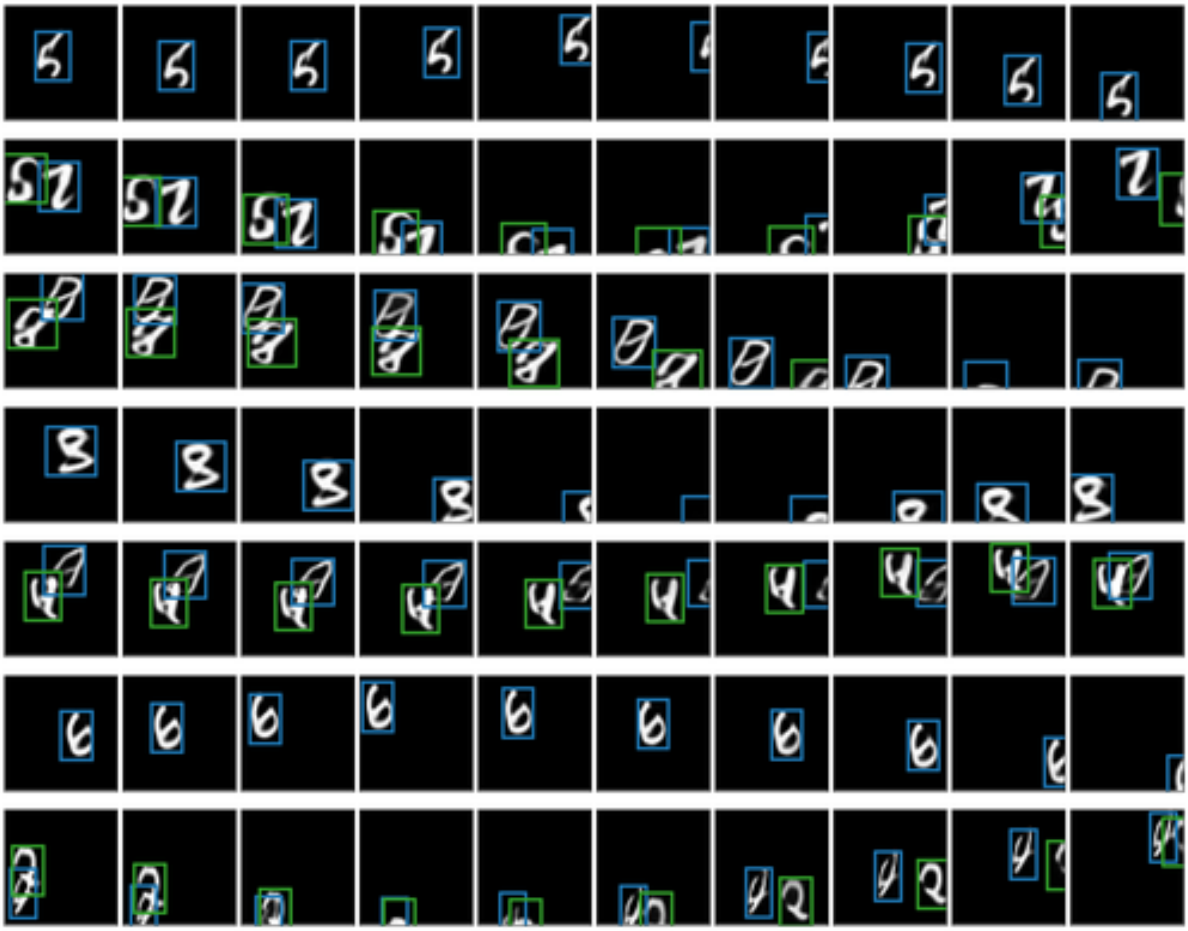


Figure 5.H.3: Samples from SQAIR. Both motion and appearance are temporally consistent. In the last sample, the model introduces the third object despite the fact that it has seen only up to two objects in training.

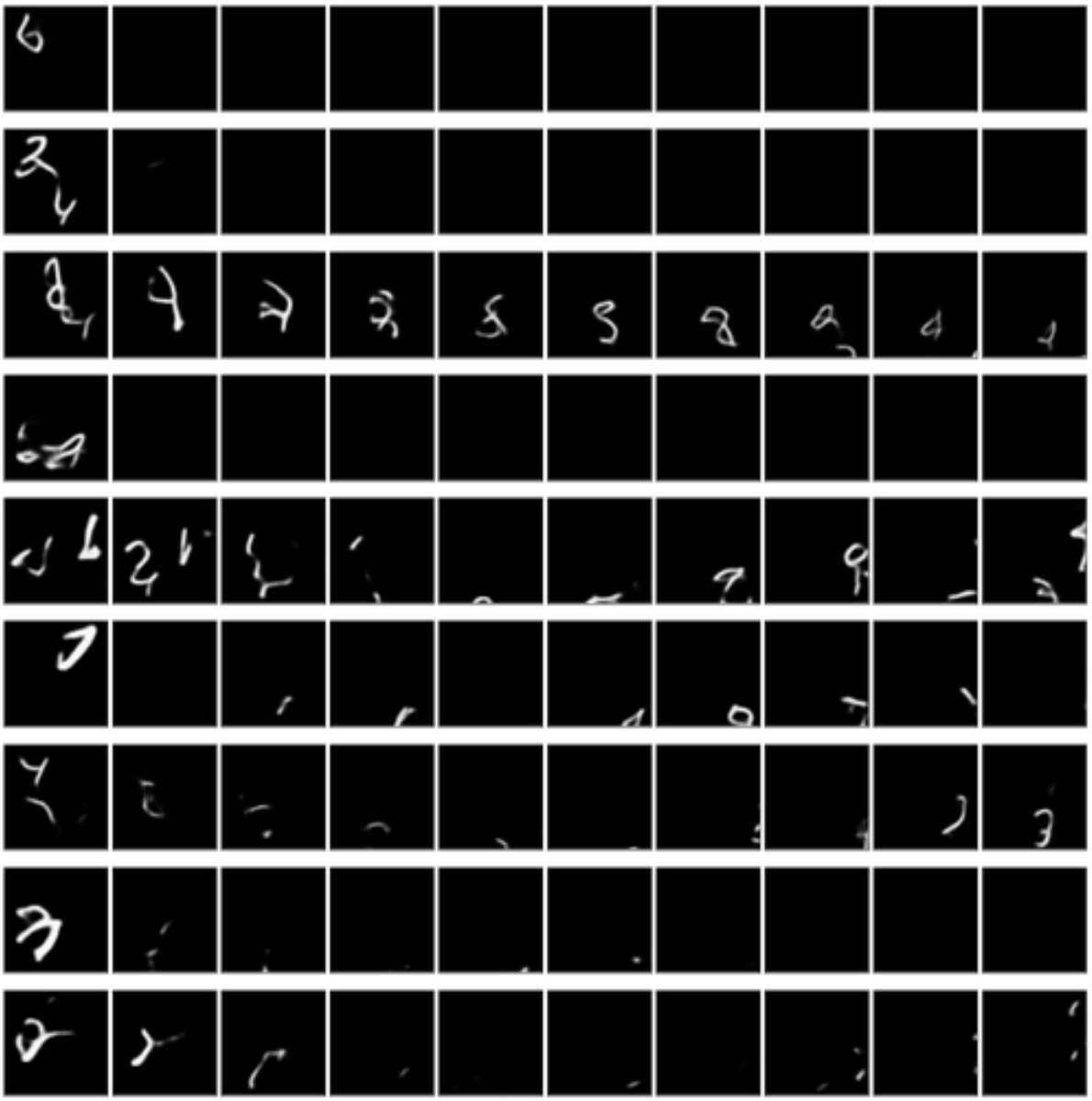


Figure 5.H.4: Samples from CONV-VRNN. They show lack of temporal consistency. Objects in the generated frames change between consecutive time-steps and they do not resamble digits from the training set.

5.H.3 Conditional Generation

Figure 5.H.5: 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.

5.I Reconstruction and Samples from the DukeMTMC Dataset

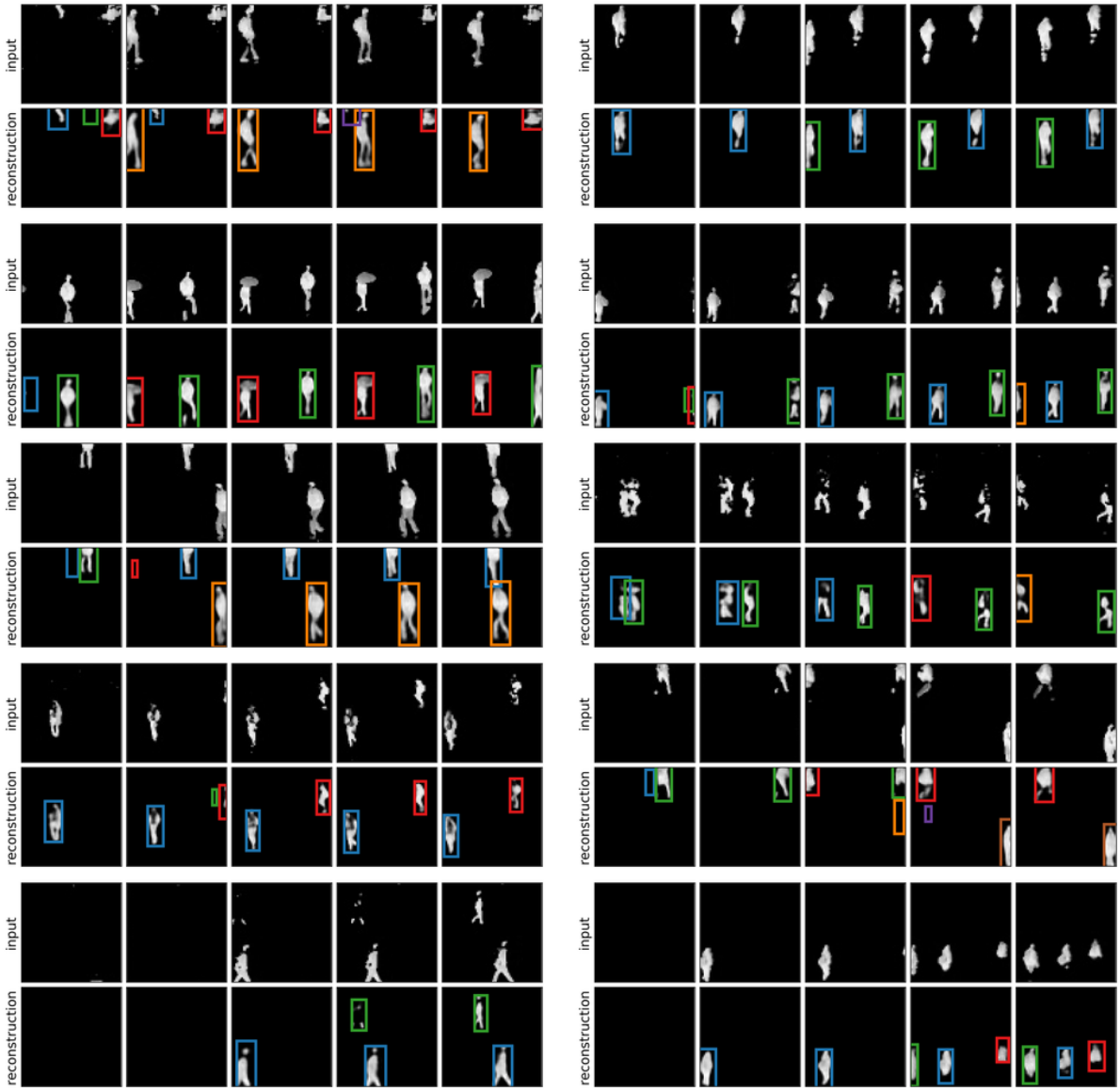


Figure 5.I.1: Sequences of input (first row) and SQAIR reconstructions with marked glimpse locations. While not perfect (spurious detections, missed objects), they are temporally consistent and similar in appearance to the inputs.

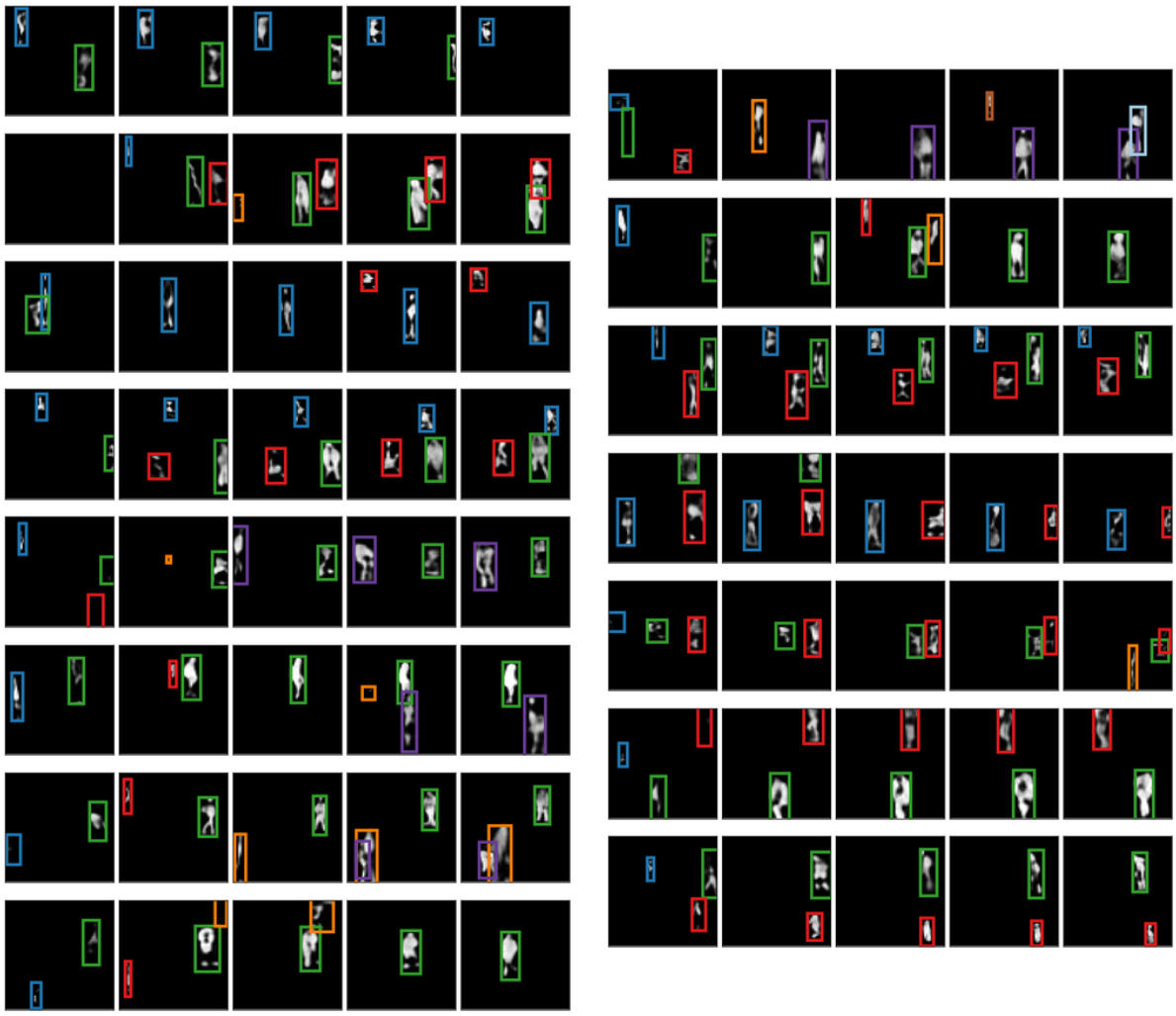


Figure 5.I.2: Samples with marked glimpse locations from SQAIR trained on the DukeMTMC dataset. Both appearance and motion is spatially consistent. Generated objects are similar in appearance to pedestrians in the training data. Samples are noisy, but so is the dataset.

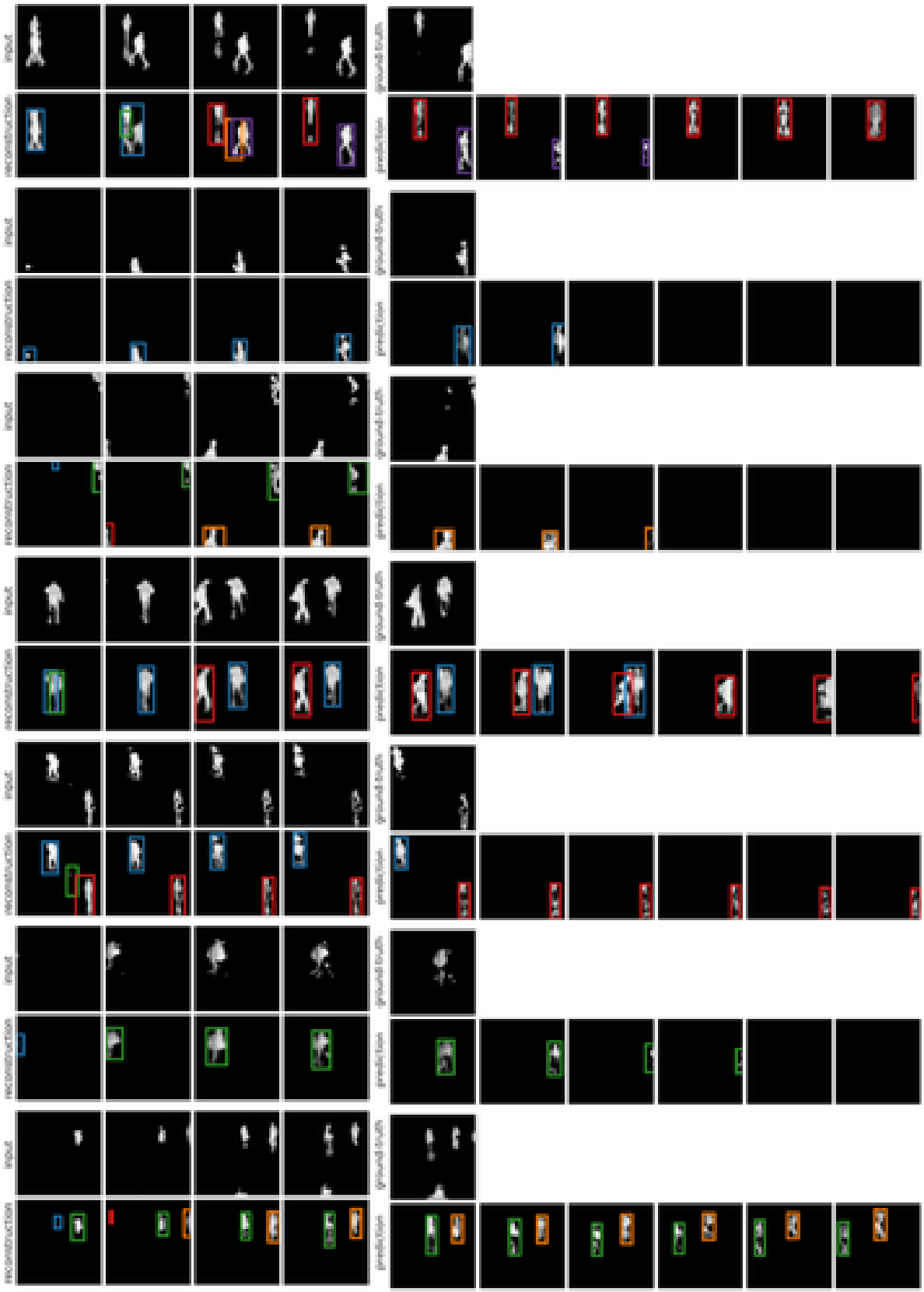

input
reconstruction
ground truth
prediction

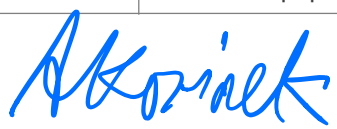
Figure 5.I.3: Conditional generation from SQAIR, which sees only the first four frames in every case. Top is the input sequence (and the remaining ground-truth), while bottom is reconstruction (first four time-steps) and then generation.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

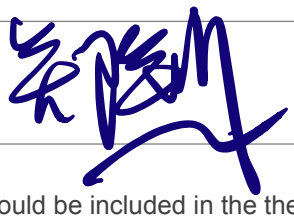
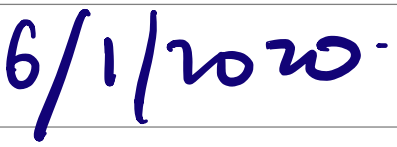
Title of Paper	Sequential Attend, Infer, Repeat: Generative modelling of moving objects
Publication Status	Published
Publication Details	A. R. Kosiorek, H. Kim, Y. W. Teh, and I. Posner. "Sequential Attend, Infer, Repeat: Generative modelling of moving objects". In: *Advances in Neural Information Processing Systems*. 2018. arXiv: 1806.01794. url: https://arxiv.org/abs/1806.01794.

Student Confirmation

Student Name:	Adam R. Kosiorek		
Contribution to the Paper	• Describe a probabilistic generative model of videos based on AIR,where every object is modelled by a sequence of location (where) and appearance (what) variables. • Develop the associated inference procedure. • Implement the model in code and demonstrate that it can work on toy data. • Demonstrate that the model can work on real-world CCTV data, where it does unsupervised detection and tracking of pedestrians. • Write the paper.		
Signature		Date	06/01/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Yee Whye Teh			
Supervisor comments			
Signature		Date	

This completed form should be included in the thesis, at the end of the relevant chapter.

6

Stacked Capsule Autoencoders

Objects are composed of a set of geometrically organized parts. We introduce an unsupervised capsule autoencoder (SCAE), which explicitly uses geometric relationships between parts to reason about objects. Since these relationships do not depend on the viewpoint, our model is robust to viewpoint changes. SCAE consists of two stages. In the first stage, the model predicts presences and poses of part templates directly from the image and tries to reconstruct the image by appropriately arranging the templates. In the second stage, SCAE predicts parameters of a few object capsules, which are then used to reconstruct part poses. Inference in this model is amortized and performed by off-the-shelf neural encoders, unlike in previous capsule networks. We find that object capsule presences are highly informative of the object class, which leads to state-of-the-art results for unsupervised classification on SVHN (55%) and MNIST (98.7%).

6.1 Introduction

CNN work better than networks without weight-sharing because of their inductive bias: if a local feature is useful in one image location, the same feature is likely to be useful in other locations. It is tempting to exploit other effects of viewpoint changes by replicating features across scale, orientation and other affine degrees of freedom, but this quickly leads to cumbersome, high-dimensional feature maps.

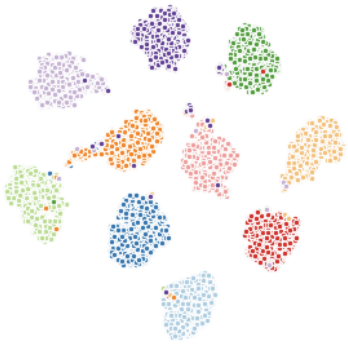


Figure 6.1.1: SCAEs learn to explain different object classes with separate object capsules, thereby doing unsupervised classification. Here, we show TSNE embeddings of object capsule presence probabilities for 10000 MNIST digits. Individual points are color-coded according to the corresponding digit class.

An alternative to replicating features across the non-translational degrees of freedom is to explicitly learn transformations between the natural coordinate frame of a whole object and the natural coordinate frames of each of its parts. Computer graphics relies on such object→part coordinate transformations to represent the geometry of an object in a viewpoint-invariant manner. Moreover, there is strong evidence that, unlike standard CNNs, human vision also relies on coordinate frames: imposing an unfamiliar coordinate frame on a familiar object makes it challenging to recognize the object or its geometry (G. E. Hinton [62] and Rock [141]).

A neural system can learn to reason about transformations between objects, their parts and the viewer, but each kind of transformation will likely need to be represented differently. An object-part-relationship (OP) is viewpoint-invariant, approximately constant and could be easily coded by learned weights. The relative coordinates of an object (or a part) with respect to the viewer change with the viewpoint (they are viewpoint-equivariant), and could be easily coded with neural activations[1].

[1] This may explain why accessing perceptual knowledge about objects, when they are not visible, requires creating a mental image of the object with a specific viewpoint.

With this representation, the pose of a single object is represented by its relationship to the viewer. Consequently, representing a single object does not necessitate replicating neural activations across space, unlike in CNNs. It is only processing two (or more) different instances of the same type of object in parallel that requires spatial replicas of both model parameters and neural activations.

In this paper we propose the Stacked Capsule Autoencoder (SCAE), which has two stages (Fig. 6.1.2). The first stage, the Part Capsule Autoencoder (PCAE), segments an image into constituent parts, infers their poses, and reconstructs the image by appropriately arranging affine-transformed part templates. The second stage, the Object Capsule Autoencoder (OCAE), tries to organize discovered parts and their poses into a smaller set of objects. These objects then try to reconstruct the part poses using a separate mixture of predictions for each part. Every object capsule contributes components to each of these mixtures by multiplying its pose—the object-viewer-relationship (OV)—by the relevant object-part-relationship (OP).

Stacked Capsule Autoencoders (Section 6.2) capture spatial relationships between whole objects and their parts when trained on unlabelled data. The vectors of presence probabilities for the object capsules tend to form tight clusters (cf. Figure 6.1.1), and when we assign a class to each cluster we achieve state-of-the-art results for unsupervised classification on SVHN (55%) and MNIST (98.7%), which can be further improved to 67% and 99%, respectively, by learning fewer than 300 parameters (Section 6.3). We describe related work in Section 6.4 and discuss implications of our work and future directions in Section 6.5. The code is available at `github.com/google-research/google-research/tree/master/stacked_capsule_autoencoders`.

6.2 Stacked Capsule Autoencoders (scae)

Segmenting an image into parts is non-trivial, so we begin by abstracting away pixels and the part-discovery stage, and develop the Constellation Capsule Autoencoder (CCAE) (Section 6.2.1). It uses two-dimensional points as parts, and their coordinates are given as the input to the system. CCAE learns to model sets of points as

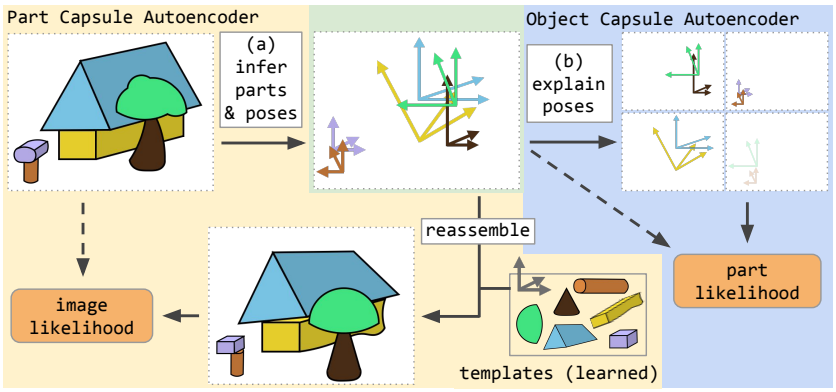


Figure 6.1.2: Stacked Capsule Autoencoder (SCAE): (a) *part* capsules segment the input into parts and their poses. The poses are then used to reconstruct the input by affine-transforming learned templates. (b) *object* capsules try to arrange inferred poses into objects, thereby discovering underlying structure. SCAE is trained by maximizing image and part log-likelihoods subject to sparsity constraints.

arrangements of familiar constellations, each of which has been transformed by an independent similarity transform. The CCAE learns to assign individual points to their respective constellations—without knowing the number of constellations or their shapes in advance. Next, in Section 2.2, we develop the Part Capsule Autoencoder (PCAE) which learns to infer parts and their poses from images. Finally, we stack the Object Capsule Autoencoder (OCAE), which closely resembles the CCAE, on top of the PCAE to form the Stacked Capsule Autoencoder (SCAE).

6.2.1 Constellation Autoencoder (ccae)

Let $\{\mathbf{x}_m \mid m = 1, \ldots, M\}$ be a set of two-dimensional input points, where every point belongs to a constellation as in Figure 6.2.1. We first encode all input points (which take the role of part capsules) with Set Transformer (Lee et al. [101])—a permutation-invariant encoder h^{caps} based on attention mechanisms—into K object capsules. An object capsule k consists of a capsule feature vector $\mathbf{c}_k$, its presence probability $a_k \in [0, 1]$ and a 3×3 object-viewer-relationship (OV) matrix, which represents the affine transformation between the object (constellation) and the viewer. Note that each object capsule can represent only one object at a time. Every object capsule uses a separate MLP h_k^{part} to predict $N \leqslant M$ part candidates from the capsule feature vector $\mathbf{c}_k$. Each candidate consists of the conditional probability $a_{k,n} \in [0, 1]$ that a given

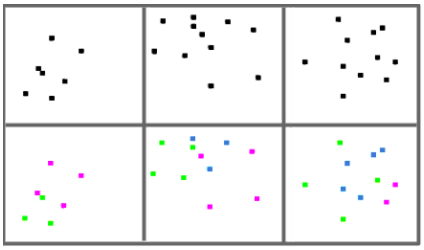


Figure 6.2.1: Unsupervised segmentation of points belonging to up to three constellations of squares and triangles at different positions, scales and orientations. The model is trained to reconstruct the points (top row) under the CCAE mixture model. The bottom row colours the points based on the parent with the highest posterior probability in the mixture model. The right-most column shows a failure case. Note that the model uses sets of points, not pixels, as its input; we use images only to visualize the constellation arrangements.

candidate part exists, an associated scalar standard deviation $\lambda_{k,n}$, and a 3×3 object-part-relationship (OP) matrix, which represents the affine transformation between the object capsule and the candidate part[2]. Candidate predictions $\mu_{k,n}$ are given by the product of the object capsule OV and the candidate OP matrices. We model all input points as a single Gaussian mixture, where $\mu_{k,n}$ and $\lambda_{k,n}$ are the centres and standard deviations of the isotropic Gaussian components. See Figures 6.1.2 and 6.2.4 for illustration; formal description follows:

$$\mathrm{OV}_{1:K}, \mathbf{c}_{1:K}, a_{1:K} = h^{\text{caps}}(\mathbf{x}_{1:M}) \qquad \text{predict object capsule parameters,} \tag{6.1}$$

$$\mathrm{OP}_{k,1:N}, a_{k,1:N}, \lambda_{k,1:N} = h^{\text{part}}_k(\mathbf{c}_k) \qquad \text{decode candidate parameters from } \mathbf{c}_k\text{'s,} \tag{6.2}$$

$$\mu_{k,n} = \mathrm{OV}_k \mathrm{OP}_{k,n} \qquad \text{decode a part pose candidate,} \tag{6.3}$$

$$p(\mathbf{x}_m \mid k, n) = \mathcal{N}(\mathbf{x}_m \mid \mu_{k,n}, \lambda_{k,n}) \qquad \text{turn candidates into mixture components,} \tag{6.4}$$

$$p(\mathbf{x}_{1:M}) = \prod_{m=1}^{M} \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{a_k a_{k,n}}{\sum_i a_i \sum_j a_{i,j}} p(\mathbf{x}_m \mid k, n)\,. \tag{6.5}$$

The model is trained without supervision by maximizing the likelihood of part capsules in Equation (6.5) subject to sparsity constraints, *cf.* Sections 6.2.4 and 6.C. The part capsule m can be assigned to the object capsule $k^\star$ by looking at the mixture component responsibility, that is $k^\star = \arg\max_k \; a_k a_{k,n}\, p(\mathbf{x}_m \mid k, n)$.[3] Empirical results

[2]Deriving these matrices from capsule feature vectors allows for deformable objects, see Section 6.D for details.

[3]We treat parts as independent and evaluate their probability under the same mixture model. While there are no clear 1:1 connections between parts and predictions, it seems to work well in practice.

show that this model is able to perform unsupervised instance-level segmentation of points belonging to different constellations, even in data which is difficult to interpret for humans. See Figure 6.2.1 for an example and Section 6.3.1 for details.

6.2.2 Part Capsule Autoencoder (pcae)

Explaining images as geometrical arrangements of parts requires 1) discovering what parts are there in an image and 2) inferring the relationships of the parts to the viewer (their pose). For the CCAE a part is just a 2D point (that is, a (x, y) coordinate), but for the PCAE each part capsule m has a six-dimensional pose $\mathbf{x}_m$ (two rotations, two translations, scale and shear), a presence variable $d_m \in [0, 1]$ and a unique identity. We frame the part-discovery problem as auto-encoding: the encoder learns to infer the poses and presences of different part capsules, while the decoder learns an image template T_m for each part (Fig. 6.2.2) similar to Eslami et al. [38] and Tieleman [164]. If a part exists (according to its presence variable), the corresponding template is affine-transformed with the inferred pose giving $\widehat{T}_m$. Finally, transformed templates are arranged into the image. The PCAE is followed by an Object Capsule Autoencoder (OCAE), which closely resembles the CCAE and is described in Section 6.2.3.

Let $\mathbf{y} \in [0, 1]^{h \times w \times c}$ be the image. We limit the maximum number of part capsules to M and use an encoder to infer their poses $\mathbf{x}_m$, presence probabilities d_m, and special features $\mathbf{z}_m \in \mathbb{R}^{c_z}$, one per part capsule. Special features can be used to alter the templates in an input-dependent manner (we use them to predict colour, but more complicated mappings are possible). The special features also inform the OCAE about unique aspects of the corresponding part (e. g., occlusion or relation to other parts). Templates $T_m \in [0, 1]^{h_t \times w_t \times (c+1)}$ are smaller than the image $\mathbf{y}$, but have an additional alpha channel which allows occlusion by other templates. We use T_m^a to refer to the alpha channel and T_m^c to refer to its colours.

We allow each part capsule to be used only once to reconstruct an image, which means that parts of the same type are not repeated[4]. To infer part capsule parameters

[4]We could repeat parts by using multiple instances of the same part capsule.

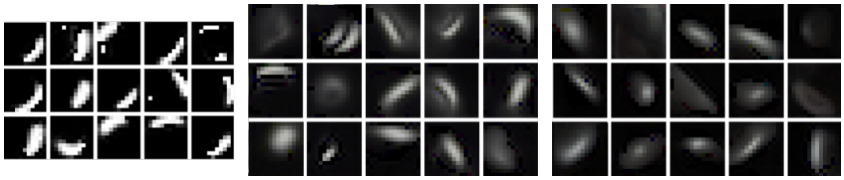


Figure 6.2.2: Stroke-like templates learned on MNIST (left) as well as sobel-filtered SVHN (middle) and CIFAR10 (right). For SVHN they often take the form of double strokes due to sobel filtering.

we use a CNN-based encoder followed by *attention-based pooling*, which is described in more detail in the Section 6.E and whose effects on the model performance are analyzed in Section 6.3.3.

The image is modelled as a spatial Gaussian mixture, similarly to Burgess et al. [16], Engelcke et al. [37], and Greff et al. [48]. Our approach differs in that we use pixels of the transformed templates (instead of component-wise reconstructions) as the centres of isotropic Gaussian components, but we also use constant variance. Mixing probabilities of different components are proportional to the product of presence probabilities of part capsules and the value of the learned alpha channel for every template. More formally:

$$\mathbf{x}_{1:M}, d_{1:M}, \mathbf{z}_{1:M} = h^{\text{enc}}(\mathbf{y}) \qquad \text{predict part capsule parameters,} \tag{6.6}$$

$$\mathbf{c}_m = \text{MLP}(\mathbf{z}_m) \qquad \text{predict the color of the } m^{\text{th}} \text{ template,} \tag{6.7}$$

$$\widehat{T}_m = \text{TransformImage}(T_m, \mathbf{x}_m) \qquad \text{apply affine transforms to image templates,} \tag{6.8}$$

$$p^y_{m,i,j} \propto d_m \widehat{T}^a_{m,i,j} \qquad \text{compute mixing probabilities,} \tag{6.9}$$

$$p(\mathbf{y}) = \prod_{i,j} \sum_{m=1}^{M} p^y_{m,i,j}\, \mathcal{N}\Big(y_{i,j} \mid \mathbf{c}_m \cdot \widehat{T}^c_{m,i,j}; \sigma_y^2\Big) \qquad \text{calculate the image likelihood.} \tag{6.10}$$

Training the PCAE results in learning templates for object parts, which resemble strokes in the case of MNIST, see Figure 6.2.2. This stage of the model is trained by maximizing the image likelihood of Equation (6.10).

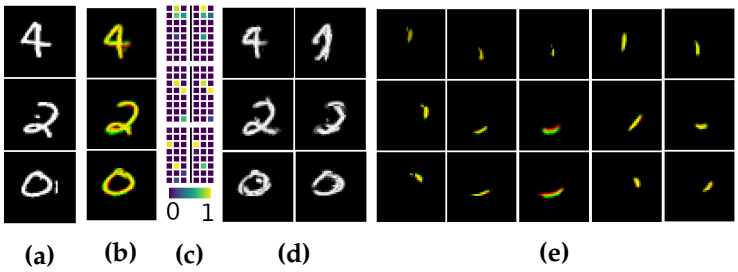


Figure 6.2.3: MNIST (a) images, (b) reconstructions from part capsules in red and object capsules in green, with overlapping regions in yellow. Only a few object capsules are activated for every input (c) a priori (left) and even fewer are needed to reconstruct it (right). The most active capsules (d) capture object identity and its appearance; (e) shows a few o f the affine-transformed templates used for reconstruction.

6.2.3 Object Capsule Autoencoder (ocae)

Having identified parts and their parameters, we would like to discover objects that could be composed of them[5]. To do so, we use concatenated poses $\mathbf{x}_m$, special features $\mathbf{z}_m$ and flattened templates T_m (which convey the identity of the part capsule) as an input to the OCAE, which differs from the CCAE in the following ways. Firstly, we feed part capsule presence probabilities d_m into the OCAE's encoder—these are used to bias the Set Transformer's attention mechanism not to take absent points into account. Secondly, d_ms are also used to weigh the part-capsules' log-likelihood, so that we do not take log-likelihood of absent points into account. This is implemented by raising the likelihood of the m^{th} part capsule to the power of d_m, *cf.* Equation (6.5). Additionally, we stop the gradient on all of OCAE's inputs except the special features to improve training stability and avoid the problem of collapsing latent variables; see e. g., Rasmus et al. [131]. Finally, parts discovered by the PCAE have independent identities (templates and special features rather than 2D points). Therefore, every part-pose is explained as an independent mixture of predictions from object-capsules—where every object capsule makes exactly M candidate predictions $\mu_{k,1:M}$, or exactly **one** candidate prediction per part. Consequently, the part-capsule likelihood is given

[5] Discovered objects are *not* used top-down to refine the presences or poses of the parts during inference. However, the derivatives backpropagated via OCAE refine the lower-level encoder network that infers the parts.

by,

$$p(\mathbf{x}_{1:M}, d_{1:M}) = \prod_{m=1}^{M} \left[\sum_{k=1}^{K} \frac{a_k a_{k,m}}{\sum_i a_i \sum_j a_{i,j}} p(\mathbf{x}_m \mid k, m) \right]^{d_m} . \tag{6.11}$$

The OCAE is trained by maximising the part pose likelihood of Equation (6.11), and it learns to discover further structure in previously identified parts, leading to learning sparsely-activated object capsules, see Figure 6.2.3. Achieving this sparsity requires further regularization, however.

6.2.4 Achieving Sparse and Diverse Capsule Presences

Stacked Capsule Autoencoders are trained to maximise pixel and part log-likelihoods ($\mathcal{L}_{ll} = \log p(\mathbf{y}) + \log p(\mathbf{x}_{1:M})$). If not constrained, however, they tend to either use all of the part and object capsules to explain every data example or collapse onto always using the same subset of capsules, regardless of the input. We want the model to use different sets of part-capsules for different input examples and to specialize object-capsules to particular arrangements of parts. To encourage this, we impose sparsity and entropy constraints. We evaluate their importance in Section 6.3.3.

We first define prior and posterior object-capsule presence as follows. For a minibatch of size B with K object capsules and M part capsules we define a minibatch of prior capsule presence $a^{\text{prior}}_{1:K}$ with dimension $[B, K]$ and posterior capsule presence $a^{\text{posterior}}_{1:K,1:M}$ with dimension $[B, K, M]$ as,

$$a^{\text{prior}}_k = a_k \max_m a_{m,k} , \qquad a^{\text{posterior}}_{k,m} = a_k a_{k,m} \, \mathcal{N}(\mathbf{x}_m \mid m, k) , \tag{6.12}$$

respectively; the former is the maximum presence probability among predictions from object capsule k while the latter is the unnormalized mixing proportion used to explain part capsule m.

Prior sparsity Let $\overline{u}_k = \sum_{b=1}^{B} a^{\text{prior}}_{b,k}$ the sum of presence probabilities of the object capsule k among different training examples, and $\widehat{u}_b = \sum_{k=1}^{K} a^{\text{prior}}_{b,k}$ the sum of object capsule presence probabilities for a given example. If we assume that training examples contain objects from different classes uniformly at random and we would like to assign the same number of object capsules to every class, then each class would

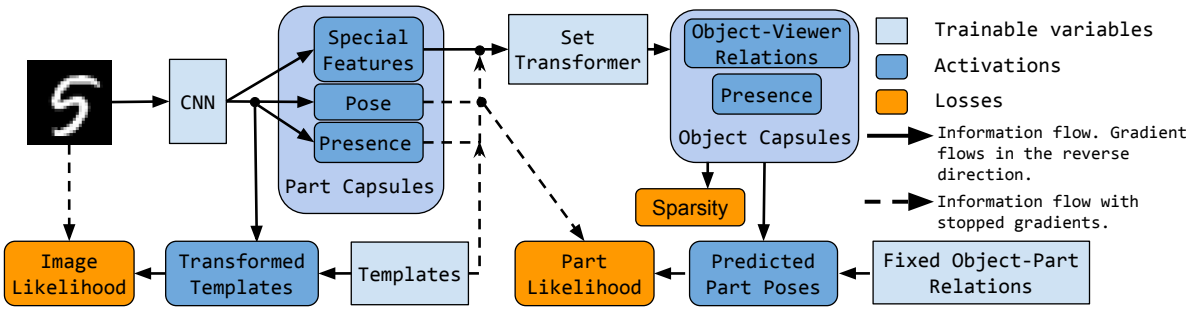


Figure 6.2.4: SCAE architecture.

obtain K/C capsules. Moreover, if we assume that only one object is present in every image, then K/C object capsules should be present for every input example, which results in the sum of presence probabilities of B/C for every object capsule. To this end, we minimize,

$$\mathcal{L}_{\text{prior}} = \frac{1}{B}\sum_{b=1}^{B}(\hat{u}_b - K/C)^2 + \frac{1}{K}\sum_{k=1}^{K}(\bar{u}_k - B/C)^2 \,. \tag{6.13}$$

Posterior Sparsity Similarly, we experimented with minimizing the within-example entropy of capsule posterior presence $\mathcal{H}(\bar{v}_k)$ and maximizing its between-example entropy $\mathcal{H}(\hat{v}_b)$, where $\mathcal{H}$ is the entropy, and where $\bar{v}_k$ and $\hat{v}_b$ are the the normalized versions of $\sum_{k,m} a^{\text{posterior}}_{b,k,m}$ and $\sum_{b,m} a^{\text{posterior}}_{b,k,m}$, respectively. The final loss reads as

$$\mathcal{L}_{\text{posterior}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{H}(\bar{v}_k) - \frac{1}{B}\sum_{b=1}^{B}\mathcal{H}(\hat{v}_b)\,. \tag{6.14}$$

Our ablation study has shown, however, that the model can perform equally well without these posterior sparsity constraints, cf. Section 6.3.3.

Fig. 6.2.4 shows the schematic architecture of SCAE. We optimize a weighted sum of image and part likelihoods and the auxiliary losses. Loss weight selection process, as well as the values used for experiments, are detailed in Section 6.A.

In order to make the values of presence probabilities (a_k, $a_{k,m}$ and d_m) closer to binary we inject uniform noise $\in [-2, 2]$ into logits, similar to Tieleman [164]. This forces the model to predict logits that are far from zero to avoid stochasticity and makes the predicted presence probabilities close to binary. Interestingly, it tends to work better in our case than using the Concrete distribution (Maddison et al. [110]).

6.3 Evaluation

The decoders in the SCAE use explicitly parameterised affine transformations that allow the encoders' inputs to be explained with a small set of transformed objects or parts. The following evaluations show how the embedded geometrical knowledge helps to discover patterns in data. Firstly, we show that the CCAE discovers underlying structures in sets of two-dimensional points, thereby performing instance-level segmentation. Secondly, we pair an OCAE with a PCAE and investigate whether the resulting SCAE can discover structure in real images. Finally, we present an ablation study that shows which components of the model contribute to the results.

6.3.1 Discovering Constellations

We create arrangements of constellations online, where every input example consists of up to 11 two-dimensional points belonging to up to three different constellations (two squares and a triangle) as well as binary variables indicating the presence of the points (points can be missing). Each constellation is included with probability 0.5 and undergoes a similarity transformation, whereby it is randomly scaled, rotated by up to 180° and shifted. Finally, every input example is normalised such that all points lie within $[-1, 1]^2$. Note that we use sets of points, and not images, as inputs to our model.

We compare the CCAE against a baseline that uses the same encoder but a simpler decoder: the decoder uses the capsule parameter vector $\mathbf{c}_k$ to directly predict the location, precision and presence probability of each of the four points as well as the presence probability of the whole corresponding constellation. Implementation details are listed in Section 6.A.1.

Both models are trained unsupervised by maximising the part log-likelihood. We evaluate them by trying to assign each input point to one of the object capsules. To do so, we assign every input point to the object capsule with the highest posterior

probability for this point, *cf.* Section 6.2.1, and compute segmentation accuracy (i. e., the true-positive rate).

The CCAE consistently achieves[6] below 4% error with the best model achieving 2.8% , while the best baseline achieved 26% error using the same budget for hyperparameter search. This shows that wiring in an inductive bias towards modelling geometric relationships can help to bring down the error by an order of magnitude—at least in a toy setup where each set of points is composed of familiar constellations that have been independently transformed.

6.3.2 Unsupervised Class Discovery in Images

We now turn to images in order to assess if our model can simultaneously learn to discover parts and group them into objects. To allow for multimodality in the appearance of objects of a specific class, we typically use more object capsules than the number of class labels. It turns out that the vectors of presence probabilities form tight clusters as shown by their TSNE embeddings (Maaten and G. E. Hinton [108]) in Figure 6.1.1—note the large separation between clusters corresponding to different digits, and that only a few data points are assigned to the wrong clusters. Therefore, we expect object capsules presences to be highly informative of the class label. To test this hypothesis, we train SCAE on MNIST, SVHN[7] and CIFAR10 and try to assign class labels to vectors of object capsule presences. This is done with one of the following methods: LIN-MATCH: after finding 10 clusters[8] with KMEANS we use bipartite graph matching (Kuhn [95]) to find the permutation of cluster indices that minimizes the classification error—this is standard practice in unsupervised classification, see e. g., Ji et al. [75]; LIN-PRED: we train a linear classifier with supervision given the presence vectors; this learns $K \times 10$ weights and 10 biases, where K is the number of object capsules, but it does not modify any parameters of the main model.

[6] This result requires using an additional sparsity loss described in Section 6.C; without it the CCAE achieves around 10% error.

[7] We note that we tie the values of the alpha channel T^a_m and the color values T^c_m which leads to better results in the SVHN experiments.

[8] All considered datasets have 10 classes.

Method	MNIST	CIFAR10	SVHN
KMEANS (Haeusser et al. [56])	53.49	20.8	12.5
AE (Bengio et al. [10])§	81.2	31.4	-
GAN (Radford et al. [127])§	82.8	31.5	-
IMSAT (Hu et al. [67])†,∇	**98.4** (0.4)	45.6 (0.8)	**57.3** (3.9)
IIC (Ji et al. [75])§,†	**98.4** (0.6)	**57.6** (5.0)	-
ADC (Haeusser et al. [56])†	**98.7** (0.6)	29.3 (1.5)	38.6 (4.1)
SCAE (LIN-MATCH)	**98.7 (0.35)**	25.01 (1.0)	**55.33 (3.4)**
SCAE (LIN-PRED)	**99.0 (0.07)**	33.48 (0.3)	**67.27 (4.5)**

Table 6.3.1: Unsupervised classification results in % with (standard deviation) are averaged over 5 runs. Methods based on mutual information are shaded. Results marked with † use data augmentation, ∇ use IMAGENET-pretrained features instead of images, while § are taken from Ji et al. [75]. We highlight the best results and those that are are within its 98% confidence interval according to a two-sided t test.

In agreement with previous work on unsupervised clustering (Haeusser et al. [56], Hjelm et al. [65], Hu et al. [67], and Ji et al. [75]), we train our models and report results on full datasets (TRAIN, VALID and TEST splits). The linear transformation used in LIN-PRED variant of our method is trained on the TRAIN split of respective datasets while its performance on the TEST split is reported.

We used an PCAE with 24 single-channel 11×11 templates for MNIST and 24 and 32 three-channel 14×14 templates for SVHN and CIFAR10, respectively. We used sobel-filtered images as the reconstruction target for SVHN and CIFAR10, as in Jaiswal et al. [74], while using the raw pixel intensities as the input to PCAE. The OCAE used 24, 32 and 64 object capsules, respectively. Further details on model architectures and hyper-parameter tuning are available in Section 6.A. All results are presented in Table 6.3.1. SCAE achieves state-of-the-art results in unsupervised object classification on MNIST and SVHN and under-performs on CIFAR10 due to the inability to model backgrounds, which is further discussed in Section 6.5.

6.3.3 Ablation study

SCAEs have many moving parts; an ablation study shows which model components are important and to what degree. We train SCAE variants on MNIST as well as a

padded-and-translated 40×40 version of the dataset, where the original digits are translated up to 6 pixels in each direction. Trained models are tested on TEST splits of both datasets; additionally, we evaluate the model trained on the 40×40 MNIST on the TEST split of AFFNIST dataset. Testing on AFFNIST shows whether the model can generalise to unseen viewpoints. This task was used by Rawlinson et al. [133] to evaluate Sparse Unsupervised Capsules, which achieved 90.12% accuracy. SCAE achieves $92.2 \pm 0.59\%$, which indicates that it is better at viewpoint generalisation. We choose the LIN-MATCH performance metric, since it is the one favoured by the unsupervised classification community.

Results are split into several groups and shown in Table 6.3.2. We describe each group in turn. Group a) shows that sparsity losses introduced in Section 6.2.4 increase model performance, but that the posterior loss might not be necessary. Group b) checks the influence of injecting noise into logits for presence probabilities, *cf.* Section 6.2.4. Injecting noise into part capsules seems critical, while noise in object capsules seems unnecessary—the latter might be due to sparsity losses. Group c) shows that using similarity (as opposed to affine) transforms in the decoder can be restrictive in some cases, while not allowing deformations hurts performance in every case.

Group d) evaluates the type of the part-capsule encoder. The LINEAR encoder entails a CNN followed by a fully-connected layer, while the CONV encoder predicts one feature map for every capsule parameter, followed by global-average pooling. The choice of part-capsule encoder seems not to matter much for within-distribution performance; however, our attention-based pooling (cf. Section 6.E) does achieve much higher classification accuracy when evaluated on a different dataset, showing better generalisation to novel viewpoints.

Additionally, e) using Set Transformer as the object-capsule encoder is essential. We hypothesise that it is due to the natural tendency of Set Transformer to find clusters, as reported in Lee et al. [101]. Finally, f) using special features $\mathbf{z}_m$ seems not less important—presumably due to effects the high-level capsules have on the representation learned by the primary encoder.

	Method	MNIST	40×40 MNIST	AFFNIST
	full model	**95.3 (4.65)**	**98.7 (0.35)**	**92.2 (0.59)**
a)	no posterior sparsity	**97.5 (1.55)**	**95.0 (7.20)**	**85.3 (11.67)**
	no prior sparsity	72.4 (22.39)	88.2 (6.98)	71.3 (5.46)
	no prior/posterior sparsity	84.7 (3.01)	82.0 (5.46)	59.0 (5.66)
b)	no noise in object caps	**96.7 (2.30)**	**98.5 (0.12)**	**93.5 (0.38)**
	no noise in any caps	**93.1 (5.09)**	78.5 (22.69)	64.1 (26.74)
	no noise in part caps	**93.9 (7.16)**	82.8 (24.83)	70.7 (25.96)
c)	similarity transforms	**97.5 (1.55)**	95.9 (1.59)	88.9 (1.58)
	no deformations	87.3 (21.48)	87.2 (18.54)	79.0 (22.44)
d)	LINEAR part enc	**98.0 (0.52)**	63.2 (31.47)	50.8 (26.46)
	CONV part enc	**97.6 (1.22)**	**97.8 (.98)**	81.6 (1.66)
e)	MLP enc for object caps	27.1 (9.03)	36.3 (3.70)	25.29 (3.69)
f)	no special features	90.7 (2.25)	58.7 (31.60)	44.5 (21.71)

Table 6.3.2: Ablation study on MNIST. All used model components contribute to its final performance. AFFNIST results show out-of-distribution generalization properties and come from a model trained on 40×40 MNIST. Numbers represent average % and (standard deviation) over 10 runs. We highlight the best results and those that are are within its 98% confidence interval according to a two-sided t test.

6.4 Related Work

Capsule Networks Our work combines ideas from Transforming Autoencoders (G. E. Hinton et al. [63]) and EM Capsules (G. E. Hinton et al. [64]). Transforming autoencoders discover affine-aware capsule *instantiation parameters* by training an autoencoder to reconstruct an affine-transformed version of the original image. This model uses an additional input that explicitly represents the transformation, which is a form of supervision. By contrast, our model does not need any input other than the image.

Both EM Capsules and the preceding Dynamic Capsules (Sabour et al. [144]) use the poses of parts and learned part→object relationships to vote for the poses of objects. When multiple parts cast very similar votes, the object is assumed to be present, which is facilitated by an interactive inference (routing) algorithm. Iterative routing is inefficient and has prompted further research (H. Li et al. [105], Wang and

Q. Liu [176], and S. Zhang et al. [190]). In contrast to prior work, we use objects to predict parts rather than vice-versa; therefore we can dispense with iterative routing at inference time—every part is explained as a mixture of predictions from different objects, and can have only one parent. This regularizes the OCAE's encoder to respect the single parent constraint when learning to group parts into objects.

Additionally, since it is the objects that predict parts, the part poses can have fewer degrees-of-freedom than object poses (as in the CCAE). Inference is still possible because the OCAE encoder makes object predictions based on *all* the parts. This is in contrast to each individual part making its own prediction, as was the case in previous works on capsules.

A further advantage of our version of capsules is that it can perform unsupervised learning, whereas previous capsule networks used discriminative learning. Rawlinson et al. [133] is a notable exception and used the reconstruction MLP introduced in Sabour et al. [144] to train Dynamic Capsules without supervision. Their results show that unsupervised training for capsule-conditioned reconstruction helps with generalization to AFFNIST classification; we further improve on their results, *cf*. Section 6.3.3.

Unsupervised Classification There are two main approaches to unsupervised object category detection in computer vision. The first one is based on representation learning and typically requires discovering clusters or learning a classifier on top of the learned representation. Eslami et al. [38] and Kosiorek et al. [91] use an iterative procedure to infer a variable number of latent variables, one for every object in a scene, that are highly informative of the object class, while Burgess et al. [16] and Greff et al. [48] perform unsupervised instance-level segmentation in an iterative fashion. While similar to our work, these approaches cannot decompose objects into their constituent parts and do not provide an explicit description of object shape (e. g., templates and their poses in our model).

The second approach targets classification explicitly by minimizing mutual information (MI)-based losses and directly learning class-assignment probabilities. IIC (Ji et al.

[75]) maximizes an exact estimator of MI between two discrete probability vectors describing (transformed) versions of the input image. DeepInfoMax (Hjelm et al. [65]) relies on negative samples and maximizes MI between the predicted probability vector and its input via noise-contrastive estimation (Gutmann and Hyvärinen [54]). This class of methods directly maximizes the amount of information contained in an assignment to discrete clusters, and they hold state-of-the-art results on most unsupervised classification tasks. MI-based methods suffer from typical drawbacks of mutual information estimation: they require massive data augmentation and large batch sizes. This is in contrast to our method, which achieves comparable performance with batch size no bigger than 128 and with no data augmentation.

Geometrical Reasoning Other attempts at incorporating geometrical knowledge into neural networks include exploiting equivariance properties of group transformations (Cohen and Welling [23]) or new types of convolutional filters (Dieleman et al. [33] and Oyallon and Mallat [125]). Although they achieve significant parameter efficiency in handling rotations or reflections compared to standard CNNs, these methods cannot handle additional degrees of freedom of affine transformations—like scale. Lenssen et al. [102] combined capsule networks with group convolutions to guarantee equivariance and invariance in capsule networks. Spatial Transformers (ST; Jaderberg et al. [73]) apply affine transformations to the image sampling grid while steerable networks (Cohen and Welling [24] and Jacobsen et al. [70]) dynamically change convolutional filters. These methods are similar to ours in the sense that transformation parameters are predicted by a neural network but differ in the sense that ST uses global transformations applied to the whole image while steerable networks use only local transformations. Our approach can use different global transformations for every object as well as local transformations for each of their parts.

6.5 Discussion

The main contribution of our work is a novel method for representation learning, in which highly structured decoder networks are used to train one encoder network that can segment an image into parts and their poses and another encoder network that can compose the parts into coherent wholes. Even though our training objective is not concerned with classification or clustering, SCAE is the only method that achieves competitive results in unsupervised object classification without relying on mutual information (MI). This is significant since, unlike our method, MI-based methods require sophisticated data augmentation. It may be possible to further improve results by using an MI-based loss to train SCAE, where the vector of capsule probabilities could take the role of discrete probability vectors in IIC (Ji et al. [75]). SCAE underperforms on CIFAR10, which could be because of using fixed templates, which are not expressive enough to model real data. This might be fixed by building deeper hierarchies of capsule autoencoders (e. g., complicated scenes in computer graphics are modelled as deep trees of affine-transformed geometric primitives) as well as using input-dependent shape functions instead of fixed templates—both of which are promising directions for future work. It may also be possible to make a much better PCAE for learning the primary capsules by using a differentiable renderer in the generative model that reconstructs pixels from the primary capsules.

Finally, the SCAE could be the 'figure' component of a mixture model that also includes a versatile 'ground' component that can be used to account for everything except the figure. A complex image could then be analyzed using sequential attention to perceive one figure at a time.

6.6 Acknowledgements

We would like to thank Sandy H. Huang for help with editing the manuscript and making Figure 6.1.2. Additionally, we would like to thank S. M. Ali Eslami and Danijar Hafner for helpful discussions throughout the project. We also thank Hyunjik

Kim, Martin Engelcke, Emilien Dupont and Simon Kornblith for feedback on initial versions of the manuscript.

Appendix

6.A Model Details

6.A.1 Constellation Experiments

The CCAE uses a four-layer Set Transformer as its encoder. Every layer has four attention heads, 128 hidden units per head, and is followed by layer norm (Ba et al. [4]). The encoder outputs three 32-dimensional vectors—one for each object capsule. The decoder uses a separate neural net for each object capsule to predict all parameters used to model its points: this includes four candidate part predictions per capsule for a total of 12 candidates. In this experiment, each object→part relationship OP is just a 2-D offset in the object's frame of reference (instead of a 3×3 matrix) and it is affine transformed by the corresponding OV matrix to predict the 2-D point.

6.A.2 Image Experiments

We use a convolutional encoder for part capsules and a set transformer encoder (Lee et al. [101]) for object capsules. Decoding from object capsule to part capsules is done with MLPS, while the input image is reconstructed with affine-transformed learned templates. Details of the architectures we used are available in Table 6.A.1.

Table 6.A.1: Architecture details. S in the last column means that the entry is the same as for SVHN.

Dataset	Constellation	MNIST	SVHN	CIFAR10
num templates	N/A	24	24	32
template size	N/A	11×11	14×14	S
num capsules	3	24	32	64
part CNN	N/A	2x(128:2)-2x(128:1)	2x(128:1)-2x(128:2)	S
set transformer	4x(4-128)-32	3x(1-16)-256	3x(2-64)-128	S

We use ReLu nonlinearities except for presence probabilities, for which we use sigmoids. (128:2) for a CNN means 128 channels with a stride of two. All kernels are 3×3. For set transformer (1-16)-256 means one attention head, 16 hidden units and 256 output units; it uses layer normalization (Ba et al. [4]) as in the original paper (Lee et al. [101]) but no dropout. All experiments (apart from constellations) used 16 special features per part capsule.

For SVHN and CIFAR10, we use normalized sobel-filtered images as the target of the reconstruction to emphasize the shape importance. Figure 6.B.1 in Section 6.B shows examples of SVHN and CIFAR10 reconstruction. The filtering procedure is as follows: 1) apply sobel filtering, 2) subtract the median color, 3) take the absolute value of the image, 4) normalize for image values to be $\in [0, 1]$.

All models are trained with the RMSProp optimizer (Tieleman and G. Hinton [163]) momentum $= .9$ and $\epsilon = (10 * \text{batch\_size})^{-2}$. Batch size is 64 for constellations and 128 for all other datasets. The learning rate was equal to 10^{-5} for MNIST and constellation experiments (without any decay), while we run a hyperparameter search for SVHN and CIFAR10: we searched learning rates in the range of $5 * 10^{-5}$ to $5 * 10^{-4}$ and exponential learning rate decay of 0.96 every 10^3 or $3 * 10^3$ weight updates. Learning rate of 10^{-4} was selected for both SVHN and CIFAR10, the decay steps was 10^3 for SVHN and $3 * 10^3$ for CIFAR10. The LIN-PRED accuracy on a validation set is used as a proxy to select the best hyperparameters—including weights on different losses, reported in Table 6.A.2. Models were trained for up to $3 * 10^5$ iterations on single Tesla V100 GPUs, which took 40 minutes for constellation experiments and less than a day for CIFAR10.

Table 6.A.2: Loss weights values. The *within* and *between* quantifiers in sparsity losses corresponds to different terms of Equations (6.13) and (6.14).

Dataset	Constellation	MNIST	SVHN	CIFAR10
part ll weight	1	1	2.56	2.075
image ll weight	N/A	1	1	1
prior within sparsity	1	1	0.22	0.17
prior between sparsity	1	1	0.1	0.1
posterior within sparsity	0	10	8.62	1.39
posterior between sparsity	0	10	0.26	7.32
too-few-active-capsules	10	0	0	0

6.B Reconstructions

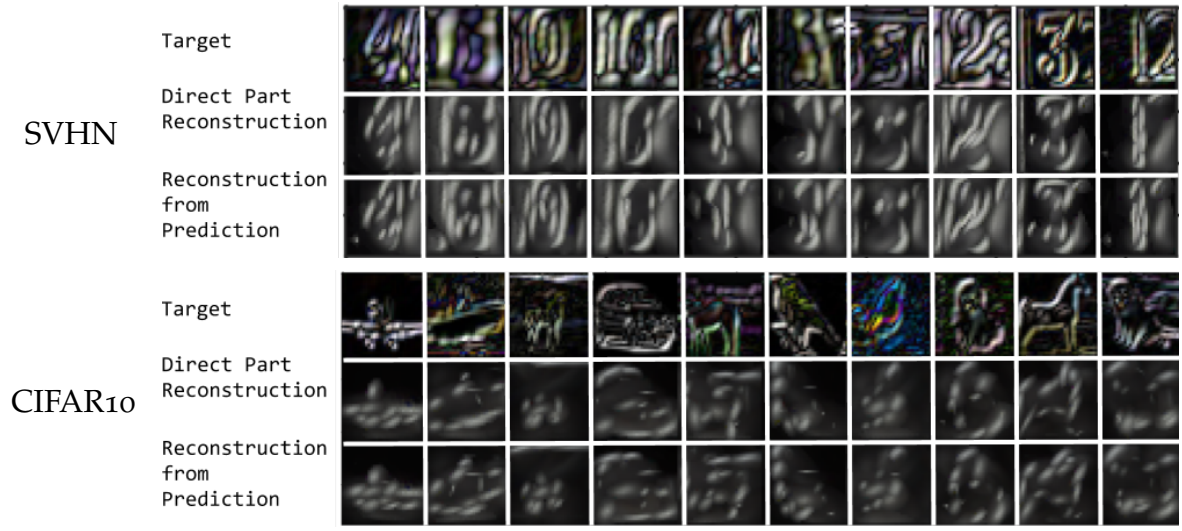


Figure 6.B.1: 10 Sample SVHN and Cifar10 reconstructions. First row shows Sobel filtered target image. Second row shows the reconstruction from Part Capsule Layer directly. Third row shows the reconstruction if we use the object predictions for the Part poses instead of Part poses themselves for reconstruction. The templates in this model has the same number of channels as the image, but they have converged to black and white templates and the reconstruction do not have color diversity. The SCAE model is trained completely unsupervised but the reconstructions tend to focus on the center digit in SVHN and filter the rest of the clutter.

6.C Constellation Capsule Sparsity

We noticed that we can get better instance segmentation results in the constellation experiment when we add an additional sparsity loss, which says that every active object capsule should explain at least two parts. We say that an object capsule has

'won' a part if it has the highest posterior mixing probability for that part among other object capsules. We then create binary labels for each of object capsules, where the label is 1 if the capsule wins at least two parts and it is 0 otherwise. The final loss takes the form of binary cross-entropy between the generated label and the prior capsule presence. This loss is used only for the stand-alone constellation model experiments on point data, *cf.* Sections 6.2.1 and 6.3.1.

6.D Modelling Deformable Objects

Each object capsule votes for part capsules by contributing Gaussian votes to mixture models, where the centers of these Gaussians are a product of an object-viewer OV matrix and an object-part OP matrix, cf. Equations (6.1) to (6.4). Importantly, the $\text{OP}_{k,n}$ matrices are a sum of a static component $\text{OP}^{\text{static}}_{k,n}$, which represents the mean shape of an object, and a dynamic component $\text{OP}^{\text{dynamic}}_{k,n} = \text{MLP}(\mathbf{c}_k)$, which is a function of data, and can model deformations in objects' shape. If an object capsule is to specialise to a specific object class, it should learn its mean shape. Therefore we discourage large deformations, which also prevents an object capsule from modelling several objects at once. Concretely, we add the weighted Frobenius norm of the deformation matrix $\alpha||\text{OP}^{\text{dynamic}}_{k,n}||^2_F$ to the loss, where α is a weight set to a high value, typically $\alpha = 10$ in our experiments.

6.E Part Capsule Encoder with Attention-based Pooling

The PCAE encoder consists of a CNN followed by a bottom-up attention mechanism based on global-average pooling, which we call attention-based pooling. When global-average pooling is typically used, a feature map with d channels is averaged along its spatial dimensions resulting into a d-dimensional vector. This is useful for e. g., counting features of a particular type, which can be useful for classification. In our case, we wanted to predict pose and existence of a particular part, with the constraint that this part can exist at at most one location in the image. In order to support this, we predict a $d+1$ dimensional feature map, where the additional dimension

represents softmax logits for attention. Finally, we compute the weighted average of the feature map along its spatial dimensions, where the weights are given by the softmax. This allows to predict different part parameters at different locations and weigh them by the corresponding confidence.

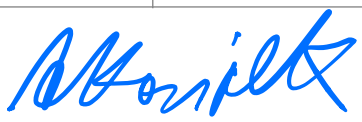
Concretely, for every part capsule k, we use the CNN to predict a feature map $\mathbf{e_k}$ of $6\,(\text{pose}) + 1\,(\text{presence}) + c_z\,(\text{special features})$ capsule parameters with spatial dimensions $h_e \times w_e$, as well as a single-channel attention mask $\mathbf{a_k}$. The final parameters for that capsule are computed as $\sum_i \sum_j \mathbf{e_{k,i,j}}\,\text{softmax}(\mathbf{a})_{\mathbf{k,i,j}}$, where softmax is along the spatial dimensions.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

Title of Paper	Stacked Capsule Autoencoders
Publication Status	Submitted for Publication
Publication Details	A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. "Stacked Capsule Autoencoders". In: *Advances in Neural Information Processing Systems*. 2019. arXiv: 1906.06818. url: https://arxiv.org/abs/1906.06818.

Student Confirmation

Student Name:	Adam R. Kosiorek		
Contribution to the Paper	• Adapt Hinton's idea to modern machine learning toolkit. • Propose architectural changes and a number of sparsity losses that led to desired model characteristics and satisfactory performance. • Implement the model in code. • Design the image experiments and carry out all experiments described in the paper. • Write the paper.		
Signature		Date	06/01/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Yee Whye The 		
Supervisor comments		
Signature 	Date	6/1/2020

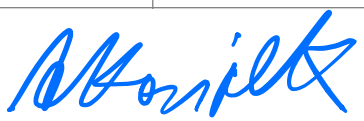
This completed form should be included in the thesis, at the end of the relevant chapter.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

Title of Paper	Stacked Capsule Autoencoders
Publication Status	Submitted for Publication
Publication Details	A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. "Stacked Capsule Autoencoders". In: *Advances in Neural Information Processing Systems*. 2019. arXiv: 1906.06818. url: https://arxiv.org/abs/1906.06818.

Student Confirmation

Student Name:	Adam R. Kosiorek		
Contribution to the Paper	• Adapt Hinton's idea to modern machine learning toolkit. • Propose architectural changes and a number of sparsity losses that led to desired model characteristics and satisfactory performance. • Implement the model in code. • Design the image experiments and carry out all experiments described in the paper. • Write the paper.		
Signature		Date	06/01/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Yee Whye Teh			
Supervisor comments			
Signature		Date	6/1/2020

This completed form should be included in the thesis, at the end of the relevant chapter.

7 Discussion

Learning object-centric representations is a relatively new subfield of machine learning. It has a potential of affecting a wide range of applications ranging from relational reasoning in images, through improving performance in locomotion tasks for e. g., autonomous vehicles, to empowering multi-agent systems and dialogue agents. With few studies available to date, this thesis constitutes an early investigation into learning object-centric representations and focuses on exploring different methods that can be used to this end.

In our investigations, we emphasise that learning object-representations from single images is an ill-posed problem, as there might exist many plausible decompositions of any image into scene components, where different components may or may not correspond to objects—this observation is corroborated by Burgess et al. [16], Engelcke et al. [37], and Greff et al. [48]. Therefore, we argue that it is essential to incorporate powerful priors, inductive biases, or structural constraints that lead to favouring some particular decompositions. Interestingly, it is still unclear what constitutes an object, and it might not be possible to come up with a consistent definition suitable for all applications. Instead, it might be necessary to define objects with respect to a specific task or a utility function. In this thesis, we focus on predictability, where objects

can be seen as *minimal* regions in an image that are sufficient to predict properties of corresponding regions in future images[1] We also impose structural constraints of spatial (Chapters 3 to 6) and temporal (Chapters 3 to 5) consistency, which allows us to learn object-centric representations. It is unclear, however, if any of the explored avenues are sufficient or necessary conditions by themselves, which needs to be determined in further studies.

7.1 Limitations of Investigated Approaches

Approaches investigated in this work are early attempts at learning object-centric representations, and they generally do not generalise to complex real-world data. The supervised-object trackers of Chapters 3 and 4 work well on relatively small datasets with a limited number of classes, but so far we were not able to train them on large datasets containing hundreds of classes like YOUTUBE8M (Abu-El-Haija et al. [2]) or IMAGENET-VID (Russakovsky et al. [143]), and the reasons for this failure remain unknown. We hypothesise that this is partly due to RNN's preference for learning smooth functions, which then cannot accommodate quickly changing object locations contrary to detection- and matching-based trackers. Additionally, our approaches are constrained to look at a relatively small region of the image, and the assumption is that this region contains the tracked object. If this is not the case, however, it is very difficult for our model to recover and resume tracking.

SQAIR of Chapter 5 can learn to detect and track objects without supervision, but in its original form it is constrained to videos from static cameras, in which background can be removed. This is because even though the model can model moving objects well, there are no model components responsible for modelling backgrounds. Moreover, inference in SQAIR exhibits sequential dependencies in the number of objects, which means that applying it to settings containing many objects can be very time-consuming. Fortunately, both of these drawbacks were addressed

[1] This is not obvious in Chapter 6, which works with single images. However, in this chapter, the task is to discover the geometric structure of objects, which can be used to synthesise those objects in different poses.

in SCALOR (Jiang et al. [76]), which works on real dynamic videos with hundreds of objects. However, the datasets used in Jiang et al. [76] are still relatively simple, with slow-moving objects and relatively simple scenes, and further evaluation on more challenging datasets should be undertaken by future research.

SCAE, the capsule-based autoencoder described in Chapter 6, learns geometric structure of objects from images containing simple objects and shows state-of-the-art unsupervised classification performance on MNIST and SVHN. As our experiments show, however, it does not generalise to real-world three-dimensional objects on cluttered backgrounds in CIFAR10. While the structure embedded in the decoder can help to learn good representations, too much structure or too rigid setup can hurt performance on more complicated data, which happens with SCAE. In future work, it would be interesting to investigate equally informative and less rigid forms of structure and inductive bias.

7.2 Evaluating Object-Centric Representations

A good representation should be easy to access and informative about the input, but should not contain the entirety of information present in the input. For example, it is undesirable to preserve information about pixel noise in image representations. Currently, the most popular evaluation protocol for representation learning is to use learned representations as an input to a supervised linear classifier, and assess the performance of that classifier; with some works using non-linear classification with an MLP (Hénaff et al. [60]) or with a support vector machine (SVM) (Hjelm et al. [65]). While these protocols are useful for evaluating the efficacy of global (describing the whole input) representations, they are not particularly useful to evaluate object-centric or other local representations. This motivates the need for novel evaluation protocols that allow to asses how well object-centric latent variables reflect properties of objects they are supposed to represent. Eslami et al. [38] use a classifier to predict the sum of digits present in an image—an approach we also adopt in Chapter 5—where the classifier takes concatenated latent variables extracted from that image; we

adopt this approach in Chapter 5. This approach evaluates object representations only indirectly since it is possible to obtain a perfect classification score even with a global representation. Burgess et al. [16] and Greff et al. [48] use the adjusted rand index (ARI) score to asses image segmentation produced by object-centric latent variables. This choice is criticised by Engelcke et al. [37], who show examples where AIR cannot discriminate between badly- and well-segmented scenes. These metrics are informative of what is captured by different latent variables, but tell us nothing about how easy it is to access this information since access is usually accomplished by a complicated neural network that acts as a decoder. Finally, Kipf et al. [89] proposes K-nearest-neighbour evaluation, where representations extracted at the current video frame are unrolled into the feature and compared to representations extracted from a future frame. The K here tells us how many representations from the dataset was closer to the target representation than the predicted one, and a low rank under this metric indicates good predictive performance. An ideal metric should take into account: (1) representation capability at the current time-step, (2) future predictive capability, (3) relational usefulness—is it possible to figure out relations between objects (spatial and otherwise) by looking at the representations alone? (4) readability—can we use linear classifiers to predict different properties of represented objects? The ability to take all of the above factors into account depends not only on the existence of respective metrics but also of a dataset that contains appropriate ground-truth labels. It would be immensely beneficial for this field of research to come up with such a dataset.

7.3 What are Objects, anyway?

This thesis is about objects and their representations, but we have not stated what an object is. While it is not machine learning that should develop such a definition, it is necessary to have a working definition for machine learning in order to be precise. Therefore, we posit that an object in an image or in a video is nothing but a set of pixels characterised by certain properties and that objects in a single image or video

coexist within a shared context. The context determines the nature of the interaction between the objects and influences their behaviour. Moreover, objects should be conditionally independent of each other, given the context. That is, they should not be informative of each other. Additionally, the total correlation between pixels belonging to one object should be high, and the total correlation between pixels belonging to different objects should be low. It is unclear what the total correlation between pixels belonging to an object and the context should be.

In a different view, an object can be seen as a spatio-temporal event. That is, if a video is analysed as a 6-dimensional signal (time, height, width, three colour channels), then objects can be seen to have a non-zero volume along the time, height and width axes. In this sense, discovering an object in a video is akin to discovering changes in any signal, and could be linked to anomaly or change-point detection. This then allows to extend object detection and object-centric representation learning to abstract events in e. g., audio signals or motion data and might allow using some of object detection techniques to discover temporal events.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, A. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. "YouTube-8M: A Large-Scale Video Classification Benchmark". In: (2016). arXiv: 1609.08675.

[3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. "Social LSTM: Human trajectory prediction in crowded spaces". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[4] J. Ba, J. Kiros, and G. E. Hinton. "Layer Normalization". In: *CoRR* abs/1607.06450 (2016).

[5] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. "Stochastic Variational Video Prediction". In: *CoRR* (2017). arXiv: 1710.11252.

[6] S.-H. Bae and K.-J. Yoon. "Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[7] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. "Emergent Tool Use From Multi-Agent Autocurricula". In: (2019). arXiv: 1909.07528.

[8] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. M. O. Heess, D. Wierstra, P. Kohli, M. M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. "Relational inductive biases, deep learning, and graph networks". In: (2018). arXiv: 1806.01261.

[9] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. "Interaction Networks for Learning about Objects, Relations and Physics". In: *Advances in Neural Information Processing Systems* (Dec. 2016), pp. 4502–4510. arXiv: 1612.00222.

[10] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. "Greedy Layer-wise Training of Deep Networks". In: *Advances in Neural Information Processing Systems*. 2007, pp. 153–160.

[11] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum learning". In: *International Conference on Machine Learning*. New York, New York, USA, 2009. arXiv: arXiv:1011.1669v3.

[12] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft. "Simple online and realtime tracking". In: *IEEE International Conference on Image Processing*. 2016, pp. 3464–3468.

[13] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. "Optimizing the Latent Space of Generative Networks". In: (2017). arXiv: abs/1707.05776.

[14] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. "Signature Verification Using A "Siamese" Time Delay Neural Network". In: *International Journal of Pattern Recognition and Artificial Intelligence*. 1993.

[15] Y. Burda, R. Grosse, and R. Salakhutdinov. "Importance Weighted Autoencoders". In: *International Conference on Learning Representations*. Sept. 2016. arXiv: 1509.00519. URL: http://arxiv.org/abs/1509.00519.

[16] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. "MONet: Unsupervised Scene Decomposition and Representation". In: *CoRR* (2019). arXiv: 1901.11390.

[17] B. Cheung. "Neural Attention for Object Tracking". In: *GPU Technology Conference*. 2016.

[18] B. Cheung, E. Weiss, and B. Olshausen. "Emergence of foveal image sampling from learning to attend in visual scenes". In: *International Conference on Learning Representations* (2017). arXiv: 1611.09430.

[19] M. Cho, S. Kwak, C. Schmid, and J. Ponce. "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals". In: *CoRR* (2015). arXiv: 1501.06170.

[20] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. "A Recurrent Latent Variable Model for Sequential Data". In: *Advances in Neural Information Processing Systems*. June 2015. arXiv: 1506.02216. URL: http://arxiv.org/abs/1506.02216.

[21] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera. "Deep Learning in Video Multi-Object Tracking: A Survey". In: (2019). arXiv: 1907.12740.

[22] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *CoRR* (2015). arXiv: 1511.07289.

[23] T. Cohen and M. Welling. "Group Equivariant Convolutional Networks". In: *International Conference on Machine Learning*. 2016, pp. 2990–2999.

[24] T. Cohen and M. Welling. "Steerable CNNs". In: *International Conference on Representation Learning*. 2017.

[25] E. Crawford and J. Pineau. "Exploiting Spatial Invariance for Scalable Unsupervised Object Tracking". In: *Association for the Advancement of Artificial Intelligence*. 2019. arXiv: 1911.09033.

[26] E. Crawford and J. Pineau. "Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks". In: *Association for the Advancement of Artificial Intelligence*. 2019.

[27] P. Dayan and L. F. Abbott. *Theoretical neuroscience : computational and mathematical modeling of neural systems*. Massachusetts Institute of Technology Press, 2001, p. 460.

[28] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool. "Dynamic Filter Networks". In: *Advances in Neural Information Processing Systems* (2016). arXiv: 1605.09673.

[29] E. Denton and V. Birodkar. "Unsupervised learning of disentangled representations from video". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4417–4426.

[30] E. Denton and R. Fergus. "Stochastic Video Generation with a Learned Prior". In: *International Conference on Machjine Learning*. 2018.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2019.

[32] N. Dhir, A. R. Kosiorek, and I. Posner. "Bayesian Delay Embeddings for Dynamical Systems". In: *NIPS Timeseries Workshop*. 2017. URL: https://pdfs.semanticscholar.org/78e4/cf6912236acc1035519595c34e35a5ee4381.pdf.

[33] S. Dieleman, J. De Fauw, and K. Kavukcuoglu. "Exploiting Cyclic Symmetry in Convolutional Neural Networks". In: *CoRR* (2016). arXiv: 1602.02660.

[34] L. Dinh, J. Sohl-Dickstein, and S. Bengio. "Density estimation using Real NVP". In: *International Conference on Learning Representations*. 2016. arXiv: 1605.08803.

[35] J. Donahue, P. Krähenbühl, and T. Darrell. "Adversarial Feature Learning". In: *International Conference on Learning Representations*. 2017. arXiv: 1605.09782.

[36] J. Donahue and K. Simonyan. In: (2019). arXiv: 1907.02544.

[37] M. Engelcke, A. R. Kosiorek, O. Parker Jones, and I. Posner. "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *International Conference on Learning Representations*. 2019. URL: https://arxiv.org/abs/1907.13052.

[38] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K. Kavukcuoglu, and G. E. Hinton. "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models". In: *Advances in Neural Information Processing Systems*. 2016. arXiv: 1603.08575.

[39] T. Fernando, S. Denman, S. Sridharan, and C. Fookes. "Soft + hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection". In: *Neural networks* (2017).

[40] F. B. Fuchs, O. Groth, A. R. Kosiorek, A. Bewley, M. Wulfmeier, A. Vedaldi, and I. Posner. "Scrutinizing and De-Biasing Intuitive Physics with Neural Stethoscopes." In: *British Machine Vision Conference*. 2019. URL: https://arxiv.org/abs/1806.05502.

[41] F. B. Fuchs, A. R. Kosiorek, L. Sun, O. P. Jones, and I. Posner. "End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning". In: *CoRR*. 2019. arXiv: 1907.12887. URL: https://arxiv.org/abs/1907.12887.

[42] J. V. Gael, Y. W. Teh, and Z. Ghahramani. "The Infinite Factorial Hidden Markov Model". In: *Advances in Neural Information Processing Systems*. 2009, pp. 1697–1704. ISBN: 9781605609492. URL: https://papers.Advances%20in%20Neural%20Information%20Processing%20Systems.cc/paper/3518-the-infinite-factorial-hidden-markov-model.

[43] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (Sept. 2013), pp. 1231–1237.

[44] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Conference on Computer Vision and Pattern Recognition* (2013), pp. 580–587.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[46] D. Gordon, A. Farhadi, and D. Fox. "Re3 : Real-Time Recurrent Regression Networks for Object Tracking". In: *CoRR*. 2017. arXiv: 1705.06368.

[47] A. Graves, 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. "Hybrid computing using a neural network with dynamic external memory". In: *Nature* 538.7626 (Oct. 2016), pp. 471–476.

[48] K. Greff, R. L. Kaufmann, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. "Multi-Object Representation Learning with Iterative Variational Inference". In: *CoRR* (2019). arXiv: 1903.00450.

[49] K. Greff, A. Rasmus, M. Berglund, T. H. Hao, H. Valpola, and J. Schmidhuber. "Tagger: Deep Unsupervised Perceptual Grouping". In: *Advances in Neural Information Processing Systems*. 2016.

[50] K. Greff, S. van Steenkiste, and J. Schmidhuber. "Neural Expectation Maximization". In: *Advances in Neural Information Processing Systems*. 2017.

[51] K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra. "Towards conceptual compression". In: *Advances in Neural Information Processing Systems*. 2016.

[52] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. "Pixelvae: A latent variable model for natural images". In: *International Conference on Learning Representations*. 2016. arXiv: 1611.05013.

[53] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[54] M. Gutmann and A. Hyvärinen. "Noise-contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models". In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 297–304.

[55] D. Ha and J. Schmidhuber. "World Models". In: *CoRR* (2018). arXiv: 1603.10122.

[56] P. Haeusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers. "Associative Deep Clustering: Training a Classification Network with No Labels". In: *German Conference on Pattern Recognition*. 2018, pp. 18–32.

[57] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: (2019). arXiv: 1911.05722.

[58] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. "Mask R-CNN". In: *IEEE Transactions n Pattern Analysis and Machine Intelligence* (2017).

[59] D. Held, S. Thrun, and S. Savarese. "Learning to track at 100 FPS with deep regression networks". In: *European Conference on Computer Vision Workshop*. Springer. 2016. arXiv: 1604.01802.

[60] O. J. Hénaff, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. "Data-Efficient Image Recognition with Contrastive Predictive Coding". In: (2019). arXiv: 1905.09272.

[61] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017.

[62] G. E. Hinton. "Some Demonstrations of the Effects of Structural Descriptions in Mental Imagery". In: *Cognitive Science* 3 (1979), pp. 231–250.

[63] G. E. Hinton, A. Krizhevsky, and S. D. Wang. "Transforming Auto-Encoders". In: *International Conference on Artifical Neural Networks*. 2011.

[64] G. E. Hinton, S. Sabour, and N. Frosst. "Matrix Capsules with EM routing". In: *International Conference on Learning Representations*. 2018.

[65] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio. "Learning Deep Representations by Mutual Information Estimation and Maximization". In: (2019). arXiv: 1808.06670.

[66] J.-T. Hsieh, B. Liu, D.-A. Huang, L. Fei-Fei, and J. C. Niebles. "Learning to Decompose and Disentangle Representations for Video Prediction". In: *Advances in Neural Information Processing Systems*. 2018.

[67] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. "Learning Discrete Representations via Information Maximizing Self-augmented Training". In: *International Conference on Machine Learning*. 2017, pp. 1558–1567.

[68] A. Ilin, I. Prémont-Schwarz, T. H. Hao, A. Rasmus, R. Boney, and H. Valpola. "Recurrent Ladder Networks". In: *Advances in Neural Information Processing Systems*. 2017.

[69] Itseez. *Open Source Computer Vision Library*. https://github.com/itseez/opencv. 2015.

[70] J.-H. Jacobsen, B. De Brabandere, and A. W. Smeulders. "Dynamic steerable blocks in deep residual networks". In: *CoRR* (2017). arXiv: 1706.00598.

[71] J.-H. Jacobsen, J. Van Gemert, Z. Lou, and A. W. M. Smeulders. "Structured Receptive Fields in CNNs". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016. URL: https://www.cv-foundation.org/openaccess/content%7B%5C_%7DIEEE%20Conference%20on%20Computer%20Vision%20and%20Pattern%20Recognition%7B%5C_%7D2016/papers/Jacobsen%7B%5C_%7DStructured%7B%5C_%7DReceptive%7B%5C_%7DFields%7B%5C_%7DIEEE%20Conference%20on%20Computer%20Vision%20and%20Pattern%20Recognition%7B%5C_%7D2016%7B%5C_%7Dpaper.pdf.

[72] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. "Reinforcement Learning with Unsupervised Auxiliary Tasks". In: *arXiv:1611.05397*. 2016. arXiv: 1611.05397.

[73] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems*. 2015. arXiv: 1506.02025v1.

[74] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan. "Capsulegan: Generative adversarial capsule network". In: *European Conference on Computer Vision*. 2018.

[75] X. Ji, J. F. Henriques, and A. Vedaldi. "Invariant Information Distillation for Unsupervised Image Segmentation and Clustering". In: *CoRR* (2018). arXiv: 1807.06653. URL: http://arxiv.org/abs/1807.06653.

[76] J. Jiang, S. Janghorbani, G. de Melo, and S. Ahn. "Scalable Object-Oriented Sequential Generative Models". In: (2019). arXiv: 1910.02384.

[77] S. E. Kahoú, V. Michalski, and R. Memisevic. "RATM: Recurrent Attentive Tracking Model". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshop* (2017). arXiv: 1510.08660.

[78] R. E. Kálmán and R. S. Bucy. "New Results in Linear Filtering and Prediction Theory". In: 1961.

[79] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data". In: *International Conference on Learning Representations*. 2017. arXiv: 1605.06432.

[80] S. Kastner and L. G. Ungerleider. "Mechanisms of visual attention in the human cortex". In: *Annual Review of Neuroscience* 23.1 (2000), pp. 315–341.

[81] C. Kemp and J. B. Tenenbaum. "The discovery of structural form". In: *Proceedings of the National Academy of Sciences* 105.31 (2008), pp. 10687–10692.

[82] A. Kendall, Y. Gal, and R. Cipolla. "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *arXiv:1705.07115* (May 2017). arXiv: 1705.07115.

[83] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. "Motion segmentation & multiple object tracking by correlation co-clustering". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[84] H. Kim and A. Mnih. "Disentangling by factorising". In: *International Conference on Machine Learning*. 2018. arXiv: 1802.05983.

[85] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (2015). arXiv: 1412.6980.

[86] D. P. Kingma and P. Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. 2018.

[87] D. P. Kingma, T. Salimans, and M. Welling. "Improving variational inference with inverse autoregressive flow". In: *Advances in Neural Information Processing Systems*. 2016. arXiv: 1606.04934.

[88] D. P. Kingma and M. Welling. "Auto-encoding variational Bayes". In: *International Conference on Learning Representations*. 2014.

[89] T. Kipf, E. van der Pol, and M. Welling. "Contrastive Learning of Structured World Models". In: (2019). arXiv: 1911.12247.

[90] A. R. Kosiorek, A. Bewley, and I. Posner. "Hierarchical Attentive Recurrent Tracking". In: *Advances in Neural Information Processing Systems*. 2017. arXiv: 1706.09262. URL: http://arxiv.org/abs/1706.09262.

[91] A. R. Kosiorek, H. Kim, Y. W. Teh, and I. Posner. "Sequential Attend, Infer, Repeat: Generative modelling of moving objects". In: *Advances in Neural Information Processing Systems*. 2018. arXiv: 1806.01794. URL: https://arxiv.org/abs/1806.01794.

[92] A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. "Stacked Capsule Autoencoders". In: *Advances in Neural Information Processing Systems*. 2019. arXiv: 1906.06818. URL: https://arxiv.org/abs/1906.06818.

[93] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojí, G. Häger, G. Nebehay, R. Pflugfelder, A. Gupta, A. Bibi, A. Lukežič, A. Garcia-Martin, A. Saffari, P. H. S. Torr, Q. Wang, R. Martin-Nieto, R. Pelapur, R. Bowden, C. Zhu, S. Becker, S. Duffner, S. L. Hicks, S. Golodetz, S. Choi, T. Wu, T. Mauthner, T. Pridmore, W. Hu, W. Hübner, X. Wang, X. Li, X. Shi, X. Zhao, X. Mei, Y. Shizeng, Y. Hua, Y. Li, Y. Lu, Y. Li, Z. Chen, Z. Huang, Z. Chen, Z. Zhang, Z. He, and Z. Hong. "The Visual Object Tracking Vot2016 challenge results". In: *European Conference on Computer Vision Workshop*. 2016.

[94] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, and C. Pal. "Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations". In: *International Conference on Learning Representations*. 2017. arXiv: 1606.01305.

[95] H. W. Kuhn. "The Hungarian Method for the Assignment Problem". In: *Naval Research Logistics Quarterly* (1955), pp. 83–97.

[96] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. "Unsupervised object discovery and tracking in video collections". In: *IEEE International Conference on Computer Vision*. IEEE. 2015, pp. 3173–3181.

[97] M. L. Lambert, M. Schiestl, R. Schwing, A. H. Taylor, G. K. Gajdon, K. E. Slocombe, and A. M. Seed. "Function and flexibility of object exploration in kea and New Caledonian crows". In: *Royal Society Open Science*. 2017. DOI: 10.1098/rsos.170652.

[98] T. A. Le, A. R. Kosiorek, N. Siddharth, Y. W. Teh, and F. Wood. "Revisiting Reweighted Wake-Sleep". In: *Conference on Uncertainty in Artificial Intelligence*. 2019. arXiv: 1805.10469.

[99] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), p. 436.

[100] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[101] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. "Set Transformer". In: *International Conference on Machine Learning*. 2019. arXiv: 1810.00825. URL: http://proceedings.mlr.press/v97/lee19d.html.

[102] J. E. Lenssen, M. Fey, and P. Libuschewski. "Group Equivariant Capsule Networks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8844–8853.

[103] A. Lerner, Y. Chrysanthou, and D. Lischinski. "Crowds by example". In: *Computer Graphics Forum*. 2007.

[104] B. Li, W. Xie, W. Zeng, and W. Liu. "Learning to Update for Object Tracking With Recurrent Meta-Learner". In: *IEEE Transactions on Image Processing* 28 (2019), pp. 3624–3635.

[105] H. Li, X. Guo, B. Dai, W. Ouyang, and X. Wang. "Neural Network Encapsulation". In: *CoRR* (2018). arXiv: 1808.03749.

[106] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. "Deep Learning for Generic Object Detection: A Survey". In: *International Journal of Computer Vision* (2018), pp. 1–58.

[107] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. "BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling". In: (2019). arXiv: 1902.02102.

[108] L. van der Maaten and G. E. Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* (2008), pp. 2579–2605.

[109] C. J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. W. Teh. "Filtering Variational Objectives". In: *Advances in Neural Information Processing Systems*. 2017. arXiv: 1705.09279.

[110] C. J. Maddison, A. Mnih, and Y. W. Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *International Conference on Learning Representations*. 2017.

[111] M. Malinowski, M. Rohrbach, and M. Fritz. "Ask Your Neurons: A Neural-Based Approach to Answering Questions about Images". In: *IEEE International Conference on Computer Vision*. 2015.

[112] J. Marino, Y. Yue, and S. Mandt. "Iterative Amortized Inference". In: *International Conference on Machine Learning*. 2018.

[113] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. "MOT16: A Benchmark for Multi-Object Tracking". In: (2016). arXiv: 1603.00831.

[114] A. Milan, S. Roth, and K. Schindler. "Continuous energy minimization for multitarget tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014).

[115] A. Mnih and K. Gregor. "Neural Variational Inference and Learning in Belief Networks". In: *International Conference on Machine Learning*. Jan. 2014. arXiv: arXiv:1402.0030v2.

[116] A. Mnih and D. J. Rezende. "Variational inference for Monte Carlo objectives". In: *International Conference on Machine Learning*. Feb. 2016. arXiv: 1602.06725. URL: http://arxiv.org/abs/1602.06725.

[117] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. "Recurrent Models of Visual Attention". In: *Advances in Neural Information Processing Systems*. 2014. arXiv: 1406.6247.

[118] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. "Playing Atari with Deep Reinforcement Learning". In: (2013). ArXiv: 1312.5602.

[119] H. Nam and B. Han. "Learning Multi-Domain Convolutional Neural Networks for Visual Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[120] W. Neiswanger and F. Wood. "Unsupervised Detection and Tracking of Arbitrary Objects with Dependent Dirichlet Process Mixtures". In: *CoRR* (2012). arXiv: 1210.3288.

[121] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang. "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking". In: *arXiv Prepr. arXiv1607.05781* (2016). arXiv: 1607.05781.

[122] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang. "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking". In: *IEEE International Symposium on Circuits and Systems* (2017).

[123] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. "WaveNet: A Generative Model for Raw Audio". In: *ISCA Speech Synthesis Workshop*. 2016.

[124] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. June 2016. arXiv: 1606.05328. URL: http://arxiv.org/abs/1606.05328.

[125] E. Oyallon and S. Mallat. "Deep Roto-Translation Scattering for Object Classification". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2865–2873.

[126] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *IEEE International Conference on Computer Vision*. 2009.

[127] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *International Conference on Learning Representations* (2016).

[128] T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh. "Tighter Variational Bounds are Not Necessarily Better". In: *International Conference on Machine Learning*. 2018. URL: https://arxiv.org/abs/1802.04537.

[129] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. "Stand-Alone Self-Attention in Vision Models". In: (2019). arXiv: 1906.05909.

[130] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. "Video (language) modeling: a baseline for generative models of natural videos". In: *CoRR* (2014). arXiv: 1412.6604.

[131] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. "Semi-supervised Learning with Ladder Networks". In: *Advances in Neural Information Processing Systems*. 2015.

[132] M. Rasouli Danesh, S. Yadav, S. Herath, Y. Vaghei, and S. Payandeh. "Deep Attention Models for Human Tracking Using RGBD". In: *Sensors* 19 (Feb. 2019), p. 750.

[133] D. Rawlinson, A. Ahmed, and G. Kowadlo. "Sparse Unsupervised Capsules Generalize Better". In: *CoRR* (2018). arXiv: 1804.06094.

[134] A. Razavi, A. van den Oord, and O. Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2". In: (2019). arXiv: 1906.00446.

[135] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[136] S. Ren, K. He, R. B. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2015), pp. 1137–1149.

[137] D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*. 2015. arXiv: 1505.05770.

[138] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *International Conference on Machine Learning*. 2014.

[139] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. "Performance measures and a data set for multi-target, multi-camera tracking". In: *European Conference on Computer Vision*. Springer. 2016, pp. 17–35.

[140] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savaresei. "Learning Social Etiquette: Human Trajectory Prediction In Crowded Scenes". In: *European Conference on Computer Vision* (2016).

[141] I. Rock. *Orientation and form*. Academic Press, 1973.

[142] A. Rudenko, L. Palmieri, and K. O. Arras. "Joint long-term prediction of human motion using a planning-based social force approach". In: *2018 IEEE International Conference on Robotics and Automation*. IEEE. 2018.

[143] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115 (2014), pp. 211–252.

[144] S. Sabour, N. Frosst, and G. E. Hinton. "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems*. 2017.

[145] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese. "Sophie: An attentive gan for predicting paths compliant to social and physical constraints". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[146] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. W. Battaglia, and T. Lillicrap. "A simple neural network module for relational reasoning". In: *Arxiv* (June 2017), pp. 1–16. arXiv: 1706.01427.

[147] R. E. Schapire. "A Brief Introduction to Boosting". In: *International Joint Conferences on Artificial Intelligence*. 1999.

[148] M. S. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. "Modeling Relational Data with Graph Convolutional Networks". In: *ESWC*. 2017.

[149] J. Schmidhuber. "Learning Factorial Codes by Predictability Minimization". In: *Neural Computation* 4 (1992), pp. 863–879.

[150] J. Schmidhuber and D. Prelinger. "Discovering Predictable Classifications". In: *Neural Computation* 5 (1993), pp. 625–635.

[151] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll. "The Simpler the Better: Constant Velocity for Pedestrian Motion Prediction". In: *arXiv preprint arXiv:1903.07933* (2019).

[152] C. Schuldt, I. Laptev, and B. Caputo. "Recognizing human actions: A local SVM approach". In: *ICPR*. IEEE, 2004. arXiv: 1505.04868.

[153] S. Schulter, P. Vernaza, W. Choi, and M. K. Chandraker. "Deep Network Flow for Multi-object Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2730–2739.

[154] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1874–1883.

[155] N. Srivastava, E. Mansimov, and R. Salakhutdinov. "Unsupervised Learning of Video Representations using LSTMs." In: *International Conference on Machine Learning*. 2015, pp. 843–852.

[156] S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber. "Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions". In: *International Conference on Learning Representations*. 2018.

[157] S. van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem. In: *Advances in Neural Information Processing Systems*. 2019. arXiv: 1905.12506.

[158] K. Stelzner, R. Peharz, and K. Kersting. "Faster Attend-Infer-Repeat with Tractable Probabilistic Models". In: *International Conference on Machine Learning*. 2019.

[159] M. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber. "Deep Networks with Internal Selective Attention through Feedback Connections". In: *arXiv Prepr. arXiv ...* 2014, p. 13. arXiv: 1407.3068.

[160] H. Su, Y. Dong, J. Zhu, H. Ling, and B. Zhang. "Crowd Scene Understanding with Coherent Recurrent Neural Networks". In: *International Joint Conferences on Artificial Intelligence*. 2016.

[161] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett. "3DOF pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data". In: *2018 IEEE International Conference on Robotics and Automation*. IEEE. 2018.

[162] P. Swerling. "First-order error propagation in a stagewise smoothing procedure for satellite observations". In: (1959).

[163] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.

[164] T. Tieleman. *Optimizing Neural Networks That Generate Images*. University of Toronto, Canada, 2014.

[165] P. Trautman and A. Krause. "Unfreezing the robot: Navigation in dense, interacting crowds". In: *International Conference on Intelligent Robots and Systems*. 2010.

[166] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. "On Mutual Information Maximization for Representation Learning". In: (2019). arXiv: 1907.13625.

[167] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. "Mocogan: Decomposing motion and content for video generation". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

[168] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. "Neural Autoregressive Distribution Estimation". In: *Journal of Machine Learning Research* 17 (2016), 205:1–205:37.

[169] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. "End-to-end representation learning for Correlation Filter based tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017. arXiv: 1704.06036.

[170] D. Varshneya and G. Srinivasaraghavan. "Human trajectory prediction using spatially aware deep attention models". In: (2017). arXiv: 1705.09436.

[171] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* (2017).

[172] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. B. Tenenbaum, and S. Levine. "Entity Abstraction in Visual Model-Based Reinforcement Learning". In: (2019). arXiv: 1910.12827.

[173] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. "Grammar as a Foreign Language". In: *Advances in Neural Information Processing Systems*. 2015. arXiv: 1412.7449.

[174] P. A. Viola and M. J. Jones. "Rapid object detection using a boosted cascade of simple features". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2001).

[175] E. Wagstaff, F. B. Fuchs, M. Engelcke, I. Posner, and M. A. Osborne. "On the Limitations of Representing Functions on Sets". In: *International Conference on Machine Learning* (2019).

[176] D. Wang and Q. Liu. "An Optimization View on Dynamic Routing Between Capsules". In: *International Conference on Learning Representations Workshop*. 2018.

[177] N. Watters, L. Matthey, C. P. Burgess, and A. Lerchner. "Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs". In: (2019). arXiv: 1901.07017.

[178] N. Watters, D. Zoran, T. Weber, P. W. Battaglia, R. Pascanu, and A. Tacchetti. "Visual Interaction Networks: Learning a Physics Simulator from Video". In: *Advances in Neural Information Processing Systems* (2017).

[179] T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. "Imagination-augmented agents for deep reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2017.

[180] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. "DETRAC: A New Benchmark and Protocol for Multi-Object Tracking". In: *arXiv* 1511.04136 (2015).

[181] Y. Xiang, A. Alahi, and S. Savarese. "Learning to Track: Online Multi- Object Tracking by Decision Making Multi-Object Tracking". In: *IEEE International Conference on Computer Vision* (2015).

[182] F. Xiao and Y. Jae Lee. "Track and segment: An iterative unsupervised approach for video object proposals". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 933–942.

[183] J. Xu, J.-F. Ton, H. Kim, A. R. Kosiorek, and Y. W. Teh. "MetaFun: Meta-Learning with Iterative Functional Updates". In: *CoRR*. 2019. arXiv: 1912.02738. URL: https://arxiv.org/abs/1912.02738.

[184] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[185] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. "Who are you with and where are you going?" In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2011.

[186] K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum. "CLEVRER: CoLlision Events for Video REpresentation and Reasoning". In: (2019). arXiv: 1910.01442.

[187] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. "UnitBox: An Advanced Object Detection Network". In: *Proc. 2016 ACM Multimed. Conf.* ACM. 2016, pp. 516–520. arXiv: 1608.01471.

[188] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, and A. Smola. "Deep Sets". In: *Advances in Neural Information Processing Systems*. 2017. arXiv: arXiv:1703.06114v3.

[189] L. Zhang, Y. Li, and R. Nevatia. "Global data association for multi-object tracking using network flows". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[190] S. Zhang, Q. Zhou, and X. Wu. "Fast Dynamic Routing Based on Weighted Kernel Density Estimation". In: *International Symposium on Artificial Intelligence and Robotics*. 2018, pp. 301–309.