University of Heidelberg

Department of Physics and Astronomy

Master's Thesis

in **Physics**

in the year **2020**

submitted by

Jannik Lukas Kossen,

born in **Nordhorn, Germany**.

Modelling Videos of Physically Interacting Objects

This thesis has been carried out by

Jannik Lukas Kossen

under the supervision of

First Supervisor **Prof. Dr. Daniel Durstewitz**
Theoretical Neuroscience
Department of Physics and Astronomy
University of Heidelberg and Central Institute of Mental Health

&

Second Supervisor **Prof. Dr. Kristian Kersting**
Artificial Intelligence and Machine Learning Lab
Department of Computer Science
Darmstadt University of Technology.

Modelling Videos of Physically Interacting Objects

A strong prior in human cognition is the notion of objects. Humans exhibit excellent predictive capabilities in dynamical settings, precisely estimating future object trajectories despite complex physical interactions. In machine learning, designing models capable of making similarly accurate estimates about future states and obtaining a useful representation of the world in an unsupervised fashion is a challenging research question. Building models with inductive biases inspired by those of human perception presents a promising path towards solving these problems. This thesis presents STOVE, a novel machine learning architecture capable of modelling videos of physically interacting objects. STOVE combines recent advances in unsupervised object-aware image modelling and relational physics modelling to yield a non-linear state space model, which explicitly reasons about objects and their interactions. It is trained from pixels without any external supervision. We demonstrate STOVE on simulated videos of physically interacting objects, where it clearly improves upon previous unsupervised and object-aware approaches to video modelling, even approaching the performance of a supervised baseline. STOVE's predictions exhibit long-term stability, and it correctly conserves the kinetic energy in one of the prediction scenarios. Additionally, STOVE is extended to action-conditioned video prediction, advancing video models towards more general interactions beyond physical simulations.

Modellierung von Videos physikalisch interagierender Objekte

Objekte sind ein wichtiger Prior menschlicher Wahrnehmung. Zusätzlich zeigen Menschen exzellente prädiktive Fähigkeiten in dynamischen Situationen. Selbst bei komplexen physikalischen Interaktionen treffen wir präzise Vorhersagen über die Trajektorien von Objekten. Im maschinellen Lernen ist die Konstruktion von Modellen, die nützliche Repräsentationen der Welt hervorbringen und zu akkuraten Vorhersagen fähig sind, Gegenstand aktueller Forschung. Das Design von Modellen mit induktiven Biases, ähnlich derer menschlicher Wahrnehmung, stellt hierbei einen vielversprechenden Weg dar. Diese Arbeit präsentiert STOVE, einen neuartigen Ansatz des maschinellen Lernens, um Videos physikalisch interagierender Objekte zu modellieren. STOVE kombiniert Fortschritte unüberwachter objektorientierter Bildmodelle und relationaler Physikmodelle, um ein nichtlineares Zustandsraummodell zu bilden, welches explizit Objekte und ihre Interaktionen miteinbezieht. Das Modell wird vollständig unüberwacht auf Bildpixeln trainiert. Wir zeigen anhand von Videos physikalischer Simulationen, dass STOVE die Ergebnisse vorheriger unüberwachter Modelle verbessern kann und sogar an die Performanz einer überwachten Baseline herankommt. Die Vorhersagen von STOVE sind über lange Zeiträume stabil und in einer der Anwendungsszenarien lernt STOVE korrekterweise die Erhaltung der kinetischen Energie. Darüber hinaus wird STOVE um die aktionskonditionierte Videovorhersage erweitert, was einen wichtigen Schritt zur Entwicklung von Videomodellen für allgemeinere Interaktionen jenseits strikter physikalischer Simulationen darstellt.

Acknowledgements

I am grateful to Prof. Kristian Kersting for supervising this thesis. I have greatly enjoyed our conversations, whose impact spreads beyond this document, and which have had a lasting effect on my personal thoughts and convictions.

I would like to extend a special thank you to Prof. Daniel Durstewitz. Without your trust and agreement to the supervision, this thesis would have been impossible.

I would like to thank Karl Stelzner for his excellent supervision and keen mathematical insights. I truly learnt a lot from our countless discussions, and I am grateful for all the useful feedback.

Thank you Karl, Claas, and Marcel for the incredible teamwork leading up to the ICLR deadline. I still cannot believe how much we got done in those few weeks. Thank you for believing in our model and thank you for your contributions, without which, I am certain, successful publication would have been much less likely. September was easily the most fun and gratifying month of this thesis.

Finally, I want to thank all the researchers and students in the Artificial Intelligence and Machine Learning Lab at Technical University Darmstadt for the perfect balance of thought-provoking conversations and distracting chatter in the office kitchen and beyond.

Contents

1 Introduction

1.1 Motivation

Humans are remarkable. Every day we solve a multitude of complex and diverse tasks with incredible skill. When picking up new tasks, we efficiently leverage prior experience to accelerate the learning process. We possess a complex mental model of the world, allowing us to reason about our decisions. Human intelligence far outranks any artificial intelligence today (Chollet, 2019; Lake et al., 2017).

Many problems of artificial intelligence may be approached with machine learning. Over the last decade, such machine learning based approaches have seen great progress with the advent of deep learning based architectures. Following the excellent overview of LeCun et al. (2015): In many areas of machine learning that require highly non-linear function approximation, such as computer vision or natural language processing, deep learning based approaches have been able to improve upon state-of-the-art performance significantly. While previous approaches relied on hand-designed feature extractors as input to shallow models, deep learning based architectures can, in many cases, automatically extract complex non-linear features. However, *deep learning* is not one *thing*, not a single model with which to solve every problem. It is a general concept, a buzzword, which best encapsulates the approaches taken by many machine learning models of the current wave. Its progress has been enabled by a mix of new model architectures, advances in optimisation methods, faster processors, and the availability of large amounts of curated data. Current state-of-the-art deep learning architectures *outperform* humans on some tasks. Considerable media attention was given to the cases of ATARI video games (Mnih et al., 2015) or the board game GO (Silver et al., 2016, 2017).

However, as lamented by Chollet (2019), many of these models are extremely far away from *true intelligence*. As defined by Legg and Hutter (2006),

> "Intelligence measures an agent's ability to achieve goals in a wide range of environments." (Legg and Hutter (2006))

If the goal of the artificial intelligence community truly is to achieve intelligent systems, the current modus operandi of large parts of the community seems problematic: Intelligence has primarily been measured as *skill*. Skill is measured as performance on some success metric, such as accuracy, on a narrowly defined task for which the model has been specifically trained. In some parts of the community, this has downright led to *benchmarking races*: State-of-the-art results on popular benchmarking tasks were released by one group and then beaten by others in quick succession.[1] To be fair, this has lead to impressive improvements on these tasks.

[1] See paperswithcode.com/sota/image-classification-on-imagenet to get an idea of the amount of papers and their timing on the ImageNet classification task (last accessed on 25th March, 2020).

However today, arguably, performance on many of the popular benchmarks is saturated. We may be close to the point where beating the previous state-of-the-art may only be achieved by implicitly overfitting the model architecture to the data set, see e. g. Recht et al. (2019).

Chollet (2019) finds fault with a different consequence of this benchmarking mentality. He argues that few of the improvements of the last years have resulted in models that are more *intelligent* as defined above. They often require large amounts of carefully curated and labelled data to allow for learning, are limited in their capability to generalise knowledge to new tasks, and have no deeper understanding of their actions, i. e. they cannot reason about their decisions or act in logically sound ways. This is problematic beyond a philosophical debate about what high-level goals the community should aim for. Instead, it has real-world consequences. In many use cases, current state-of-the-art machine learning models may not be applicable. For example, the creation of high-quality data sets with sufficient labelled data may be prohibitively expensive. Of course, the problems mentioned above have been given attention by the community – some relevant areas are transfer learning, multi-task learning, explainable or interpretable models, learning from weak supervisory signals, semi-supervised learning, data-efficient reinforcement learning, or neuro-symbolic approaches combining reasoning and data – but Chollet (2019) argues that time is ripe for a widespread disruptive change in the community towards *building more intelligent models*.

It should be obvious that none of the current machine learning models can match the versatility of human intelligence. However, Chollet (2019) proposes that we should focus research efforts on models with similarly desirable properties. Adding to the definition of Legg and Hutter (2006), he posits that intelligence, in the context of artificial intelligence research, should be measured as follows:

> "The intelligence of a system is a measure of its skill-acquisition efficiency over a scope of tasks, with respect to priors, experience, and generalization difficulty." (Chollet (2019))

Lake et al. (2017) argue that seeking inspiration from humans is the way to build such models. The artificial intelligence community may be far from creating true intelligence, but cognitive science, neuroscience, psychology, and related disciplines have a good understanding of many processes related to human intelligence. With careful design, such priors of human cognition may be translated into mathematics and computer code. In this way, machine learning models inspired by powerful human priors can be built, going beyond the strained biological motivations of neural networks. The manifest of Lake et al. (2017) is called "Building machines that learn and think like people", and we see the work presented here directly following its philosophy.

Human vision is inherently based on the notion of *objects*. This strong innate prior can already be found in infants (Zelazo and Johnson, 2013). From a physics perspective, our vision apparatus, the eyes, is constantly bombarded with large amounts of photons. From a machine learning perspective, this photon shower corresponds to vast amounts of unstructured data as input to our vision model. As humans, we do not perceive this unstructured data. Instead, we perceive a highly structured and complex representation of our world. A fundamental entity of this representation is the *object*. We perceive objects and endow them with properties or

argue about relations between them. The notion of objects is a central ingredient to intelligence, intricately connected to human vision. Recently, this has lead to the development of machine learning models of images which include this notion of objects at their core, see Section 3.1. Such models hold the promise of extracting a variety of knowledge in an unsupervised fashion. They can infer segmentation masks, obtain latent object representations useful for reasoning in downstream tasks, or learn to count the total number of objects present in a scene. Importantly, they have been shown to improve upon the generalisation capabilities of previous approaches, e. g. generalising to unseen numbers of objects without requiring any extra training (Eslami et al., 2016).

Another strong innate prior to human cognition is *intuitive physics* (McCloskey, 1983). Humans intuitively anticipate the outcome of collisions between objects or the movement of a ball falling in gravity. In other words, we effortlessly make precise predictions about future trajectories of physically acting objects. These physical reasoning capabilities are essential to our successful navigation of the world. We should therefore seek to create machine learning models, which can likewise predict the outcomes of complex physical interactions. In the machine learning community, a mountain of research trying to teach physics to machines has been established over the last years, see Section 3.2. Examples of *physics* scenarios include teaching a model to play billiards, non-linear physics simulations, fluid dynamics, rigid body physics, or even real-life robotics applications. As with object-based image models, critical innovations in model architectures were necessary to obtain models suited to physical predictions. However, many of these approaches rely on strong supervisory signals, such as ground truth trajectories or a simulator. While their performance is impressive, such work often does not scale efficiently to many interesting real-world applications, e. g. robotics, because the collection of sufficient supervisory signals is not feasible.

Our work combines the priors of object-based vision and intuitive physics to yield a generative model of videos trained directly on raw pixels. It seems obvious that the inclusion of an object-based representation for image modelling is helpful for the task of physics prediction and vice versa. Rather than restricting our modelling capabilities in comparison to more general approaches, the inclusion of these inductive biases seems to empower the performance of the model.

1.2 Contribution

This thesis presents a machine learning model which learns about objects and their interactions from raw pixels in an unsupervised fashion. We call the model *STOVE*, loosely abbreviating *Structured Object-Aware Physics Prediction for Video Modelling and Planning*. A first overview of its capabilities is given in Fig. 1.1. More specifically, a generative model of videos is proposed as a non-linear state space model. We construct the video model as the composition of a generative unsupervised model of static images and a graph neural network for modelling the relational dynamics between object states. Skipping many critical details, one may give an intuitively appealing description of this model in three sentences: The image model infers object-specific states from single images. The dynamics model can propagate these states forward in time, predicting future object trajectories. The image model can also render future frames of video from the predicted states. Model training is

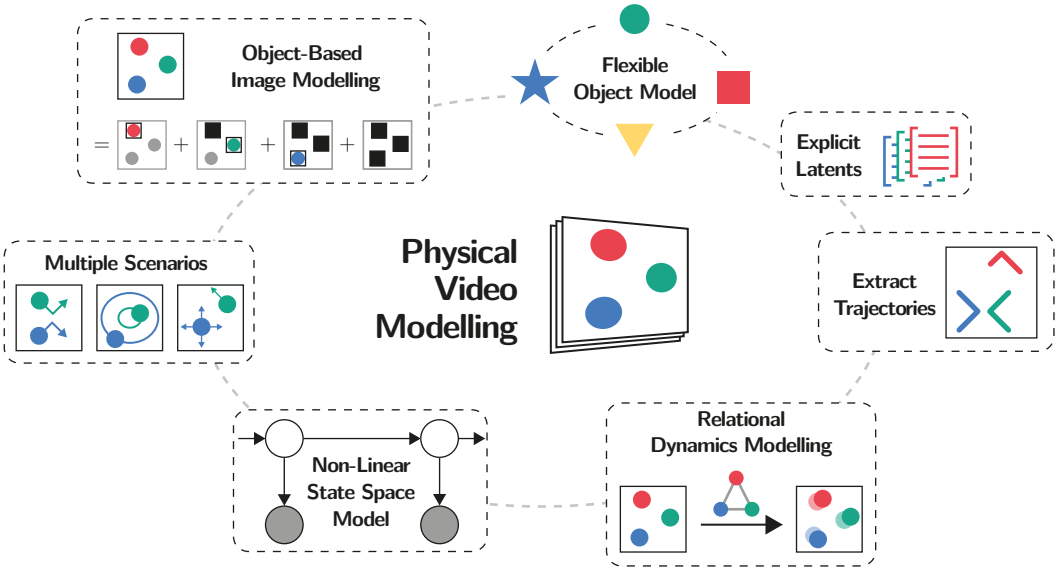


Figure 1.1.: This thesis introduces STOVE, a generative model of videos of physically interacting objects. STOVE models images in terms of their constituent objects using a Sum-Product Attend-Infer-Repeat image model. The image model infers object-specific latent states, part of which explicitly correspond to object positions, velocities, and sizes. From this interpretable latent space, future latent states may be predicted using a relational dynamics model, a graph neural network. Image and object appearance are not hard coded but learned instead. Likewise, STOVE can accommodate a variety of physical simulation scenarios. The STOVE architecture corresponds to a probabilistic non-linear state space model. Learning in this model is made tractable using amortised variational inference.

accomplished using amortised variational inference in the probabilistic state space model. This conceptually simple approach significantly improves upon previous work in future frame or state prediction tasks. Notably, in one of the tasks, our model manages to correctly preserve the kinetic energy of the system, leading to predictions with impressive stability over long timeframes. This is in stark contrast to previous approaches whose predictions quickly diverge to unrealistic trajectories.

More philosophically speaking and connecting to the previous discussion, STOVE presents a counterpoint to the prevailing wisdom of approaching complex problems with large unstructured models and lots of data. While the achievements of deep and mostly unstructured neural networks on classic tasks of computer vision or natural language processing are impressive, the explicit specification of model structure informed by prior knowledge massively benefits performance and training efficiency. Discovering structure in data remains central to machine learning, and one may not always want to predefine models as much as we have done with STOVE. However, STOVE demonstrates that with minimal model building, we can achieve predictive performance drastically superior to unstructured approaches. Note that this approach to structured problem-solving is undoubtedly distinct from old-fashioned, hand-crafted machine learning. STOVE consists entirely of flexible neural networks and deep probabilistic models and is adaptable to a wide range of scenarios. It is the careful delegation of tasks among these components, the design of the communication interfaces between them, and the formulation of the probabilistic model, that are key components of a new wave of structured machine

learning architectures, of which STOVE is an example.

Part of my work on this thesis resulted in the publication *Structured Object-Aware Physics Prediction for Video Modeling and Planning* (Kossen et al., 2020), which is to be published in the proceedings of the *Eighth International Conference of Learning Representations* (ICLR), a highly-ranked conference in machine learning. I am grateful for the work of my co-authors, without whom successful publication would not have been possible. Some of the results in the publication can also be found in this thesis. However, unless otherwise mentioned, they were obtained by me. To avoid any uncertainties, I will attribute my co-authors whenever I discuss results from the paper which cannot be unambiguously attributed to my own work. Additionally, since the publication in the proceedings coincides with the submission of this thesis, and a preprint of the paper has been available since 25th September, 2019, I will cite the paper whenever necessary to avoid self-plagiarism. The thesis can be seen as an extended version of the paper, containing additional experiments and insights, as well as a more thorough treatment of the theoretical background, related works, and derivations.

1.3 Outline

Chapter 2 introduces generally useful concepts from machine learning crucial to this thesis, focussing on machine learning for sequences, variational inference, and sum-product networks. The chapter lays the theoretical foundation for STOVE and the discussed related work. Chapter 3 then serves as an introduction to image, physics, and video modelling. Previous approaches influential and tangential to this thesis are discussed. Of special importance is Section 3.1.2, which introduces SuPAIR, a modified version of which is used as the image model in STOVE. Likewise, Section 3.2.2 introduces relational physics modelling using graph neural networks, a variant of which comprises our dynamics model. Chapter 4 then formally introduces STOVE with all the components of its non-linear state space model. Chapter 5 presents a thorough experimental evaluation of STOVE with a comparison against baselines, an ablation study, as well as more model-specific considerations. Chapter 6 then discusses some exciting ideas for future research, before closing this thesis with a final summary.

2 Concepts

This chapter introduces recurrent neural networks, variational inference and the variational autoencoder, latent state space models, as well as sum-product networks. These concepts present the essential theoretical foundation for STOVE as well as the discussed related works. Throughout this chapter, we will assume that the reader is familiar with machine learning on the level of an introductory university course, i. e. foundational knowledge of probability theory as well as basic classification, regression, clustering, and dimensionality reduction methods. Additionally, some basic knowledge of fully connected neural networks and their training by backpropagation is helpful. This information may be found in any machine learning textbook, e. g. Bishop (2006) or Barber (2012).

Rather than introducing each concept wherever it appears first, we introduce them all together here. While this leads to a more structured approach, it also results in a decently long chapter on machine learning methods without constantly visible links to this thesis. You may of course skip parts you are familiar with. You may even skip concepts you are *not* acquainted with, as they will be referenced when needed, such that you can then come back here.

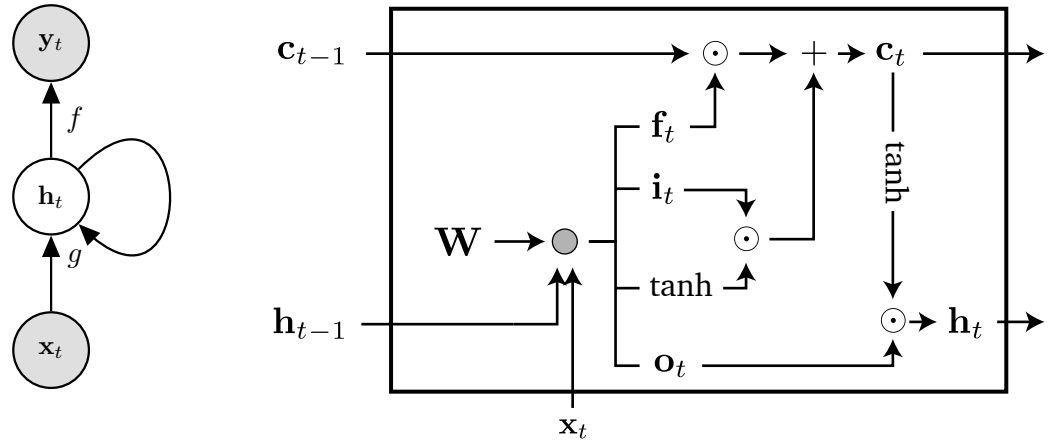
2.1 Recurrent Neural Networks

Feed-forward neural networks are powerful function approximators. Given a *single* input $\mathbf{x} \in \mathbb{R}^D$, they learn to approximate the desired output $\mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^M$, where $\mathbf{y}$ might be the prediction for a regression or classification task. However, they have no natural way to process a *sequence* of observations $\mathbf{x}_{1:T} = \{\mathbf{x}_t\}_{t=1}^T \in \mathbb{R}^{T\times D}$. While one could reshape the sequence to form a single observation $\mathbf{x} \in \mathbb{R}^{T\cdot D}$, this approach does not scale to long sequences, cannot handle sequences of varying length, and discards temporal structure. Recurrent Neural Networks (RNNs) remedy this by introducing *recurrent* connections, i. e. self-connections, to neural networks (Elman, 1990). Now, input sequences are consumed sequentially, e. g. such that

$$\mathbf{h}_t = g(\mathbf{x}_t,\, \mathbf{h}_{t-1}) \tag{2.1}$$

$$\mathbf{y}_t = f(\mathbf{h}_t)\,, \tag{2.2}$$

where f and g are non-linear transformations as in a feed-forward network. Often, this is a single linear transformation followed by a non-linearity. Figure 2.1a shows a graphical depiction of such a network. At each timestep t, the mapping g updates a hidden state $\mathbf{h}_t$ by considering the current position of the input $\mathbf{x}_t$ as well as the previous hidden state $\mathbf{h}_{t-1}$. An output $\mathbf{y}_t$ may be produced from the hidden state with the mapping f. This hidden state allows the RNN to model dependencies over time, retaining and accessing information from previous timesteps. If the desired output is again a sequence, $\mathbf{y} = \{\mathbf{y}_t\}_{t=1}^T$ may be concatenated, if not, $\mathbf{y} = \mathbf{y}_T$ does the trick. Of course, RNNs may also be applied when the input is not a sequence but



(a) (b)

Figure 2.1.: (a) Illustration of a simple Recurrent Neural Network (RNN). At time t, the function g uses the input $\mathbf{x}_t$ to update a hidden state $\mathbf{h}_{t-1}$. Output $\mathbf{y}_t$ is produced using f and the current hidden state $\mathbf{h}_t$. (b) Procedural flow of a Long-Short Term Memory (LSTM) network: The previous hidden state $\mathbf{h}_{t-1}$ and current input $\mathbf{x}_t$ are used to construct the gates $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$, which are then used to update the hidden and cells states $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$. Figure adapted from Bonse (2019).

the output should be, by setting $\mathbf{x}_t = \mathbf{x}, \forall t$. See Goodfellow et al. (2016, Chap. 10) or Lipton et al. (2015) for excellent ground-up introductions to RNNs, the latter including a more thorough review of current literature.

LSTMs. While immediately intriguing, early implementations of RNNs were plagued with problems, preventing their widespread use. Mainly, they struggled to model *long-term dependencies*. As the hidden state is propagated through time, it is multiplied with the state transition matrix. In many cases, this will either shrink (or increase) the modulus of the hidden state. Repeated applications will lead to exponentially decaying (or exploding) states and gradients. This became known as the "vanishing gradient problem" in RNNs (Hochreiter, 1991, 1998). The introduction of the Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) architecture essentially solved this problem by introducing fixed unit-weight recurrent edges (Lipton et al., 2015). Now, for the first time, RNNs could successfully model long-term dependencies (Hochreiter and Schmidhuber, 1997). The use of LSTM-based models in time series modelling, natural language processing, and even computer vision exploded. They became the de-facto standard for dealing with sequences in neural networks, achieving state-of-the-art performance on many tasks (Lipton et al., 2015). Formally, the update equations of the LSTM are given as

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{W}_{xc}\mathbf{x}_t + \mathbf{b}_c) \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)\,, \end{aligned} \tag{2.3}$$

where $\odot$ signifies element-wise multiplication. Here, $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ implement gating mechanisms, given by

$$\begin{aligned}\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f)\\ \mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i)\\ \mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)\,,\end{aligned} \tag{2.4}$$

where all matrices $\mathbf{W}$ and vectors $\mathbf{b}$ in the above Eqs. (2.3) and (2.4) are learnable parameters and $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid function. Figure 2.1b visualises the above operations. The gates $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ explicitly control the flow of information between all cells of the LSTM. The forget gate $\mathbf{f}_t$ controls the amount of information kept from the previous cell state, the input gate $\mathbf{i}_t$ controls how much is added to the cell state, and the output gate $\mathbf{o}_t$ controls how much of the current cell state is copied to the hidden state. This architecture mitigates the problem of vanishing gradients, because the cell state is no longer subject to an exponential decrease due to repeated multiplication with weights $w < 1$. Instead, the forget gate $\mathbf{f}_t$ dynamically modulates, how much of the previous cell state is kept. In fact, the original LSTM did not have a forget gate, i. e. $\mathbf{f}_t = \mathbf{1}$. This formulation makes the constant connection between cell states across time, and thereby the non-vanishing of the gradients, more visible. The forget gate was added later by Gers et al. (1999). The capability to explicitly forget states was found to be beneficial to learning, without reintroducing vanishing gradients. See Lipton et al. (2015) for a further discussion of this.

Many variations of the above Eqs. (2.3) and (2.4) exist. The exact form displayed was introduced by Zaremba and Sutskever (2014). Other, similarly designed architectures such as Bidirectional Recurrent Neural Networks (BRNNs) (Schuster and Paliwal, 1997), Neural Turing Machines (NTMs) (Graves et al., 2014), and Gated Recurrent Units (GRUs) (Cho et al., 2014) exist. GRUs simplify the design of the LSTM by discarding the output gate, which may sometimes speed up learning but has adversarial effects on modelling complexity (Chung et al., 2014). GRUs have never reached the popularity of LSTMs.

Only recently has the reign of LSTM-supremacy in natural language processing begun to topple as purely *self-attention*-based architectures, introduced by the Transformer (Vaswani et al., 2017), have been able to surpass previous performance records. However, LSTMs are still frequently used, e. g. as building blocks in larger architectures, where they are often employed to model factorisations of distributions $p(\mathbf{z}) = \prod_j p(z_j \mid \mathbf{z}_{1:j-1})$, where j indexes dimensions, as is the case in this thesis. Section 2.3 below will give some examples of LSTMs as part of probabilistic sequence models.

Data from many domains are inherently stochastic. The RNNs discussed so far are deterministic architectures, and as such, they struggle to model naturally stochastic processes, such as speech (Chung et al., 2015). Embracing this stochasticity in model building leads to more expressive and powerful models. Recently, hybrid architectures combining probabilistic modelling and the flexibility of RNNs have been proposed. These approaches are enabled by recent advances in Variational Inference (VI), which we will introduce now, before returning to our discussion of sequence models in Section 2.3.

2.2 Variational Inference and Variational Autoencoders

This introduction to Variational Inference (VI) will follow the textbooks of Bishop (2006) and Barber (2012) for a classical perspective of VI. The review papers of Zhang et al. (2018) and Blei et al. (2017) supplement this view and additionally shed light on more recent developments. Named after Euler's 18th-century calculus of variations, an early application of VI is given by Mézard et al. (1987) in the theoretical statistical physics community, where it was used to describe the global behaviour of a lattice of locally interacting atomic spins. VI has since enjoyed an important standing in the Bayesian modelling community to allow for posterior inference in complex models.

Given vector-valued observations $\mathbf{x}$ and latent variables $\mathbf{z}$, it is often feasible to specify a conditional likelihood $p(\mathbf{x} \mid \mathbf{z})$, generating the data, as well as a prior over the latent variables $p(\mathbf{z})$. A recurrent object of desire in Bayesian modelling is the distribution over the posterior of the latents

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{\int p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})\, \mathrm{d}\mathbf{z}}\,, \tag{2.5}$$

which, using Bayes' rule and assuming continuous latent variables $\mathbf{z}$, may be written in terms of the conditional likelihood and prior only. In Eq. (2.5), the computation of the evidence, or marginal likelihood, $p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$ is exponential in the number of latent states and its evaluation quickly grows intractable if no closed-form solution to the integral exists (Blei et al., 2017). Even in the simple setting of learning conditional probability tables in belief networks, latent (or missing) variables introduce dependencies which complicate inference (Barber, 2012, p. 248). Variational inference is a popular remedy for this intractability. Instead of the unwieldy true posterior $p(\mathbf{z} \mid \mathbf{x})$, a variational distribution $q(\mathbf{z} \mid \mathbf{x})$ from a less complex family of distributions is introduced. Then, a divergence between the true posterior and the variational approximation is minimised. Such approaches lead to a variety of inference algorithms, which turn the problem of intractable integration into one of optimisation. Alternatively, approaches such as Markov Chain Monte Carlo (MCMC) or Gibbs sampling are a popular way to approximate the integral with samples. Unlike VI, sampling-based approaches usually offer convergence guarantees with respect to the true posterior. However, many proclaim VI to be faster and more easily scalable to larger data sets (Blei et al., 2017).

The Kullback-Leibler Divergence $\mathrm{KL}(q \parallel p)$ between two distributions $q(\mathbf{z})$ and $p(\mathbf{z})$ is defined as

$$\mathrm{KL}(q \parallel p) = \mathbb{E}_{q(\mathbf{z})}\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z})}\right] = \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})}\, \mathrm{d}\mathbf{z}\,, \tag{2.6}$$

where the last equality holds only for $q(\mathbf{z})$ and $p(\mathbf{z})$ defined over continuous spaces. The KL divergence is non-negative, and it is zero if and only if $q(\mathbf{z}) = p(\mathbf{z})\, \forall \mathbf{z}$. It is however not symmetric $\mathrm{KL}(q \parallel p) \neq \mathrm{KL}(p \parallel q)$, which is one of the reasons it is a *divergence* rather than a *metric*. In variational inference, we minimise the KL

divergence between the variational distribution and the true posterior[1]

$$\begin{aligned}
&\mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})) && (2.7)\\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{q(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z} \mid \mathbf{x})}\right] && (2.8)\\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z} \mid \mathbf{x})\right] - \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}, \mathbf{z})\right] + \log p(\mathbf{x})\,, && (2.9)
\end{aligned}$$

where we have used $p(\mathbf{x})$'s independence of $\mathbf{z}$ for the last term. We want to minimise the LHS of the equation but cannot evaluate it directly due to the intractabilities in $p(\mathbf{z} \mid \mathbf{x})$. Likewise, the RHS of the equation contains $p(\mathbf{x})$, which, lest we forget, is the reason for this mathematical endeavour in the first place. However, because $p(\mathbf{x})$ is irrelevant to our optimisation in $\mathbf{z}$, we may omit it and instead maximise

$$\mathrm{ELBO(q)} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z} \mid \mathbf{x})\right]\,, \tag{2.10}$$

where we introduce the Evidence Lower Bound (ELBO). The negative ELBO is equivalent to the KL divergence we seek to minimise up to a constant. Maximising the ELBO therefore corresponds to minimising an upper bound to the above KL divergence.

A different but equally popular derivation for this lower bound is given by maximum likelihood learning in models with unobserved latent variables: We seek to maximise the marginal likelihood, or evidence, $p(\mathbf{x})$ of the data under our model. This requires marginalisation over the latent variables $\mathbf{z}$. Again, one introduces the variational distribution $q(\mathbf{z} \mid \mathbf{x})$ to modify the intractable integral

$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z})\,\mathrm{d}\mathbf{z} && (2.11)\\
&= \log \int q(\mathbf{z} \mid \mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\,\mathrm{d}\mathbf{z} && (2.12)\\
&= \log \left(\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right]\right) && (2.13)\\
&\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right] && (2.14)\\
&= \mathrm{ELBO}(q)\,, && (2.15)
\end{aligned}$$

where Eq. (2.14) results from Jensen's inequality (Jensen et al., 1906). We see that both the minimisation of the KL divergence between the true posterior and the variational approximation, as well as the maximisation of the marginal likelihood, may be used as motivation to derive the ELBO, Eq. (2.10).

By rewriting Eq. (2.9)

$$\log p(\mathbf{x}) = \mathrm{ELBO}(q) + \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x}))\,, \tag{2.16}$$

we see that the lower bound is tight if the KL divergence is 0, and therefore $q = p$. In practice, this may only occur if the variational approximation $q(\mathbf{z} \mid \mathbf{x})$ is expressive enough to model the true posterior $p(\mathbf{z} \mid \mathbf{x})$. This will often not be the case. For example, the popular mean field approach specifies $q(\mathbf{z} \mid \mathbf{x}) = \prod_j q_j(z_j \mid \mathbf{x})$, where j

[1] The asymmetry of the KL divergence results in various subtleties not discussed here but e. g. in Barber (2012, Sec. 28.3).

indexes the latent space dimensions. This leads to tractable inference but cannot account for dependencies between the latent dimensions. Mean-field Variational Inference (MFVI) leads to iterative optimisation algorithms, which require iteration over the complete data set and therefore do not scale well to large data settings (Zhang et al., 2018). Stochastic Variational Inference (SVI) (Hoffman et al., 2013) eliminates this constraint by applying stochastic optimisation to variational inference, essentially yielding a mini-batch compatible VI variant. However, in all of the above, a variational distribution has to be computed *for each input datum* $\mathbf{x}$, either by evaluating expectations with respect to $q(\mathbf{z} \mid \mathbf{x})$, if they are tractable, or performing optimisations, if they are not. Therefore, scaling MFVI to large data and complex architectures is not without problems, i. e. see discussions in Zhang et al. (2018), Farquhar et al. (2020), or Wu et al. (2019a).

Amortised variational inference presents a promising proposal to solve these issues. Instead of optimising the variational distribution for each data point separately, in amortised VI, a powerful model $f_{\boldsymbol{\phi}}(\mathbf{x})$ *directly predicts* the variational distribution. If $q(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\lambda})$ is parametrised by $\boldsymbol{\lambda}$, e. g. $\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ for a mean-field Gaussian, evaluation of $f_{\boldsymbol{\phi}}$ directly yields $f_{\boldsymbol{\phi}}(\mathbf{x}) = \boldsymbol{\lambda}$. Here, $\boldsymbol{\phi}$ are the parameters of $f_{\boldsymbol{\phi}}$ to be optimised. As the parameters $\boldsymbol{\phi}$ are shared over all data instances, variational inference is said to be *amortised*. Amortised VI hinges on the assumption that $f_{\boldsymbol{\phi}}$ is flexible enough to predict the correct distribution for all $\mathbf{x}$ reliably. Neural networks are a popular choice for such approximators, and amortised VI has quickly become the default approach to combine neural networks and probabilistic frameworks (Zhang et al., 2018). The parameters $\boldsymbol{\phi}$ then correspond to the weights and biases of the network $f_{\boldsymbol{\phi}}$.

The Variational Autoencoder. Amortised VI was first proposed by Dayan et al. (1995), see Marino et al. (2019), while the term was coined in Gershman and Goodman (2014). However, its use in the Variational Autoencoder (VAE), attributed to both Kingma and Welling (2014) and Rezende et al. (2014), popularised it. Traditionally, VI is used to allow for inference in complex probabilistic models, which specify intricate dependencies between variables. The VAE formulates a simple probabilistic model and pushes complexities into encoding and decoding networks. See Doersch (2016) for a welcoming and easy-to-digest tutorial on VAEs and Tschannen et al. (2018) for a more recent review paper focussed on unsupervised representation learning. VAEs are a probabilistic architecture relying on two neural networks, a probabilistic *encoder* $q(\mathbf{z} \mid \mathbf{x})$ and a probabilistic *decoder* $p(\mathbf{x} \mid \mathbf{z})$, where $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{z} \in \mathbb{R}^L$, and usually $D \gg L$. Usually, a normally distributed prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$ on the latent space is assumed, and both the encoder and the decoder parametrise the means $\boldsymbol{\mu}$ and variances $\boldsymbol{\sigma}^2$ of factorised Gaussian distributions,

$$q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{z};\, \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x})\right) \quad \text{and} \tag{2.17}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mathbf{x};\, \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\mathbf{z})\right) , \tag{2.18}$$

where $\boldsymbol{\phi}$ (and $\boldsymbol{\theta}$) are the parameters of the encoding (and decoding) networks.[2] VAEs may be used for unsupervised representation learning as well as generative

[2] We often omit the parameters $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$ for the sake of readability. Neural networks have been used to parametrise conditional distributions well before the introduction of VAEs. An early example of this are Mixture Density Networks (MDNs) (Bishop, 1994), which already share a lot of the philosophy of amortised inference, i. e. for each incoming data point, a neural network directly infers the parameters of a probabilistic model. However, MDNs predict deterministic estimates of the latent parameters.

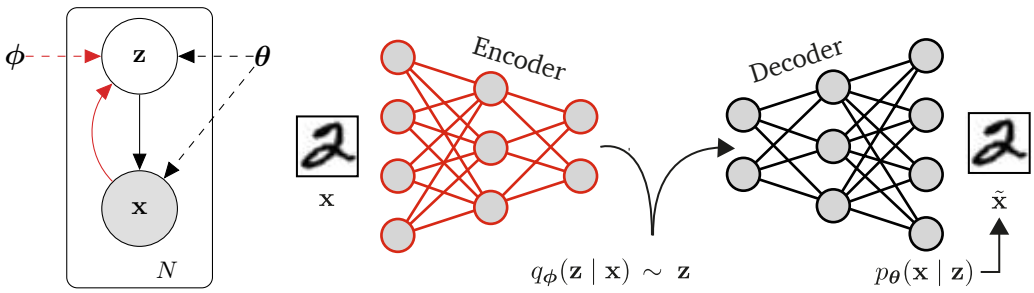


Figure 2.2.: (a) Graphical model of a Variational Autoencoder (VAE). Black edges indicate the generative model $p(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$ and red ones the inference model $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$. Solid lines indicate dependency structures between random variables, dashed lines the dependencies on parameters. Shaded nodes indicate observed variables, transparent nodes latent variables. Plate notation is used to indicate amortisation, i. e. sharing of $\boldsymbol{\phi}$ over all data instances N. For the sake of simplicity, we will not display plate notation or dependencies on parameters in future graphical models in this thesis. Figure adapted from Kingma and Welling (2014). (b) Illustration of a typical VAE architecture and computational flow. Given an input image $\mathbf{x}$, the encoder infers the latent distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$. From this, we sample a latent state $\mathbf{z}$, which the decoder uses to predict $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$. Next, a reconstruction $\tilde{\mathbf{x}}$ of the input image may be obtained by sampling. In fully generative mode, samples from the prior $\mathbf{z} \sim p(\mathbf{z})$ are used as input to the decoder. Figure inspired by Tschannen et al. (2018).

modelling of complex data.

In the setting of image modelling, a VAE may be employed to (a) produce a low-dimensional encoding $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$ given an image $\mathbf{x}$, and (b) to generate an image by sampling $\mathbf{z} \sim p(\mathbf{z})$ from the prior and decoding it to an image $\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{z})$. Depending on the scenario, one or both of these applications may be desirable. Figure 2.2 displays the graphical model of to the VAE as well as an additional illustration of a typical architecture. VAE training proceeds by stochastic optimisation of the ELBO, Eq. (2.10), for the parameters of the encoding and decoding networks jointly. In the context of VAEs, the ELBO is often rewritten as

$$\mathrm{ELBO}(q) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x} \mid \mathbf{z})\right] - \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))\,, \tag{2.19}$$

which highlights the VAE components more clearly. As the expectations with respect to $q(\mathbf{z} \mid \mathbf{x})$ are not tractable, they are usually approximated with a single sample.[3] The first term $\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x} \mid \mathbf{z})\right]$ can be interpreted as a reconstruction loss of the current image $\mathbf{x}$ given the latent code $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$ and resulting decoding $p(\mathbf{x} \mid \mathbf{z})$. To see this more clearly, let $\mathbf{x}$ be an image and $p(\mathbf{x} \mid \mathbf{z})$ a pixel-wise independent

[3] Approaches such as the IWAE (Burda et al., 2015) or AESMC (Le et al., 2018) use multiple samples $\mathbf{z}$ to approximate the expectations with respect to q. If q is flexible enough, this results in tighter bounds with respect to the true posterior. However, the interplay between the encoding and decoding networks is complex and "tighter variational bounds are not necessarily better" (Rainforth et al., 2018). See Rainforth et al. (2018) for an excellent discussion of why more samples may hinder learning of the inference network.

Gaussian distribution with fixed variances. Then

$$\log p(\mathbf{x} \mid \mathbf{z}) = \sum_j \frac{(\mu_{\boldsymbol{\theta}}(z_j) - x_j)^2}{2\,\sigma^2} + C\,, \tag{2.20}$$

which is the common squared loss between predicted and observed pixel values. Maximising the ELBO minimises $\mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))$, which penalises deviations between $q(\mathbf{z} \mid \mathbf{x})$ and $p(\mathbf{z})$. If the KL divergence is minimal and therefore 0, then $q(\mathbf{z} \mid \mathbf{x}) = p(\mathbf{z})$ and the encodings are pure noise and cannot contain any information. This is, of course, detrimental to achieving good reconstructions from $\mathbf{z}$. On the other hand, without the KL-penalty, the network will not learn encodings which resemble the prior distribution $p(\mathbf{z})$, and therefore makes sampling from $p(\mathbf{z})$ to obtain images $p(\mathbf{x} \mid \mathbf{z})$ impossible. Intuitively the ELBO can be seen as representing a trade-off between good reconstructions and conforming to the prior distribution of the latent space.[4]

Optimisation of VAEs usually proceeds via (mini-batch) gradient descent, i. e. a random subset of the data $\{\mathbf{x}_i\}_{i=1}^{N_b}$ is propagated through the VAE and the parameters $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ of the encoding and decoding distributions $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ are updated using the cumulative gradient $N_b^{-1} \sum_i \nabla_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathrm{ELBO}_i$, e. g. with the ADAM optimiser (Kingma and Ba, 2015).

This then finally leads us to the computation of

$$\nabla_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathrm{ELBO} = \nabla_{\boldsymbol{\phi},\boldsymbol{\theta}}\, E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] - \nabla_{\boldsymbol{\phi},\boldsymbol{\theta}}\, \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))\,. \tag{2.21}$$

For the encoding distribution, the gradient with respect to $\boldsymbol{\phi}$ is problematic because

$$\begin{aligned} \nabla_{\boldsymbol{\phi}}\, \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[f_{\boldsymbol{\phi}}(\mathbf{z})\right] &= \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{z}) f_{\boldsymbol{\phi}}(\mathbf{z})\, \mathrm{d}\mathbf{z} && (2.22)\\ &\neq \int q_{\boldsymbol{\phi}}(\mathbf{z}) \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{z})\, \mathrm{d}\mathbf{z} && (2.23)\\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[\nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{z})\right]\,. && (2.24) \end{aligned}$$

This means that, in the current form, we may not sample $\mathbf{z} \sim q$, evaluate the ELBO, and calculate its gradient, because this would correspond to a calculation of $\mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{z})\right]$. In other words, generally, the gradient of an expectation is not the same as the expectation of a gradient. One way of solving this, is writing

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[f_{\boldsymbol{\phi}}(\mathbf{z})\right] &= \int \left((\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}) \frac{q_{\boldsymbol{\phi}}}{q_{\boldsymbol{\phi}}} f_{\boldsymbol{\phi}} + q_{\boldsymbol{\phi}} (\nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}) \right) \mathrm{d}\mathbf{z} && (2.25)\\ &= \int q_{\boldsymbol{\phi}} \left(f_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}} + \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}} \right) \mathrm{d}\mathbf{z} && (2.26)\\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[f_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}} + \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}} \right]\,, && (2.27) \end{aligned}$$

[4] Again, many subtleties beyond the scope of this thesis are involved here. Many works on the effects and side-effects of the KL-divergence and reconstruction loss exist, and in turn, many models have been proposed to overcome their limitations. This discussion is often tightly linked with the aim to construct VAEs which learn *disentangled* representations. A *disentangled* representation in a VAE is loosely defined as a representation for which the individual dimensions of the latent space correspond to the *true* generative factors of the data. The interested reader may consult Higgins et al. (2017), Burgess et al. (2017), Mathieu et al. (2019), or Zhao et al. (2019) for a further discussion of these topics.

which does however have a high variance (Krishnan et al., 2015).

This is where the *reparametrisation trick* (Kingma and Welling, 2014) comes into play. For some distributions, we may *reparametrise* $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$, such that the inequality in Eqs. (2.22) to (2.24) becomes an equality. For example, if $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x}))$, we may write $\mathbf{z}$ as a deterministic function, where noise is injected via the auxiliary variable $\boldsymbol{\epsilon}$,

$$\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}(\mathbf{x}) = g(\mathbf{x}, \boldsymbol{\epsilon}) \,, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) = p(\boldsymbol{\epsilon}) \,. \tag{2.28}$$

Note that for vector-valued $\mathbf{z}$ and $\boldsymbol{\epsilon}$, multiplication is element-wise. This allows us to rewrite the expectation Eq. (2.22) as

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[f_{\boldsymbol{\phi}}(\mathbf{z}) \right] &= \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{z}) f_{\boldsymbol{\phi}}(\mathbf{z}) \, \mathrm{d}\mathbf{z} && (2.29) \\
&= \nabla_{\boldsymbol{\phi}} \int p(\boldsymbol{\epsilon}) f_{\boldsymbol{\phi}}(\mathbf{z}) \, \mathrm{d}\boldsymbol{\epsilon}, \text{ using } q(\mathbf{z}) \, \mathrm{d}\mathbf{z} = p(\boldsymbol{\epsilon}) \, \mathrm{d}\boldsymbol{\epsilon} \,, && (2.30) \\
&= \int p(\boldsymbol{\epsilon}) \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{z}) \, \mathrm{d}\boldsymbol{\epsilon} && (2.31) \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\phi}}(\mathbf{z}) \right] \,. && (2.32)
\end{aligned}$$

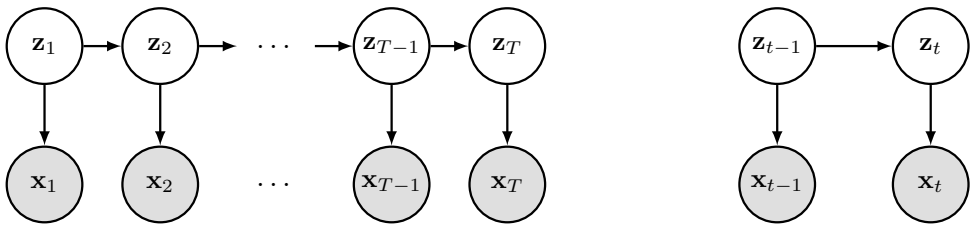
This is the Stochastic Gradient Variational Bayes (SGVB) estimator (Kingma and Welling, 2014). We may therefore now sample $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, compute the ELBO, and evaluate its gradient, obtaining a correct estimate Eq. (2.21).[5] See Kingma and Welling (2014) for a discussion of other distributions for which this is possible. Note that this problem does not arise for the gradient with respect to $\boldsymbol{\theta}$ for the decoding network because q does not depend on $\boldsymbol{\theta}$ and the above inequality Eq. (2.22) becomes an equality. For the calculation of the KL divergence in Eq. (2.19), this problem may be circumvented entirely if closed-form solutions of the KL divergence exist, e. g. if both p and q are Gaussian. Such closed-form solutions additionally lead to lower variance estimates of the gradient, which may benefit training, see e. g. Burda et al. (2015).

2.3 State Space Models

2.3.1 Classical State Space Models

Before the rise of LSTMs, Probabilistic Graphical Models (PGMs) were the dominant approach for modelling sequences. The presentation below largely follows the works of Barber (2012), Bishop (2006), and Murphy (2012). Given a sequence of observations $\mathbf{x}_{1:T} = \{\mathbf{x}_t\}_{t=1}^T$ and latent states $\mathbf{z}_{1:T} = \{\mathbf{z}_t\}_{t=1}^T$, we formulate a joint

[5] The reparametrisation trick was explained in detail because of the apparent confusion surrounding it. Often, the question of "Why do we need the reparametrisation trick?" is answered with "Because TensorFlow, PyTorch, etc. cannot backpropagate through stochastic variables. The reparametrisation trick makes the latent variables deterministic and automatic backpropagation will then work fine." This answer is incomplete, as it reduces the reparametrisation trick to an implementation detail, when, in fact, it is a neat and necessary mathematical trick. Without it, we would either (a) naïvely and wrongly evaluate Eq. (2.24), which is not the desired quantity, or (b) need to resort to the high-variance estimator Eq. (2.27).



(a) (b)

Figure 2.3.: Graphical model of the Hidden Markov Model (HMM) and Linear Gaussian State-Space Model (LG-SSM) in (a) full and (b) compact representation. The dynamics of the latent state $\mathbf{z}_t$ are given by the *transition distribution* $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$. Observations $\mathbf{x}_t$ may be generated from latent states via the *emission distribution* $p(\mathbf{x}_t \mid \mathbf{z}_t)$. The latent state $\mathbf{z}_t$ fully explains $\mathbf{x}_t$, such that $\mathbf{x}_t$ is independent of all other variables given $\mathbf{z}_t$. For the HMM, the latent states are discrete, giving a transition matrix, while the emission distribution may be specified with respect to discrete or continuous variables. For the LG-SSM, both transition and emission distributions are Gaussian. From now on, we will only draw graphical models in their compact representation.

probability distribution over both as

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p(\mathbf{x}_1 \mid \mathbf{z}_1)\, p(\mathbf{z}_1) \prod_{t=2}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t)\, p(\mathbf{z}_t \mid \mathbf{z}_{t-1})\,. \tag{2.33}$$

This factorisation encodes a variety of conditional (in-)dependence assumptions between variables, which can be displayed as a *graphical model*, see Fig. 2.3. Namely, at time t and given $\mathbf{z}_t$, the observation $\mathbf{x}_t$ is independent of all other observations $\mathbf{x}_{1:t-1,t+1:T}$ and latent states $\mathbf{z}_{1:t-1,t+1:T}$, i. e. $p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1,t+1:T}, \mathbf{z}_{1:T}) = p(\mathbf{x}_t \mid \mathbf{z}_t)$. Likewise, $\mathbf{z}_{t-1}$ and $\mathbf{z}_{t+1}$ are conditionally independent given $\mathbf{z}_t$. In other words, the latent state $\mathbf{z}_t$ *fully* explains the current observation $\mathbf{x}_t$. The model is 1st order Markov in the latent states, i. e. the latent state $\mathbf{z}_t$ suffices to predict the distribution over future latent states and observations. Previous latent states $\mathbf{z}_{t'<t}$ provide no additional information. However, without knowledge of the latents, the observations $\mathbf{x}_t$ on their own do *not* exhibit such independence structure, i. e. $p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})$ may not be simplified any further. We can sample a sequence of latent states from this model if the *transition distribution* $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ is known. For each $\mathbf{z}_t$, we may then sample an observation $\mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{z}_t)$ from the *emission distribution*. The prior $p(\mathbf{z}_1)$ encodes assumptions about the distribution of states in absence of previous observations.

For discrete latent states, the above model is known as a Hidden Markov Model (HMM),[6] which has been successful in many sequence modelling applications such as object tracking, bioinformatics, or speech recognition (Barber, 2012). In speech recognition, the observations correspond to a representation of a snippet of audio and the latent states represent the phonemes needed to transcribe the audio. In HMMs, the transition distribution is given by a matrix $p(\mathbf{z}_t = i \mid \mathbf{z}_{t-1} = j) = A_t(i, j)$. If this matrix is independent of time, the sequence is called *stationary*. Note that

[6] Sometimes, the term Hidden Markov Model is used to describe all sequence models with Markovian latent states. However, most often, such as in the textbooks of Barber (2012), Bishop (2006), and Murphy (2012), the term is used solely for models with *discrete* latent states. This is the convention that we will conform to.

HMMs are capable of modelling long-range dependencies because no independence assumptions on the observations exist, i. e. the predictive distribution $p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})$ depends on all previous observations. This dependency is mediated by the latent states. Without the inclusion of latent states, long-term dependency modelling could only be achieved by adding many individual dependencies – additional backwards arrows $\{\mathbf{x}_t \to \mathbf{x}_{t-b}\}_{b=2}^{B}$ in the graphical model – which would lead to a combinatorial explosion in the conditional distribution $p(\mathbf{x}_t \mid \mathbf{x}_{t-1:t-B})$ (Bishop, 2006, p. 608).

For continuous latent states, the above factorisation is called a Linear Gaussian State-Space Model (LG-SSM), also called Linear Dynamical System (LDS). It corresponds to the same graphical model, Fig. 2.3, only this time, continuous Gaussian latent states are assumed and linear equations govern the transition and emission distributions, such that, following Murphy (2012, Chap. 18),

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{Q}_t) \tag{2.34}$$

$$\mathbf{x}_t = \mathbf{B}_t \mathbf{z}_t \quad + \delta_t, \quad \delta_t \sim \mathcal{N}(0, \mathbf{R}_t)\,. \tag{2.35}$$

The matrices $\mathbf{A}_t$ and $\mathbf{B}_t$ give the transition and emission behaviour, and noise representing the uncertainty of observations δ_t and noise intrinsic to the process $\boldsymbol{\epsilon}_t$ is included (Durstewitz, 2017, p. 152f.). It follows from Eqs. (2.34) and (2.35) that $p(\mathbf{x}_t \mid \mathbf{z}_t)$ and $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ are also Gaussian. Often, the LG-SSM is formulated in additional dependency on an external control parameter or action $\mathbf{a}_t$. One of the LG-SSM's main application is the estimation of the state $\mathbf{z}_t$ given a sequence of uncertain measurements $\mathbf{x}_{1:t}$, i. e. the position of a rocket given only noisy estimates (Barber, 2012). Estimation of $p(\mathbf{z}_t \mid \mathbf{x}_{1:t})$ is referred to as filtering. Given the parameters of the system $\gamma_t = \{\mathbf{A}_t, \mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t\}$, filtering in LG-SSMs is tractable and leads to the *Kalman filter equations*. The LG-SSM is colloquially referred to as the *Kalman filter*, as *filtering* is its predominant use. LG-SSMs are also used heavily in traditional forecasting, even though they might be called by a different name. For example, the popular Autoregressive-Moving Average (ARMA) models used in time series analysis can be represented as a special case of the LG-SSM (Murphy, 2012, p. 640).

Table 2.1 lists classic inference tasks for both HMMs and LG-SSMs. Given the parameters of the model, all tasks are tractable for both models, assuming the size of the HMM's state space is reasonable. This means that, if the model assumptions hold, the HMM and LG-SSM perform these inference tasks optimally. Often for LG-SSMs, a physical model describing the measurement and transition process exists, and all parameters in Eqs. (2.34) and (2.35) are known. However, parameter learning of all or some parameters in the above equations may be performed. Learning in both models is trivial if $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ are observed during training. For example, when training an HMM for speech recognition, one may create a training set that contains both the audio recordings (observations) and the corresponding correct phonemes (latents). If only $\mathbf{x}_{1:T}$ is observed, algorithms such as expectation maximisation may be used to estimate stationary parameters.

Both HMMs and LG-SSMs present a classical alternative to modern RNNs such as the LSTM. However, HMMs do not scale well to larger state spaces, as the transition matrix and inference is quadratic in the size of the latent space (Lipton et al., 2015). Furthermore, LG-SSMs are unsuited for applications where the assumption of linear transitions does not hold. Extensions of LG-SSMs to non-linear transition and emission functions exist (Wan and Van Der Merwe, 2000). However, they

Table 2.1.: Relevant inference tasks in probabilistic sequence modelling. Exact estimation of the below distributions is tractable in the Hidden Markov Model (HMM) and Linear Gaussian State-Space Model (LG-SSM). Adapted from Barber (2012).

Filtering	$p(\mathbf{z}_t \mid \mathbf{x}_{1:t})$	
Prediction	$p(\mathbf{z}_t \mid \mathbf{x}_{1:t'})$	$t' < t$
Smoothing	$p(\mathbf{z}_t \mid \mathbf{x}_{1:t'})$	$t' > t$
Likelihood Estimation	$p(\mathbf{x}_{1:T})$	
Most Probable Latent Sequence	$\operatorname{argmax}_{\mathbf{z}_{1:T}} p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$	

rely on sampling and assume that a linearisation of the functions up to first or second order is sufficient. Additionally, they either demand emission and transition matrices to be known upfront or require a tractable posterior to allow for learning (Karl et al., 2017a). LSTMs on the other hand, allow for learning highly non-linear transition functions and can model long-term dependencies between latent states. The relationship between RNNs and state space models have been discussed as early as in DeCruyenaere and Hafez (1992). Until recently, it mostly seemed like the two approaches were opposing. In the following section, we will give an introduction to recent architectures that defy this assumption.

2.3.2 Neural State Space Models

We concluded the last section by remarking that LG-SSMs and HMMs may be unsuited to model complex processes in large data settings. However, LSTMs often *can* accommodate such scenarios. Why would we want to combine recurrent neural networks and state space models? As discussed at the end of Section 2.1, RNNs are not well suited to modelling processes where stochasticity is inherent and dominant. While we may interpret the *output* of an RNN as the parametrisation of a probability distribution, the hidden state *transitions* in the RNN remain entirely deterministic. Chung et al. (2015) note that the introduction of stochasticity helps the LSTM to model "the kind of variability observed in highly structured sequential data". Adding to that, LSTMs are not well suited to the unconditional generation of novel sequences. Principled approaches to generate novel sequences require a prior distribution, from which to draw samples, which standard LSTMs lack. The above statements are supported by the fact that the works introduced below routinely outperform deterministic recurrent approaches or classical state space models.

Over the last years, a cornucopia of approaches marrying state space models and recurrent neural networks have been proposed. This has lead to sequence models capable of dealing with non-linear, stochastic, and high-dimensional data such as audio or video in a fully probabilistic way. The topic of this thesis is object-aware modelling of physical video *sequences*. By specialising general-purpose sequence models to be better suited to physical video modelling, e. g. by including object-based image models and physical prediction modules, approaches such as ours, STOVE, may be obtained. The non-linear sequence models presented below therefore correspond to important baselines, but may also yield inspiration for further developments in object-aware modelling of physical video sequences. Examples of such models that we are aware of include Bayer and Osendorfer (2014), Fabius and van Amersfoort (2015), Chung et al. (2015), Watter et al. (2015), Krishnan et al. (2015), Johnson et al. (2016), Karl et al. (2017a), Krishnan et al. (2017), Maddison et al. (2017), and Rangapuram et al. (2018). They enjoy the advantage of using neural networks

to model highly non-linear functions, while simultaneously benefitting from the stochasticity, structure, interpretability, data-efficiency, and the explicit inclusion of structure in state space models (Rangapuram et al., 2018).

Such models can be obtained by modifying the emission and transition behaviour of the LG-SSM in numerous ways. Regarding the emission function, it is common to replace the linear model with (a) a complex non-linear likelihood model, e. g. a neural network. Additionally, most approaches implement one of the following tactics to allow for more flexible transition models: (b) model the latent transition with a neural network, (c) keep the transition dynamics linear but predict their parameters with a (recurrent) neural network, or (d) introduce a separate deterministic RNN which interfaces with stochastic latent states. The approaches also differ by how many of the strict independence assumptions of the LG-SSM they retain. Many of the above approaches leverage amortised variational inference, see Section 2.2, to achieve tractable learning in these highly non-linear hybrid approaches between state space models and neural networks. VI is usually applied for models which implement strategies (a), (b), or (d), because the resulting non-linear transition models make classical inference intractable. For future reference, STOVE, our approach, falls into the categories (a) and (b), where (a) is achieved by using Sum-Product Attend-Infer-Repeat (SuPAIR), Section 3.1.2, and (b) is realised with a Graph Neural Network (GNN), Section 3.2.2. The above approaches may be framed in terms of a variety of applications, such as time series prediction, modelling of speech, handwriting, or videos. While this may have informed model design, any approach is relevant for this section as long as it is a general approach capable of modelling high-dimensional data with complex dependencies. Here, *modelling* refers to the specification of a fully generative model as well as the capability to perform inference therein.

A useful strategy for understanding neural state space models is to seek answers to the following questions.

(i) How are images $\mathbf{x}_t$ generated?
(ii) How are future latent states $\mathbf{z}_{t+1}$ predicted?
(iii) How are present latent state $\mathbf{z}_t$ inferred?

Questions (i) and (ii) are directed at the form of the emission and transition distributions, the generative model, Eq. (2.33) for the LG-SSM. As posterior inference of latent variables $\mathbf{z}_t$ is intractable in non-linear state space models, some form of amortised approximation is usually employed, the exact form of which is asked by question (iii).

Due to different ways of interlacing the RNNs and SSMs, the cited approaches implement observation likelihoods, transition functions, or inference networks in different ways. Note that the above approaches may have been designed with different applications in mind. While one might focus on accurate inference of latent states, e. g. by designing a neural smoother, another might focus on obtaining a good generative model.

VRNNs. While all of the above approaches have reached some level of popularity in the community, the Variational Recurrent Neural Network (VRNN) (Chung et al., 2015) is most common. In our evaluation in Chapter 5, as well as in other papers in the literature, it is the chosen representative of general-purpose non-linear and

stochastic sequence models. Let us quickly introduce the defining characteristics of the VRNN. The VRNN employs the strategies (a) and (d) mentioned above. It introduces an LSTM whose hidden state $\mathbf{h}_t = f_1(\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{z}_t)$ depends on both the current observation $\mathbf{x}_t$, the current latent state $\mathbf{z}_t$, and the previous hidden state $\mathbf{h}_t$. At time t, the LSTM therefore expresses dependencies on *all* previous images $\mathbf{x}_{1:t-1}$ and latents $\mathbf{z}_{1:t-1}$ through its hidden state. This hidden state is used as input to Multilayer Perceptrons (MLPs) to predict the means and variances of the relevant Gaussian conditional distributions. In all subsequent approaches and unless otherwise mentioned, distributions are likewise to be assumed factorised Gaussians, whose means and variances are predicted by neural networks. To answer questions (i) through (iii), let us assume that we have inferred, observed, or calculated all previous variables $\mathbf{z}_{t-1}$, $\mathbf{x}_{t-1}$, and $\mathbf{h}_{t-1}$. To ease notation, we will also assume that the VRNN and all subsequent approaches correctly handle prior distributions. Figures 2.4a to 2.4d illustrate the following operations. (i) Following Fig. 2.4a, to generate an image $\mathbf{x}_t$, we require the previous hidden state $\mathbf{h}_{t-1}$, available by assumption, and current latent state $\mathbf{z}_t$, which is available from (ii) or (iii). This gives us $\mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = f_2(\mathbf{h}_{t-1}, \mathbf{z}_t)$, where f_2 and all following f_i are realised as MLPs. (ii) Still looking at Fig. 2.4a, we can sample future latent states as $\mathbf{z}_t \sim p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = f_3(\mathbf{h}_{t-1})$ from the previous hidden state of the RNN. (iii) Following Fig. 2.4b, given the current observation $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_{t-1}$, we may infer $\mathbf{z}_t \sim q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = f_4(\mathbf{x}_t, \mathbf{h}_{t-1})$ without any extra steps. Here, q is the amortised variational approximation of the true posterior. Now that we have obtained both $\mathbf{x}_t$ and $\mathbf{z}_t$, we can update the hidden state and move on to the next timestep, see Fig. 2.4c. As $\mathbf{h}_{t-1}$ can realise a dependence on *all* past states and observations, the recognition model's estimate of $\mathbf{z}_t$ may be seen as a neural and implicit version of the Kalman filtering algorithm. The deterministic transition of hidden states is shared over inference and generation. The VRNN does not implement a way to obtain *smoothed* estimates of $\mathbf{z}_t$. The above procedures and Figs. 2.4a to 2.4d correspond to the generative and inference models

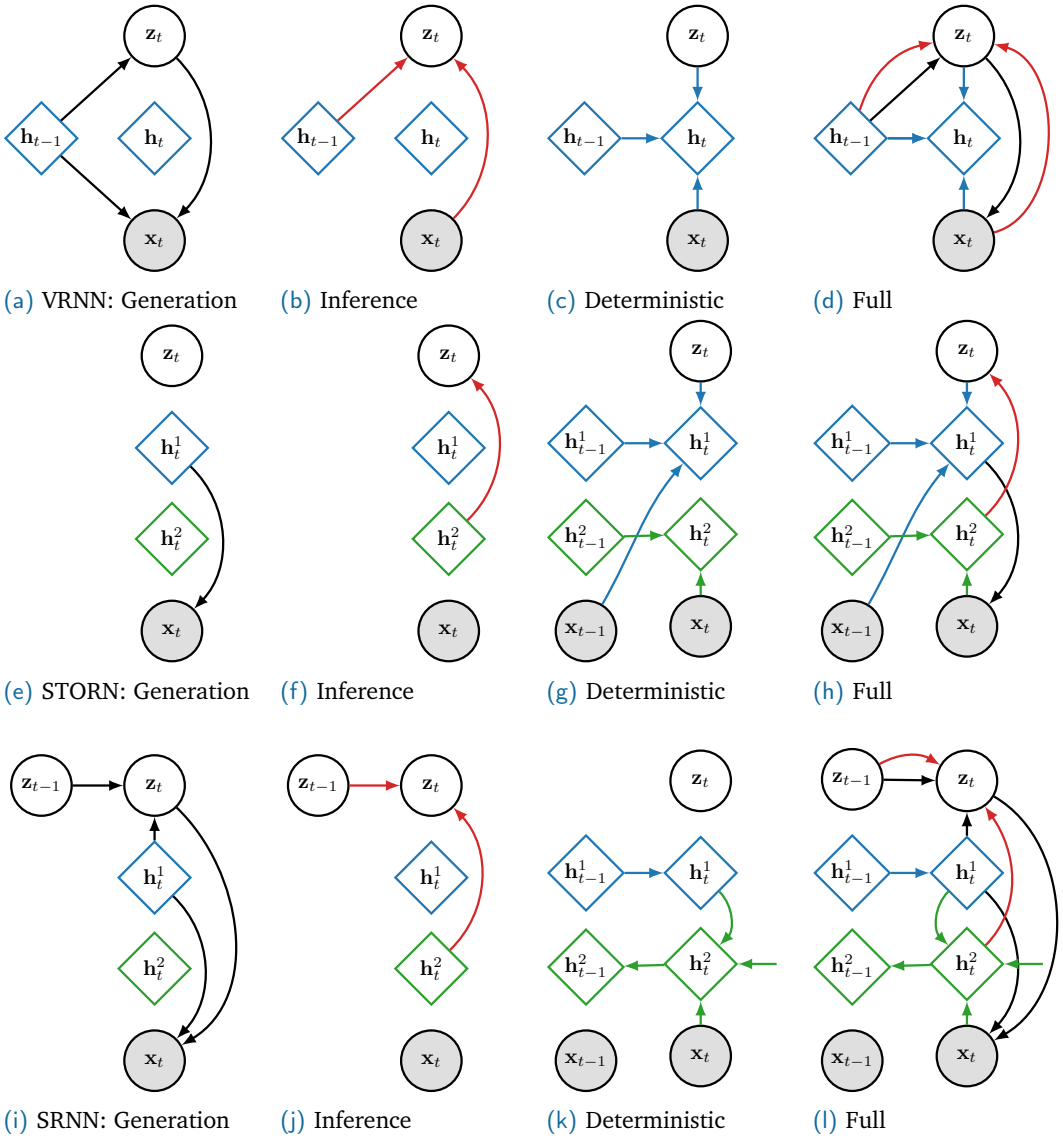
$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_t p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\, p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) \tag{2.36}$$

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1})\,, \tag{2.37}$$

from which the ELBO may be assembled for parameter learning. In contrast to the LG-SSM, the VRNN includes *no* conditional independence assumptions on previous observations or latent states.[7]

STORN. The Stochastic Recurrent Network (STORN) (Bayer and Osendorfer, 2014) is one of the first approaches combining SSMs and LSTMs. It is a predecessor to VRNNs, also uses strategies (a) and (d), and its graphical model is given in Figs. 2.4e to 2.4h. STORN's approach of integrating RNNs and LSTMs is straightforward. STORN includes two RNNs f_1 and f_2. They realise the time-factorisation of the generating and recognition distributions, where the output of the RNN at time t

[7] While not immediately relevant to this thesis, I find it instructive to briefly discuss some other approaches to neural state space models. Sadly, a comprehensive review and comparison of these models does currently not exist. The only thing that comes close is Lesort et al. (2018), which has a heavy focus on reinforcement learning/control and misses many of the works cited herein. The busy reader may skip these last paragraphs. In addition to presenting interesting background knowledge, their primary relevance is for the discussion of STOVE and future work in Section 6.1.



(a) VRNN: Generation (b) Inference (c) Deterministic (d) Full

(e) STORN: Generation (f) Inference (g) Deterministic (h) Full

(i) SRNN: Generation (j) Inference (k) Deterministic (l) Full

Figure 2.4.: Graphical models for (a-d) Variational Recurrent Neural Networks (VRNNs), (e-h) Stochastic Recurrent Networks (STORNs), and (i-l) Stochastic Recurrent Neural Networks (SRNNs). The first column illustrates the generation of novel observations and/or latents. Observed variables are shaded, latents transparent, and deterministic variables are displayed as diamonds. Inference operations are drawn with red arrows. Deterministic nodes refer to hidden states of RNNs, which model dependencies on all of their previous input variables. Emission and transition distribution of stochastic nodes are realised by MLPs. In the original papers, the dependence on external control variables is sometimes considered, which is omitted here. Figures (a-c) adapted from Chung et al. (2015).

directly gives the current conditional distribution

$$p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}) = \prod_t p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \qquad p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = f_1(\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{h}^1_{t-1}) \tag{2.38}$$

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \qquad q(\mathbf{z}_t \mid \mathbf{x}_{1:t2}) = f_2(\mathbf{x}_t, \mathbf{h}^2_{t-1})\,. \tag{2.39}$$

Similar to the procedure for the VRNN, one may check how the questions (i-iii) are answered for STORN. (i) $\mathbf{x}_t$ is generated from Eq. (2.38), requiring the previous images and current latent state, and (iii) $\mathbf{z}_t$ is inferred from Eq. (2.39), requiring only the current observation to update the hidden state. Unlike the VRNN, STORN infers latent state $\mathbf{z}_t$ independent of all previous latent states. (ii) A major drawback of STORN is, that it provides no way to predict future latent states $\mathbf{z}_t$. The updated latent $\mathbf{z}_t$ would also be needed to update $\mathbf{h}^1_t$ to generate a new observation $\mathbf{x}_t$. In the absence of such a mechanism, we sample $\mathbf{z} \sim p(\mathbf{z})$ from the prior at each step of the prediction, which leads to lower quality predictions, as the amount of stochasticity is fixed and is independent of all previous states and observations (Chung et al., 2015).

DMMs. To generate an observation $\mathbf{x}_t$, the generative model of the VRNN, Eq. (2.36), depends on both the stochastic variable $\mathbf{z}_t$ as well as the history $\mathbf{h}_{t-1}$, encompassing all previous observations $\mathbf{x}_{1:t-1}$ and latents $\mathbf{z}_{1:t-1}$. Likewise, the generation of novel latent states $\mathbf{z}_t$ depends on all $\mathbf{x}_{1:t-1}$ and $\mathbf{z}_{1:t-1}$ through the hidden state $\mathbf{h}_{t-1}$. This breaks with Markovian assumptions of the LG-SSM, which the authors of the Deep Markov Model (DMM) (Krishnan et al., 2017) retain.[8] DMMs employ strategies (a) and (b), and their generative model is identical in factorisation to the LG-SSM,

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_t p(\mathbf{x}_t \mid \mathbf{z}_t)\, p(\mathbf{z}_t \mid \mathbf{z}_{t-1})\,, \tag{2.40}$$

where the emission distribution is parametrised by an MLP and the transition distribution is inspired by the Gated Recurrent Unit. Notably, they formulate the inference network as

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_t q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_{1:T})\,, \tag{2.41}$$

using all previous *and future* observations to infer the latent states. This is unlike VRNNs and STORNs, which make the simplifying assumption that $\mathbf{z}_t$ is inferred using only past information.[9] Unlike the VRNN, the DMM relies on *all* observations, including future observations $\mathbf{x}_{t+1:T}$, to infer $\mathbf{z}_t$. Both formulations are valid, however, without any further additions, this *smoothing-like* formulation requires

[8] DMMs were previously published as Deep Kalman Filters (DKFs) by Krishnan et al. (2015). A less fitting name, especially given that the Deep Kalman *Filter* performs smoothing in its default configuration.

[9] In addition to $(z_{t-1}, x_{1:T})$, the authors also consider other dependencies of the inference network on the observations, such as conditioning on $(x_{1:T})$, $(x_{1:t})$, $(z_{t-1}, x_{1:T})$, $(z_{t-1}, x_{1:t})$, or $(z_{t-1}, x_{t:T})$. From the perspective of a Markovian state space model, the previous latent state and all *future* observations should lead to optimal smoothing inference given the independence assumptions. Indeed, they show that the inclusion of past observations is negligible if the last latent state is included. If it is not included, then past observations contribute significantly to the overall performance. Conditioning on all observations is performed using a bi-directional RNN, see e. g. (Salehinejad et al., 2017).

the entire sequence to be available at inference time and complicates the reuse of transition functions as in the VRNN, which shares its LSTM between inference and generation. The authors show strong performance of the DMM inference model compared to STORN and others, but show no comparisons for the generative model. One may assume that their generative performance will deteriorate if the assumed independencies no longer hold. However, the simplified generative model Eq. (2.40) does lead to tractable evaluation of the KL divergence in the ELBO.

SRNNs. Fraccaro et al. (2016) provide yet another variant of stochastic recurrent networks with strategies (a) and (d), which more clearly separates stochastic from deterministic nodes, i. e. separately models transitions for deterministic and stochastic nodes. All distributions are then predicted by MLPs with dependence on both the stochastic as well as the deterministic nodes. They have found this to give better performance and to be easier to train. They name their approach Stochastic Recurrent Neural Network, which should not be confused with Stochastic Recurrent Networks (STORNs), but mercifully choose the more distinct abbreviation SRNN. Like Krishnan et al. (2017), they use information from future observations during inference. We have displayed them in Figs. 2.4i to 2.4l alongside the others for comparison, but will not go into any more detail.

DVBF. Indeed, the reuse of the generative transition in the inference model, as in VRNNs, is paramount according to the authors of Deep Variational Bayes (DVBF) (Karl et al., 2017a), who claim that it leads to favourable gradients during training and therefore improves performance. Like the DMM, they keep the independence assumptions Eqs. (2.40) and (2.41) of the generative model. Additionally, Karl et al. (2017a) claim that many sequence models are stuck in local optima of good reconstruction but bad transition behaviour because optimal reconstructions may be obtained without incorporating knowledge about the transition. This biases the latents towards reconstruction and not transition. The transition function then cannot modify the latents because this would decrease the reconstruction quality, leading to a model biased towards reconstruction. To instead bias the model towards useful transition functions, they posit a special form of the transition function and inference network. Instead of directly inferring latent states, the inference network infers parameters of the transition function, making it deterministic, from the previous latent state and current observation. A Bayesian treatment of the inferred parameters makes it possible to penalise the heavy use of the current observation to predict the parameters of the current latent state, ensuring that the transition function is used as intended. Karl et al. (2017a) observe that their model can recover derivative quantities, such as velocities, in the latent space, while Krishnan et al. (2017) fail to do so. As our model, STOVE, also reuses the generative transition model during inference and strictly enforces latent space assumptions, we will discuss the effect of these assumptions further in Chapter 4.

Quasi-Linear Approaches. Other recent approaches using strategies (c), such as Fraccaro et al. (2017) and Rangapuram et al. (2018), are of special interest because they retain the ability to tractably compute all quantities in Table 2.1. Fraccaro et al. (2017) achieve this by predicting *time-dependent* parameters $\gamma_{1:T}$ of a LG-SSM using (recurrent) neural networks. As the equations of Table 2.1 also apply to non-stationary processes, once the linear transition and emission parameters have been inferred, this model is identical to a time-dependent LG-SSM. Here,

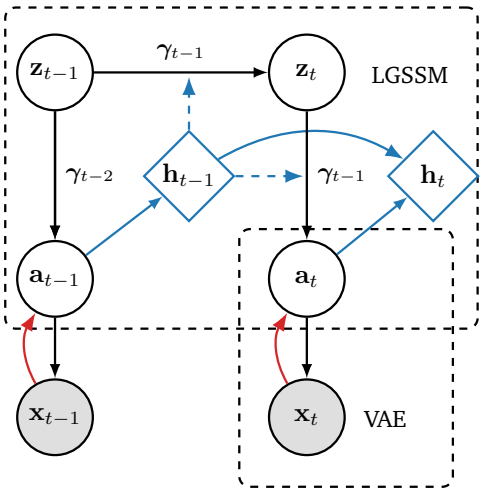


Figure 2.5.: Graphical model of the Kalman Variational Autoencoder (KVAE). The deterministic LSTM predicts the parameters $\gamma_{1:T}$ of a Linear Gaussian State-Space Model (LG-SSM). Given these parameters, the resulting model is a LG-SSM in the latents $\mathbf{z}_{1:T}$ and pseudo-observations $\mathbf{a}_{1:T}$. This allows for tractable calculation of all tasks in Table 2.1. To allow the KVAE to model visually complex observations $\mathbf{x}_t$, the pseudo observations $\mathbf{a}_t$ are used as latents to a time-independent VAE, $\mathbf{a}_t \rightarrow \mathbf{x}_t$. Observed nodes are shaded, deterministic nodes diamonds, and the inference network is shown as red arrows.

Fraccaro et al. (2017) rephrase the linear observation model of the LG-SSM as pseudo-observations $\mathbf{a}_t$. These serve as input to a time-independent VAE, which can then model complex and high-dimensional distributions, such as images. The parameters γ_t, which model the transition of $\mathbf{z}_{t-1}$ to $\mathbf{z}_t$ and emission of $\mathbf{a}_t$ from $\mathbf{z}_t$, are predicted from the previous pseudo observations $\mathbf{a}_{1:t-1}$. Other approaches, such as Watter et al. (2015) and Karl et al. (2017a), also infer locally linear transitions $\gamma_{1:T}$. However, due to the direct dependence of $\gamma_{1:T}$ on the latent states $\mathbf{z}$, they lose the linear dependence of consecutive latent states and cannot employ the exact algorithms (Fraccaro et al., 2017). See Fig. 2.5 for an illustration of their architecture. This approach presents an elegant and disentangled combination of VAEs and LG-SSMs. In contrast to VRNN and STORN, their model does not necessitate the expensive generation of high-dimensional images to generate a sequence of latent states. It can instead restrict itself to lower-dimensional pseudo observations $\mathbf{a}_t$. A drawback of this approach is that, to make the prediction of γ_t tractable, it parametrises them as a mixture of fixed, learned global parameters. Therefore, at each timestep, the LSTM can only predict a mixture of linear actions, which limits the non-linearity this design can achieve. Nevertheless, such approaches provide promising avenues of future work for STOVE, see Section 6.1.

2.4 Sum-Product Networks

Sum-Product Networks (SPNs) are a deep probabilistic graphical model introduced to the world by Poon and Domingos (2011). One may view them as a hierarchical generalisation of mixture models. As such, they inherit the property of universal approximation from mixture models (Peharz et al., 2019). SPNs are an instance of

tractable probabilistic models. In contrast to most other graphical models, they allow for tractable and exact evaluation of many relevant inference tasks. All tractable graphical models have a SPN-representation (Poon and Domingos, 2011), and every SPN may be converted into a Bayesian Network in linear time (Zhao et al., 2015). They have successfully been employed to on complex tasks, such as modelling images (Poon and Domingos, 2011; Dennis and Ventura, 2012), speech (Peharz et al., 2014), or language (Cheng et al., 2014), and classification e. g. of images (Gens and Domingos, 2012), often producing state-of-the-art results at the time of publication. SPNs are relevant to this thesis because STOVE employs a variant of Sum-Product Attend-Infer-Repeat (SuPAIR) to model the distribution over individual images. SuPAIR uses Random Sum-Product Networks (Random SPNs) (Peharz et al., 2019) to achieve this. Random SPNs are a variant of SPNs fully amendable to end-to-end differentiable optimisation. Historically, SPNs are an instance of Arithmetic Circuits (Darwiche, 2003), for which parameters are traditionally specified and not learned. The interested reader is referred to Vergari et al. (2019) for a recent and extensive review of SPNs.

A Sum-Product Network (SPN) may be represented by a rooted directed acyclic graph, containing sum nodes, product nodes, and leaf distributions. The scope of an SPN is given by the set of variables over which it is defined. Following this, a recursive definition of SPNs is given:

> "(1) A tractable distribution is an SPN; (2) a product of SPNs defined over different scopes is an SPN; and (3), a convex combination of SPNs over the same scope is an SPN." (Molina et al. (2018))

While this definition should technically suffice, we will illustrate SPNs and their properties a bit further. At the root, the joint probability $p(\mathbf{x})$ over all $\mathbf{x} = (x_1, \ldots, x_D) \in \mathcal{X}$ is obtained. Each leaf defines a distribution over a subset of the input variables. Any other node in the tree is either a sum node or a product node. Sum nodes propagate (upwards) a weighted sum over their children, which all have identical scope. In other words, each sum node is a mixture model over its children. The mixture weights are learned during training, and Peharz et al. (2015) show that they may be normalised locally for each sum node without losing expressiveness. Product nodes compute the product over inputs from different, non-overlapping scopes. This allows the SPN to implement hierarchies over variables by modelling independencies between them. As we traverse the tree of a typical SPN, going down from the full joint distribution at the root, product nodes are used to iteratively split the joint distribution into less complex distributions over fewer variables. Intuitively, we want to use product nodes whenever we can find disjoint subsets of statistically independent variables at the current node. If we cannot find such independencies, we use sum nodes to model more complex distributions. These may then, in turn, induce further independencies. Usually, sum and product nodes are employed in an alternating pattern. Evaluation of an SPN is performed bottom-up, i. e. given an input $\mathbf{x}$, we evaluate all leaf distributions at the corresponding subvectors of $\mathbf{x}$ and then pass up the results, performing weighted summation over the children at sum nodes and multiplication of children at product nodes. Figure 2.6 shows an illustration of a simple SPN, where each leaf node defines a distribution over a single variable only. Note that for SPNs, the graph structure may be seen as a deterministic computation graph of summation and multiplication operations from the leaves to the root. In contrast, in graphical depictions of other models, such as Bayesian

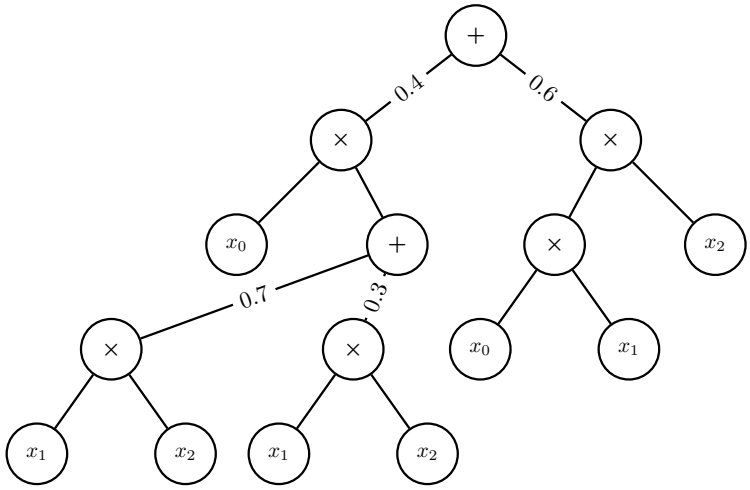


Figure 2.6.: Illustration of a simple SPN. Sum nodes implement normalised mixtures over their child distributions and product nodes model independencies. The joint probability over all variables $p(\mathbf{x})$ is obtained at the root by evaluating the leaf node distributions and propagating the results up through sum and product nodes. In this illustration, the leaves contain the univariate distributions of the variables $\mathbf{x} = (x_0, x_1, x_2)$. Figure adapted from Molina et al. (2019).

Networks, graph nodes usually directly illustrate random variables. In an SPN, this is only true for the leaf nodes. See the original publication (Poon and Domingos, 2011) for further information, as well as an airtight theoretical introduction of SPNs.

An attractive property of SPNs is that they are capable of performing many inference tasks in linear time with respect to the size of the model. Computation of marginals $p(\mathbf{x}_{-y}) = \int p(\mathbf{x}) \mathrm{d}\mathbf{y}$ simply corresponds to marginalisation at the respective leaf nodes, where $\mathbf{y} \in \mathcal{Y} \subset \mathcal{X}$, $\mathbf{x}_{-y} \in \mathcal{X}_{-\mathbf{y}} \subset \mathcal{X}$, and $\mathcal{Y} \cup \mathcal{X}_{-\mathbf{y}} = \mathcal{X}$, $\mathcal{Y} \cap \mathcal{X}_{-\mathbf{y}} = \emptyset$.[10] Each leaf node is defined over a subvector $\mathbf{y}'$ of $\mathbf{x}$. If all $\mathbf{y}' \in \mathcal{Y}'$ are in the marginalisation set for a specific leaf node $\mathcal{Y}' \subset \mathcal{Y}$, then marginalisation simply corresponds to setting this node to 1. Similarly, any conditional $p(\mathbf{y} \mid \mathbf{x}_{-\mathbf{y}})$ may be constructed via marginalisation $p(\mathbf{y}, \mathbf{x}_{-\mathbf{y}})/p(\mathbf{x}_{-\mathbf{y}})$ and is therefore obtained in linear time. A linear-time approximation exists for MPE-inference $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}}\, p(\mathcal{X}_{-\mathbf{y}} = \mathbf{x}_{-\mathbf{y}}, \mathbf{y})$, for which exact computation is NP-hard. Vergari et al. (2018) demonstrate that SPNs have built-in autoencoding and unsupervised representation learning capabilities.

If the structure of an SPN is given, e. g. because it is designed with prior knowledge of the data or specific applications in mind, the weights of an SPN may be learned using expectation maximisation or gradient-based approaches (Poon and Domingos, 2011). However, often, the tree structure of an SPN is learned from data using specialised algorithms such as in Gens and Pedro (2013), Poon and Domingos (2011), Dennis and Ventura (2012), or Adel et al. (2015). To obtain a valid SPN, the algorithms need to ensure that the structure obeys the *completeness* (i. e. for each sum node, all children have identical scope) and *decomposability* (i. e. for each product node, all children have non-overlapping scope) properties, see Poon and Domingos (2011) and Peharz et al. (2015) for further details.

[10] Set operations are meant as operations on the dimensionality of the input vectors, i. e. the $\cup$ operator corresponds to concatenation.

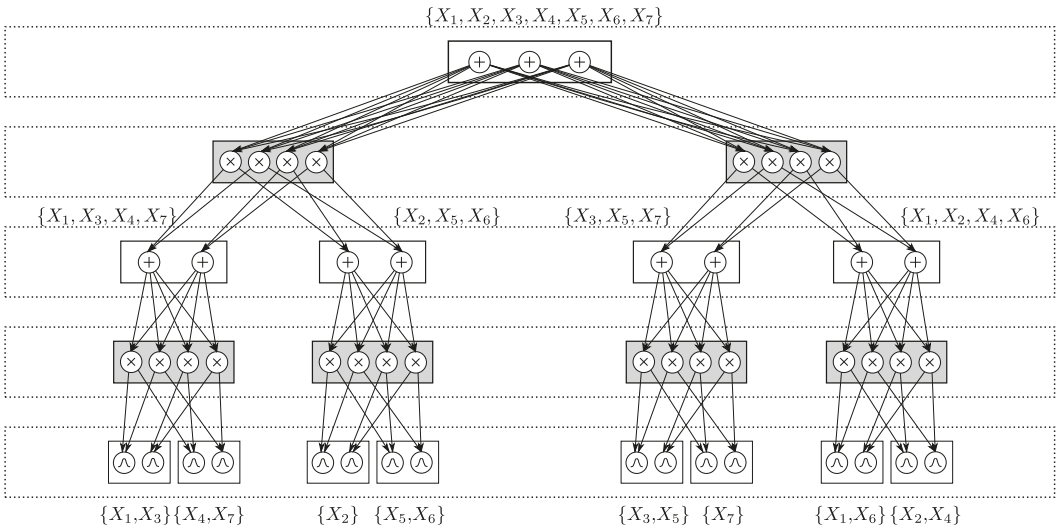


Figure 2.7.: Illustration of a Random Sum-Product Network (Random SPN) over seven variables $(X_1, \ldots, X_7)$ with hyperparameters $D = 2$, $R = 2$, $C = 3$, $S = 2$, and $I = 2$. Here, D is the number of times the scope is partitioned (downwards in the tree), R the number of times we repeat partitioning from full scope, C the number of sum nodes at the root, S the number of sum nodes at intermediate levels, and I the number of leaf distributions. Figure from Peharz et al. (2019), where further details may be found.

Random SPNs. Peharz et al. (2019) argue that the need for structure learning may have hindered widespread adoption of SPNs, as their learning algorithms are incompatible with the prevalent notion of general-purpose architectures and learning using Stochastic Gradient Descent (SGD). They propose Random Sum-Product Networks (Random SPNs), which forego structure learning entirely. Instead, a large set of *random* structures is generated, which essentially represent an over-parametrisation of the problem, similar in spirit to deep neural networks. After random initialisation, Random SPNs are fully end-to-end trainable using SGD. Such simple-to-use deep probabilistic architectures are desirable because they retain the capabilities of SPNs – exact likelihood evaluation, efficient computation of arbitrary conditioning or marginalisation queries, and a principled treatment of uncertainties and missing data – while eliminating the need for structure learning. In addition, Random SPNs, also called Random *Tensorised* SPNs (RATs), integrate naturally into popular deep learning frameworks and are easily combinable with other end-to-end differentiably trainable architectures such as neural networks. Peharz et al. (2019) demonstrate that Random SPNs are capable of modelling complex distributions, allow for a continuous trade-off between discriminative and generative learning, maintain well-calibrated predictive uncertainties, and can be used in hybrid neural-SPN architectures.

We will briefly discuss the construction of a Random SPN and direct the interested reader to the original publication by Peharz et al. (2019) for further details. Random SPNs abandon structure learning in favour of a randomly created graph structure, the *region graph*. The creation of the region graph can be defined succinctly recursively, see Peharz et al. (2019, Algorithm 2). The overarching idea is that, starting at the root, the full scope $\mathcal{X}$ is randomly divided into two disjunct and equally-sized regions $\mathcal{X}_1$ and $\mathcal{X}_2$, such that $\mathcal{X}_1 \cup \mathcal{X}_2 = \mathcal{X}$ and $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, a 2-partition. These are then further and randomly subdivided down the tree, where a hyperparameter D governs

how often this splitting procedure is recursively repeated. The whole procedure itself (partitioning from full scope) is also repeated a specified number of times R. Given these regions, a Random SPN may then be created. A region will either be a sum region or a leaf region, where each sum region will be equipped with S sum nodes and each leaf region with I leaf nodes. For class-conditional distribution modelling $p(\mathcal{X} \mid \mathcal{C})$, C different roots may be used in parallel. Between two sum regions (or a leaf and a sum region), product regions are created which realise all possible products between the sum nodes of their child regions (the outer product of the child regions). Figure 2.7 shows an example of a Random SPN. Instead of learning the correct partitioning of variables from data, the Random SPN contains many different deep hierarchical mixtures over the variables. If sufficiently large random structures are generated, SGD-training automatically reinforces useful substructures, such that Random SPNs can successfully model complex data.

3 Related Work

While the previous chapter has introduced general concepts, relevant to the machine learning community at large, this chapter focuses on related work in more specific subfields. Our approach, STOVE, mainly combines two related topics: image modelling and physics prediction. The combination of these necessarily yields a model capable of reasoning over sequences, i. e. *videos*. While we have presented general-purpose sequence models in Sections 2.1 and 2.3, this chapter will also devote a section to models catered to sequence prediction, albeit with a focus on videos, object-awareness, or physics. Each of the three sections below will first give a high-level overview of related works. Then, some approaches will be presented in-depth as they provide direct inspiration for STOVE, are used as components in STOVE, or are compared to as baselines. Specifically, STOVE uses Sum-Product Attend-Infer-Repeat (SuPAIR), Section 3.1.2, as its vision model and a Graph Neural Network (GNN), Section 3.2.2, as its physics model. Lastly, each section then contains a few words on tangential but nevertheless related developments in the respective fields.

3.1 Object-Aware Image Modelling

3.1.1 Motivation and Overview

Teaching machines to *see* as humans do is the ultimate goal of computer vision. Ideally, this entails learning a model which can adapt to and solve a wide variety of tasks in real-world visual environments with little supervision using a complex mental representation of the world. In practice, however, this ambitious goal is often broken down into a variety of narrow tasks, which necessitate the expensive curation of labelled training data for supervised learning. Most popular is image classification, where each image is an instance of a class to be determined by the model, see e. g. Rawat and Wang (2017). Also prevalent are segmentation task, where each pixel belongs to a class in the image, see e. g. Ronneberger et al. (2015). Then, there are hybrid tasks between computer vision and natural language processing, such as image captioning, wherein the model describes the contents of a given image with a short sentence. Other tasks of computer vision, such as object tracking (e. g. Redmon et al. (2016)) or motion estimation (e. g. Dosovitskiy et al. (2015)), are related to videos rather than singular images.

Classically, hand-designed approaches, such as designing specific filter kernels to detect edges, were used to engineer solutions to some of the above tasks. However, currently, many state-of-the-art models in computer vision rely on some form of machine learning. Of those, many use (deep) convolutional neural networks (CNNs). This applies to *all* examples cited above. CNNs specifically have been an instrumental component in achieving state-of-the-art performance over the last decade, where the publication of Krizhevsky et al. (2012) marks a starting point. Only recently

have purely *self-attention-based* architectures begun to take off in computer vision (Ramachandran et al., 2019), challenging the prevalent dominance of convolutional architectures.

Because the collection of supervised data for the models above is expensive and time-consuming, the computer vision community has put approaches forward to alleviate this problem, such as data augmentation, weak supervision, semi-supervision, or training on surrogate tasks. However, generally speaking, the problem remains unsolved. A promising avenue of research is that of image (and video) modelling.

Image Models. An *image model* aims to learn a probabilistic model of the likelihood distribution $p(\mathbf{x})$ of images $\mathbf{x}$ in unsupervised fashion from a data set $(\mathbf{x}_1, \dots, \mathbf{x}_N)$. Ideally, such a model then allows for both sampling of novel images $\mathbf{x} \sim p(\mathbf{x})$ and exact evaluation of the likelihood $p(\mathbf{x})$. Additionally, the model might extract a latent representation $\mathbf{z} \sim p(\mathbf{z} \mid \mathbf{x})$, or a variational approximation thereto, which may be useful for reasoning in downstream tasks. At first inspection, image models seem unrelated to the tasks of computer vision mentioned above. However, as we shall soon see, depending on their specification, they may directly or indirectly help solve tasks entirely without supervisory signals. In fact, generative modelling-based approaches to computer vision have a longstanding history in the community and are known as *vision as inverse graphics* (Grenander, 1976). Such approaches require the formulation of an image generating process, such as a renderer, and *vision* then corresponds to parameter inference in the renderer. Classically, this renderer is non-differentiable, and parameter inference is attempted using sampling–based approaches. We will not discuss such models here. Instead, we focus on models that *learn* the data-generating process. Generally, image models are trained in an unsupervised fashion, although learning may be conditioned on supervisory signals if desired.

Before we focus on *object-centric* approaches to image modelling, we briefly discuss the use of general-purpose probabilistic models for image modelling. Variational Autoencoders (VAEs) are a popular example of deep generative models, which may be used for image modelling, and have been introduced in detail in Section 2.2. They allow for the generation of novel images in a two-step process: First, a latent sample $\mathbf{z}$ is drawn from the prior $\mathbf{z} \sim p(\mathbf{z})$, after which an image $\mathbf{x}$ is generated from the conditional likelihood $\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{z})$. VAEs do not allow for direct evaluation of the likelihood $p(\mathbf{x})$, although sampling-based approximations exist (Kingma and Welling, 2014). However, a latent representation $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$ may be obtained. Generative Adversarial Networks (GANs) allow for the generation of highly realistic samples but lack an explicit likelihood as well as posterior inference (Goodfellow, 2016). Autoregressive approaches, such as Van den Oord et al. (2016), omit the latent space but allow for tractable likelihood evaluation. We employ a Random Sum-Product Network (Random SPN), which allows for tractable density estimation, sampling, and encoding, but is, in essence, a linear architecture, see Section 2.4. Recently, flow-based approaches, see Rezende and Mohamed (2015), Kobyzev et al. (2019), or Papamakarios (2019), have shown promise. They use a series of non-linear invertible transformations to yield neural density estimators that allow for sampling, tractable likelihood evaluation, and inference of latent states. However, depending on the flow architecture, either sampling is slow and likelihood evaluation is fast, sampling is fast and likelihood evaluation is slow, or both are fast but at the cost of decreased modelling flexibility (Papamakarios, 2019). That being said, most image

models relevant to this thesis are VAE-based architectures. Although there are no obvious obstacles for using flows or other approaches for image modelling, we are not aware of many works that do so. One may specualte that this is due to the wealth of work dealing with interpretable or disentangled latent spaces in VAEs, see Section 2.2, compared to only very few works that discuss this in the context of flows, see Sorrenson et al. (2020) for a rare example.

Attend-Infer-Repeat. As motivated in Chapter 1, the notion of objects is an elementary building block to human cognition (Lake et al., 2017). This strong human prior of object-oriented vision motivated Eslami et al. (2016) to introduce the Attend-Infer-Repeat (AIR) model. Unlike the previously discussed *unstructured* generative models, AIR has a baked-in notion of objects. This object-centrality is achieved using a special architecture, entirely unlike the predominant approaches to supervised scene segmentation (Burgess et al., 2019). We employ a variant of the AIR-successor SuPAIR (Stelzner et al., 2019), which we introduce in full detail below in Section 3.1.2, in our approach to object-aware video modelling. In addition to SuPAIR, many other variants of AIR have since been introduced, some of which are mentioned in Section 3.1.3. Before that, we would like to give a brief and non-technical introduction to the idea behind the AIR image models.

For AIR, an image is modelled as a superposition of objects. An LSTM, see Section 2.1, the attention network, is given the image as input and iteratively attends to each of the objects, inferring their positions and sizes. These are used as input to spatial transformer layers (Jaderberg et al., 2015), which obtain rectangular, differentiable cutouts at the proposed object locations. A VAE is then used to model the appearances of these cutouts individually. To obtain image reconstructions, first, reconstructions of the object cutouts are obtained from the object VAE and then pasted one by one to the correct position at the correct scale on the, initially empty, image canvas using the inverse transformation for the spatial transformer. Therefore, AIR is also referred to as a *cut-and-paste* image model (Stelzner et al., 2019).

The original AIR has some fairly restricting assumptions: Rectangular bounding boxes are assumed, the total number of objects is assumed small, and the background is assumed to be black. Additionally, AIR is only demonstrated to work on visually simple data. However, it is in many ways closer to the elementary goal of computer vision than the supervised, task-specific approaches discussed above. Instead of training on a specially curated data set for a narrow task, AIR obtains a general-purpose, object-oriented representation of a given scene. This representations may be used directly for object localisation, (rudimentary) image segmentation, or object counting, and is easily adaptable to other downstream tasks. Scaling AIR-like approaches to more natural scenes of higher visual complexity may provide solutions to key problems in computer vision. Motivated by this, there is a large interest from the robotics and reinforcement learning community, where the creation of manually labelled datasets is often infeasible.

3.1.2 Sum-Product Attend-Infer-Repeat

With Sum-Product Attend-Infer-Repeat (SuPAIR), Stelzner et al. (2019) suggest a variation to the original AIR architecture: Instead of using a VAE to model the appearance over the objects, SuPAIR employs a Random Sum-Product Network (Random SPN), see Section 2.4, to model the object distributions directly. We will

now thoroughly introduce SuPAIR, relying heavily on the formalism and motivation of Section 2.2, and then discuss improvements over the original AIR. Although the original formulations of AIR and SuPAIR can handle varying numbers of objects, in this work, a fixed and given object count will be assumed throughout for simplicity. SuPAIR with fixed object count is used as the vision component in STOVE, as presented in Chapter 4.

Like AIR, SuPAIR is framed as a Bayesian scene model: A data generating process $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$ is assumed, where $\mathbf{x}$ is an image and $\mathbf{z}$ are latent variables. Maximum likelihood learning of $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z})\mathrm{d}\mathbf{z}$ and posterior inference $p(\mathbf{z} \mid \mathbf{x}) = p(\mathbf{x}, \mathbf{z})/p(\mathbf{x})$ are intractable, and the recognition network $q(\mathbf{z} \mid \mathbf{x})$ is introduced as the amortised variational approximation to the true posterior. Learning is performed jointly in the generative process and the recognition distribution by maximising the ELBO, Eq. (2.10), as a lower bound to the true image likelihood. So far, this is identical to the VAE. The specification of the generative and recognition distributions will now give the formalism by which *the notion of objects* is included in the model. SuPAIR is illustrated in Fig. 3.1a.

The recognition network $q(\mathbf{z} \mid \mathbf{x})$ infers a distribution over the per-object latent states $(\mathbf{z}^1, \dots, \mathbf{z}^O) = \{\mathbf{z}^o\}_{o=1}^O = \mathbf{z}^{1:O}$. Each object is characterised by a two-dimensional position and size parameters for the horizontal and vertical scale of the object $\mathbf{z}^o = (\mathbf{z}^o_{\text{pos}}, \mathbf{z}^o_{\text{size}}) = (\mathbf{z}^o_{\text{x-pos}}, \mathbf{z}^o_{\text{y-pos}}, \mathbf{z}^o_{\text{x-size}}, \mathbf{z}^o_{\text{y-size}})$. The size parameters give the scale of the object's bounding box with respect to the size of the canvas, i. e. $0 \leq \mathbf{z}^o_{\text{x/y-size}} \leq 1$. The posterior approximation $q(\mathbf{z} \mid \mathbf{x})$ is given by a factorised Gaussian

$$q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x}))\,, \tag{3.1}$$

whose means and variances are predicted by the attention network, an LSTM. It is given the image $\mathbf{x}$ as constant input and outputs the parameters $(\boldsymbol{\mu}(\mathbf{x})^o, \boldsymbol{\sigma}^2(\mathbf{x})^o)$ for the latent state $\mathbf{z}^o$ distribution at each iteration $o = (1, \dots, O)$.

Let us now turn towards the specification of the generative distribution $p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$. The prior

$$p(\mathbf{z}) = \prod_{o=1}^{O} p(\mathbf{z}^o_{\text{pos}})\, p(\mathbf{z}^o_{\text{size}}) \tag{3.2}$$

is assumed to be a simple uniform distribution over the object positions and sizes. The conditional likelihood $p(\mathbf{x} \mid \mathbf{z})$ also factorises over objects as

$$p(\mathbf{x} \mid \mathbf{z}) = p_{\text{bg-SPN}}(x^{\text{bg}} \mid \mathbf{z}^{1:O}) \prod_{o=1}^{O} p_{\text{obj-SPN}}(\text{ST}(\mathbf{x}; \mathbf{z}^o, A) \mid \mathbf{z}^{1:o})^{\mathbf{z}^o_{\text{x-size}}\mathbf{z}^o_{\text{y-size}}}\,. \tag{3.3}$$

This equation warrants a longer explanation. The conditional likelihood of an image $\mathbf{x}$ is given by the product of the background likelihood and the individual object likelihoods. The likelihood of an object is modelled directly using the *obj-SPN*, a Random Sum-Product Network (Random SPN), see Section 2.4. Likewise, the likelihood of the background is modelled using the *bg-SPN*. Object sizes may vary between objects and images, but the *obj-SPN* expects a fixed number of input variables, or equivalently, a fixed object pixel resolution. Therefore, for each object o, a spatial transformer layer $\text{ST}(\mathbf{x}; \mathbf{z}^o, A)$ extracts the contents of $\mathbf{x}$ at $\mathbf{z}^o_{\text{pos}}$ via a rectangular bounding box with size $\mathbf{z}^o_{\text{size}}$ and scales them to fixed quadratic resolution (A, A).

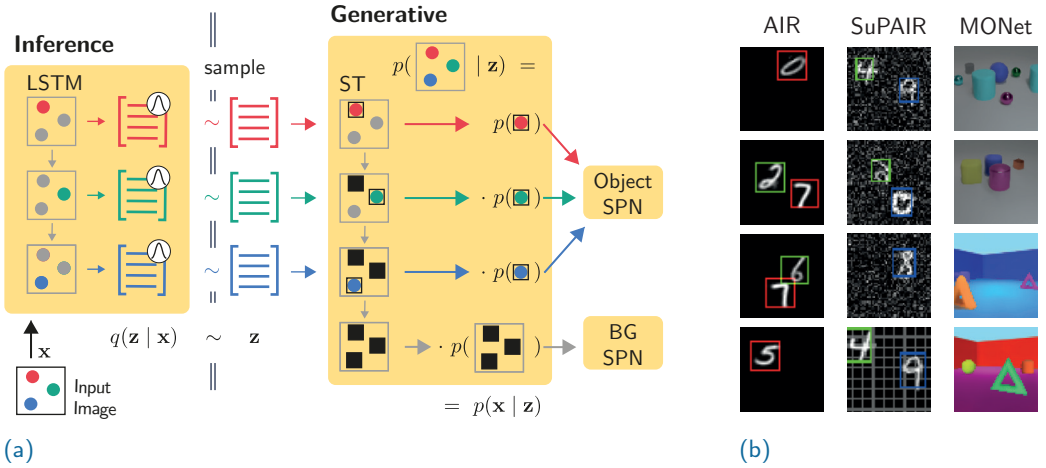


Figure 3.1.: (a) Overview of the Sum-Product Attend-Infer-Repeat (SuPAIR) architecture for image modelling. An attention LSTM attends to one object after the other, inferring their latent positions and sizes $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$. A spatial transformer layer (ST) extracts rectangular object patches from the image $\mathbf{x}$. The object and background SPNs directly assign patch likelihoods from which we assemble $p(\mathbf{x} \mid \mathbf{z})$. Generative and inference model are trained jointly by maximising the ELBO. (b) Example images from AIR, SuPAIR, and MONet. Unlike AIR, SuPAIR can handle structured backgrounds and accommodate noisy inputs. The spatial mixture model formulation of MONet allows for modelling of more complex scenes, although MONet relies heavily on colour cues. Images for (b) reassembled from Eslami et al. (2016), Stelzner et al. (2019), and Burgess et al. (2019).

The object likelihoods may then be directly evaluated using the *obj-SPN*. In Eq. (3.3), they need to be rescaled by an exponent of $\mathbf{z}^o_{\text{x-size}}\mathbf{z}^o_{\text{y-size}}$ in order to account for the transformation. As we expect the overall image sizes to be constant throughout the data, no such rescaling is needed for the evaluation of the background likelihood in Eq. (3.3). The evaluation of any object likelihood component is always conditioned on *all previous* latent variables. This becomes necessary when the bounding boxes of different objects overlap. If this is the case, we must avoid modelling the same pixels twice, once for object o_1 and again for $o_2 > o_1$. Therefore, pixels that have been modelled in o_1 need to be *marginalised over* when evaluating the likelihood $p_{\text{obj-SPN}}(\text{ST}(\mathbf{x}; \mathbf{z}^{o_2}, A) \mid \mathbf{z}^{o_1:o_2})$, implementing *explaining away*. This marginalisation is trivial to achieve in random SPNs, see Section 2.4. Essentially, if we evaluate an object $o_2 > 1$, we need to check if any of the pixels of its bounding box in $\mathbf{x}$, parametrised by $\mathbf{z}^{o_2}$, have been part previous of bounding boxes $o_i < o_2$, and if this is the case, we must marginalise over these pixels when evaluating the conditional likelihood. Here, we let the order of the objects as discovered by the attention LSTM induce a natural ordering. For the background, we must marginalise over all pixels which are part of object bounding boxes, to ensure that the *bg-SPN* does in fact model background pixels only. In Stelzner et al. (2019), the generative model, Eq. (3.3), is written down using a formulation relying on mask and indicator matrices, so that they can explicitly handle the marginalisation information that we have dealt only informally with here. Otherwise, their presentation of the algorithm is identical.

The parameters of the inference network, i. e. the parameters of the LSTM attention network, and the generative network, i. e. the weights and leaf distribution parameters of the object and background random SPNs, are optimised jointly us-

ing mini-batch gradient descent on the standard ELBO, Eq. (2.10). Single samples $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$ and the reparametrisation trick are used to approximate the gradient of the expectation, as presented in Section 2.2.

While SuPAIR uses amortised variational inference to obtain the position and size parameters of the objects, unlike AIR, it does *not* use a variational autoencoder to model the appearances over the objects. In the original AIR, an encoder network extracts a latent description $\mathbf{z}^o_{\text{what}}$ from the object glimpse, from which a decoder then generates a reconstruction. Much of the model complexity of AIR lies in this encoding-decoding detour to obtain the loss on the generative model $p(\mathbf{x} \mid \mathbf{z})$ (Stelzner et al., 2019). By explicitly modelling the distribution over the objects, SuPAIR is able to assign conditional likelihoods directly, without resorting to expensive reconstructions. The authors find that this improves the speed of learning in SuPAIR by an order of magnitude in comparison to AIR. AIR assumes the background to be black, while SuPAIR explicitly models the distribution over the background. Therefore, with SuPAIR, one obtains a fully probabilistic model over the conditional likelihoods. Stelzner et al. (2019) demonstrate that this leads to more robust performance in scenarios with non-uniform backgrounds or observation noise. Note that, while SuPAIR does not *need* a per-object latent encoding, if desired, such an encoding can still be obtained directly using the autoencoding procedure specified in Vergari et al. (2018). Figure 3.1b showcases applications of AIR and SuPAIR.

3.1.3 Tangential Developments

We will briefly touch on some historical and recent developments related to AIR and SuPAIR, some of which may inspire future work.

Preceding AIR, models such as RAM (Mnih et al., 2014) and DRAM (Gregor et al., 2015) have used attention-based mechanisms for modelling images as a sequential drawing process. In each drawing step, the attention model selects a region of the image canvas upon which the model then draws. Unlike in AIR, these steps are not supposed to correspond to objects but rather separate strokes of a pen, mimicking the human drawing process.

Succeeding AIR, a variety of approaches have been proposed that aim to overcome its limitations. Crawford and Pineau (2019) target AIR's inability to accommodate scenes with large and varying amounts of objects by replacing the recurrent attention mechanism with a convolutional architecture. MONet (Burgess et al., 2019) relaxes AIR's assumption of rectangular bounding boxes. Instead of predicting bounding box parameters, MONet postulates each image as a pixel-wise mixture of components. For this, MONet uses a U-Net-like (Ronneberger et al., 2015) architecture to predict *pixel-wise masks* for each object iteratively and in an unsupervised fashion. The image *components*, or objects, are then modelled with a component-VAE at full resolution, where the masks are used to block out unrelated pixels for the current component. While the number of components C, also called slots, is fixed for MONet, the approach is able to handle a variable number of objects $C' \leq C$ by ignoring the superfluous slots using the masking mechanism. In addition to modelling more complex sceneries, this allows MONet to accommodate partially occluded objects, see Fig. 3.1b. MONet is shown to generalise to unseen numbers of objects as well as novel feature combinations and object co-occurences (Burgess et al., 2019). The extracted latent scene representation corresponds to interpretable quantities such as object shape, position, scale, or colour. However, MONet's attention mechanism

is deterministic, and it lacks a proper probabilistic treatment. Specifically, there is no principled way to allow for unconditional generation of scenes because the component latent states are assumed to be independent. Two approaches that remedy this are GENESIS (Engelcke et al., 2020) and IODINE (Greff et al., 2019). With IODINE, Greff et al. (2019) introduce an iterative inference scheme. At each step, component latents are inferred jointly over all objects. However, the iterative procedure allows for interactions between the latents and, therefore, unconditional scene generation. GENESIS takes a different route and introduces dependencies between the object-wise latents through an autoregressive formulation using a RNN. Most recently, SPACE (Lin et al., 2020) has proposed a combination of spatial attention and scene mixture model, attempting to solve scaling issues in MONet and its derivatives, similar to how Crawford and Pineau (2019) did for AIR.

3.2 Physics Modelling

3.2.1 Motivation and Overview

As with the previous section, we will now give a brief introduction to physics modelling in the machine learning community. Sometimes the lines between a *physics* model and *video* model become blurry because both may be able to predict future observations. For us, *physics* models are those (a) whose main motivation comes from modelling physics, (b) explicitly use a physics engine, or (c) rely on strong supervisory signals such as object positions and properties, or the pre-specification of the governing types of physics (e. g. gravity or collisions).

Humans are capable of robustly and precisely predicting the result of intricate physical processes, such as objects falling under gravity, colliding with walls or other objects, or rolling down slopes. We can do this with incredible flexibility for a variety of scenarios and objects. This facet of human ability is not only relevant for children playing with toy bricks. Instead, it seems to be a core component of human intelligence, relevant for a variety of every-day tasks. It is called *intuitive physics*. For a long time, the predominant explanation of this phenomenon has been that humans rely on a broad set of heuristics (McCloskey, 1983). However, recent developments in the cognitive and psychological sciences suggest that a better explanation is given by assuming that we possess some variant of a mental Newtonian physics simulator (Hamrick et al., 2011; Sanborn et al., 2013; Battaglia et al., 2013). To predict future trajectories, we perform probabilistic inference in this mental physics engine given our noisy sensory observations.

Simulator-Based Approaches. Early approaches in physics modelling, such as Galileo (Wu et al., 2015), were heavily driven by such cognitive motivations, designing machine learning models to mimic this newfound explanation of intuitive physics. Galileo proposes a model which can infer the parameters of a single object – such as mass, position, or shape – from observations of real videos of physics experiments. While certainly impressive, early approaches, such as Galileo, often contain a lot of hand-designed structure and supervision: A standard tracking algorithm is used to infer the object trajectories, e. g. of an object rolling down a slope. A physics simulation software is initialised with the type and geometry of the experiment, such that it matches the observed experiment exactly. The parameters of the simulation correspond to the object properties, we wish to infer. Then, Markov Chain Monte

Carlo (MCMC) is used to find the parameters of the simulator which result in simulations that best fit the observations. Once a good fit is obtained, the parameters may be compared to ground truth or the simulator is used to predict future trajectories.

Robotics. In Fragkiadaki et al. (2016) and later Agrawal et al. (2016), the cognitive roots are overshadowed and replaced by hopes of intuitive physics in robotics, teaching an agent to predict the effects of its actions in novel environments. In Fragkiadaki et al. (2016), the effects of hitting billiard balls on a simulated two-dimensional table are predicted. Object positions and applied forces are given. To predict the effect of collisions, a *glimpse* around the current object position is taken, from which a CNN extracts useful visual information about nearby objects or walls. The approach struggles to simulate scenarios with more than one ball realistically. In Agrawal et al. (2016) a similar approach is used to teach a real robot to push objects to goal locations from video input alone.

3.2.2 Relational Physics using Graph Neural Networks

Neural Prediction Engine. Chang et al. (2017) present the Neural Prediction Engine (NPE), which uses a neural network to predict future states for each object individually. Instead of relying on visual attention via glimpses, the NPE models the interactions for object o by applying an *interaction network* to each nearby object o'. While not explicitly written down by the original authors, the next-state prediction for each object o is given by

$$\Delta \mathbf{v}^o = A\left(\sum_{o' \neq o} \mathbb{1}\left[\|\mathbf{r}^o - \mathbf{r}^{o'}\| < d\right] \; I(\mathbf{z}^o, \mathbf{z}^{o'}),\, \mathbf{z}^o\right), \tag{3.4}$$

the sum of its interactions $I(\cdot)$ and the previous state $\mathbf{z}^o$ aggregated by $A(\cdot)$, where both functions are parametrised using MLPs. Here, $\mathbf{r}^o \subset \mathbf{z}^o$ are two-dimensional position vectors, and the states $\mathbf{z}^o$ contain extrinsic properties (positions, velocities, orientations, angular velocity) and intrinsic properties (mass, object type, object size). The result $\Delta \mathbf{v}^o$ is the change in velocity for object o between two consecutive, fixed-length timesteps. The neighbourhood mask $\mathbb{1}[\cdot]$ is a *hard* and prespecified selection criterion, which posits that only interactions between objects that are within Euclidean distance d of one another are considered. This hard masking allows NPE to scale to many-object scenarios. The approach relies on complete state supervision and does not feature any visual component.

Interaction Networks. On the very same day as NPE, Battaglia et al. (2016) introduced Interaction Networks (INs). Like, NPE they process object states and not visual input, and they require full supervision with respect to object properties and physics parameters. Unlike NPE, the authors demonstrate that INs can handle a variety of physics scenarios in addition to the bouncing ball data of NPE, such as n-body gravitational simulations, invisible springs between objects, or non-rigid dynamics (soft strings – realised as a chain of springs – falling under gravity and wrapping around hard objects). Gravitational interactions and invisible springs act at a distance, such that NPE's local neighbourhood assumption necessarily leads to modelling failure. Instead, INs require the supervised specification of a $O \times O$, non-symmetric, binary interaction matrix, indicating which objects interact with one another. Again, INs rely on complete state supervision and do not feature a visual

component. NPE and INs have been shown to outperform general-purpose neural networks such as MLPs or LSTMs on trajectory prediction tasks.

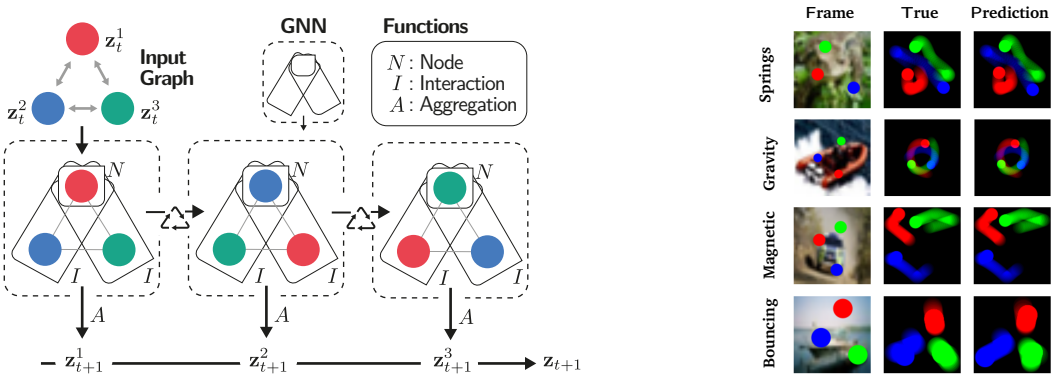
Visual Interaction Networks. Watters et al. (2017) then introduce Visual Interaction Networks (VINs), which extend INs in multiple meaningful ways. VINs can infer latent object positions, velocities, and properties from a stack of visual observations using a CNN. A dynamics model, similar to INs, then predicts future object states, but not observations. The model performs well on a number of physics scenarios and requires only image-state pairs for learning, where states contain object positions and velocities. Figure 3.2b illustrates results obtained by VIN. No additional supervisory signals, such as object properties or relation matrices as in NPE or IN, are needed. However, VIN needs to rely on colour cues such that the object order with respect to the MSE loss used for training between predicted and ground truth states is consistent. Instead of making predictions using only the previous state, VIN predicts future states by aggregating over predictions made using the last C states. At time t, for each offset $c \in \{0, \dots, C-1\}$, a different *interaction core* predicts $\mathbf{z}_{t+1}$ from the observation $\mathbf{z}_{t-c}$. From Watters et al. (2017, App. 8.2), we can deduce that this prediction is given by

$$\mathbf{z}^o_{t+1} = U^c \left(A^c \left(\sum_{o' \neq o} I^c(\mathbf{z}^o_{t-c}, \mathbf{z}^{o'}_{t-c}) + N^c(\mathbf{z}^o_{t-c}) \right), \mathbf{z}^o_{t-c} \right), \tag{3.5}$$

where the similarities to Eq. (3.4) are apparent and U^c, A^c, I^c, N^c are MLPs shared over all objects. Unlike IN and NPE, VIN generally assumes interactions between *all* pairs of objects. An aggregation MLP aggregates over the predictions from the individual offsets. Notably, VINs outperform INs, which have access to ground truth states. In addition to the added flexibility from multiple prediction cores, the authors speculate that the observation noise may have a regularising effect, a discussion we will pick up later in Chapter 5.

The above approaches exploit much prior knowledge related to our understanding of physics. Equation (3.5) contains all of the crucial ideas: The prediction for the next state of object o should contain a *sum* of interaction terms of object o with all other objects $o' \neq o$, as well as a term relying only on the object itself, which allows to model movement in the absence of interactions. The function which computes the interactions is shared over all objects. Likewise, the same state prediction Eq. (3.4) function is applied to all objects. Building a physics predictor in such a way is an excellent example of how to leverage prior knowledge to build structured models, which outperform general-purpose equivalents. Because of its modular design, this approach trivially generalises to unseen numbers of objects. While Fragkiadaki et al. (2016) also predict future states for each object independently, their handling of interactions using the visual attention glimpses is somewhat implicit, which likely explains the observed deficient performance when modelling collisions between objects.

Graph Neural Networks. Approaches such as the above may be seen as instances of Graph Neural Networks (GNNs). This view was popularised by Battaglia et al. (2016). For VIN, we can imagine a fully connected, directed graph between all objects. At time t, each object state $\mathbf{z}^o_t$ is associated with a node in that graph, and edges indicate the possibility of interactions between two objects. In graph neural



(a) (b)

Figure 3.2.: (a) Overview of the Graph Neural Network (GNN) architecture for relational physics modelling. In our case, a fully connected graph between all object states $(\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{z}_t^3)$ is assumed. The GNN makes future state predictions using the same set of functions for each object, where each object is chosen to be the *focus node* once. The node function $N(\mathbf{z}_t^o)$ is applied to the focus node o, and for each neighbour of the focus node, the interaction function $I(\mathbf{z}_t^o, \mathbf{z}_t^{o'})$ is applied for all focus-node–neighbour-node pairs (o, o'). The aggregation function yields the next state by aggregating over the results of I and N, $\mathbf{z}_{t+1}^o = A(N(\mathbf{z}_t^o), \sum_{o'} I(\mathbf{z}_t^o, \mathbf{z}_t^{o'}))$. While the procedure is depicted as a sequence in the graphic, i. e. one object is predicted after the other, prediction can easily be parallelised in implementation. (b) Example simulations obtained from the Visual Interaction Network (VIN). The left column shows an example input frame to the VIN architecture, the centre column visualises true trajectories taken by the balls in the various physics simulations, and the final column shows VIN's well-matching predictions. Simulation images in (b) reconstructed from Watters et al. (2017).

networks, the same set of operations is applied locally at each node $\mathbf{z}^o$. A set of GNN operations matching the above would be: (a) the neural network $N(\mathbf{z}^o)$ is applied to the node itself, (b) then $I(\mathbf{z}^o, \mathbf{z}^{o'})$ is applied for each neighbour o' of o on the graph, and (c) the results of the operations (a) and (b) are aggregated with $A(\cdot)$, which finally yields the output of the GNN at node $\mathbf{z}^o$. This architecture is demonstrated in Fig. 3.2a. Comparing this to the above models, we can now clearly see how VIN corresponds to a GNN with edges between all objects, IN corresponds to a GNN with edges specified by the interaction matrix, and NPE corresponds to a GNN where edges are assigned dynamically depending on the current distances between objects. For a thorough introduction and review of GNNs, the reader is referred to either Zhou et al. (2018) or Wu et al. (2019b). STOVE uses a GNN similar to Eq. (3.5) as its physics prediction module, which we will present in Section 4.2.

3.2.3 Tangential Developments

Zheng et al. (2018) extend INs to extract physical properties from object state trajectories. Simulator-based approaches remain popular, and Wu et al. (2016, 2017) make Galileo more efficient and performant by replacing expensive sampling-based inference routines with neural networks. Jaques et al. (2020) use a differentiable physics engine, which enables end-to-end gradient-based learning. Interaction mechanisms similar to $I(\cdot, \cdot)$ in Eq. (3.5) present a useful inductive bias to other tasks in machine learning which require reasoning about relations between entities. Santoro

et al. (2017) introduce Relation Networks (RNs) and show that such interaction mechanisms are useful to solve a variety of visual or text-based question answering tasks.

Physical Tracking. Ehrhardt et al. (2018) learn a multi object-tracker using the physically motivated principles of *causality* and *equivariance*, resulting in an adversarial style training with data augmentation. For each stack of input frames, a randomly warped version is created. Equivariance then ensures that the warped version of the input images results in an appropriately warped trajectory. Causality is motivated by the desired smoothness of the obtained trajectories. Random permutations of real trajectories, which violate smoothness, are generated, and a discriminator is trained to discern between real and permutated trajectories produced by the tracker. Also relevant is their later work (Ehrhardt et al., 2019), where a single model learns to accommodate a variety of physics simulations by explicitly inferring the currently active scenario from a stack of past observations.

GNNs for Particle Simulations. GNNs have been applied to physical simulations of increasing complexity in supervised scenarios. Sanchez-Gonzalez et al. (2018) propose a generalisation of INs and apply it to the *DeepMind Control Suite* (Tassa et al., 2018). Recently, Li et al. (2019) and Sanchez-Gonzalez et al. (2020) have used large GNNs with up to ten-thousand particles to obtain realistic simulations of fluids, rigid objects, or deformable objects in three-dimensional environments.

Neural ODEs. Another interesting and related line of work are Neural Ordinary Differential Equations (Neural ODEs) (Chen et al., 2018), where neural networks are used to parametrise the time derivative of some unknown dynamical system $\mathrm{d}f(\mathbf{z}(t))/\mathrm{d}t = \mathrm{NN}(\mathbf{z}(t), t; \boldsymbol{\theta})$. This ODE is then solved using differentiable versions of standard ODE integrators and gradient-based learning optimises the parameters $\boldsymbol{\theta}$ of the network to produce time derivatives which result in trajectories $\mathbf{z}(t)$ in agreement with the training set. Neural ODEs can model continuous-time processes and easily incorporate irregularly sampled data. They can be seen as infinite depth normalising flows. Lutter et al. (2019), Cranmer et al. (2020), Greydanus et al. (2019), and Toth et al. (2020) explicitly constrain neural ODEs to modelling *physical systems* by interpreting the neural ODE in relation to a physical Hamiltonian or Lagrangian. They show successful application on dynamical systems such as chaotic pendulums or n-body systems, where they achieve energy conservation. Kolter and Manek (2019) similarly constrain a neural ODE to produce stable dynamics using the concept of Lyapunov functions from chaos theory. The above approaches will likely fail, if the underlying dynamics may not be well approximated by an ODE. It is unclear how well they scale to discontinuous dynamics or more general scenarios. However, extensions of neural ODEs to stochastic processes exist (Jia and Benson, 2019).

3.3 Video Modelling

3.3.1 General-Purpose Video Models

We refer to *video prediction* as the conditional prediction of future frames $\tilde{\mathbf{x}}_{t+1}$ from a history of observations $(\mathbf{x}_1, \ldots, \mathbf{x}_t)$, where observations correspond to high-dimensional images. Often, such models are trained in self-supervised fashion by comparing the prediction $\tilde{\mathbf{x}}_{t+1}$ to the true and readily observable $\mathbf{x}_{t+1}$. *Video*

modelling is treated akin to image modelling, where, depending on the application, unconditional sampling of new videos, tractable video likelihood evaluation, or the inference of informative sequence- or image-level latents is desired. General-purpose non-linear latent state *sequence models*, such as the VRNN (Chung et al., 2015), see Section 2.3.2, may of course also be used to process and predict future frames in videos. In fact, the VRNN is a popular baseline in the object-aware video modelling community. However, other architectures purpose-built for videos also exist. They are often motivated by applications in (model-based) reinforcement learning, prediction, classification, or content creation.

The discussion of generative models of video is similar to that for sequence models, see Chapter 2, and image models in Section 3.1 above. Srivastava et al. (2015) present an early approach to multi-step video prediction. Simple enough, they use an LSTM which consumes the sequence of images and is trained in a self-supervised fashion. The fully connected layers of a standard LSTM are unsuited to accommodate complex visual scenarios. This motivates Xingjian et al. (2015) to introduce the Convolutional LSTM (ConvLSTM), combining CNNs and LSTMs, which they apply to precipitation forecasting, outperforming state-of-the-art alternatives. Oh et al. (2015) extend this and feed a small stack of frames to a CNN-based autoencoding architecture to produce action-conditioned frame predictions in ATARI video games. For up to 100 frames, they achieve stable predictions useful for applications in model-based control. Multiple models (Finn et al., 2016; Liu et al., 2017; Mathieu et al., 2016) reframe video prediction as pixel-wise motion prediction, i. e. optical flow. A MSE-based loss does not differentiate well between a model that has learned visually appealing behaviour and one that has not. Likewise, MSE-style losses are incompatible with multiple futures, i. e. multiple plausible future frame predictions given the history of observations. This can lead to blurry or implausible output. To approach these problems Vondrick et al. (2016), Mathieu et al. (2016), and Lee et al. (2018) employ a GAN-style adversarial loss. Similar to our argument for the advantage of including stochasticity in sequence models, many have argued that deterministic models fail to capture the stochasticity inherent to *videos* as well. Therefore, many in the community have shifted towards probabilistic models of videos such as Kalchbrenner et al. (2017), Babaeizadeh et al. (2018), Lee et al. (2018), or Denton and Fergus (2018). Many of these approaches can be set in relation to the non-linear sequence models of Section 2.3. Finally and recently, Kumar et al. (2020) apply normalising flows to video modelling, while Weissenborn et al. (2019) present an autoregressive approach based on self-attention networks.

3.3.2 Object-Aware Approaches

An object-based representation of videos learned in an unsupervised fashion could be hugely beneficial to applications, for example in reinforcement learning and robotics. Reasoning about the effects of actions and understanding the interactions between the objects present is crucial to learning intelligent agents. While such reasoning capability is difficult to obtain from raw pixels directly, object-based video models and their latent representations might provide a compelling inductive bias. This argument for object-aware video models is similar to that made for object-aware image models. For object-aware video models, one can additionally make the case that future frame predictions should be superior to unstructured models in suitable scenarios. If the objects are clearly discernible and move predictably, e. g. determined by physical laws or external actions, it is much easier to understand

and predict the movement of a limited number of objects rather than that of many individual, unstructured pixels. This idea is at the core of object-aware image modelling. Additionally, learning object-based representations from *video* should in fact be simpler than learning them from single *images*. Compared to object-aware image models, the presence of *time* in video has been argued to be a helpful bias for object detection in video models (Kosiorek et al., 2018). After all, if a mass of pixels moves coherently and predictably throughout the frames of the video, it is likely to be part of the same object. Kosiorek et al. (2018) and Hsieh et al. (2018) present two approaches which extend the Attend-Infer-Repeat framework to sequences. Both are highly related work to our approach, and we will compare against them in Chapter 5.

SQAIR. Kosiorek et al. (2018) propose Sequential Attend-Infer-Repeat (SQAIR) to extend AIR to sequences. Much of the complexity in SQAIR is in its aspiration to handle a varying number of objects between different frames of the same video. At each timestep, SQAIR not only has to discover all objects, but it has to decide if any object can be matched with an object from the previous frame, or if it instead belongs to a new object which has just appeared in the video. While AIR uses a single LSTM to attend to the objects of a static image, SQAIR uses a total of three different LSTMs for its inference routine: One updates object states between frames, another models relations between objects, and lastly, a third LSTM is used to discover new objects. This *mesh* of RNNs across time and space leads to $\mathcal{O}(O \cdot T)$ algorithmic complexity. The processes of old-object propagation and new-object discovery are likewise present in the generative model. The authors find that SQAIR is indeed more successful than AIR at discerning partially or heavily occluded objects, demonstrating that video does indeed have a regularising effect on object detection. SQAIR is applied to a simulated dataset of moving handwritten numbers – an animated version of MNIST (LeCun et al., 1998) – and simple but realistic CCTV video footage of walking pedestrians.

DDPAE. Concurrently to SQAIR, Hsieh et al. (2018) introduce the Decompositional Disentangled Predictive Autoencoder (DDPAE). Like SQAIR, DDPAE presents an extension of AIR to videos. Compared to SQAIR, the architecture of DDPAE is conceptually simpler. It does not include a fully generative model of videos and focuses on conditional generation instead. Also, it lacks the complex discovery and propagation networks, instead assuming a constant number of objects with constant appearance per sequence. The constant appearance is inferred from an aggregation RNN which smoothes over all appearance vectors from the individual frames. This mechanism results in good modelling of overlapping objects. For inference, DDPAE also uses a two-dimensional recurrence over objects and time, such that they arrive at the same algorithmic complexity as SQAIR. For future state generation, a RNN-based model is proposed, which the authors claim is able to handle interactions. Here, the DDPAE is again a bit more explicit than SQAIR, as future states are predicted as offsets to previous states. Like the original AIR, the authors demonstrate generalisation to novel numbers of objects. DDPAE is demonstrated on moving MNIST data and bouncing balls.

NEM. A distinctive approach is presented by Greff et al. (2017). Neural Expectation Maximisation (NEM) learns a pixel-wise mixture model, using a *neural* variant of the expectation maximisation (EM) algorithm. Essentially, a neural network predicts

the pixel-wise mixture distribution from latent cluster representations. They unroll and backpropagate through the EM steps. Unlike in AIR, *objects* in NEM do not necessarily correspond to a localised collection of pixels. Its iterative procedure needs multiple steps, about ten, for the clusters to converge, which is undesirable for many applications. Additionally, NEM requires binarised input with added noise to avoid unstable behaviour. Notably, the successor, Relational Neural Expectation Maximisation (R-NEM), employs a relational reasoning module between objects similar to the approaches of Battaglia et al. (2016) and Santoro et al. (2017).

Object-aware sequence models have not been shown to be capable of modelling realistic videos. However, currently, unstructured approaches show only limited performance in this domain as well. Nevertheless, on low-complexity videos with clearly defined objects, object-based approaches regularly outperform unstructured models. This is currently restricted to simulated and visually simple, two- or three-dimensional environments. One can only speculate that some sort of object-inspired structure should also be beneficial in modelling real videos. Therefore, scaling object-aware approaches to such environments presents a fruitful avenue of future work. We will now discuss some recent and tangential developments related to object-aware video models.

3.3.3 Tangential Developments

Approaches such as Finn et al. (2016) and Goel et al. (2018) model the movement between two frames of video in terms of the movement of a fixed number of pixel-wise movement masks. While these approaches are not necessarily motivated by future frame *prediction*, they may still obtain an object-like representation and segmentation, which they demonstrate to be beneficial to reinforcement learning in ATARI video games and robotics. As many approaches to object-oriented video prediction are motivated by such applications, a rising amount of work fuses reinforcement learning models and object-based representations. For example, Watters et al. (2019) use object-representations obtained from a MONet-like vision module for data-efficient model-based reinforcement learning in a simple, sprite-based, two-dimensional simulation.

Since the proposal of our model, a number of new and related approaches have been published. Jiang et al. (2020) parallelise the attention mechanism in SQAIR, such that they are able to demonstrate the successful application of their method to scenes with more than 60 objects. Another recent proposal by Akhundov et al. (2019), assumes a Markovian state space for dynamics modelling, similar to our approach. Kipf et al. (2020) learn an object-oriented latent embedding on sequences without relying on pixel reconstructions using a *contrastive loss* objective, which solely applies to the latent space embeddings. They employ a GNN to model transitions in this latent space. In Fuchs et al. (2019), the authors use GNNs to extend their attention-based tracking algorithm (Kosiorek et al., 2017) to the multi-object setting. Veerapaneni et al. (2019) introduce OP3, making first steps towards generalising a MONet-like image model to sequences with a GNN-based transition model. OP3 is applied for planning in a *discrete* prediction task.

4 Structured Object-Aware Physics Prediction for Video Modelling and Planning

After the previous chapters have introduced the motivation for object-aware video modelling, relevant concepts, and related works, we may finally turn our attention to Structured Object-Aware Physics Prediction for Video Modelling and Planning (STOVE).[1,2] The theoretical formulation of STOVE in this chapter and its experimental evaluation in the following chapter comprise the main contribution of this thesis. STOVE presents a self-supervised approach to learning models of videos of physically interacting objects. Given a sequence of images $\{\mathbf{x}_t\}_{t=1}^T$, we assume the existence of Markovian latent states $\{\mathbf{z}_t\}_{t=1}^T$, which fully explain the observations and, given the right transition function, allow for the prediction of future states and observations. STOVE is obtained as the composition of a Sum-Product Attend-Infer-Repeat (SuPAIR) image model, see Section 3.1.2, and a Graph Neural Network (GNN) for relational physics modelling, predicting transitions in latent space, see Section 3.2.2. The above combination gives a non-linear latent state space model (SSM), see Section 2.3. Unlike in linear state space models, the conditional distributions are not parametrised by linear operations but rather neural networks or deep probabilistic models. To allow for tractable learning, we introduce an amortised variational approximation to the true posterior and optimise all model components jointly via the resulting ELBO. The SuPAIR-like image model interfaces with the observations $\mathbf{x}_t$ in pixel space, inferring latent state distributions $q(\mathbf{z}_t \mid \mathbf{x}_t)$ and evaluating conditional image likelihoods $p(\mathbf{x}_t \mid \mathbf{z}_t)$. The physics model then propagates these states forward in time, realising the transition distribution $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$. At the time of publication, STOVE presented the first extension of AIR image models to relational physical reasoning in videos. While reading this chapter, the reader is referred to the Figs. 4.1 and 4.2 for reference. Figure 4.1 gives a detailed overview of the workings of STOVE, while Fig. 4.2 displays the underlying graphical model and the structure of the latent space. For a first overview of STOVE, let us introduce its defining characteristics: *structure*, *object-awareness*, *physics prediction*, *video modelling*, and *planning*.

Structure. In comparison to other approaches, STOVE exhibits a large amount of predefined structure. For one, the assumption of Markovian latent states in both the generative model and inference routine is decidedly restricting, compared to many of the much more general approaches in Section 2.3 and Chapter 3, such

[1] A PyTorch implementation of STOVE is provided at github.com/jlko/STOVE.

[2] Concurrently to this thesis, a paper on STOVE will be published in the proceedings of the Eighth International Conference of Learning Representations. This thesis and the paper share some of the results and figures presented. While the work of my co-authors has been immeasurably important for the creation of the paper, unless otherwise mentioned, the figures presented and experiments described in this thesis have been created and performed by me.

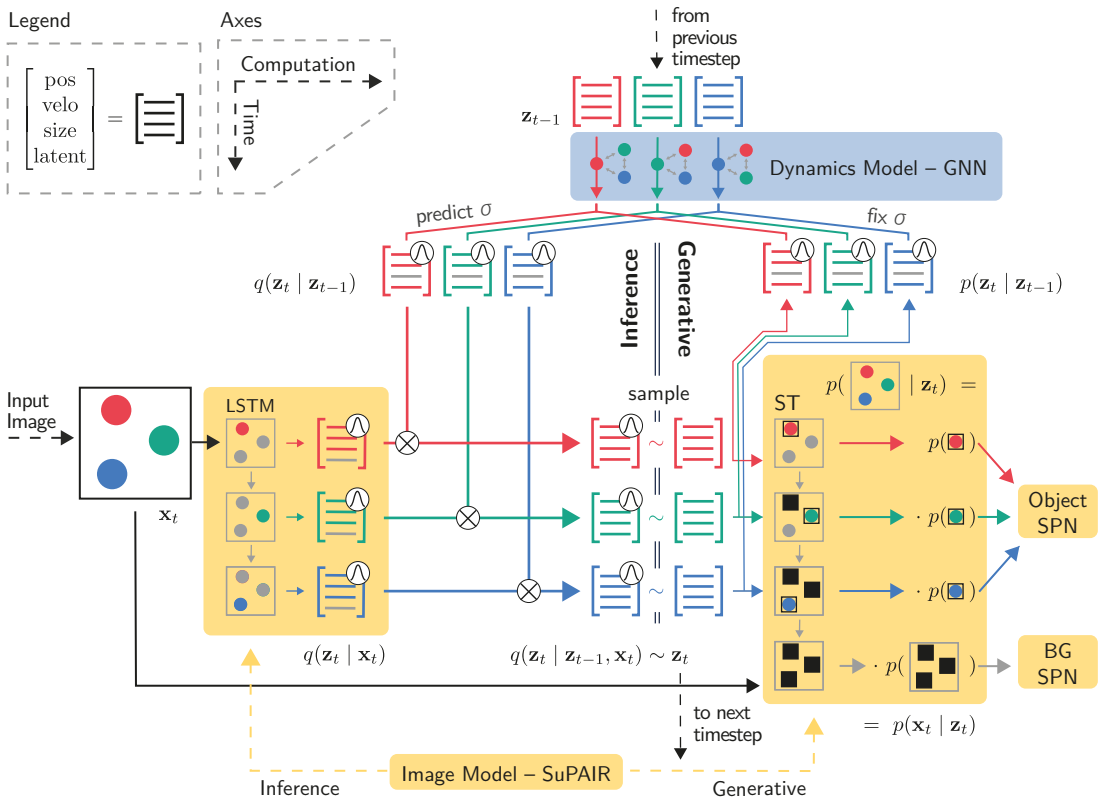


Figure 4.1.: Overview of STOVE's architecture. (Centre left) At time t, the input image $\mathbf{x}_t$ is processed by an LSTM in order to obtain a proposal distribution over object states $q(\mathbf{z}_t \mid \mathbf{x}_t)$. (Top left) A separate proposal $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ is obtained by propagating the previous state $\mathbf{z}_{t-1}$ using the dynamics model. (Centre) The multiplication of both proposal distributions yields the final variational distribution $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t)$. (Right) We sample $\mathbf{z}_t$ from this distribution to evaluate the generative distribution $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})p(\mathbf{x}_t \mid \mathbf{z}_t)$, where $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ shares means – but not variances – with $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$, and $p(\mathbf{x}_t \mid \mathbf{z}_t)$ can be obtained by direct evaluation of $\mathbf{x}_t$ in the sum-product networks, using the spatial transformer operations, *ST*, for the objects. Not shown is the dependence on $\mathbf{x}_{t-1}$ in the inference routine, which allows for the explicit inference of velocities from position differences. Figure and caption also appearing in Kossen et al. (2020).

as VRNN and SQAIR. Additionally, the latent space itself is structured. Competing approaches in video prediction often assume a largely unstructured latent space, usually with a normally-distributed prior. *After* training their models, they then look for *meaning* in the resulting latent space. In contrast to this, we directly specify a meaningful latent space, consisting of per object positions, velocities, and sizes. This further injects prior knowledge into the model, directly benefitting model stability, performance, and interpretability. In our experiments, this structure does not hinder generalisation on the data explored. Instead, the right kind of structure benefits performance and interpretability due to a clear allocation of responsibilities between model components. As the SuPAIR-like image model does *not* rely on latent states to model the object appearances, appearance information does not compete with dynamics information in the latent space. Of course, the assumptions of clearly defined object positions, velocities, and sizes hold in all our experiments and would, depending on the application, have to be revisited. The following paragraph lists a further structural constraint of STOVE.

Object-Awareness. STOVE reasons about the world in terms of objects. As is common for AIR-derivatives, see Section 3.1, images are constructed as the superposition of objects atop a background. An image may be described in terms of object-specific latent states $\mathbf{z}_t = \{\mathbf{z}_t^o\}_{o=1}^O$. A video is a sequence of T images, which likewise has a corresponding sequence of latent states $\mathbf{z} = \{\mathbf{z}_t\}_{t=1}^T$. At each timestep, per object positions, velocities, and sizes are inferred from the given images using the scene model or propagated from previous latent states by the dynamics model. While object-centric modelling of images is a novelty, object-centrality in *physics modelling*, where the goal is to approximate a sequence of object states, is an obvious prerequisite. For videos, only few approaches to object-oriented modelling exist, see Chapter 3.

Physics Prediction. The movement of a single object in a video is rarely independent of its surrounding. Instead, complex interactions between objects affect individual object trajectories. This is true for real-world scenarios such as the movements of a self-driving car, as well as simpler applications such as video games. In physics prediction and unsupervised video modelling, toy data sets of physically interacting objects, such as bouncing or gravitationally attracted balls, are often used to mimic some of this complexity. In such simulations, the trajectories of the individual objects are highly non-linear and dependent on interactions with other objects. In STOVE, the transition of the per-object latent states $p(\mathbf{z}_t^o \mid \mathbf{z}_{t-1})$ is realised by a state-of-the-art physics prediction model, a Graph Neural Network (GNN). GNNs are special relational architectures which can be used to facilitate the modelling of interactive behaviour in physics simulations. At the time of publication and to the best of our knowledge, STOVE is the first AIR-video model capable of generatively modelling complex physical interactions. Our experiments show that models which do not include such a relational component in their transition model fail to appropriately model interactions between objects and exhibit unrealistic trajectory predictions. We suspect that such relational modelling has meaning beyond toy datasets and will likely be important for applications in real-world scenarios.

Video Modelling. A *video* model naturally arises from the components above. The image model allows STOVE to infer latent states from images $q(\mathbf{z}_t \mid \mathbf{x}_t)$ as well as to render images from states $p(\mathbf{x}_t \mid \mathbf{z}_t)$, while the sequence model allows for the propagation of latent states through time $p(\mathbf{z}_t \mid \mathbf{z}_{t+1})$. Therefore, the combination of the model over the images and the physics model over the states yields a model over a sequence of images – a model of videos. The dynamics model and the image model only interface via the object positions and velocities, see discussion below. While STOVE is expressive enough for the scenarios which we investigate, it would need to be adapted for more complex situations where objects may change their visual appearance during the video sequence.

Planning. We propose an extension of STOVE from videos to video *games*, i. e. to the setting of model-based control. For this, we modify STOVE such that it can predict future frames, states, and rewards conditioned on external control inputs, or actions. In other words, STOVE is employed as a world model, a full simulation of the interactive environment. One can then use STOVE in tandem with planning methods, acting in the environment to maximise rewards. The extension of STOVE beyond strict physical simulation is an important step towards obtaining video models for more general scenarios with wider varieties of interactions and dynamics.

By simply going over the main characteristics that, more or less, spell out the letters of STOVE, we have already introduced a significant proportion of what sets STOVE apart from other approaches. We will now introduce the details of the STOVE architecture.

4.1 Sum-Product Attend Infer Repeat for Image Modelling

We use a variation of the Sum-Product Attend-Infer-Repeat (SuPAIR) image model, as introduced in Section 3.1.2 and by Stelzner et al. (2019), to model the distribution of images $\mathbf{x}$ in object-aware fashion. The focus of STOVE is the proposal of a video model capable of modelling complex interactions between objects. We therefore avoid the complexities of dealing with a varying number of objects during the sequence, see SQAIR (Kosiorek et al., 2018) and Section 3.3 for further details. The original SuPAIR architecture is simplified by constraining SuPAIR to a fixed number of objects O. In Section 3.1.2, we have directly proposed this constrained SuPAIR variant. The assumption of constant O simplifies inference, as we do not need to evaluate the expectation with respect to O in the ELBO during training. Extending STOVE to a varying object count, which stays constant over the sequence, similar to Hsieh et al. (2018), would be possible without much effort. Additionally, the original SuPAIR employs an auxiliary loss term, which penalises overlapping bounding boxes between objects. Without it, SuPAIR can predict superfluous objects without being penalised. This is not needed in the O-constrained version of SuPAIR.[3]

Otherwise, there are deliberately few changes from the original SuPAIR architecture. Therefore, the components in STOVE take on clearly designed roles and have few but meaningful interactions between each other. In Section 3.1.2, we have introduced the conditional likelihood model $p(\mathbf{x} \mid \mathbf{z})$, Eq. (3.3), which models the appearance of an image $\mathbf{x}$ using an SPN to assess object likelihoods, an SPN for background likelihoods, and spatial transformer layers. Likewise, we have introduced the recognition model $q(\mathbf{z} \mid \mathbf{x})$, Eq. (3.1), which infers object latent positions and sizes $\mathbf{z}^o_{(\text{pos, size})}$ in recurrent fashion. For STOVE, we use these components without any further changes. However, as we are now considering sequences of images, we endow the variables with an index t, writing $p(\mathbf{x}_t \mid \mathbf{z}_t)$, $q(\mathbf{z}_t \mid \mathbf{x}_t)$, and $\mathbf{z}^o_{t,(\text{pos, size})}$ instead.

As a further inductive bias, we want to introduce velocities $\mathbf{z}^o_{t,\text{velo}}$ to the latent state. While the experiments presented in Section 5.10 show that STOVE can indeed *discover* velocities and does not *need* them a priori, explicit velocities do provide an important bias to improve model training. Also, they are straightforward for us to

[3] In the original SuPAIR, if a superfluously predicted object o' is completely on top of a previous bounding box, all pixels of o' are marginalised over, such that it does not contribute to the ELBO, but leads to a wrong object count. One therefore needs an auxiliary loss term to combat this behaviour. In the O-constrained version, such behaviour may not occur because each bounding box *needs* to cover an object. If a bounding box completely overlaps with a previous box, such that it is completely marginalised over, an object of the scene is missed by the attention model. This object, which is necessarily not covered by any bounding box, then causes a loss because it must be part of the background, and the background SPN is likely to assign a low background likelihood.

implement from position differences as

$$q(\mathbf{z}^o_{t,\text{velo}} \mid \mathbf{x}_t, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{z}^o_{t,\text{velo}}; \boldsymbol{\mu}^o_{t,\,\text{pos},\,i} - \boldsymbol{\mu}^o_{t-1,\,\text{pos},\,i}, \\ (\boldsymbol{\sigma}^2)^o_{t,\,\text{pos},\,i} + (\boldsymbol{\sigma}^2)^o_{t-1,\,\text{pos},\,i}) \,, \tag{4.1}$$

where the uncertainty results from the additive combination of the two independent Gaussian position variables in a principled probabilistic way. The subscript i refers to the fact that the mean and variance proposals for the latent state are inferred by the *i*mage and not the *d*ynamics model.

A few hyperparameters guide the predictions of SuPAIR. A possible failure mode of SuPAIR is that the background model may learn to model full scenes on its own, disregarding the object mechanism. To avoid this behaviour, Stelzner et al. (2019) have found it helpful to bias the structure of the background SPN towards less complex appearances. Another possible failure is the prediction of a single image-sized object or the prediction of zero-sized objects. To avoid such problems, the size of objects are lower- and upper-bounded to be $10\,\%$ and $90\,\%$ of the total image size. See Appendix A.4 for an exhaustive list of hyperparameters and default values.

In addition to training speed, there is an unexpected advantage of using SuPAIR over AIR in the context of physics prediction. We require the obtained object positions from the attention network to be as precise as possible. For this, it is essential that the object is always well-centred within the bounding box. In SuPAIR, we can observe this property and notice a decrease in likelihood for off-centre bounding boxes. In other AIR-like models, this is not necessarily the case. Due to the un-structure of the latent space, AIR may learn latent states which encode the position of the object *within* the bounding box, yielding accurate reconstructions even for off-centre boxes. While additional information in the latent state will incur a loss in the KL divergence, such behaviour is still observed in AIR models. Because SuPAIR avoids the unstructured appearance latent, it *cannot* encode object positions within the bounding box and consequently learns to predict well-centred bounding boxes.

As SuPAIR is already a probabilistic model framed in terms of amortised variational inference, its integration into the STOVE architecture is almost effortless, see Section 4.3.

4.2 Graph Neural Networks for Physics Modelling

In Section 3.2.2, relational physics modelling was presented from the perspective of supervised machine learning, where learning is usually accomplished by minimising a squared loss between predicted and observed trajectories from the training set. In STOVE, we need to embed such a physics model into the probabilistic state space model. Instead of directly predicting future states $\mathbf{z}_{t+1}$, the physics model needs to predict transition *probabilities* $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$. A simple approach to accomplish this is to use the predictions from the physics model as means of a factorised Gaussian with

fixed standard deviations

$$\tilde{\boldsymbol{\mu}}^{o}_{t+1,\,(\text{pos, velo, latent})} = A\left(N(\mathbf{z}^{o}_{t}) + \sum_{o' \neq o} \alpha(\mathbf{z}^{o}_{t}, \mathbf{z}^{o'}_{t})\, I(\mathbf{z}^{o}_{t}, \mathbf{z}^{o'}_{t})\right) \tag{4.2}$$

$$\boldsymbol{\mu}^{o}_{t+1,\,(\text{pos, velo, latent})} = U\left(\tilde{\boldsymbol{\mu}}^{o}_{t+1,\,(\text{pos, velo, latent})},\, \mathbf{z}^{o}_{t}\right) . \tag{4.3}$$

We have introduced the governing ideas behind relational physics modelling already in Section 3.2.2. The prediction of the future latent state is given by a concatenation of functions, Eqs. (4.2) and (4.3), which is applied to each object independently. For each object, the prediction is given by a function $N(\mathbf{z}^{o}_{t})$ applied to the current focus node, modelling object movement independent of interactions, plus a sum over its interactions $I(\mathbf{z}^{o}_{t}, \mathbf{z}^{o'}_{t})$ with neighbouring objects o'. The function $A(\cdot)$ is used to aggregate over these terms. Similarly, $U(\cdot)$ represents aggregation in addition to a skip connection to the input state and, for reasons that will become obvious in Section 4.4, we introduce the intermediate results $\tilde{\boldsymbol{\mu}}^{o}_{t+1,\,(\text{pos, velo, latent})}$. The interaction function is reused over all objects pairs. The different functions are parametrised by MLPs. In addition to the familiar functions mentioned above, we introduce an attention term $\alpha(\mathbf{z}^{o}_{t}, \mathbf{z}^{o\prime}_{t}) > 0$. Like the interaction module, it is applied to each pair of focus object o and neighbouring object o' and shared over predictions for all objects $o \in (1, \ldots, O)$. The attention term strictly increases the modelling flexibility of STOVE. In simulation settings where not all possible interactions between objects might be relevant at all times, $\alpha(\mathbf{z}^{o}_{t}, \mathbf{z}^{o\prime}_{t})$ presents a learned version of the hard and prespecified mechanism employed by NPE in Eq. (3.4). This allows STOVE to model scenarios with discrete interactions, such as collisions, in a similar fashion as NPE, where intuitively, the attention term can switch these interactions on and off, while the interaction term models their outcome. However, STOVE can also accommodate scenarios where a hard attention mechanism is not appropriate, combining the advantages of VIN and NPE. A similar attention mechanism is used by R-NEM, introduced in Section 3.3. Unlike VIN, we find that a single prediction core suffices for convincing performance.

Instead of predicting new states directly, NPE predicts velocity differences and uses those to update states, see Section 3.3. Similarly, we bias STOVE towards predicting new positions as offsets to old ones. This inductive bias is useful for scenarios with smoothly moving objects. However, this offset is not given by $\mathbf{z}_{t,\text{velo}}$, as our velocity estimate at time t relies on past information from $t-1$ and is therefore inaccurate during collisions, i. e. at the time of collision, the true, instantaneous velocity changes are not yet reflected in our observational velocity. Instead, the offset to the last position is predicted directly by the GNN. Otherwise, much of the structure of the MLPs of the dynamics GNN is straightforward. They consist of a couple of layers of fully connected neural networks, with skip connections and ReLU non-linearities. Equation (4.3) is implemented efficiently and computed for all objects in parallel using tensorised code. This is true also for the interactions, which are computed in parallel using stacking and replicating operations. Appendix A.2 gives the full details.

So far, we have introduced the object position, velocity, and size as latent space components. For physics modelling, we extend the latent space by unstructured

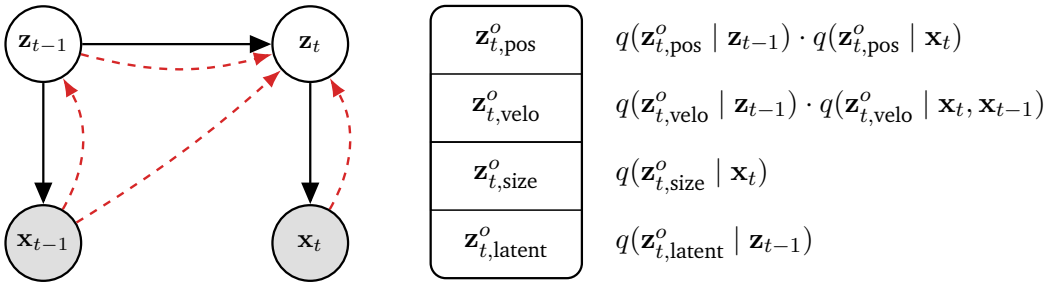


Figure 4.2.: (a) Depiction of the graphical model underlying STOVE. Black arrows denote the generative mechanism and red arrows the inference procedure. The variational distribution $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1})$ is formed by combining predictions from the dynamics model $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ and the object detection network $q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1})$. (b) Components of the per-object latent state $\mathbf{z}^o_t$ and corresponding variational distributions. Note that the velocities are estimated based on the change in positions between timesteps, inducing a dependency on $\mathbf{x}_{t-1}$. Figures adapted from the original graphics by Karl Stelzner. Figures and caption first produced for Kossen et al. (2020).

latent space $\mathbf{z}^o_{t,\text{latent}}$. In VAE fashion, this latent space is regularised towards

$$\mathbf{z}^o_{1,\text{latent}} \sim p(\mathbf{z}^o_{1,\text{latent}}) = \mathcal{N}(\mathbf{z}^o_{1,\text{latent}}; \mathbf{0}, \mathbf{1}) \,. \tag{4.4}$$

In essence, the motivation for this latent space is to give the GNN extra modelling capabilities: Additional information about the object states not included in position, approximate velocity, or size can be stored in the latent space by the GNN. For example, the latent state space may be used to account for different dynamical behaviour between objects. This may be needed because the *same* function Eq. (4.3) is used to predict the dynamics of *all* objects. Therefore, without the latent space, if two objects have the same position, velocity, and size, the GNN *must* predict the same future state distribution. It cannot account for differences between objects that are not present in the latent state. With the introduction of unstructured latent space, such differences can now be reflected, which then allows the GNN to account for them as well. Experiments have indicated that the unstructured latent is indeed essential for modelling performance. In addition, preliminary experiments have shown that the unstructured latents may contain higher order derivatives, i. e. accelerations. The following reflections indicate that this may indeed be necessary: We have already mentioned issues with the observational velocity during collisions. Because we observe a discretised sequence of images, the position differences observed during collisions usually do not correspond to the true, instantaneous object velocities. It follows that the original latent space of positions and observational velocities is, in a first order Markov model, not enough to accurately model the dynamics because we cannot predict the outcome of collisions. The unstructured latent space allows to account for that. See Section 5.10 for further discussion. Even in the worst case, if the model does not need the latent space in some scenario, it can easily be ignored and should not negatively affect performance.

4.3 Joint State Space Model

Above, we have established the two defining components of STOVE, the Sum-Product Attend-Infer-Repeat (SuPAIR) image module and the Graph Neural Network (GNN) relational physics module. We now identify them with the corresponding distributions in our non-linear state space model, see Fig. 4.2.

For the generative model, we decompose the joint likelihood over observations and latent states as

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p(\mathbf{x}_1 \mid \mathbf{z}_1)p(\mathbf{z}_1) \prod_{t=2}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t)\, p(\mathbf{z}_t \mid \mathbf{z}_{t-1})\,, \tag{4.5}$$

using the chain rule of probability and the Markov property of the latent states. Here, $p(\mathbf{x}_t \mid \mathbf{z}_t)$ is given by the generative model of SuPAIR, Eq. (3.3), and $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ is as described by Eq. (4.3). The prior $p(\mathbf{z}_0)$ is assembled from the uniform priors over object positions and sizes of SuPAIR and the normal prior over the unstructured latents of the dynamics model.

As inference in the non-linear state space model is intractable, we formulate a recognition model for amortised inference of $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})$, see Section 2.2. At time t, we can use SuPAIR's inference routine to infer latent states $q(\mathbf{z} \mid \mathbf{x}_t, \mathbf{x}_{t-1})$.[4] Such an approach is common in non-linear state space models, see the detailed discussion in Section 2.3. However, in doing so, we would disregard all information from the dynamics model. Instead, we can construct a joint estimate of the current latent state position and velocity using information from both the image and the dynamics model, by introducing a variational approximation to the dynamics model $q(\mathbf{z}_{t,\,(\text{pos, velo})} \mid \mathbf{z}_{t-1})$. A naïve approach to do this would be to construct a second dynamics model with identical architecture but separate parameters. Instead, we efficiently *reuse* the generative GNN for inference, i. e. the means of $q(\mathbf{z}_{t,\,(\text{pos, velo})} \mid \mathbf{z}_{t-1})$ and $p(\mathbf{z}_{t,\,(\text{pos, velo})} \mid \mathbf{z}_{t-1})$ are identical and as given by Eq. (4.3). The benefits of this reuse are discussed below. The joint estimate from both models is given as

$$\begin{aligned}
&q(\mathbf{z}_{t,\,\text{pos}} \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{z}_{t-1}) \\
&\qquad \propto q(\mathbf{z}_{t,\,\text{pos}} \mid \mathbf{x}_t, \mathbf{x}_{t-1}) \cdot q(\mathbf{z}_{t,\,\text{pos}} \mid \mathbf{z}_{t-1}) && (4.6)\\
&\qquad = \mathcal{N}(\mathbf{z}_{t,\,\text{pos}};\, \boldsymbol{\mu}_{t,\,\text{pos},\,i},\, \boldsymbol{\sigma}^2_{t,\,\text{pos},\,i}) \cdot \mathcal{N}(\mathbf{z}_{t,\,\text{pos}};\, \boldsymbol{\mu}_{t,\,\text{pos},\,d},\, \boldsymbol{\sigma}^2_{t,\,\text{pos},d}) \\
&\qquad \propto \mathcal{N}(\mathbf{z}_{t,\,\text{pos}}; \boldsymbol{\mu}_{t,\,\text{pos}},\, \boldsymbol{\sigma}^2_{t,\,\text{pos}}) && (4.7)\\
&\boldsymbol{\mu}_{t,\,\text{pos}} = \frac{\boldsymbol{\sigma}^2_{t,\,\text{pos},\,d}\; \boldsymbol{\mu}_{t,\,\text{pos},\,i} + \boldsymbol{\sigma}^2_{t,\,\text{pos},\,i}\; \boldsymbol{\mu}_{t,\,\text{pos},\,d}}{\boldsymbol{\sigma}^2_{t,\,\text{pos},\,d} + \boldsymbol{\sigma}^2_{t,\,\text{pos},\,i}} && (4.8)\\
&\frac{1}{\boldsymbol{\sigma}^2_{t,\,\text{pos}}} = \frac{1}{\boldsymbol{\sigma}^2_{t,\,\text{pos},\,d}} + \frac{1}{\boldsymbol{\sigma}^2_{t,\,\text{pos},\,i}}\,, && (4.9)
\end{aligned}$$

where the subscripts i and d denote the respective parameters from the *i*mage and *d*ynamics model, and we proceed the same way for $q(\mathbf{z}_{t,\,\text{velo}} \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{z}_{t-1})$.[5]

[4] Dependence on $\mathbf{x}_{t-1}$ serves as a reminder that we use the position distribution previously predicted by SuPAIR to assemble an estimate of the velocity, see Eq. (4.1). As $\mathbf{x}_{t-1}$ is not used any further, e. g. for filtering estimates about the current position or size, we might sometimes omit the dependency to avoid confusion.

[5] In principle, one could also obtain estimates of $\mathbf{z}_{t,\text{size}}$ in identical fashion. However, as this seemed to impact model training negatively, we rely on SuPAIR only for the estimate of the object sizes.

Because both models output factorised Gaussian distributions, their product is again proportional Gaussian with means and variances as given by Eqs. (4.8) and (4.9). Equation (4.6) presents a principled probabilistic way to combine uncertain Gaussian information from two sources, similar to the measurement step in a Kalman filter (Murphy, 2012, p. 641). The predicted state mean in Eq. (4.8) is weighted by the uncertainty in the two source distributions, naturally preferring certain information with small variances over uncertain information. Uncertainty in one measurement increases the overall uncertainty. For the generative dynamics model $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$, means are predicted by Eq. (4.3) and variances fixed. To allow the dynamics model to make use of the probabilistic combination, we predict variances for $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ as additional outputs of the GNN, Eq. (4.3).

This approach is inspired by Becker-Ehmck et al. (2019) and Karl et al. (2017b), who have, however, used it in single-object settings only. In addition to modelling efficiency and more precise estimates of $\mathbf{z}_t$, they give another justification for the reuse of the generative dynamics model for inference: The dependency of $\mathbf{z}_t$ on the dynamics model leads to a favourable backpropagation of the image model through the dynamics model. In other words, because the quality of the samples $\mathbf{z}_t$ now *depends* on the dynamics model, the dynamics model becomes *crucial* for obtaining a well-working model. If the dynamics model provides bad or uncertain estimates, samples are bad, and learning is significantly hindered for *all* model components, including the image model. Without this mechanism, a bad dynamics model would lead to a penalty in the transition prior, Eq. (4.5), but the image model could still be trained well, as samples would be unaffected. Furthermore, this mechanism is an algorithmically efficient way to couple inference for dynamics and image models without introducing computational burden, as SuPAIR inference may still be performed for all images in parallel. SQAIR and DDPAE instead introduce a costly double recurrence, with complexity $\mathcal{O}(O \cdot T)$ over objects and time to form estimates $\mathbf{z}_t$, see Section 3.3.2. STOVE instead requires O inference operations for all images in parallel, followed by T prediction steps for the dynamics model for all objects in parallel, keeping the total complexity to only $\mathcal{O}(O + T)$.

For the sequence of observations $\mathbf{x}_{1:T}$ and corresponding latent states $\mathbf{z}_{1:T}$, we decompose the recognition distribution as

$$q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) = q(\mathbf{z}_1 \mid \mathbf{x}_1) \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{z}_{t-1}) \,, \tag{4.10}$$

where $q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{z}_{t-1})$ is given by Eq. (4.6). Note that similar to the discussion in Section 2.3, we would also be allowed to condition on future observations in the recognition network. However, similar to Krishnan et al. (2017), we have found the above formulation to be sufficient. The initialisation of the model is given by $q(\mathbf{z}_1 \mid \mathbf{x}_1)$, which is obtained from the image model only, see Appendix A.3 for full details of model initialisation.

We train the generative model and the recognition model jointly by maximising the

ELBO, Eq. (2.10), which for a sequence of images $\mathbf{x}_{1:T}$ is given by

$$\begin{aligned}\log p(\mathbf{x}_{1:T}) &\geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left\{\log \frac{p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})}\right\} \\ &= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\Bigg\{\log\left\{\frac{p(\mathbf{x}_1 \mid \mathbf{z}_1)\, p(\mathbf{z}_1)}{q(\mathbf{z}_1 \mid \mathbf{x}_1)}\right\} \\ &\qquad + \sum_{t=2}^{T} \log\left\{\frac{p(\mathbf{x}_t \mid \mathbf{z}_t)\, p(\mathbf{z}_t \mid \mathbf{z}_{t-1})}{q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{z}_{t-1})}\right\}\Bigg\}, \end{aligned} \tag{4.11}$$

where we approximate expectations $\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\{\cdot\}$ with a single chain of samples $\mathbf{z}_{1:T}$. In practice, STOVE is trained using mini-batch gradient descent, i. e. gradients from Eq. (4.11) are accumulated over multiple sequences $\{(\mathbf{x}_{1:T})_n\}_{n=1}^{N_b}$ before parameter updates are applied. We use the reparametrisation trick (Kingma and Welling, 2014; Rezende et al., 2014), see Section 2.2, to correctly evaluate the gradient with respect to the expectation over $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T})$. Equation (4.11) is not modified with ad-hoc auxiliary losses, as are common practice in supervised physics models, and all model components are trained from scratch without pretraining.

The attention LSTM of STOVE's image model is not guaranteed to preserve the object order between individual timesteps. We therefore face the challenge of matching the order of the object proposals between dynamics and image model and timesteps. A simple but effective solution to this is to match states based on the Euclidean distance $\|\cdot\|$ between their positions. More precisely, we first ensure a consistent object order in the recognition distributions $q(\mathbf{z}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{z}_t;\, \boldsymbol{\mu}_{t,i}, \boldsymbol{\sigma}^2_{t,i})$ from the image model i by matching objects between timesteps such that $\|\boldsymbol{\mu}^o_{t,\text{pos},i} - \boldsymbol{\mu}^{o'}_{t-1,\text{pos},i}\|$ is minimised. Because the dynamics model is initialised from states inferred with the attention LSTM, see Appendix A.3, we now have consistently sorted states over time between the dynamics and object model. A naïve implementation of this sorting problem has complexity $\mathcal{O}(O!)$. However, using the Hungarian algorithm (Kuhn, 1955), the assignment problem can be solved exactly in polynomial time $\mathcal{O}(O^3)$. For small $O \leq 3$, the naïve implementation suffices. For large O, a greedy approximation to the exact solution can be obtained in $\mathcal{O}(O^2)$, which may however lead to adverse effects on model training. Even for an exact implementation of matching, the solutions obtained are correct only if objects move sufficiently little between timesteps, which should, however, be a fair assumption for any reasonable discretisation of continuous physics simulations. Note that many supervised approaches such as VIN (Watters et al., 2017) circumvent the need for matching by assigning the objects a fixed order based on their, presumed to be distinctive, visual appearance. This is not required for STOVE, although we will investigate appearance-based matching, in addition to the position distances, see Section 4.4. Section 6.1 discusses ways of modifying the model structure to omit the need for matching.

As part of our attempt to fully disentangle dynamics and visuals, STOVE, like e. g. DDPAE (Hsieh et al., 2018), cannot model changes in object appearance over time. During rollouts, approximate MPE in the SPNs is used to render novel images (Vergari et al., 2018). Karl et al. (2017a) have argued that a latent space which competes between modelling visuals and dynamics is a significant cause for flawed dynamics performance. STOVE does not struggle with this per construction, as the latent space does not model object appearances. Extensions to STOVE that do

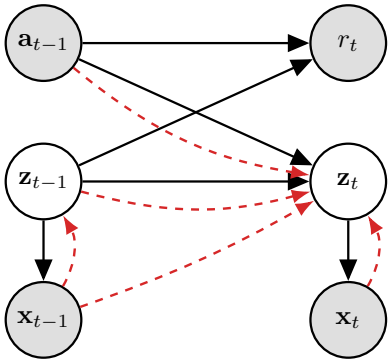


Figure 4.3.: (a) Depiction of the graphical model underlying the action-conditioned variant of STOVE. State predictions of the dynamics model $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ are now conditioned on actions $\mathbf{a}_{t-1}$ of the agent. The dynamics model is extended to predict a distribution over the reward $p(r_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$. Black arrows denote the generative mechanism and red arrows the inference procedure. The variational distribution $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1})$ is formed by combining predictions from the dynamics model $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ and the object detection network $q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1})$. Figure adapted from original graphic by Karl Stelzner. Figure and caption first produced for Kossen et al. (2020).

consider dynamically changing visuals are left for future work, although first steps have already been taken by Voelcker (2020).

4.4 Conditioning on Actions

In reinforcement learning, an agent executes actions $\mathbf{a}_t$ to maximise an expectation over future rewards r in an environment with observations $\mathbf{x}_t$. Recently, much attention has been given to *model-based* approaches to reinforcement learning in visually complex domains, see e. g. Schrittwieser et al. (2019), Kaiser et al. (2020), Watters et al. (2019), or Veerapaneni et al. (2019). For such approaches, a sophisticated simulator of the environment, or other relevant quantities, is required. After such a simulator has been learned, it can, for example, be used for *planning*: The simulator is used to predict the cumulative rewards of many different possible sequences of actions, after which the most promising action is executed in the real environment. This is in contrast to *model-free* approaches, which learn to act directly in the environment. The acquisition of observation samples $\mathbf{x}_t$ is often expensive in real-world settings. A main motivation for model-based approaches is their presumed sample efficiency, see the citations above for support of this.

An extension of STOVE allows its use as a world model in model-based control.[6] Figure 4.3 reflects the changes to the graphical model of STOVE for action-conditioning.

[6]Model-based reinforcement learning remains a footnote in this thesis. The extension of STOVE as an action-conditioned world-model was done in collaboration between all co-authors. The experiments on STOVE's predictive capabilities presented in this thesis were performed by me. Kossen et al. (2020) contains additional experiments, where Monte-Carlo Tree Search (MCTS) (Coulom, 2006) was used for planning with STOVE as a world-model and its performance compared to suitable baselines. Those experiments were performed by Claas Voelcker and Marcel Hussing, and they are not presented in this thesis. Further experiments with STOVE in the context of model-based reinforcement learning may be found in Voelcker (2020) and Hussing (2020). For the scope of this thesis, we will only evaluate STOVE's capabilities as a generative action-conditioned video model, isolated from applications in reinforcement learning.

STOVE's dynamics prediction is extended to consider actions $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$, and likewise for the variational approximation $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$. The dynamics model predicts states *per object*, Eq. (4.3), but we want STOVE to make no a priori assumptions about which objects are affected by actions. Therefore, actions are concatenated to the state representations of *all* objects in Eq. (4.3). This allows the GNN to infer when actions are relevant for accurate prediction. Additionally, the dynamics model is extended to provide predictions of rewards $p(r_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ via

$$\boldsymbol{\mu}_{t+1,\,\text{reward}} = R_2 \left(\sum_{o=1}^{O} R_1 \left(\tilde{\boldsymbol{\mu}}^o_{t+1,\,(\text{pos, velo, latent})} \right) \right) . \tag{4.12}$$

In other words, we reuse intermediate results $\tilde{\boldsymbol{\mu}}^o_{t+1,\,(\text{pos, velo, latent})}$ from the dynamics GNN, Eq. (4.2). The agent receives a reward per observation, not per observation and object. We therefore need to transform the per-object predictions of the GNN to a single reward prediction per observation. A simple and order-invariant mechanism to achieve this is the summation over all per-object terms. For added modelling flexibility, R_1 transforms the intermediate prediction before and R_2 after summation, see Appendix A.2 for details. It may seem like an arbitrary decision to reuse parts of the dynamics model for reward prediction. However, similar to the prediction of object dynamics, the observed reward is likely to depend on interactions between objects. It is therefore plausible that an intermediate representation $\tilde{\boldsymbol{\mu}}^o_{t+1,\,(\text{pos, velo, latent})}$ presents a suitable starting point for both reward and dynamics prediction. The weight of the reward loss in the total ELBO is linearly increased during training until a maximum weight is reached. This corresponds to a decrease in the variance of the distribution $p(r_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$.

We stress that this extension of STOVE to action-conditioned video prediction presents a departure from its original use for modelling of videos of *physically* interacting objects. In physics simulation, all objects are subject to the same laws of motion. In reinforcement learning environments, interactions between objects are more general. Even for the simple application of action-conditioned video prediction presented in Chapter 5, actions discretely affect the motion of a single object, upsetting the balance of physical simulations. This presents a challenge to the dynamics GNN, which models the motion of each object individually using the *same* function. To lighten the burden on the GNN, we extend the object-representation $\mathbf{z}^o_{t,\,(\text{pos, velo, latent})}$ in the dynamics model with visual input $\mathbf{v}^o_t$, such that the GNN may directly discern if objects have different visual appearances. This latent appearance encoding may be obtained from the representation encoded in the object SPN of SuPAIR, see Vergari et al. (2018). Alternatively, a VAE, possibly with convolutional layers, may be trained on the object glimpses to extract a latent encoding, as done in AIR. We opt for a simple alternative and construct $\mathbf{v}^o_t$ from simple summary statistics: the mean pixel value per colour channel of the object glimpse defined by the bounding box.

5 Evaluation and Discussion

This chapter contains a thorough experimental evaluation of STOVE. As introduced in Chapter 4, STOVE's primary capabilities are the inference of object states from a sequence of observed images, the reconstruction of a sequence of observed images, the prediction of future object state trajectories, and the prediction of future video frames conditioned on a sequence of observed images. The prediction of future object positions or video frames is a much more demanding task than reconstructing or inferring from observations, as it much more relies on an adequate dynamics model in addition to the image model. Therefore, the centrepiece of this evaluation will be a comparison of STOVE's predictive performance to previously proposed unsupervised video prediction models, upon which STOVE can improve.[1] We evaluate STOVE on three tasks. Two of those tasks, *billiards* and *gravity*, are common benchmarks for evaluating the capabilities of physics models. Additionally, we extend STOVE to consider action-conditioned state transitions, evaluating STOVE's capabilities as a world model for planning. We further show that STOVE is able to conserve the average kinetic energy in the billiards scenario. This leads to visually convincing rollouts for extremely long timeframes, which have not been demonstrated by previous approaches. Also, experimental considerations suggest that STOVE performs optimally given the chaoticity of the billiards environment. We come to relevant conclusions concerning the general applicability of the billiards environment as a benchmark for physics models. A discussion of the effect of observation noise on the supervised baseline is followed by a qualitative illustration of the stability of the predictions, STOVE's performance on scenarios with different object visuals, an ablation study, and some considerations with respect to STOVE's scaling behaviour. The environments are fully introduced in the following section.

5.1 Introducing the Environments[2]

Figure 5.1 shows frames of videos from all environments.

Billiards Environment. The *billiards* environment consists of multiple objects moving on a bounded, quadratic, (two-dimensional) plane. The objects are circular and, going with the metaphor of billiards, will be referred to as *balls*. Unless a collision occurs, balls move in straight lines with constant velocities, since no frictional forces are considered. If the distance between two balls is less than their combined radius, elastic collisions occur, preserving the total momentum and kinetic energy of the system. The boundaries of the canvas are modelled as walls. If the outer radius

[1] The repository at github.com/jlko/STOVE contains animated versions of the static sequences displayed here. The reader is strongly encouraged to watch them, as they much better convey a visual impression of STOVE's qualitative performance.

[2] Our implementation of the environments is released as part of the official STOVE repository and builds on the original code by Sutskever et al. (2009). I want to thank my co-authors Karl Stelzner, Claas Voelcker, and Marcel Hussing for contributing valuable code to the environment's codebase.

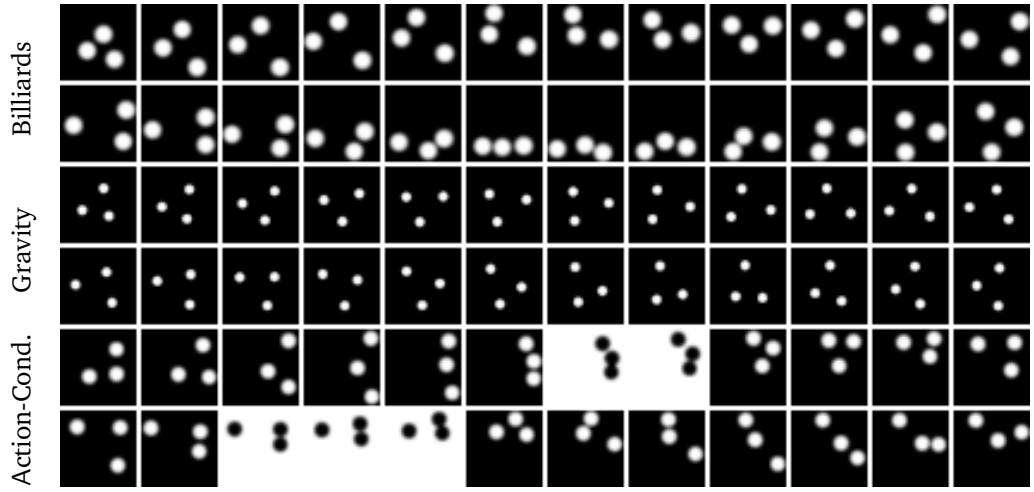


Figure 5.1.: Exemplary future frame predictions from STOVE for the relevant environments. The first two rows show consecutive frames from the billiards environment, followed by frames from the gravity simulations, and the action-conditioned setting. To save space, the above frames show 1 out of every 3 frames in the simulation and 8 of the above frames correspond to 1 second of video. See github.com/jlko/STOVE for animated versions. For the action-conditioned settings, we artificially invert the pixel values, whenever STOVE predicts a reward of -1.

of a ball reaches one those boundaries, it is reflected without losing momentum or energy. This can be seen as an elastic collision where the boundary is modelled with infinite mass and zero velocity. Objects are initialised with random positions drawn from a uniform distribution over the area of the billiards table. If object positions should overlap, the initialisation is invalid, and new positions are drawn until a valid initialisation is produced. Object velocity components are initialised as draws from a zero-mean normal distribution. Velocities are normalised so that the total kinetic energy of the system is constant across all sequences, as is standard in the literature, see e. g. Sutskever et al. (2009). Average velocities are such that the entire length of the canvas may be crossed in about 20 simulation steps. Unless otherwise mentioned, the billiards environment is rendered at a resolution of 32 by 32 pixels, with ball radii of about 8 pixels. While this environment can be seen as the top view of a quadratic billiards table without pockets and friction, it really does not bear any resemblance to the game of billiards. This may be why it is sometimes also given the more descriptive, albeit somewhat silly, name of *bouncing balls*.

Gravity Environment. The gravity environment consists of multiple, gravitationally interacting objects on a (two-dimensional) plane. Objects are modelled as point masses but rendered with finite and circular size. They interact gravitationally, i. e. any pair of objects is subject to a force proportional to the inverse square of their distance and proportional to the product of their masses. No hard object-object or object-wall collisions are modelled. Compared to the billiards environment, reliably obtaining stable sequences of gravitationally interacting objects requires a fairly large amount of additional modifications to the system. In line with Watters et al. (2017), we apply a small additional parabolic force towards the centre to prevent objects from drifting apart. The simulation is also transformed such that the centre of mass of the system always corresponds to the centre of the frame to

account for object configurations whose centre of mass drifts. As gravitational forces diverge for object distances tending to zero, great accelerations may occur during the simulation. These accelerations are too large to be stabilised using the parabolic attraction towards the centre. We therefore additionally clip the gravitational force to dampen the impact of such events. We allow objects to partially leave the frame. However, sequences where objects fully leave the frame or occlude each other are dropped from the data sets. The gravity environment is highly sensitive to the initialisation of object positions and velocities. This complicates the search for a broad range of parameters to which we can randomly and safely initialise the system. After extensive experimentation, we opted for a random initialisation of positions, avoiding points too close to the centre, the boundary of the frame, or to other objects. Velocities are initialised as orthogonal to the object's position vector measured from the centre of the frame and corrupted by component-wise random noise. This initialises the objects on a rotating path around the centre of the frame. Unless otherwise mentioned, the gravity environment is rendered at a resolution of 50 by 50 pixels, with ball radii of about 7 pixels.

Action-Conditioned Environment. The billiards scenario is extended for application to model-based control. One of the balls is now actively steered by the actions of an agent; the others continue to move according to the physics of the billiards environment. When one of the two passive balls collides with a ball controlled by the agent, the active ball is simulated as having infinite mass and zero velocity, and their collision is identical to that of a physical ball and a wall: The *passive ball* is elastically reflected, leaving its absolute momentum and energy unchanged, while the *active ball*'s state is unchanged. The agent may choose between 9 discrete actions: Moving the ball in one of 4 cardinal directions (up, down, left, right), one of the 4 diagonals, or not moving the ball at all. Excluding the non-action, the actions result in an instantaneous movement of the active ball in that direction. Unlike a physical force, the action *does not* additively modify previous velocity values. Instead, an action sets the velocity of the active ball to a fixed value, resulting in a direct movement of the ball in that direction for one timestep only. Therefore, the active ball does not exhibit constant velocity motion. In other words, to move it across the screen left-to-right, the "right" action has to be chosen not once but on each frame. This scheme was chosen because it proved to be the most natural and easiest to play from the perspective of a human player. All 8 movements are normalised to equal length. A *game* is obtained by the construction of the following reward function: Negative reward of -1 is given when the active ball collides with one of the passive balls. If no collisions occur, a reward of 0 is returned. The goal of the game is therefore to steer the active ball such that collisions with the passive balls are avoided. The size of the movements are such that the active ball may cross the entire length of the canvas with about 10 actions. Successful multi-step planning is needed to avoid collisions. A random action selection policy is used to generate data for training STOVE in this setting. At time $t = 0$, a random action a_0 is chosen uniformly from the set of available actions. At time t, action $a_t = a_{t-1}$ is chosen with probability $1 - \alpha$, and with probability α, a random action is chosen uniformly from the set of all actions.

The environments support a varying numbers of objects, as well as different masses or sizes between objects of the same sequence. Unless otherwise mentioned, experiments are performed with three identically sized, circular objects of equal mass. The environment also offers the option to render objects from a set of prespecified sprites. While the environment supports colour images, unless otherwise mentioned,

greyscale-transformed images were used as input to the models. This results in images, for which all balls are white, while the background is black. The balls are rendered using a Gaussian kernel smoother for anti-aliasing, which results in smooth-looking objects at low resolutions. Coloured images are usually for illustrative purposes only, although we do use colour input for the action-conditioned scenario. While the laws governing the billiards and gravity environments are straightforward to describe, their emulation using a function approximator, e. g. a neural network, is not trivial, as the objects' overall behaviour is highly non-linear and chaotic.

5.2 Training Details

For all video modelling tasks, 1000 test and 300 training sequences with 100 frames each are generated. A random subset of 8 frames is used to train STOVE via the sequence ELBO, Eq. (4.11). Unless otherwise stated, STOVE was trained for a total of $83\,000$ steps with a batch size of 256. We used the ADAM optimizer (Kingma and Ba, 2015) with a learning rate schedule of $2 \times 10^{-3} \exp(-4 \times 10^{-2} \cdot \text{step})$. STOVE is implemented using the PyTorch library (Paszke et al., 2019). Training of a default STOVE model on a single desktop machine with an AMD Ryzen 7 3800X processor and a Nvidia Geforce RTX 2080 Ti converges in about 10 hours. See Appendix A.4 for an exhaustive list of hyperparameters and their settings.

5.3 Video and State Modelling

We compare the rollout performance of STOVE to relevant and recently published baselines. As *rollout*, we define the prediction of future frames of video or object states conditional on a sequence of observed images. Here, the models observe and reconstruct 8 frames of video from the test set. Conditioned on those, a prediction for the following 92 frames is made and compared to the ground truth. For models which are able to extract object positions, we obtain a position prediction error as the mean Euclidean distance between the predicted and the ground truth positions averaged over all objects. For models able to render future video frames, we obtain a prediction error as the mean squared error between predicted and ground truth pixel values.

Introducing Baselines. We choose VRNN (Chung et al., 2015), DDPAE (Hsieh et al., 2018), and SQAIR (Kosiorek et al., 2018), which have been introduced in Sections 2.3 and 3.3, as baselines against which we compare.[3] VRNNs are an established baseline for unstructured non-linear state space models and present the non-AIR baseline in the experiments. SQAIR and DDPAE are obvious contenders because, like us, they extend AIR to videos. To allow for a fair comparison to STOVE, we fix the number of objects in both DDPAE and SQAIR to the correct amount. A supervised baseline is constructed by isolating STOVE's dynamics core and directly using ground truth object positions and velocities as input. As this baseline does not suffer from any visual uncertainties, it serves as an upper bound to the performance obtainable with the dynamics core. Additionally, a linear ablation is constructed by using STOVE to infer object states from the first 8 observable frames of video. For future frame and

[3]The experiments with SQAIR have been performed by Karl Stelzner.

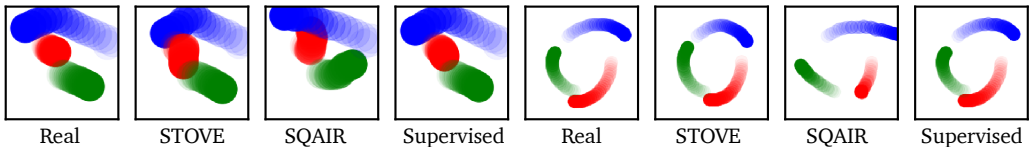


Figure 5.2.: Visualisation of object positions from the real environment and predictions made by STOVE, SQAIR, and the supervised baseline for the billiards and gravity environments after the first 8 frames were given. STOVE achieves realistic behaviour, outperforms the unsupervised models, and approaches the quality of the supervised baseline, despite being fully unsupervised. For full effect, the reader is encouraged to watch animated versions of the sequences at github.com/jlko/STOVE. Objects coloured for illustrative purposes only. Figure and caption first produced for Kossen et al. (2020).

state prediction, the dynamics core is substituted by a linear model which fixes the velocity of each object to the last observed values, resulting in straight trajectories. In the billiards environment, this model will predict sensible behaviour until the first collision occurs or an object reaches a frame boundary. For the linear baseline, we clip maximum positions to the dimensions of the canvas. This results in pixel and position error curves, which serve as lower bounds that should be outperformed by any well-functioning dynamics model. A similar linear baseline has been used by Battaglia et al. (2016). As VRNNs have no inherent notion of objects, one cannot directly obtain object positions from them. Therefore, for VRNN, we only compare the pixel MSE. As DDPAE and SQAIR are both object-aware AIR-derived models, we can obtain pixel and position errors. The supervised baseline directly operates on states, and therefore only position errors can be specified. As the linear ablation is a modification of STOVE, pixel and position errors may be calculated. For additional details on the training and hyperparameter selection of the baselines, the reader is referred to Appendix B.

Result Summary. The qualitative behaviour of the rollouts is visualised in Fig. 5.2. Figure 5.3 displays the rollout errors for STOVE in comparison to the baselines. Table 5.1 underlines the results with concrete numbers. STOVE is able to clearly improve upon the baselines against which we compare. In addition, STOVE performs strikingly close to the supervised model.

Reconstructions. STOVE, DDPAE, SQAIR, and VRNN all reconstruct the observed frames with similarly high accuracy. The same holds for the positions inferred by STOVE, DDPAE, and SQAIR. STOVE's average accuracy for inferred positions is at $1/3$ pixel accuracy. As previously, the accuracy of the object positions obtained by SuPAIR (Stelzner et al., 2019) or AIR (Eslami et al., 2016) has not been reported, this result in itself is an encouraging addition to the literature. However, as no notable differences between STOVE and the baselines may be found in the reconstruction tasks, our focus on a predictive evaluation of the models is further motivated. Note that the position error for the observed frames for the supervised baseline is 0, as the model directly observes object positions and velocities.

Predictions. In the predictive regime, Fig. 5.3 clearly shows that STOVE is able to improve upon previous approaches. No model is significantly outperformed by the linear baseline. Notably, in the billiards scenario STOVE's performance comes close to that of the upper bound given by the supervised baseline. The gravity task

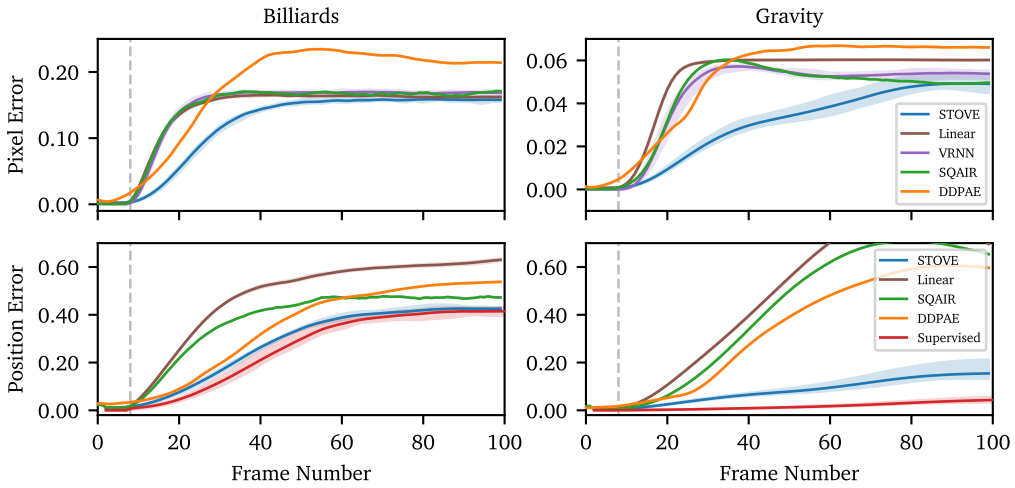


Figure 5.3.: Mean test set performance of STOVE compared to baselines. STOVE clearly outperforms all unsupervised baselines and is almost indistinguishable from the supervised dynamics model on the billiards task. (Top) Mean squared errors over all pixels in the video prediction setting (the lower, the better). (Bottom) Mean Euclidean distances between predicted and true positions (the lower, the better). All position and pixel values are in $[0, 1]$. In all experiments, the first eight frames are given, all remaining frames are then conditionally generated. The shading indicates the min and max values over multiple training runs with identical hyperparameters. Figure and caption first produced for Kossen et al. (2020).

seems easier, because even though the objects move at similar speeds, prediction errors grow slower for STOVE and the supervised baseline. Here, larger differences between STOVE and the supervised baseline are observable, suggesting that the gravity task may further benefit from the precise inference of ground truth states.

Qualitative Discussion. A visual inspection of the produced animated rollouts confirms STOVE's improvements over the baselines.[4] DDPAE, SQAIR, and VRNN are able to model freely drifting objects and objects colliding with walls. However, they are unable to correctly and consistently account for interactions between objects. The videos they predict quickly show unstable and unphysical behaviour. Among the baselines, DDPAE performs best and sometimes produces interactions between objects which resemble realistic collisions. In contrast, STOVE's predictions look strikingly similar to the true underlying simulation. Indeed, preliminary experiments have shown that the quality of STOVE's rollouts is such that humans struggle to consistently discriminate between STOVE's predictions and simulations from the environment. STOVE is able to uphold this prediction quality for a large number of timesteps. We tested this for up to $100\,000$ prediction steps in the billiards environment. This already hints at STOVE's capability of correctly preserving the kinetic energy in the billiards environment, as discussed in Section 5.5.

Additional Observations. It seems worth discussing some general observations concerning the error curves. All models exhibit similarly shaped error curves, resembling logistic growth. Errors for the fully observed initial frames are low. With the onset of

[4] We invite the reader to watch the animated rollouts in the repository at github.com/jlko/STOVE.

Table 5.1.: Predictive performance of our approach, the baselines, and ablations (the lower, the better, best unsupervised values are bold). STOVE outperforms all unsupervised baselines and is almost indistinguishable from the supervised model on the billiards task. The values are computed by summing the prediction errors presented in Fig. 5.3 in the time interval $t \in [9, 18]$, i. e. the first 10 predicted timesteps. In parentheses, standard deviations across multiple training runs are given. The ablations are further discussed in Section 5.10. Table and caption first produced for Kossen et al. (2020).

	Billiards (pixels)	Billiards (positions)	Gravity (pixels)	Gravity (positions)
STOVE (ours)	**0.240(14)**	**0.418(20)**	**0.040(3)**	**0.142(7)**
VRNN	0.526(14)	–	0.055(12)	–
SQAIR	0.591	0.804	0.070	0.194
DDPAE	0.405	0.482	0.120	0.298
Linear	0.844(5)	1.348(15)	0.196(2)	0.493(4)
Supervised	–	0.232(37)	–	0.013(2)
Abl: Double Dynamics	0.262	0.458	0.042	0.154
Abl: No Velocity	0.272	0.460	0.053	0.174
Abl: No Latent	0.338	0.050	0.089	0.235

prediction, the error rises until a maximum value is reached. This plateau *must* be reached, as the environments are inherently chaotic and accurate prediction over long time periods is impossible, as further discussed in Section 5.7. The maximum value for the pixel errors is due to the boundedness of pixel values, while the maximum value of the position error can be explained by the bounds on the allowed or practically observed object positions in the billiards and gravity environment. When this plateau is reached, none of the predictions contain any more useful information with respect to the true positions of the objects. For a reader unacquainted with the problem at hand, the differences between models based on the position error curves may look unremarkable. However, a visual inspection of the animated rollout graphics clearly shows substantial differences between the models, as discussed above. Section 5.5 therefore proposes an additional measure of performance to account for STOVE's ability to predict visually convincing rollouts over long periods of time. Section 5.7 then discusses the chaoticity of the billiards environment as a factor for the curve shapes and the implications this has for the environment's popular application as a benchmark for machine learning models.

Additionally, peculiarities for the different models may be revealed from the error curves. For example, the pixel error curve for DDPAE reaches a higher plateau compared to the other models. A simple explanation can be given: DDPAE tends to increase the object sizes during rollouts in both scenarios, which leads to a larger pixel error when the highly incorrect object predictions are compared to a black background. Furthermore, SQAIR and VRNN display a decrease of the long-term pixel error, which is most pronounced for the gravitational data. This decrease is due to a tendency of the models to merge objects, i. e. to decrease the total object count or predict completely overlapping objects. This leads to less area where incorrect object predictions are compared to the background and therefore to lower pixel error values. Without knowledge of the peculiarities of the environment and the pixel error, one may erroneously conclude that VRNN and SQAIR outperform DDPAE, when actually, DDPAE's non-merging behaviour should be seen as preferable and more realistic, even though it leads to higher pixel errors in the chaotic regime. The position error for the linear baseline converges to a higher plateau than all other models. Its predictions quickly stick to the boundary, which results in higher average

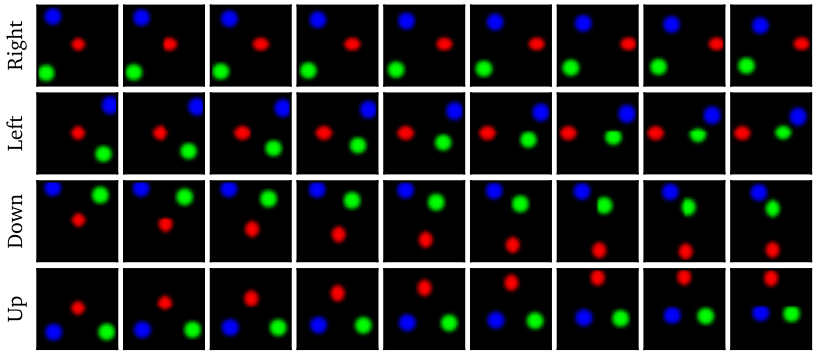


Figure 5.4.: Visualisation of action-conditioned video prediction in STOVE. STOVE correctly and precisely predicts the effect of the actions corresponding to movements in cardinal directions on the active ball, displayed in red. Actions are repeatedly applied on each frame. Images are rendered by STOVE and coloured for illustrative purposes.

distances to the uniformly distributed ground truth positions than for the other models.

5.4 Action-Conditioned Prediction

The action-conditioned variant of STOVE, introduced in Section 4.4, is applied to the avoidance game in the billiards environment. Given a list of actions, STOVE predicts future rewards of the environment in addition to the object states. STOVE therefore learns a full simulation of the environment, which is combined with a planning algorithm to obtain a policy. In Kossen et al. (2020), we show that competitive and data-efficient performance is obtained when using STOVE's action-conditioned reward prediction with MCTS (Coulom, 2006) for planning in comparison to state-of-the-art model-free reinforcement learning baselines. MCTS with STOVE achieves a performance of about 1 collision per 100 frames of video.[5] We will give a detailed discussion of the predictive performance of STOVE when conditioned on actions. For the action-conditioned experiments, the environment was used in full RGB colour mode, where each ball is given a different colour, and appearance information is appended to the object states for dynamics modelling, as described in Section 4.4. However, the SPNs continue to model greyscale images for simplicity.

The first experiment demonstrates that STOVE successfully learns the effect of actions on the active ball. The model is initialised with 4 frames, null action, and the active ball placed in the centre of the frame. Then follows a sequence of repeating a single action 10 times. The passive balls were initialised such that collisions between the active ball and one of the passive balls were prevented. Conditioning STOVE's prediction on a sequence of a single action, repeated over time, yields a precise movement of the active ball in that direction, while the passive balls move unaffected. For the four cardinal directions, this is displayed in Fig. 5.4. Figure 5.5 (left) demonstrates this for all 9 available actions. The precision with which the movements are predicted is high, especially given the low resolution of the input

[5] A detailed discussion of this application in the context of reinforcement learning is out of scope for this thesis, as the corresponding experiment in the publication has mainly been performed by my co-authors, see Kossen et al. (2020) for details.

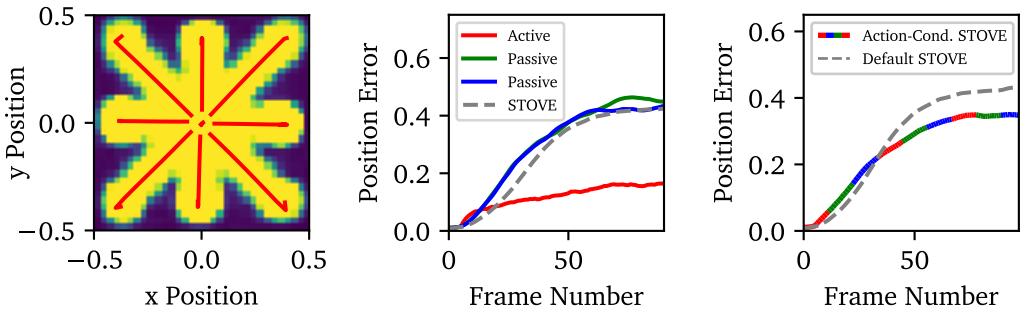


Figure 5.5.: Further visualisation of STOVE conditioned on actions. (Left) STOVE precisely predicts the effects of all 9 actions on the movement of the controlled ball at sub-pixel resolution. Actions are applied for 10 steps, the first of which is not displayed, such that the effect of the null action is clearly visible. 9 separate experiments were performed and combined in a single plot. The movement of the controlled ball as a result of the action sequences is both rendered as pixels (yellow, normalised sum of images displayed) as well as plotted in terms of the predicted positions (red). The passive balls are not displayed and did not affect the movement of the active ball. (Centre) Per-object prediction errors for the action-conditioned STOVE on the test set with random actions. The position error for the controlled ball on the test set is far lower than the error for the passive balls, suggesting that STOVE fully exploits the available information. (Right) The total position error averaged over all objects for action-conditioned STOVE (red-green-blue) is similar to STOVE in the default billiards scenario (dashed-grey). Additional information about the experiments may be found in the text.

images. Figure 5.5 (centre) demonstrates that predicting the movement of the active ball can be achieved with much lower error than the prediction of the passive balls. This is unsurprising, since, given a list of actions, the movement of the active ball is largely deterministic. That is, it is deterministic until its trajectory is affected by a passive ball, whose position is chaotically determined. We speculate that such collisions mostly account for the prediction errors of the active ball.

Figure 5.5 (right) shows that the performance of STOVE in the action-conditioned scenario is similar to the performance of STOVE in the default billiards scenario. This is surprising. As presented in Chapter 4, in STOVE, a graph neural network models the transition function. By design, this graph neural network uses the *same* function to predict the state transition for *all* objects. This is an intuitive approach for physics simulations where, usually, all objects follows the same laws of physics, i. e. in the billiards and gravity environments the implicit assumption holds that the prediction function is identical for each object. Differences between the objects which affect the transition function, such as different masses, are usually assumed to be small and reflected in the latent state. However, in the action-conditioned environment, these assumptions of largely equal dynamics are violated: The controlled ball moves entirely dependent on actions, while the passive balls continue to bounce around. Excitingly, the dynamics function continues to predict correct behaviour for *all* of the balls in the simulation. The *single* function of the GNN is therefore able to approximate *multiple* entirely different dynamics. Essentially, the dynamics model learns to map characteristics of the specific object's latent state to indicator functions in the GNN, which then determine the dynamics. Note that this behaviour is learned

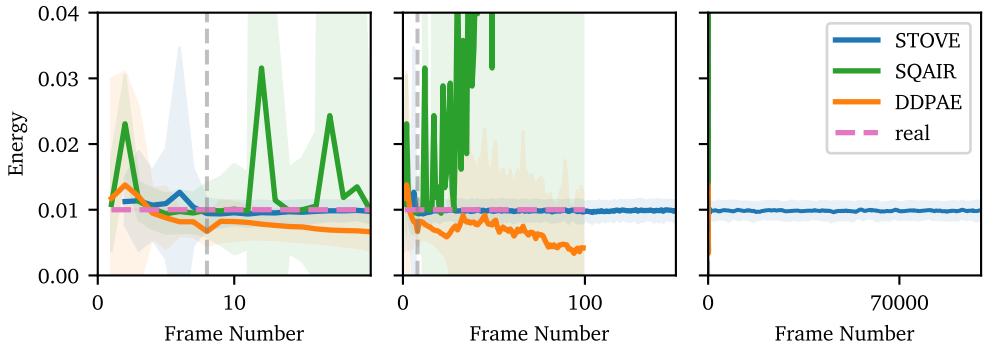


Figure 5.6.: Comparison of kinetic energies in the rollouts predicted by the models, computed based on position differences between successive states. Only STOVE's predictions reflect the conservation of total kinetic energy in the billiards data set. This is a quantitive measure of the convincing physical behaviour in the rollout videos. (Left) All models recover noisy but roughly correct estimates of the system's energy. (Dashed vertical line demarcates the end of observations.) (Centre) STOVE continues to predict trajectories with constant energy, while the energies of SQAIR and DDPAE quickly diverge. (Right) STOVE manages to predict these constant energy rollouts for possibly infinite amounts of time – displayed here are $100\,000$ frames. (Left & Centre) Averages are over 300 trajectories from the test set. Shaded regions indicate one standard deviation. (Right) Rolling average over a single, extremely long-term run. Figure and caption first produced for Kossen et al. (2020).

and not pre-specified. Such behaviour is only possible if the latent space *allows* for reliable differentiation between the object types, e. g. by including object masses, appearance information, or other distinctive properties. As we append a visual representation to each object state, here, the GNN *can* differentiate between passive and active objects from the input representations. However, preliminary experiments have shown that even without such appearance information, i. e. when objects are not discernable a priori, the model learns to predict adequately different dynamics for active and passive objects. This is only possible if the GNN learns to assign different signatures in the unstructured latent $\mathbf{z}^o_{t,\,\text{latent}}$ to different object dynamics. However, model performance does deteriorate in this scenario, such that we chose to append appearance information in the default case.

In the context of physics simulations, the action-conditioned environment may seem a bit contrived and without real-world relevance. However, the application of graph neural networks as transition functions in broader contexts such as reinforcement learning or robotics is interesting. There, it is never the case that *all* objects follow the *same* dynamical laws: The paddles in Pong behave differently than the ball, and Mario's interaction with Princess Peach is different from Donkey Kong's. Our simple extension of the billiards environment to action-conditioning can be seen as a first proof of concept that relational dynamics models continue to function well, even if implicit assumptions are broken. This further motivates their use outside of physics modelling domains, and Section 6.1 contains a further discussion of possible future directions of research implied by theses results.

5.5 Energy Conservation

STOVE predicts visually convincing rollouts for extremely long timeframes, while the predictions from DDPAE, SQAIR, and VRNN quickly show unrealistic behaviour such as teleporting, decelerating, wobbling, or overlapping balls. For the billiards scenario, a reliable quantification of this observation is demonstrated. Here, the energy of the system should be conserved, as no friction is modelled and collisions are fully elastic. Because no potential energies are present, the total energy E is equal to the kinetic energy K and

$$E = K = \sum_o \frac{1}{2} m^o (\mathbf{v}^o)^2 \,, \tag{5.1}$$

where m^o and $\mathbf{v}^o$ are the object masses and velocities. As DDPAE and SQAIR do not explicitly model the object velocities, the velocity is approximated for all models by the differences in positions between consecutive frames $\mathbf{z}^o_{t,\,\text{pos}} - \mathbf{z}^o_{t-1,\,\text{pos}} = \mathbf{v}^o \cdot \Delta t$.

STOVE correctly conserves the total kinetic energy, while the other models exhibit divergent behaviour after only few predicted frames. Note that, while the energy should be strictly conserved for any given sequence, STOVE achieves this conservation on average – rolling average over the sequence or average over different sequences. This is displayed in Fig. 5.6. STOVE's qualitative superiority in terms of visual rollout quality is clearly reflected by its ability to preserve the average kinetic energy of the system. We therefore find energy conservation to be a quantity well-reflective of the qualitative model behaviour. We have experimentally verified the constancy of STOVE's energy predictions for up to $100\,000$ frames, although we see no reason why this behaviour should not continue indefinitely given the Markovian properties of STOVE's underlying graphical model. Note that we have no measures in place to directly optimise for a conservation of energy. This is further discussed in Section 5.6.

During the initial iterations of training, the rollout *position error* decreases quickly and reliably. Figure 5.7 shows that the convergence to the correct *energy value* takes much longer, converging only in the late stages of training. This is in line with the observation that STOVE often predicts decelerating objects in the initial phase of training, leading to a near standstill of the dynamics.

The above result contains a caveat. As is common in the literature, but not often stated, the randomly initialised velocities of the objects are normalised such that the total kinetic energy of the system is constant over all sequences.[6] Therefore, in the above experiments, STOVE does not need to actually *observe* the velocities of any given sequence with high accuracy, as long as it internalises the total kinetic energy shared by all sequences in the test and training set. Still, regressing to the correct energy is impressive and certainly preferable to the catastrophic breakdown of other methods. As far as we are aware, this constant energy choice has never been explicitly discussed. This is partially explained by the fact that learning generative sequence models which exhibit energy conservation has likewise not been a focus in the video modelling community. A hint that this regression to the mean does indeed occur, is given by the fact that in Fig. 5.6, the standard deviation over the energies is higher for observed values than for future predictions. The only sensible

[6] See e. g. code for simulations of Sutskever et al. (2009).

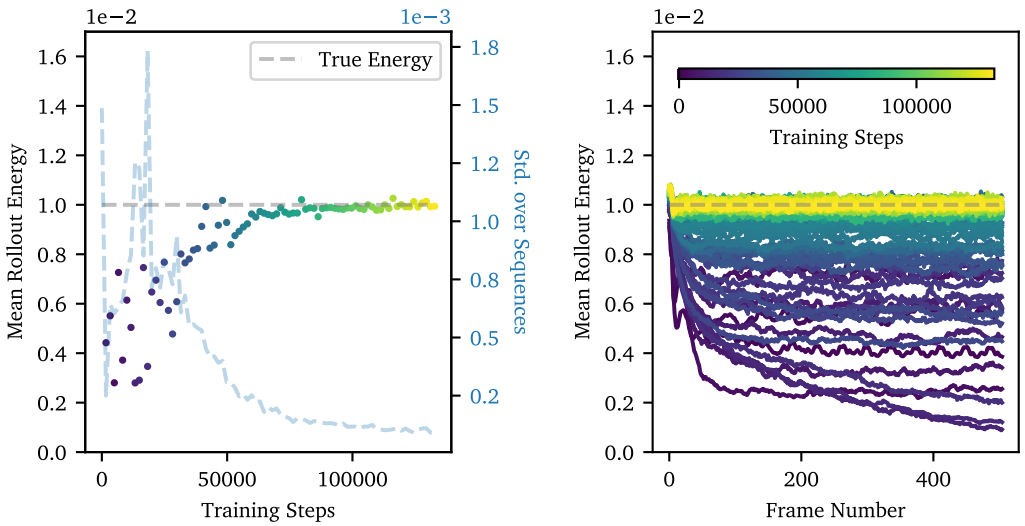


Figure 5.7.: Emergence of energy conservation during the training of STOVE for a single, reproducible run. Mean energies slowly converge to the correct value indicated by the grey dashed line. (Left) Mean energies over the 300 test set sequences of lengths 100 during training. As the mean energies converge, their standard deviations over the different sequences likewise decrease, converging to a low value. (Displayed in dashed blue, y-axis on the right.) (Right) Quantitative behaviour of the rollout energy over the number of predicted frames. During the inital stages of training (blue), the predicted energies quickly decrease over the rollout duration, corresponding to a near stand still of the objects, as if frictional forces were at work. This behaviour is remedied later in training (yellow) when energies are correctly conserved. Per curve, a mean was computed over the 300 sequences in the test set. The colourbar maps each curve to a training step.

explanation for this is that these prediction energies are actually *not* conditioned on the observed sequence at all and instead stem from internalised behaviour learned over all sequences. Therefore, due to the peculiarities of the model and data, we predict the energy of the system with lower variance than we can observe.

This then leaves the question of whether STOVE is able to observe and predict sequences of *varying* energies when (a) trained on a set of single energy sequences, as is default, or (b) trained on a set of sequences with energies sampled uniformly in a reasonable interval. Note that the physics of the environment remain unchanged such that the kinetic energy remains constant throughout a sequence. Questions (a) and (b) are answered in Fig. 5.8 (left and right). If STOVE is trained on constant energy data (left), it is able to infer new energies from observed frames, albeit with some negative bias and decent amount of noise at high energies. However, stable forward predictions of these energies are not possible, and the predicted energies quickly (after around 10 frames) diverge to the training set value. This value is then stably propagated without any regard for the input sequence. This behaviour further supports the hypothesis that STOVE learns and internalises a global kinetic energy. If STOVE is trained on a diverse set of energies, said energies can be observed more accurately and precisely. Prediction also improves, and after 10 predicted frames only small divergences from the input energies are visible. However, after predicting 100 frames, the model seems to again converge to some non-zero energy regardless of the input energy. The conservation of arbitrary energies cannot be achieved and is left for future work. Nevertheless, the improved behaviour for the diverse set of

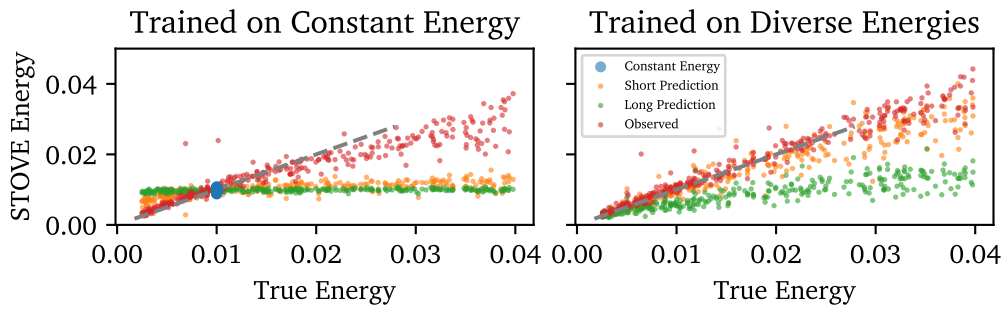


Figure 5.8.: Mean kinetic energy observed/predicted by STOVE over true energy of the sequences. (All) The dashed diagonal line represents perfect predictive behaviour. The observed values refer to energies obtained as the mean energy value over the six initially observed frames. The short (long) timeframe refers to the mean energy over the first 10 (100) frames of prediction. 300 points per label are displayed. (Left) STOVE is trained on sequences of constant kinetic energy. This is the default procedure for other experiments of this thesis and in the literature. As can be seen from the blue scatter points, STOVE predicts sequences which comply with energy conservation, when the sequence energy exactly equals the single energy observed during training. All 300 points are concentrated at $(0.01, 0.01)$. When STOVE is applied to sequences of different energies – but trained on sequences with a single energy – it manages to infer these energies from observed frames fairly well, with inaccuracies compounding at larger energies (red). In conditional prediction, however, the mean predicted energies quickly diverge to the energy value observed in the training set (orange and green scatters). (Right) STOVE is now trained on sequences of varying energies. Compared to the constant energy training, energy observation and prediction improves drastically. Energy observation (red) does not exhibit a strong bias at larger energies anymore. The short-term predictions (orange) no longer immediately regress towards a specific value. However, after 100 frames, the predicted energies still regress to a wrong value (green). Figure and caption first produced for Kossen et al. (2020).

training energies is promising.

5.6 Noisy Supervision

This section investigates the influence of observation noise on the prediction error and energy conservation capabilities on the billiards data set. The initial motivation for this experiment was the inability to obtain supervised baselines which conserve the kinetic energy of the rollouts. As previously introduced, the supervised model is obtained by isolating SuPAIR's dynamics model and inputting ground truth object positions and velocities. In STOVE, SuPAIR passes object positions and velocities to the dynamics model. It is therefore initially unclear why STOVE's dynamics model should achieve energy conservation while the dynamics model on its own, referred to as the supervised baseline, does not. A difference between the two is the fact that, while the supervised baseline has access to the floating point precise ground truth positions from the simulation, STOVE observes positions uncertainly from low-resolution images. We therefore investigate the hypothesis that *adding* noise to the ground truth data makes the supervised baseline behave more similar to STOVE. Normally distributed noise with zero mean and standard deviations equal to STOVE's

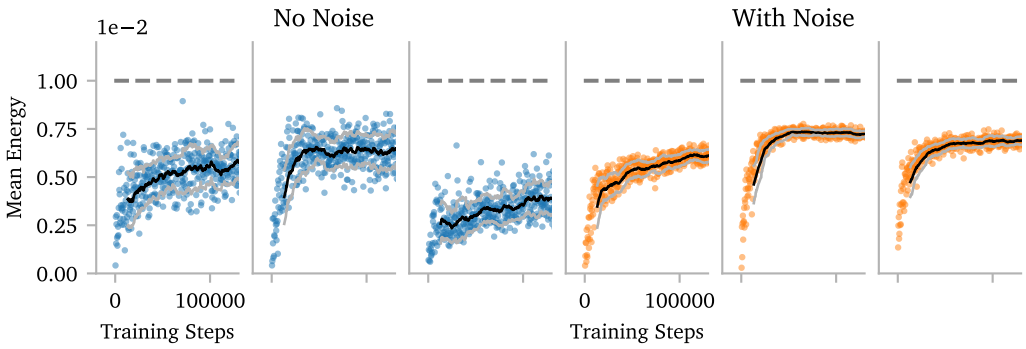


Figure 5.9.: Mean predicted energies as training progresses for the supervised dynamics prediction of six different runs. For the three blue runs, no noise was added to the ground truth data. For the three orange runs, noise the size of STOVE's observation uncertainty has been added. Adding noise seems to benefit the decrease of the standard deviation of the mean energy predictions. Energy conservation for the supervised baseline could not be achieved. Rolling means and standard deviations over the training steps are plotted alongside in black and grey. True energy is shown with the dashed horizontal lines.

observational uncertainty is applied to the training set.

Figure 5.9 visualises the effect of adding noise to the input data on the mean predicted energies during rollout as training progresses. Without noise, the mean predicted energies of the supervised model are far from the true value, which is again constant over all sequences. Additionally, a large variance over the mean predicted energies between training steps is observed (dashed grey lines). For the supervised models trained with noise, this variance decreases and additionally, the mean predicted energies edge closer to the real value. The model trained with noise does indeed appear to behave closer to STOVE than the model trained without noise. Experiments were repeated three times to highlight differences between runs due to random model initialisation. Again, variances between models are higher for the models trained without noise than for those trained with. An explanation for the difference in variance is suggested as follows: For a chaotic environment such as billiards, see Section 5.7, exact prediction over long periods is impossible for any model. However, a model which observes the uncorrupted object states may still try to predict future states exactly rather than capture general environment behaviour, as this is possible to some extent for shorter timeframes. However, over the long rollouts presented here, small deviations in the predictions add up exponentially, creating predictions with large variations in object states and therefore energies. As noise is added to the training data, it becomes harder for the model to predict future states exactly. This may be beneficial to the regression to the mean energy effect observed. In the face of uncertain data, the model quickly regresses to its internalised mean dynamics. This suggests that adding noise may indeed benefit the reliable conservation of a single mean energy.

Figure 5.10 (left) corroborates this result. It displays the progression of the average predicted energies over the number of predicted rollout frames for the models trained with (orange) and without (blue) noise from Fig. 5.9. The models trained without noise display a tendency of slowly decreasing energies during rollout. This leads to a visual impression similar to frictional forces acting on the objects. As can be seen from Fig. 5.7, similar behaviour is observed during the initial training phases

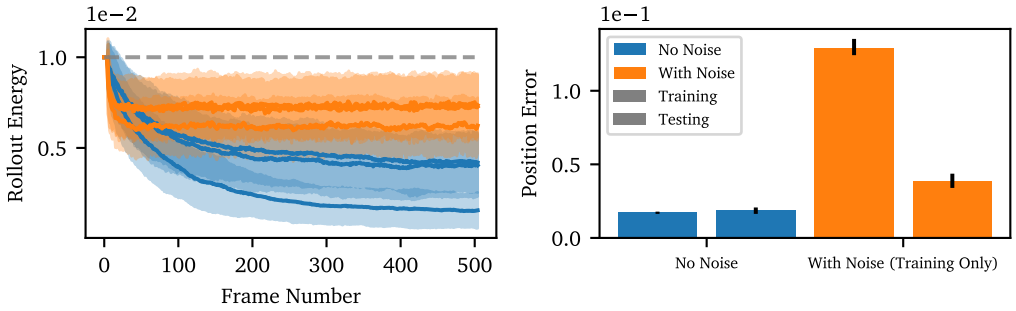


Figure 5.10.: (Left) Progression of predicted energies over rollout frames. The supervised models without noisy data (blue) display a slow but steady decrease in the predicted energies. Adding noise (orange) increases the average predicted energies, which edge closer to the true value. Energies now decrease sharply once at the beginning of prediction, and then they stay constant. The dashed grey line indicates the true, constant energy of all sequences in the data. Standard deviations are given over the 300 runs in the test set. Eight runs are displayed, a superset of those in Fig. 5.9. (Right) Average train and test set position error for supervised models trained without (blue) and with (orange) noise. Without noise, train and test set errors are low. With noise, train errors (measured against noisy training set) are higher due to the noisy observations. Test errors (not corrupted with noise) are considerably lower than training set errors. This suggests that the model is able to extrapolate from noisy data. Test errors do, however, not quite reach the level of the model without noise. Error bars indicate the min/max values over four runs each.

of STOVE. The supervised model can however not escape this undesirable behaviour as training progresses. Adding noise to the training data noticeably affects the shape of this curve. Instead of the slow, steady decrease of the predicted energies, a sharp decrease in energy is observed in the transition from reconstruction to prediction, after which the mean predicted energies stay *constant*. This suggests that adding noise does indeed benefit a conservation of energy during rollout. That being said, the supervised models predict energies lower than the true energy in both scenarios. Additionally, the initial sharp energy decrease for the model trained with noise is noticeable in animations and appears unrealistic. The energies predicted by the models augmented with noise are, however, closer to the true energies compared to the noiseless variant. Figure 5.10 (right) shows a comparison of the position prediction errors for the two supervised baselines with and without noise on the training and test sets. As was to be expected, the model without noisy training data outperforms its noisy equivalent on the position error. However, if the model trained with noise is benchmarked on the *uncorrupted* test data, its prediction error is much lower than the training error against the noisy data. This suggests that, in addition to producing rollouts with energetic properties closer to that of STOVE, this model successfully learns to generalise model behaviour beyond the noise in the training set, since the prediction error significantly drops when the model is applied to data without noise.

Some experimental steps towards closing the gap between the behaviour differences of the supervised dynamics model and the unsupervised STOVE have been taken. Often in the literature, a suspicion towards the beneficial effects of observation noise on training models from visual data is voiced, see e. g. in Watters et al. (2017).

This assumption is rarely tested. The strict compositionality of STOVE allowed us to realize specific experiments testing the effects of observation noise on model behaviour. Adding noise to the training data does indeed seem to benefit the prediction of a stable mean energy over prolonged rollout periods. However, we were unable to achieve a mean predicted energy equal the true energy of the data in the supervised setting. Future work could alleviate this difference by further tuning the artificial noise, the training procedure, or model hyperparameters.

5.7 Chaos in the Billiards Environment

The billiard environment is chaotic. This implies that, on average, states which are initially *infinitesimally close* show *exponentially* diverging trajectories. The same holds for the gravity environment. Yet, the billiards environment is more commonly used in the literature, and stable simulations may be obtained without much effort. For the purposes of brevity, we therefore restrict ourselves to a discussion of the billiards environment. The chaoticness of the environments entails that it is virtually impossible to learn a model which perfectly emulates training set trajectories, because even noise with magnitudes as small as the limits of floating-point precision will lead to an exponential divergence of states. Here, this exponential growth is, of course, limited by the boundedness of the simulation due to the hard collisions with the walls of the environment. This section studies the effects of noise on the chaotic environment, i. e. the data *generating* program itself, and not directly the performance of any model, such as STOVE, trying to emulate it. The insights obtained here have, as far as we are aware, not been discussed in the machine learning community so far. They have implications on the use of the billiards environment as a benchmark for learning physics simulators.

Perfect Simulator with Observation Noise. The main result of this investigation is obtained by answering the following question: How does the rollout error behave, assuming the model's dynamics prediction has learned a perfect simulation, but its initial observation is subject to noise? To answer this question, we investigate the effect of artificial observational noise purely in simulation. We first create a random reference trajectory from the simulation, remembering the initial state of the environment. The environment is then reinitialised using the same initial state. However, this time, the initial state is corrupted by adding normally distributed noise, before rerunning the simulation from this minimally altered state. We then compare the trajectories of the two curves and produce the familiar position error rollout plots. This procedure is repeated to obtain summary statistics. The size of the noise added is identical to the observation noise to which STOVE is subject to, i. e. roughly $1/3$ of a pixel. Surprising results are obtained and displayed in Fig. 5.11 (top left). Alongside the error curve obtained from the experiment, a reference curve for STOVE's rollout error is plotted. The two curves look almost identical. STOVE's error curves are indiscernible from those of a perfect model of the environment corrupted by appropriately sized initial observation noise. As is discussed further below, this does not imply that STOVE's dynamics model is perfect, only that it is good enough to not drastically affect the error curves. Reductions of the observation noise may only be achieved by increasing the pixel resolution of the data, assuming that SuPAIR as STOVE's image model performs optimally. As the observational error is already in the subpixel regime, we think it fair to judge this assumption to be approximately

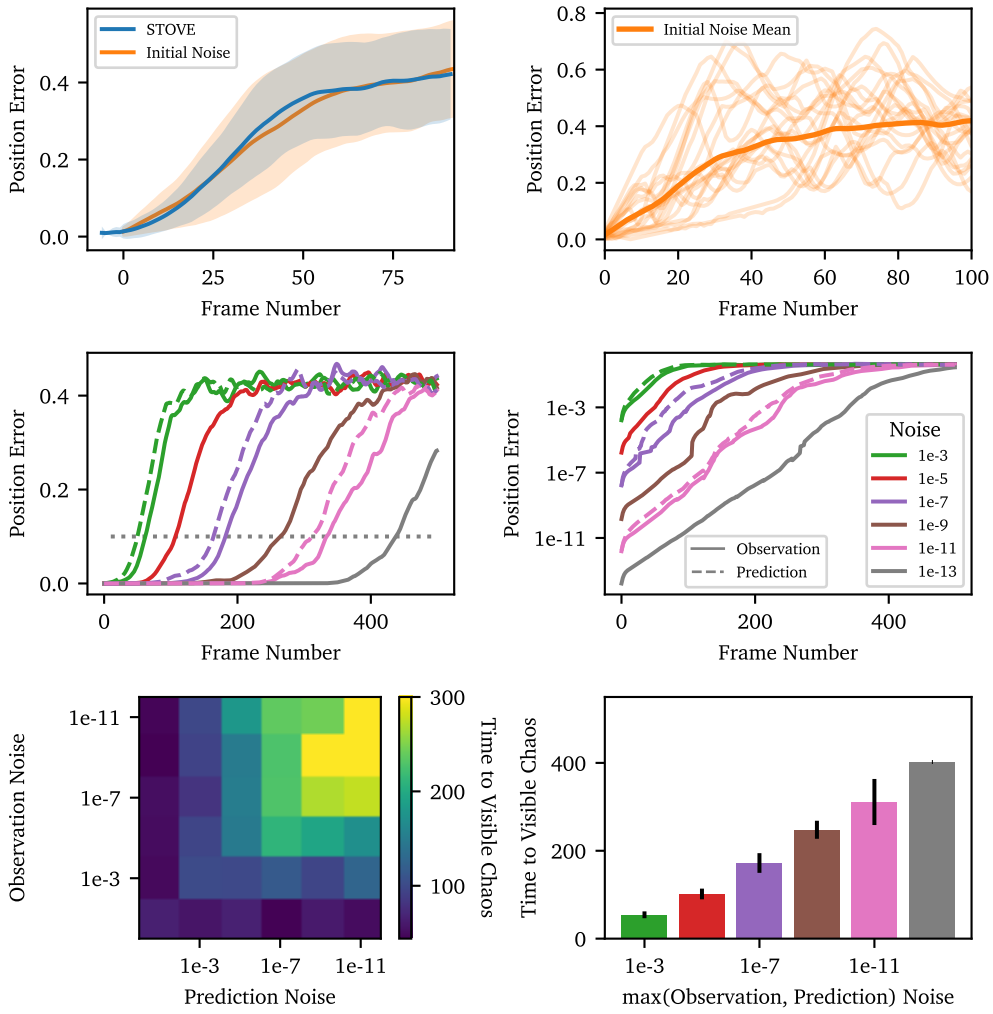


Figure 5.11.: The environment's chaoticness is a direct cause of the familiar shape of the position error curve. This is revealed by investigating the effect of artificial noise on the trajectories generated by the environment. (Top left) A perfect simulation of the environment with added initial noise results in a mean error curve strikingly similar to that of STOVE. (Top right) The familiar shape of the error curve is obtained only by averaging, and individual trajectories (displayed with transparency) are heavily influenced by collisions. (Centre left) Decreasing the initial observation errors only shifts the time until visible chaos. (Centre right) A logarithmic plot reveals that the errors do in fact accumulate in exponential fashion. The initial observation noise merely determines the initial position error, after which the rate of exponential growth is inherent to the simulation. (Centre and bottom) There are two types of noise which may be applied to the simulation: (a) noise applied once, corresponding to the noise of an initial observation, conditioned on which predictions are made, and (b) predictive noise applied continuously at each timestep, corresponding to the errors of the dynamics model. Whichever noise is larger dominates the shape of the error curve, succinctly described by the time until visible chaos. If the observation error dominates the dynamics error, the performance of the dynamics prediction cannot be assessed from the position error curve. The curves are averaged over 300/100/10 individual sequences for the curves in the top/centre/bottom. Description is displayed here in abbreviated form. See text for a full discussion of all figures displayed. Legend in centre right plot indicates the standard deviation of the applied zero-mean Gaussian noise. Colours for centre and bottom right plot are consistent with the legend. Dashed grey line in centre left plot indicates the threshold used to estimate the time until visible chaos for the bottom plots.

true. In other words, at the given resolution, the billiards environment may be considered solved optimally by STOVE with respect to the rollout error, simply due to the lower limit on the observation noise and the chaoticity of the environment. The standard deviation displayed corresponds to the standard deviation between the individual error curves over 300 sequences. This standard deviation is fairly large, as the process of error accumulation in the individual sequences is highly stochastic and dependent on *when* exactly collisions take place. This is illustrated by Fig. 5.11 (top right), which displays the progression of individual error curves alongside the mean value. The impact of individual collisions on the error is clearly visible, and even though the initial observational error is small, individual trajectories diverge quickly and result in entirely different error curves. The familiar shape of the error curve emerges only by mean aggregation. Note that in previous rollout plots, the standard deviation between the average rollout errors over different training *runs* were displayed.

Decreasing the Observation Noise. One could argue, that a simple increase in resolution should alleviate the problem, as it allows for higher resolution position inference which in turn allows for lower error curves. The solid lines in Fig. 5.11 (centre left, matching legend on centre right) show the error curves which emerge for different levels of the simulated observation noise. Between each solid line displayed in Fig. 5.11 (centre left), the initial noise is decreased 100-fold. This decrease can be considered extreme, as a similarly sized increase in the video resolution would be needed to achieve these lower observational errors in practice. Even initial noise as low as 1×10^{-13} leads to the same familiar chaotic error curve after about 500 prediction steps. Decreasing the level of noise merely *shift* the curves right, delaying the time to the apparent chaotic explosion. This may be counterintuitive, as the naïve expectation may be that error curves with smaller noise show *slower* error growth, i. e. smaller gradients in the error curves. An explanation to this is given by Fig. 5.11 (centre right). Indeed, in line with chaos theory, the average position errors grow exponentially. That is, until the state divergence reaches the order of the simulation boundary and errors converge to their maximum value. The gradient of the log-transformed curves seems to be an intrinsic property of the simulation, and the initial value of the error merely shifts its initial value, setting a baseline value from which the exponential growth proceeds. Therefore, the size of the error of observation merely affects the time until chaotic explosion becomes visible in the linear plot (centre left). This confirms our observation in Fig. 5.3 that the shape of the error curve is largely independent of model choice.

Adding Prediction Noise. Besides initial observational errors, additional prediction errors can occur with each step. We visually observe these errors clearly, e. g. in the rollouts of the unsupervised baselines. However, it is fair to assume that STOVE also does not perfectly emulate the physics environment given noisy observations, as was assumed for Fig. 5.11 (top left). We qualitatively validate this assumption in Section 5.8. What is the effect of the errors of the dynamics simulation on the shape of the error curves? As these errors are incurred anew at each prediction step, we call them *prediction* errors in Fig. 5.11. Figure 5.11 (centre) shows with dashed lines the error curves resulting from applying additional noise to the predicted states *at each step* of the prediction. The size of this continuous noise was assumed to be the same as the initial uncertainty for the sake of simplicity. Again, an initially surprising result is obtained. The added continuous noise barely affects the time to visible chaos. We

define the time until visible chaos by the time when the position error exceeds a value of 0.1 as indicated by the dashed grey line in Fig. 5.11 (centre left). Adding 1×10^{-3} dynamics noise to a process which incurred 1×10^{-3} initial noise reduces the time to visible chaos only by about 10 frames. These numbers do not change for a process with noise levels of 1×10^{-11}. This is especially surprising as the dynamics noise is applied each step along the simulation. Again, these observations may be explained by the chaotic behaviour of the environment. As the states are already exponentially diverging with a rate inherent to the environment, the additional noise does not have a large impact on the time until visible chaos. In the logarithmic view of the error curve, each additional dynamical prediction error simply contributes a constant offset to the curve.

Observation versus Dynamics Noise. If the two noises, observation and dynamics noise, have different magnitude, the time until visible chaos is dominated by whichever noise is larger. Figure 5.11 (bottom left) compares a grid of combinations of initial and dynamical noise with respect to their time to visible chaos. The above assumption that the larger uncertainty dominates the time to chaos is validated by the relation found in Fig. 5.11 (bottom right). This investigation of the environment shows that a perfect simulation of the environment subject to no uncertainty in the dynamics prediction but initial observation noise equal to that of STOVE results in error curves indiscernible from those produced by STOVE. Unless the dynamics error becomes significantly larger than the observational error, it does not influence the shape of the error curve, as this is dominated by the larger of the two errors. To improve the shape of the error curve, one must first decrease the observation error.

Model Comparison in Chaotic Environments. More generally, the shape of the error curves is largely a product of the environment characteristics. Without additional metrics such as the conservation of energy, the dynamics prediction error is our only proxy for assessing the *quality* of the generated trajectories. The pixel resolution presents a lower bound to the observation error. As the observation and prediction errors of the presented models likely do not differ over magnitudes, they all exhibit similar error curves. We find this experimentally corroborated in Fig. 5.3, as errors for the different unsupervised models look strikingly similar, even though the visual appeal of their predictions differs drastically. For STOVE, good reconstructions can be obtained after a few thousand iterations of training. However, for good future state predictions, it is necessary to train up to $100\,000$ steps. In this respect, the above results are somewhat disappointing, as we spent most of the training on improving the prediction error, which will have little effect on the shape of the error curve.

We consider the above results useful to anyone reading about or performing experiments on chaotic environments. Maybe the experiments presented herein inspire future work to consider and investigate the effects that chaos has on prediction errors and model comparison. The dominating behaviour of the observational error suggests that supervised approaches may be preferred when testing dynamics prediction. As supervised approaches do not suffer any observational noise, the error obtained by them directly corresponds to the error of the dynamics prediction. However, many models, such as SQAIR and VRNN, do not possess the strict compositional structure of STOVE, and it is not possible to obtain a dynamics module, which can be trained in isolation, from them. While STOVE's performance may be considered optimal under the above assumptions, this does not imply the obsolescence of the billiards

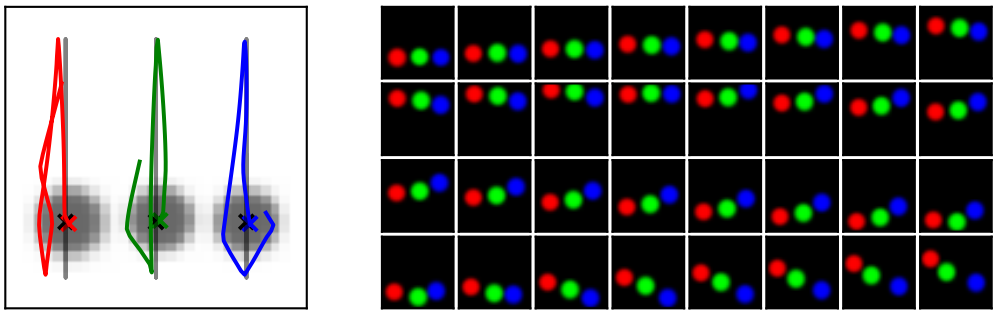


Figure 5.12.: STOVE's rollouts appear visually convincing to the naked eye. Here, a setup is tested which should result in the balls moving infinitely up and down, without drifting off to either side. The trajectories generated by STOVE do not meet this requirement. (Left) Example trajectory generated by STOVE given the setup. (Right) Rendered sequence of 64 frames corresponding to the trajectories on the left. Images are displayed with a frame skip of 2.

task. Instead, the shortcomings of the position error strongly motivate the default use of additional metrics, such as conservation of energy, to assess the performance of physics prediction models applied to chaotic dynamical systems. Other quantities of global or local conservation may be equally suited. Besides, such metrics are more in line with long-term goals of the field: discovering general truths, such as conservation laws, from the data, instead of solely solving prediction tasks.

5.8 Stability of Predictions

In the previous sections, we have established the insufficiencies of the rollout error to fully assess the performance of the dynamics model. STOVE's conservation of the kinetic energy is a quantitative measure to demonstrate the stability of the rollouts. However, STOVE exhibits this stability only on average. And even for exact energy conservation, one can imagine a variety of scenarios which conserve energy but do not conform to realistic physical behaviour. Therefore, we present an additional qualitative investigation into the physical plausibility of the predicted trajectories. We initialise STOVE with a simple constellation of object states, for which the ideal behaviour is obvious: Three balls are placed equidistant (x-axis) at equal height (y-axis) and with identical velocity of $\mathbf{v} = (v_x, v_y) = (0, v_y)$, where v_y is chosen such that the total energy results in the familiar value. Given this initial state, the behaviour of a perfect simulation are vertically bouncing balls that do not drift horizontally and keep constant absolute vertical velocity. As is visible from Fig. 5.12, STOVE does not meet this expectation. Here, STOVE predicts future states $\mathbf{z}_{t+1}$ directly from the means of $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$ without sampling. Of course, STOVE's predictions are subject to observation noise of the initial state as discussed extensively before. However, the errors made are not attributable to observation noise alone. If they were, the objects would continue on straight paths and otherwise fully obey the physics of elastic collisions. From Fig. 5.12 (left) it is obvious that this is not the case. Instead, the trajectories contain distortions. Especially collisions with the walls seem to set the objects on wrong paths. Additionally, the velocity constancy is violated, as the objects are on different heights after 64 frames, while keeping trajectories relatively straight, see Fig. 5.12 (right). While this demonstrates

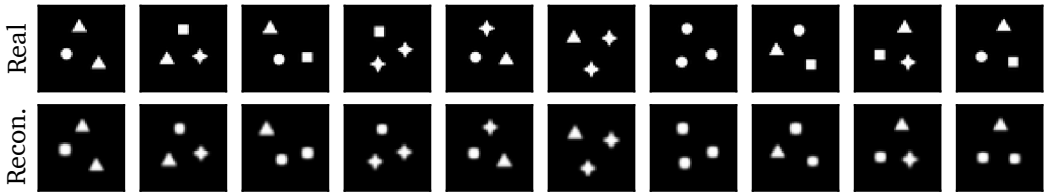


Figure 5.13.: Reconstructions obtained from the SuPAIR image model of STOVE when applied to more complex object shapes. Figure and caption first produced for Kossen et al. (2020).

that STOVE does not perfectly emulate the physics behind billiards, the simulations still look convincing to the human eye. The sequence displayed in Fig. 5.12 (right) corresponds to 64 frames of video, which equate $8/3$ seconds at 24 frames per second. This is long enough for the errors to accumulate gradually such that they do not appear jarring. This is reasonable, especially when one considers the deviations in the trajectories in Fig. 5.12 (left) in relation to the low pixel resolution of the images.

5.9 Flexibility of the Object Model

STOVE's image model SuPAIR is capable of rendering images with higher complexity than the white balls on black background employed throughout this thesis. While the modelling of complex scenes is decidedly not the focus of this thesis, we perform a proof of concept, or sanity check, showing STOVE's application to scenes where object shapes are drawn at random from a set of sprites. Multiple object shapes may be present in any single sequence and shapes are kept constant throughout the sequence. As can be seen in Fig. 5.13, STOVE does a decent job of reconstructing this slightly more complex scene. The dynamics performance is not affected by this change in appearance, as it is entirely oblivious to it. All objects were modelled using a single SPN, as suggested by Stelzner et al. (2019). We refer to the original publication (Stelzner et al., 2019) for a thorough investigation of the capabilities of SuPAIR as a generative model of images, such as its performance when dealing with noisy data or complex backgrounds.

5.10 Ablation Study

The subsequent basic ablations to the full model were performed to investigate the assumptions of the STOVE architecture: STOVE was trained (a) without the reuse of the dynamics networks, (b) without the inclusion of velocities in the latent, and (c) without the inclusion of the unstructured latent. Ablation (a) was realised by constructing two identical but separate (no weight sharing) dynamics networks, one for inference of $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ and one for generation of $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ as is common in other approaches. Figure 5.14 demonstrates the progression of the rollout position error over training. Table 5.1 gives the total decrease in performance for the ablations. The ablation study suggests that all investigated model choices are beneficial to STOVE's performance. As these experiments contain only single runs, this ablation study should be regarded as preliminary, although the low variance observed for the final position errors of repeated runs of identical STOVE configurations and the high

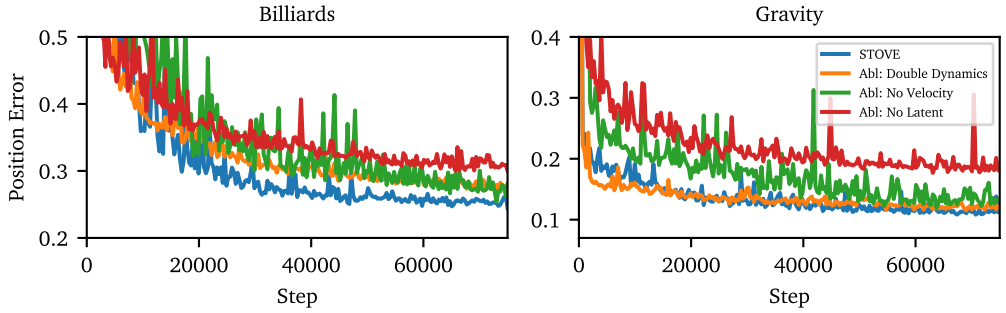


Figure 5.14.: Displayed is the mean predicted position error over a rollout length of 8 frames as training progresses for the billiards (left) and gravity (right) scenario for STOVE and its ablations: STOVE trained (orange) without the reuse of the dynamics networks, (green) without inclusion of velocities in the latent, and (red) without the inclusion of the unstructured latent. All ablated components are likely beneficial to STOVE's performance. A similar figure and caption were first produced for Kossen et al. (2020).

computational efforts of training multiple models somewhat justify this choice. We now offer some thoughts regarding the results of the ablation study.

a) No Dynamics Reuse. Even if the performance of this approach should come close to the original configuration, as is maybe the case for the gravity environment, a single, reused dynamics network should be preferred as it reduces the number of parameters of the model as well as the complexity of the model structure.

b) No Explicit Velocities. The final performance of STOVE without the explicit inclusion of velocities is not far away from the full model, which implies that STOVE learns to account for velocities in the unstructured latent space. Perfect velocity information is valuable to future state prediction. However, the observational velocity included in STOVE is not perfect, as discussed below, and the unstructured latent space likely has to make up some of the difference even for the full STOVE model. It is therefore not completely surprising that the unstructured latent can also learn to account for *all* of the velocity. The authors of Karl et al. (2017a) also show the successful recovery of velocities in a non-linear state space model, albeit for single object scenarios. Training does, however, progress much slower without velocities, justifying the choice of their inclusion.

c) No Unstructured Latent. STOVE's performance suffers worst from the omission of the unstructured latent space in the dynamics prediction. Strictly speaking, object positions and velocities fully describe the state of the billiards and gravity system. But, as discussed before in Chapter 4, the velocities obtained by STOVE are mere approximations to the true velocity. Especially around collisions, these values are imprecise, and additional latent information is necessary to predict the next object state. For example, if an object collides perpendicularly with a wall in the billiards environment, this is observed by STOVE as two snapshots at t_0 and t_1 from the simulation. Let the true component velocities be v_0 before and $v_1 = -v_0$ after the collision. Assuming that the ball moves with constant velocity before the collision, i. e. $x_{-1} = x_0 - v_0 \Delta t$, STOVE's estimate of $v_0 \propto x_0 - x_{-1}$ is accurate. However, at t_1, STOVE will not be able to accurately estimate the velocity v_1 from

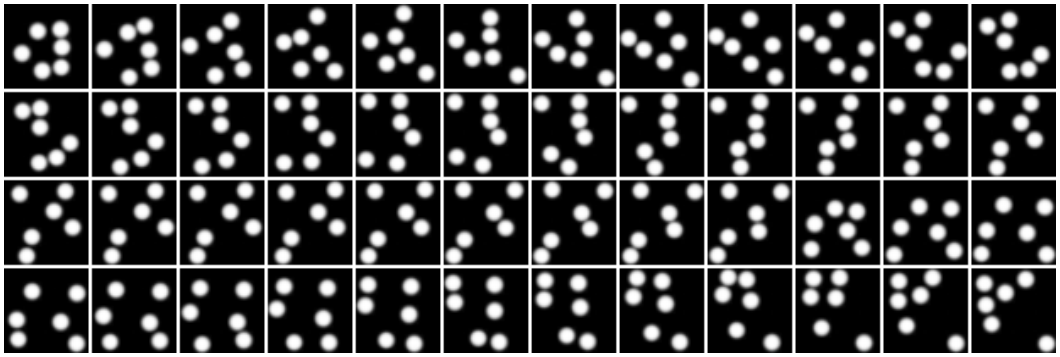


Figure 5.15.: Scaling of STOVE to scenes with six objects. Shown are frames of rollout predicted by STOVE. We display every third frame. Rollouts exhibit realistic collisions at all times, although a decrease in the kinetic energy of the system is noticeable. This is best seen in the animated version available at github.com/jlko/STOVE.

the difference in positions because a collision occurred. A collision between t_0 and t_1 will lead to $x_1 - x_0 \neq \Delta t\, v_1$ because the collision abruptly changes the ball's position and velocity. In the extreme case, the snapshot at t_1 may be taken such, that the ball is in the exact same position before and after the collision, i. e. in the time $t' \in [t_0, t_1]$ the ball moves away from its original position, collides with the wall, and then moves back to its original position. An analysis of the billiards data does in fact show these patterns. STOVE's estimates of the velocities are tightly linked to the position differences observed by SuPAIR. As $x_1 = x_0$ it follows that STOVE's estimate of v_1 must incorrectly conclude $v_1 \propto x_1 - x_0 = 0$. While the approximated velocity information is generally helpful for predicting future states, it provides wrong information around collisions, which demonstrates the need for an abstract latent space. This abstract latent may at t_1 choose to remember v_0 from t_0 and help the dynamics network to predict the collision $v_1 = -v_0$ correctly. The above argument for ball-wall collisions holds similarly for collisions between balls. This line of reasoning should also make it clear, why we do not predict new positions from offsets given by the observed velocities in STOVE, but rather opt for the more general formulation described in Chapter 4.

5.11 Scaling to Additional Objects

Lastly, we briefly explore the scaling capabilities of STOVE to scenes with more objects. We create a new billiards data set with six objects, increasing the resolution to 50 by 50 pixels to fit all objects on the screen. Adaptations to STOVE were necessary to model this scenario successfully: The exact matching routine between timesteps was replaced with a greedy matching routine, which is computationally cheaper. However, the main bottleneck for STOVE in this scenario seems to be the image model, which tends to merge nearby objects, predicting one large bounding box for two or more balls. To avoid this, we restrict the maximum allowable object size to $1/5$ of the image size. We also reintroduce an overlap penalty for the bounding boxes as in Stelzner et al. (2019). With these modifications, STOVE successfully scales to six objects, producing rollouts which do not merge objects and display convincing collision behaviour. Figure 5.15 displays example frames of the obtained model rollouts. A conservation of energy could not be obtained in this scenario, and

the object velocities decrease noticeably during rollouts. The problems with merging objects are likely an indication of the limits of the sequential attention mechanism in the image model. The number of parameters in the dynamics network is independent of the number of objects. However, the size of the tensors propagated through the GNN scales with $\mathcal{O}(O^2)$. While the above scaling test of STOVE may be considered successful, it is also likely that any scaling of STOVE to scenes with more objects or larger canvasses will require further adaptations to the model architecture.

6 Concluding Thoughts

This final chapter contains a closing discussion of the main findings of the thesis with a particular emphasis on resulting directions for future work, as well as a summary.

6.1 Discussion and Future Work

Many of the previous unstructured sequence prediction work focuses on scenarios of high visual dimensionality but with simple dynamics. This remains true even for many of the object-specific approaches such as SQAIR and DDPAE. Conversely, much of the previous work on physics prediction targets low observation dimensionality but complicated dynamics. The novelty and strength of STOVE lies in the combination of high-dimensional visual input and non-linear relational dynamics in an unsupervised fashion.

As observed by Kosiorek et al. (2018), the extension of AIR to videos is natural, and the sequential nature of videos aids the consistent discovery of objects. With the introduction of STOVE, this thesis has demonstrated the successful combination of AIR-like image models and powerful relational physics modelling. Capable transition models are an important component of video modelling. They allow for accurate future state predictions and planning in downstream applications. However, if the transition model underfits the true dynamics of the environment, a model may still obtain decent reconstructions in the simple scenarios presented in this thesis and many other publications. In other words, here, dynamics information is not crucial to obtaining good reconstructions. Nevertheless, in future applications and more complex modelling scenarios, powerful transition models may prove essential for inference of noisy and complex observations. This further motivates the reuse of the dynamics model for both generation and inference. As interactions with the environment are important in almost all real-world applications, we argue for the proliferation of relational reasoning mechanisms in video models.

Many preceding video models have not concentrated on the quality of *rollout predictions*, preferring to investigate reconstructions instead. Furthermore, if they do show rollouts, often, the absence of animated visualisations makes it hard to judge the predictive performance of the model. Likewise, the presented timeframes should be long enough to allow the viewer to come to a conclusion concerning the prediction stability. Additionally, our discussion of the chaoticness of the frequently used billiards environment has exposed subtleties with regards to simple error metrics based on the prediction performance. Most notably, rollout error curves look strikingly similar between models with entirely different dynamics performance. For example, STOVE, with convincing and stable predictions, exhibits error curves closely related to the linear baseline, a most unrealistic model, simply because they have the same

initial observation uncertainty. Therefore, we argue for the inclusion of other quantitative measures, such as physical conservation laws, when applicable to demonstrate the plausibility of the learned model. Alternatively, the model's usefulness may be demonstrated by applying it on downstream tasks such as planning.

A challenge in many object-aware models is the need for careful design of model complexity. If the object models are made too powerful, they may learn to approximate the whole scene instead of single objects. Only by introducing artificial capacity bottlenecks in the object model can we entice the architecture to use the object mechanism. This reliance on bottlenecks to elicit objects is unsatisfactory, and we hope that videos, together with capable transition models, provide a strong enough object prior to break the above failure modes. Our work has provided first steps in this direction. However, so far, reliable object-aware video modelling and future state prediction remain unsolved in scenarios with complex visuals.

SQAIR in particular has a very complex and overparametrised structure, as the authors themselves note (Kosiorek et al., 2018). Where STOVE performs a simple multiplication of distributions, see Eq. (4.6), SQAIR uses general-purpose MLPs that are provided with as much information as available. Similarly contrasting design choices are made in almost all other components of STOVE and SQAIR, see Kosiorek et al. (2018, Algorithm 2-3). While this might make SQAIR hard to train, debug, or interpret, STOVE contains many assumptions which are too restricting for some applications. How can we build models whose structure guides them with useful prior knowledge but does not restrict them? For STOVE, we may have achieved this with respect to the simple physical simulation data. However, it is unclear, how to successfully extend this to more complex settings. We will now present some more concrete directions for future work. They are loosely sorted from *easy to implement and likely to succeed* to *challenging or vague albeit interesting*.

Varying Object Counts. Extending STOVE to accommodate sequences with varying object numbers should be straightforward, as long as the number of objects per sequence is fixed. This extension would reintroduce the total number of objects O as a random variable, and we could proceed similarly as in SuPAIR. If the number of objects should be allowed to vary within the sequence, additional complexity due to modules for re-recognising objects from previous frames and discovering new objects, similar to SQAIR, could likely not be avoided. For an extension to large object counts, parallel attention mechanisms such as in Crawford and Pineau (2019) or Jiang et al. (2020) would need to be adopted.

Object Matching. While the object state matching routine of STOVE is a simple and effective solution, its algorithmic complexity of $\mathcal{O}(O^3)$ may present a limiting factor in future applications. Also, matching states based on Euclidean position distances is not desirable when objects overlap heavily. Since the conception of STOVE, different approaches to handling similar problems have emerged. Inspired by Miladinović et al. (2019), the matching problem could be solved by first creating state *proposals* from an unchanged dynamics model. Then, the image model could get these proposals as input and predict correctional *offsets* to the proposed states, conditioned on both the proposal and the observation. An explicit matching is not required anymore, because the image model is aware of the object order proposed by the dynamics model. Veerapaneni et al. (2019) proceed similarly, using the image model to refine predictions from the dynamics model. Instead of directly predicting offsets to

states, Veerapaneni et al. (2019) refine parameter guesses using iterative variational inference. By entangling the image and dynamics model, we could proceed similarly to the above for STOVE and obviate the need for a matching routine. However, the trade-off between matching complexity and component entanglement needs to be investigated.

Extensions to the SSM. The position estimates obtained by the SuPAIR image model of STOVE are completely independent of all previous observations. While these strict independence assumptions are first and foremost a strength of STOVE and lead to a fast inference routine, it is likely that the inclusion of smoothing or filtering mechanisms is necessary for scenarios with high observation noise or visual complexity. Section 2.3 has introduced many ways of constructing such non-linear state space models. Even VRNN, which is an early example of such models and which has flaws that have been discussed by subsequent work, has shown somewhat admirable performance in Chapter 5 on complex data that it was not designed for. We are therefore confident that the approaches of Section 2.3 would likely yield strong results when adapted to object-based image modelling and relational physics. Because these approaches already expect highly non-linear emission and transition distributions, the extension of STOVE to such models should be straightforward. That being said, the discussions in Section 2.3 and Karl et al. (2017a) reveal that the interaction of appearance and dynamics modelling is non-trivial. Thus, approaches such as Karl et al. (2017a) or Fraccaro et al. (2017), which can regularise the use of the dynamics model, are particularly promising.

Conservation of Energy. Section 5.5 has shown that STOVE correctly conserves the average kinetic energy of the rollouts. Unfortunately, we can only offer speculation as to how STOVE achieves this conservation and must leave everything else for the future. More importantly, future work should focus on building models which *reliably* achieve a conservation of relevant physical quantities. Neural ODEs, introduced in Chapter 3, offer a promising path towards specifying models, which, depending on the exact specification, may conserve relevant quantities by design. Successfully applying these models to videos of discontinuous processes, such as the billiards environment, or more general and complex dynamics has not been done, but is a fruitful avenue for future work.

More Complex Object Models. The SuPAIR-like image model in STOVE is derived from AIR and therefore limited to rectangular object bounding boxes. While SuPAIR already presents some improvements over AIR, such approaches are not appropriate for more complex data encountered in real-world applications. In Chapter 3, we introduced MONet and follow-up work, which presents a more flexible approach to object-aware image modelling, predicting pixel-wise object masks. Extensions of such models to relational video prediction are interesting future work. After the publication of Kossen et al. (2020), Voelcker (2020) performed experiments with an adaptation of STOVE to MONet-like image models. While the results are promising, in general, the approach is brittle and requires careful hyperparameter tuning. Unlike STOVE, MONet features object-wise unstructured latents, which leads to the entangling of appearance and dynamics information in the latent space. This may be a reason for the observed instabilities and, therefore, the approaches of Section 2.3 may provide inspiration on how to move forward. Other recent work by Veerapaneni et al. (2019) or Watters et al. (2019) extends MONet to discrete

dynamics, e. g. block-stacking tasks or moving sprites. Scaling object-aware image models and relational physics modules to realistic videos is an excellent challenge for future work.

Low-Dimensional Dynamics. For applications in reinforcement learning or robotics, full-scale video modelling of future *frames* in complex visual settings might not be the only research direction. Depending on the application, it might be preferable to directly focus dynamics modelling efforts on lower-dimensional latents, from which quantities relevant to acting in the environment can directly be predicted, similar to Dosovitskiy and Koltun (2017), Farquhar et al. (2018), or Schrittwieser et al. (2019). These quantities should be easier to model than high-dimensional visuals. Here, relational dynamics models could prove beneficial to obtaining good transitional models. However, it is not always obvious, what should constitute an object in such abstract transition models. A possible answer to this question is provided by Kipf et al. (2020), who learn a useful object-specific latent embedding without relying on a reconstruction loss.

General Relational Prediction. The application of Graph Neural Networks (GNNs) for learning physics simulations is well-established. However, in Section 5.4, we were able to show that, excitingly, the GNN continues to perform well even when not all objects follow the same dynamical laws. This could pave the way for relational reasoning modules for more general kinds of interactions. In the real world or video games, object trajectories are often determined by interactions with the environment or between objects. Relational reasoning modules in object-aware video models may provide a way to efficiently learn about these interactions, even when they are more abstract than physical laws and dynamics are allowed to differ between objects. For example, future models may be able to learn the interactions that govern the movement of objects in ATARI video games. In turn, knowledge of these interactions is then invaluable for acting in the environment. While our initial experiments in Section 5.4 proved successful without any modifications to the architecture, it seems likely that modifications to the GNN are needed in more complex settings. For example, one could explicitly introduce latent variables that assign different dynamics submodules, inspired by work on continual or multi-task learning. Alternatively, the modelling choice could be made implicit by including additional attention or gating mechanisms, which let the GNN infer the appropriate dynamics mode between object pairs. A similar approach is chosen in Ehrhardt et al. (2019), albeit not in the context of relational physics prediction.

6.2 Summary

Objects and intuitive physics are of central importance to human reasoning. With the motivation to bring such capabilities to machine learning models, we have introduced STOVE. STOVE presents a structured, object-aware approach to video modelling, extending attend-infer-repeat models to videos of physically interacting objects. We assemble STOVE as a non-linear state space model: The transition distribution is given by a relational physics module, a graph neural network, and the emission distribution is realised with a SuPAIR-like image module. STOVE is trained in a self-supervised fashion directly from pixels. Unlike generic approaches, STOVE makes use of a variety of structuring assumptions, such as the object-awareness, explicit positions, velocities, and object sizes in the latent space, the reuse of the

generative dynamics model for inference, as well as the structural assumptions imposed by the graph neural network. These assumptions seem to empower rather than restrict STOVE, as the experimental evaluation has shown clear improvements over previous, less structured video models. In addition, STOVE produces remarkably stable rollouts, which correctly conserve the kinetic energy of the simulation in the billiards scenario. There, investigations into the chaoticness of the environment suggest that STOVE is close to optimal performance considering the resolution of the data.

Many exciting directions for future work have been presented. While some strive for increased visual modelling complexity, others attempt generalisations of the dynamics model. However, they all share the overarching vision that object-awareness and relational reasoning present important inductive biases. They should be seen as general-purpose mechanisms with broad relevancy for many areas of machine learning. STOVE presents a valuable contribution to this vision.

A Model Details[1]

A.1 Recognition Model Architecture

The object detection network for $q(\mathbf{z}_{t,\,(\text{pos, size})} \mid \mathbf{x}_t)$ is realised by an LSTM (Hochreiter and Schmidhuber, 1997) with 256 hidden units, which outputs the means and variances of the object's latent state, i. e. $2 \cdot 2 \cdot 2 = 8$ parameters per object.

The Random SPN for the objects is created with parameters $(R = 6, D = 2, S = 10, C = 1, I = 10)$, see Section 2.4. Similarly, the Random SPN for the background is created with parameters $(R = 3, D = 1, S = 6, C = 1, I = 3)$. Leaf distributions are modelled as factorised Gaussians, where means and variances are learnable parameters. The structure for the background SPN is simpler to avoid STOVE relying entirely on the background model for reconstructions, neglecting object-abstractions entirely, see Stelzner et al. (2019). The above values are similar to those in Stelzner et al. (2019) and are valid for all experiments of the thesis. Appendix A.4 lists further hyperparameter choices.

A.2 Generative Model Architecture

For $\mathbf{z}^o_{t,\text{latent}}$, we choose dimensionality 12, such that a full state

$$\mathbf{z}^o_t = \left(\mathbf{z}^o_{t,\,\text{pos}}, \mathbf{z}^o_{t,\,\text{size}}, \mathbf{z}^o_{t,\,\text{velo}}, \mathbf{z}^o_{t,\,\text{latent}}\right) \tag{A.1}$$

is 18-dimensional.

The dynamics prediction is given by the following series of transformations applied to each input state of shape (N_b, O, l), where N_b is the batch size, O the number of objects, and $L = 16$ the total size of the latent space per object for the dynamics prediction without the *size* parameters.

- S_1: Encode input state with linear layer $[L, 2L]$.
- S_2: Apply linear layer $[2L, 2L]$ to S_1 followed by ReLU non-linearity.
- S_3: Apply linear layer $[2L, 2L]$ to S_2 and add result to S_2. This gives the dynamics prediction without relational effects, corresponding to $N(\mathbf{z}^o_t)$ in Eq. (4.3).
- C_1: The following steps obtain the relational aspects of dynamics prediction, corresponding to $I(\mathbf{z}^o_t, \mathbf{z}^{o'}_t)$ in Eq. (4.3). Concatenate the encoded state S^o_1 pairwise with all state encodings, yielding a tensor of shape $(N_b, O, O, 4L)$.
- C_2: Apply linear layer $[4L, 4L]$ to C_1 followed by ReLU.
- C_3: Apply linear layer $[4L, 2L]$ to C_2 followed by ReLU.
- C_4: Apply linear layer $[2L, 2L]$ to C_3 and add to C_3.

[1] Appendices A.1 to A.3 appear in similar form in the concurrently published Kossen et al. (2020).

- A_1: To obtain attention coefficients $\alpha(\mathbf{z}_t^o, \mathbf{z}_t^{o'})$, apply linear layer $[4L, 4L]$ to C_1 followed by ReLU.
- A_2: Apply linear layer $[4L, 2L]$ to A_1 followed by ReLU.
- A_3: Apply linear layer $[2L, 1]$ to A_2 and apply exponential function.
- R_1: Multiply C_4 with A_3, where diagonal elements of A_3 are masked out to ensure that R_1 only covers cases where $o \neq o'$.
- R_2: Sum over R_1 for all o', to obtain tensor of shape $(N_b, O, 2L)$. This is the relational dynamics prediction.
- D_1: Sum relational dynamics R_2 and self-dynamics S_3, obtaining the input to A in Eq. (4.3).
- D_2: Apply linear layer $[2L, 2L]$ to D_1 followed by tanh non-linearity.
- D_3: Apply linear layer $[2L, 2L]$ to D_2 followed by tanh non-linearity and add result to D_2.
- D_4: Concatenate D_3 and S_1, see U in Eq. (4.3), and apply linear layer $[4L, 2L]$ followed by tanh.
- D_5: Apply linear layer $[2L, 2L]$ to D_4 and add result to D_4 to obtain the final dynamics prediction.

The output D_5 has shape $(N_b, O, 2L)$, twice the size of L to account for the variances of the variational state prediction $q(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$. For the generative dynamics $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$, the predicted variances are ignored and fixed ones used instead.

For the model-based control scenario, the one-hot encoded actions of shape (N_b, A), where A is the number of actions, are transformed with a linear layer $[A, O \cdot E]$, where E is the size of the action-encoding, and reshaped to (A, O, E). We set $A = 9$ and $E = 4$ for the experiments in this thesis. The action embedding and the object appearances $(N_b, O, 3)$ are then concatenated to the input state. The rest of the dynamics prediction follows as above. The reward prediction consists of the following steps:

- H_1: Apply linear layer $[2L, 2L]$ to D_1 followed by ReLU.
- H_2: Apply linear layer $[2L, 2L]$ to H_1.
- H_3: Sum over object dimension to obtain tensor of shape $(N_b, 2L)$.
- H_4: Apply linear layer $[2L, L]$ to H_3 followed by ReLU.
- H_5: Apply linear layer $[L, L/2]$ to H_4 followed by ReLU.
- H_5: Apply linear layer $[L/2, 1]$ to H_4 followed by a sigmoid non-linearity.

H_5 then gives the final reward prediction.

A.3 Model Initialisation

In the first two timesteps, we cannot yet apply STOVE's main inference step $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1})$ as described above. In order to initialise the latent state over the first two frames, we apply a simplified architecture and only use a partial state at $t = 0$.

At $t = 0$, $\mathbf{z}_0 \sim q(\mathbf{z}_{0,\,(\text{pos, size})} \mid \mathbf{x}_0)$ is given purely by the image model, since no previous states which could be propagated, exist. This initial $\mathbf{z}_0$ is incomplete insofar as it does not contain velocity information or unstructured latents. At $t = 1$, $q(\mathbf{z}_{1,\,(\text{pos, size})} \mid \mathbf{x}_1, \mathbf{x}_0)$ is still given purely based on the object attention network. Note that for a dynamics prediction of $\mathbf{z}_1$, velocity information at $t = 0$ would need to be available. However, at $t = 1$, velocities can be constructed based on

the differences between the previously inferred object positions. We sample $\mathbf{z}_{1,\text{latent}}$ from the normally distributed prior to assemble the first full initial state $\mathbf{z}_1$. At $t \geq 2$, the full inference network can be run: States are inferred both from the attention network $q(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1})$ as well as propagated using the dynamics model $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$.

In the generative model, similar adjustments are made: $p(\mathbf{z}_{0,\,(\text{pos, size})})$ is given by a uniform prior, velocities and latents are omitted. At $t = 1$, velocities are sampled from a uniform distribution in planar coordinates $p(\mathbf{z}_{1,\,\text{velo}})$ and positions are given by a simple linear dynamics model

$$p(\mathbf{z}_{1,\text{pos}} \mid \mathbf{z}_{0,\text{pos}}, \mathbf{z}_{1,\text{velo}}) = \mathcal{N}(\mathbf{z}_{0,\text{pos}} + \mathbf{z}_{1,\text{velo}}, \boldsymbol{\sigma}^2)\,. \tag{A.2}$$

Latents $\mathbf{z}_{1,\text{latent}}$ are sampled from a Gaussian prior. Starting at $t = 2$, the full dynamics model is used.

A.4 Hyperparameter Settings

The following list gives the default settings for hyperparameters in STOVE. They are also part of `config.py` in the provided code base.

Data Parameters:

```
num_visible = 8  # Number of visible frames per sequence.
num_episodes = 1000  # Number of generated sequences.
num_frames = 100  # Number of frames per sequence.
channels = 1  # Use black and white images.
num_obj = 3  # Number of objects.
```

Training Configuration:

```
batch_size = 256
cl = 32 # Twice the latent space dimension for the dynamics model.
# See adjust_learning_rate() in train.py for further information.
learning_rate = 0.002  # Inital and maximum learning rate.
min_learning_rate = 0.0002  # Minimum learning rate, see above.
debug_anneal_lr = 40000.0  # Learning rate decrease, see above.
num_epochs = 400 # Total number of epochs.
debug_amsgrad = True # Fix Adam gradients.
```

Model Config: Action-Conditioned:

```
# Reward trade-off for combined loss, inverse to variance on reward dist.
debug_reward_factor = 15000
# No. of steps during which reward loss linearly increases.
debug_reward_rampup = 20000
# Add object appearance to dynamics prediction.
debug_core_appearance = False
debug_appearance_dim = 3
```

Model Config – Dynamics:

```
# Standard deviations on the generative dynamics model.
transition_lik_std = [0.01, 0.01, 0.01, 0.01]
debug_nonlinear = 'relu'  # Nonlinearity used in dynamics core.
# Bound on inferred standard deviation on latents.
debug_latent_q_std = 0.04
```

Model Config – SPN:

```
debug_bw = True  # Do not model colours in SPN.
# No. of pixels used to model objects in SPN.
patch_height = 10
patch_width = 10
# Bounds on variance for leaves in object-SPN.
obj_min_var = 0.12
obj_max_var = 0.35
# Bounds on variance for leaves in background-SPN.
bg_min_var = 0.002
bg_max_var = 0.16
# Maximum object variances for SuPAIR q-distributions.
scale_var = 0.3
pos_var = 0.3
# Bounds for mean of width of bounding box relative to canvas width.
min_obj_scale = 0.1
max_obj_scale = 0.8
# Bounds for mean of height of bounding box relative to width.
min_y_scale = 0.75
max_y_scale = 1.25
# Max and min object position relative to canvas.
obj_pos_bound = 0.9
# Number of leaf distributions and sums.
obj_spn_num_gauss = 10
obj_spn_num_sums = 10
```

B Baselines[1]

Following Kosiorek et al. (2018), we experimented with different hyperparameter configurations for VRNNs. We varied the sizes of the hidden and latent states $[h, z]$, experimenting with the values $[256, 16]$, $[512, 32]$, $[1024, 64]$, and $[2048, 32]$. We found that increasing the model capacity beyond $[512, 32]$ did not yield large increases in performance, which is why we chose the configuration $[512, 32]$ for our experiments. Our VRNN implementation is written in PyTorch and based on github.com/emited/VariationalRecurrentNeuralNetwork. VRNN and all models below were trained until convergence.

SQAIR can handle a variable number of objects in each sequence. However, to allow for a fairer comparison to STOVE, we fixed the number of objects to the correct number. This means that in the first timestep, exactly three objects are discovered, which are then propagated in all following timesteps, without further discoveries. Our implementation is based on the original implementation provided by the authors at github.com/akosiorek/sqair. Experiments with SQAIR in this thesis were performed by Karl Stelzner. We thank Adam Kosiorek for his help with the implementation.

The DDPAE experiments were performed using the implementation available at github.com/jthsieh/DDPAE-video-prediction from the original authors. Default parameters for training DDPAE with the billiards datasets are provided by the authors. However, the resolutions of our billiards (32 pixels) and gravity (50 pixels) data are different from the resolution DDPAE expects (64 pixels). While we experimented with adjusting DDPAE parameters, such as the latent space dimension, to fit our different resolutions, the best results were obtained when bilinearly scaling our data to the resolution DDPAE expects. DDPAE was trained for $400\,000$ steps, which sufficed for convergence of the models' test set error.

The linear baseline was obtained as follows: For the first 8 frames, we infer the full model state using a fully trained STOVE model. We then take the last inferred positions and velocities of each object and predict future positions by assuming constant uniform motions for each object. We do not allow objects to leave the frame, i. e. when objects reach the canvas boundary after some timesteps, they stick to it.

Since our dynamics model requires only object positions and velocities as input, it is trivial to construct a supervised baseline for physics prediction by replacing the SuPAIR-inferred states with real, ground-truth states from the environment. On these, the model can then be trained in a supervised fashion. As with STOVE, we allow the supervised baseline extra modelling capabilities by introducing unstructured latent

[1] Chapter appears in similar form in the concurrently published Kossen et al. (2020).

space in addition to the ground-truth input. The loss function per input sequence is given by

$$\frac{1}{T}\sum_{t=1}^{T}\gamma^{t}\left\|\tilde{\mathbf{z}}_{t}-\mathbf{z}_{t}\right\|^{2}, \tag{B.1}$$

where $\tilde{\mathbf{z}}_t$ are position and velocity predictions for all objects, $\mathbf{z}_t$ true states, $\gamma = 0.98$ exponentially discounts predictions, and $T = 8$ as for the full STOVE.

Acronyms

AESMC Auto-Encoding Sequential Monte Carlo.

AIR Attend-Infer-Repeat.

ARMA Autoregressive-Moving Average.

BRNN Bidirectional Recurrent Neural Network.

CNN Convolutional Neural Network.

ConvLSTM Convolutional LSTM.

DDPAE Decompositional Disentangled Predictive Autoencoder.

DKF Deep Kalman Filter.

DMM Deep Markov Model.

DVBF Deep Variational Bayes.

ELBO Evidence Lower Bound.

EM Expectation Maximisation.

GAN Generative Adversarial Network.

GNN Graph Neural Network.

GRU Gated Recurrent Unit.

HMM Hidden Markov Model.

IN Interaction Network.

IWAE Importance-Weighted Autoencoder.

KL divergence Kullback-Leibler Divergence.

KVAE Kalman Variational Autoencoder.

LDS Linear Dynamical System.

LG-SSM Linear Gaussian State-Space Model.

LHS Left-Hand Side.

LSTM Long-Short Term Memory.

MCMC Markov Chain Monte Carlo.

MCTS Monte-Carlo Tree Search.

MDN Mixture Density Network.

MFVI Mean-field Variational Inference.

MLP Multilayer Perceptron.

MONet Multi-Object Network.

MPE Most Probable Explanation.

MSE Mean Squared Error.

NEM Neural Expectation Maximisation.

Neural ODE Neural Ordinary Differential Equation.

NPE Neural Prediction Engine.

NTM Neural Turing Machine.

PGM Probabilistic Graphical Model.

R-NEM Relational Neural Expectation Maximisation.

Random SPN Random Sum-Product Network.

RHS Right-Hand Side.

RN Relation Network.

RNN Recurrent Neural Network.

SGD Stochastic Gradient Descent.

SGVB Stochastic Gradient Variational Bayes.

SPN Sum-Product Network.

SQAIR Sequential Attend-Infer-Repeat.

SRNN Stochastic Recurrent Neural Network.

SSM State-Space Model.

STORN Stochastic Recurrent Network.

STOVE Structured Object-Aware Physics Prediction for Video Modelling and Planning.

SuPAIR Sum-Product Attend-Infer-Repeat.

SVI Stochastic Variational Inference.

VAE Variational Autoencoder.

VI Variational Inference.

VIN Visual Interaction Network.

VRNN Variational Recurrent Neural Network.

Bibliography

T. Adel, D. Balduzzi, and A. Ghodsi. Learning the structure of sum-product networks via an svd-based algorithm. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 32–41, 2015.

P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

A. Akhundov, M. Soelch, J. Bayer, and P. van der Smagt. Variational tracking and prediction with generative disentangled state-space models. *arXiv preprint arXiv:1910.06205*, 2019.

M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. In *Proceedings of the International Conference on Learning Representations*, 2018.

D. Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016.

P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110 (45):18327–18332, 2013.

J. Bayer and C. Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

P. Becker-Ehmck, J. Peters, and P. Van Der Smagt. Switching linear dynamics for variational bayes filtering. In *Proceedings of the International Conference on Machine Learning*, pages 553–562, 2019.

C. M. Bishop. Mixture density networks. *Neural Computing Research Group Report*, 1994.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

M. Bonse. Machine larning based atmosphere prediction for extreme adaptive optics. *Master Thesis, Technical University Darmstadt*, 2019.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. In *Neural Information Processing Systems Workshop on Learning Disentangled Representations*, 2017.

C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *Proceedings of the International Conference on Learning Representations*, 2017.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

W.-C. Cheng, S. Kok, H. V. Pham, H. L. Chieu, and K. M. A. Chai. Language modeling with sum-product networks. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

F. Chollet. The measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Neural Information Processing Systems Workshop on Learning Disentangled Representations*, 2014.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pages 2980–2988, 2015.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of International Conference on Computers and Games*, pages 72–83. Springer, 2006.

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. In *International Conference on Learning Representations Workshop on Deep Differential Equations Workshop*, 2020.

E. Crawford and J. Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420, 2019.

A. Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305, 2003.

P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

J. DeCruyenaere and H. Hafez. A comparison between kalman filters and recurrent neural networks. In *IJCNN International Joint Conference on Neural Networks*, volume 4, pages 247–251. IEEE, 1992.

A. Dennis and D. Ventura. Learning the architecture of sum-product networks using clustering on variables. In *Advances in Neural Information Processing Systems*, pages 2033–2041, 2012.

E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *Proceedings of the International Conference on Machine Learning*, pages 1174–1183, 2018.

C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. In *Proceedings of the International Conference on Learning Representations*, 2017.

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt,

D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2758–2766, 2015.

D. Durstewitz. Advanced data analysis in neuroscience. *Springer International Publishing*, 2017.

S. Ehrhardt, A. Monszpart, N. Mitra, and A. Vedaldi. Unsupervised intuitive physics from visual observations. In *Proceedings of the Asian Conference on Computer Vision*, pages 700–716. Springer, 2018.

S. Ehrhardt, A. Monszpart, N. J. Mitra, and A. Vedaldi. Unsupervised intuitive physics from past experiences. *arXiv preprint arXiv:1905.10793*, 2019.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

M. Engelcke, A. R. Kosiorek, O. P. Jones, and I. Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *Proceedings of the International Conference on Learning Representations*, 2020.

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

O. Fabius and J. R. van Amersfoort. Variational recurrent auto-encoders. In *International Conference on Machine Learning Workshop Track*, 2015.

G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2018.

S. Farquhar, M. Osborne, and Y. Gal. Radial bayesian neural networks: Robust variational inference in big models. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2020.

C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2016.

M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pages 2199–2207, 2016.

M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. In *Proceedings of the International Conference on Learning Representations*, 2016.

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. *arXiv preprint arXiv:1907.12887*, 2019.

R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 3239–3247, 2012.

R. Gens and D. Pedro. Learning the structure of sum-product networks. In *International Conference on Machine Learning*, pages 873–880, 2013.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. In *Proceedings of International Conference on Artificial Neural Networks*. IET, 1999.

S. Gershman and N. Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2014.

V. Goel, J. Weng, and P. Poupart. Unsupervised video object segmentation for deep

reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5683–5694, 2018.
I. Goodfellow. Generative adversarial networks. In *Advances in Neural Information Processing Systems Tutorial*, 2016.
I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
K. Greff, S. Van Steenkiste, and J. Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6691–6701, 2017.
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 *Proceedings of the International Conference on Learning Representations*, pages 2424–2433, 2019.
K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the International Conference on Machine Learning*, pages 1462–1471, 2015.
U. Grenander. *Lectures in Pattern Theory: Vol. 2 Pattern Analysis*. Springer-Verlag, 1976.
S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15353–15363, 2019.
J. Hamrick, P. Battaglia, and J. B. Tenenbaum. Internal physics models guide probabilistic judgments about object dynamics. In *Proceedings of the 33rd annual conference of the cognitive science society*, volume 2. Cognitive Science Society Austin, TX, 2011.
I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations*, 2017.
S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.
S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.
S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.
M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles. Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526, 2018.
M. Hussing. Object-aware state representation learning. *Master Thesis, Technical University Darmstadt*, 2020.
M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.
M. Jaques, M. Burke, and T. Hospedales. Physics-as-inverse-graphics: Joint unsupervised learning of objects and physics from video. In *Proceedings of the International Conference on Learning Representations*, 2020.
J. L. W. V. Jensen et al. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193, 1906.
J. Jia and A. R. Benson. Neural jump stochastic differential equations. In *Advances*

in Neural Information Processing Systems, pages 9843–9854, 2019.

J. Jiang, S. Janghorbani, G. De Melo, and S. Ahn. Scalor: Generative world models with scalable object representations. In *Proceedings of the International Conference on Learning Representations*, 2020.

M. J. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pages 2946–2954, 2016.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. In *Proceedings of the International Conference on Learning Representations*, 2020.

N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. In *Proceedings of the International Conference on Machine Learning-Volume 70*, pages 1771–1779, 2017.

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 *Proceedings of the International Conference on Learning Representations*, 2017a.

M. Karl, M. Soelch, P. Becker-Ehmck, D. Benbouzid, P. van der Smagt, and J. Bayer. Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv:1710.05101*, 2017b.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*, 2014.

T. Kipf, E. van der Pol, and M. Welling. Contrastive learning of structured world models. In *Proceedings of the International Conference on Learning Representations*, 2020.

I. Kobyzev, S. Prince, and M. A. Brubaker. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*, 2019.

J. Z. Kolter and G. Manek. Learning stable deep dynamics models. In *Advances in Neural Information Processing Systems*, pages 11126–11134, 2019.

A. Kosiorek, A. Bewley, and I. Posner. Hierarchical attentive recurrent tracking. In *Advances in Neural Information Processing Systems*, pages 3053–3061, 2017.

A. 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*, pages 8606–8616, 2018.

J. Kossen, K. Stelzner, M. Hussing, C. Voelcker, and K. Kersting. Structured object-aware physics prediction for video modeling and planning. In *Proceedings of the International Conference on Learning Representations*, 2020.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

R. G. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma. Videoflow: A conditional flow-based model for stochastic video generation. In *Proceedings of the International Conference on Learning Representations*, 2020.

B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-encoding sequential monte carlo. In *Proceedings of the International Conference on Learning Representations*, 2018.

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

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

S. Legg and M. Hutter. A formal measure of machine intelligence. In *Proceedings of 15th Annual Machine Learning Conference of Belgium and The Netherlands*, 2006.

T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *Proceedings of the International Conference on Learning Representations*, 2019.

Z. Lin, Y.-F. Wu, S. V. Peri, W. Sun, G. Singh, F. Deng, J. Jiang, and S. Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *Proceedings of the International Conference on Learning Representations*, 2020.

Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4463–4471, 2017.

M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019.

C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6573–6583, 2017.

J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *Proceedings of the International Conference on Machine Learning*, pages 3403–3412, 2019.

E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling disentanglement in variational autoencoders. In *Proceedings of the International Conference on Learning Representations*, 2019.

M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *Proceedings of the International Conference on Learning Representations*, 2016.

M. McCloskey. Intuitive physics. *Scientific american*, 248(4):122–131, 1983.

M. Mézard, G. Parisi, and M. Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.

Đ. Miladinović, W. Gondal, B. Schölkopf, J. M. Buhmann, and S. Bauer. Disentangled state space models: Unsupervised learnign of dynamics across heterogenous

environments. In *International Conference on Learning Representations Workshop*, 2019.

V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

A. Molina, A. Vergari, N. Di Mauro, S. Natarajan, F. Esposito, and K. Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

A. Molina, A. Vergari, K. Stelzner, R. Peharz, P. Subramani, N. D. Mauro, P. Poupart, and K. Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.

G. Papamakarios. Neural density estimation and likelihood-free inference. *arXiv preprint arXiv:1910.13233*, 2019.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

R. Peharz, G. Kapeller, P. Mowlaee, and F. Pernkopf. Modeling speech with sum-product networks: Application to bandwidth extension. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3699–3703. IEEE, 2014.

R. Peharz, S. Tschiatschek, F. Pernkopf, and P. Domingos. On theoretical properties of sum-product networks. In *Artificial Intelligence and Statistics*, pages 744–752, 2015.

R. Peharz, A. Vergari, K. Stelzner, A. Molina, M. Trapp, K. Kersting, and Z. Ghahramani. Probabilistic deep learning using random sum-product networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2019.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *IEEE International Conference on Computer Vision Workshops*, pages 689–690. IEEE, 2011.

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 *Proceedings of the International Conference on Machine Learning*, pages 116–124, 2018.

P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems*, pages 68–80, 2019.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *Proceedings of the International Conference on Machine Learning*, pages 5389–5400, 2019.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1278–1286, 2014.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015.

H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

A. N. Sanborn, V. K. Mansinghka, and T. L. Griffiths. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review*, 120(2):411, 2013.

A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of the International Conference on Machine Learning*, pages 4470–4479, 2018.

A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks. *arXiv preprint arXiv:2002.09405*, 2020.

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, pages 4967–4976, 2017.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

P. Sorrenson, C. Rother, and U. Köthe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). In *Proceedings of the International Conference on Learning Representations*, 2020.

N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *Proceedings of the International Conference on Machine Learning*, pages 843–852, 2015.

K. Stelzner, R. Peharz, and K. Kersting. Faster attend-infer-repeat with tractable probabilistic models. In *Proceedings of the International Conference on Machine Learning*, 2019.

I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2009.

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Ab-

dolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins. Hamiltonian generative networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

M. Tschannen, O. Bachem, and M. Lucic. Recent advances in autoencoder-based representation learning. In *Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2018.

A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

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 *Proceedings of the International Conference on Robot Learning*, 2019.

A. Vergari, R. Peharz, N. Di Mauro, A. Molina, K. Kersting, and F. Esposito. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

A. Vergari, N. Di Mauro, and F. Esposito. Visualizing and understanding sum-product networks. *Machine Learning*, 108(4):551–573, 2019.

C. Voelcker. Unsupervised object detection and sequence modeling for control. *Master Thesis, Technical University Darmstadt*, 2020.

C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems*, pages 613–621, 2016.

E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158. IEEE, 2000.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, pages 2746–2754, 2015.

N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems*, pages 4539–4547, 2017.

N. Watters, L. Matthey, M. Bosnjak, C. P. Burgess, and A. Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

D. Weissenborn, O. Täckström, and J. Uszkoreit. Scaling autoregressive video models. In *Proceedings of the International Conference on Learning Representations*, 2019.

A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt. Deterministic variational inference for robust bayesian neural networks. In *Proceedings of the International Conference on Learning Representations*, 2019a.

J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems*, pages 127–135, 2015.

J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *Proceedings of the British Machine Vision Conference*, 2016.

J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via

visual de-animation. In *Advances in Neural Information Processing Systems*, pages 153–164, 2017.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019b.

S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

W. Zaremba and I. Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

P. D. Zelazo and S. P. Johnson. *Object Perception*. Oxford University Press, 2013.

C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in variational inference. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 41, pages 2008–2026. IEEE, 2018.

H. Zhao, M. Melibari, and P. Poupart. On the relationship between sum-product networks and bayesian networks. In *Proceedings of the International Conference on Machine Learning*, pages 116–124, 2015.

S. Zhao, J. Song, and S. Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5885–5892, 2019.

D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum. Unsupervised learning of latent physical properties using perception-prediction networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018.

J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Declaration

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Berlin, den 20. April 2020 .....

Jannik Lukas Kossen