Deep Generative Models of Images and Video

by

Emily Denton

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2018

Professor Rob Fergus

Dedication

To my best friend and partner Miles Rees-Spear.

Acknowledgements

First and foremost, I would like to thank my advisor, Rob Fergus, for his continuous support and guidance throughout my Ph.D. I have greatly enjoyed working with Rob over the past 5 years and am sincerely grateful for his patience, immense knowledge and encouragement. Besides my advisor, I would like to thank the rest of my committee: Kyunghyun Cho, Yann LeCun, Arthur Szlam and Marc'Aurelio Ranzato. I am especially grateful for Kyunghyun Cho's constructive and valuable feedback on my thesis draft.

I have met many brilliant mentors, collaborators and colleagues through my time at the NYU CILVR lab, Facebook AI Research and DeepMind. Special thanks to Sainbayar Sukhbaatar, William Whitney, Ross Goroshin, Michael Mathieu, Mikael Henaff, Alexander Rives, Jake Zhao, Martin Arjovsky, Roberta Raileanu, Cinjon Resnick, Wojciech Zaremba, Jordan Ash, Soumith Chintala, Jason Weston, Simon Osindero, and Oriol Vinyals for many stimulating discussions.

I would like to thank my family for their encouragement and support. I am especially grateful to my father for his continued confidence in me. Thanks to my partner, Miles, for his continued love, support and understanding through the course of my Ph.D. This work would not have been possible without him and I am forever grateful for his patience and encouragement and for keeping me well fed during deadline crunches.

Finally, I grateful for the financial support of an NSERC fellowship and a Google Ph.D fellowship.

Preface

The chapters 4, 5 and 6 of this thesis appeared in the following publications respectively:

- Denton, E., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*
- Denton, E. and Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems (NIPS)*
- Denton, E. and Fergus, R. (2018). Stochastic video generation with a learned prior. In *International Conference on Machine Learning (ICML)*

Source code can be downloaded from:

- `https://github.com/edenton/drnet-py`
- `https://github.com/edenton/svg`

Abstract

Deep neural networks have seen wide success in the supervised setting in recent years. Many of these successes rely heavily on large training sets of manually annotated data. Given the difficulty of obtaining enough labeled data to scale many deep learning approaches, it is increasingly important to look for better methods of utilizing large amounts of unlabeled data.

Building generative models of images and video is a fundamental paradigm of learning from unlabeled data. Unsupervised criterion based on generating or reconstructing images drive many representation learning frameworks. Video is a particularly appealing domain for unsupervised learning due to the inherent temporal structure of the data. This structure lends itself to representation learning approaches based on extracting invariances and predicting future frames, given the past.

Additionally, building accurate models of the world that facilitate future prediction can be useful for model based reinforcement learning, planning, and more generally, endowing an agent with the capacity to reason about its environment. Incorporating predictive models can potentially help alleviate the sample inefficiency of many reinforcement learning systems.

In this thesis, we review the challenges associated with generating images and videos. We then introduce a multi-scale image generation framework that demonstrates impressive performance on real world image datasets. This method was the first to demonstrate empirically the potential of generative adversarial networks and has since sparked a significant interest in this area. We also address two challenging aspects of video generation: learning a latent space that affords easier prediction and modeling the uncertainty in video sequences.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep neural networks have seen great success in a wide variety of domains in recent years. Convolutional networks have shown to be effective at discriminative tasks such as image classification (He et al., 2016; Huang et al., 2017; Krizhevsky et al., 2012), text classification (Zhang et al., 2015a) and image segmentation (He et al., 2017). Recurrent neural networks have been successfully applied to language modeling (Mikolov, 2012), image captioning (Karpathy and Li, 2015; Kiros et al., 2014) machine translation (Bahdanau et al., 2015; Sutskever et al., 2014), and many other tasks.

The vast majority of these successes involve a fully *supervised learning* set-up whereby a target label is given to the learning algorithm. The target may take a different form depending on the application. For example, classification tasks require data points be annotated with class labels, image segmentation necessitates a per-pixel labeling, and translation typically requires paired source-target language sentences. Many of the successes outlined above rely heavily on large training sets of manually annotated data. Given the difficulty of obtaining enough labeled data to scale many deep learning approaches, it is increasingly important to look for better methods of utilizing large amounts of unlabeled

data.

Deep neural networks have also revolutionized the field of reinforcement learning. Deep reinforcement learning methods have been successfully applied in a variety of game-playing domains (Mnih et al., 2013, 2015; Silver et al., 2016), continuous control tasks (Kalashnikov et al., 2018; Lillicrap et al., 2016), and real-world navigation (Mirowski et al., 2018, 2017). In contrast to supervised learning, reinforcement learning does not require labeled data. Instead, an agent learns through interaction with an environment how to take actions that maximize reward. While this type of trial-and-error learning does not rely on costly human annotated data, unsupervised learning still plays a crucial role. Video prediction models are particularly relevant to reinforcement learning since they can be used to build action-conditional forward models of the environment. In short, this involves predicting how the state of the world changes in response to different actions. Accurate forward models can facilitate planning (Fragkiadaki et al., 2016; Gu et al., 2016; Leibfried et al., 2017; Levine et al., 2016; Schmidhuber, 1990; Weber et al., 2017) and exploration (Oh et al., 2015; Pathak et al., 2017; Stadie et al., 2015), which in turn can accelerate learning. Incorporating unsupervised objectives based on predicting various environmental factors can also speed up learning and improve overall performance even when the predictions are not used for planning (Jaderberg et al., 2016).

Unsupervised learning aims to characterize the underlying structure or distribution of unlabeled data. *Generative models* of images are a fundamental class of unsupervised learning algorithms. In short, this approach involves learning the data-generating process of a collection of training data such that new samples from the same distribution can be synthesized. Broadly speaking, generative models of images can be viewed from two different perspectives. On the one hand, they have many direct image processing

applications such as super resolution (Ledig et al., 2016), image editing (Brock et al., 2017; Dolhansky and Ferrer, 2018; Lample et al., 2017; Ma et al., 2017; Perarnau et al., 2016), image-to-image translation (Isola et al., 2015; Kim et al., 2017; Zhu et al., 2017), conditional image synthesis (Nguyen et al., 2017; Zhang et al., 2017), etc. On the other hand, they provide a flexible way of learning representations about the visual world in the absence of labeled data.

The central aim of an unsupervised *representation learning* framework is to uncover a feature representation that is applicable across a wide range of tasks. This can be contrasted with supervised feature learning where labeled data is utilized to learn a representation with a particular task in mind. Unsupervised representation learning methods may also be combined with supervised training in what is commonly referred to as *semi-supervised learning*. This type of learning is useful when a small amount of labeled data and large reserves of unlabeled data are available.

The ultimate goal of unsupervised learning is to uncover a generically useful representation. Since this objective cannot be directly optimized, an unsupervised criterion must be defined to guide learning. An effective unsupervised objective should extract meaningful high level features that describe the underlying structure of the data. With this in mind, the intuition behind generation as an unsupervised training criterion is simple: a prerequisite to synthesizing new examples is understanding the underlying structure of the data. Consider a child tasked with drawing a cat. They must first have an internal representation of the qualities that make up a cat before they can draw a new example of one. Similarly, a neural network trained with a generative objective must first learn to represent the data before being able to accurately synthesize new examples. This involves exploiting structure and redundancy in the input images and learning salient features of different objects in the data. Many generative models provide an ex-

plicit mechanism for extracting a low-dimensional description of an image and are thus well suited to unsupervised representation learning.

Video data is a particularly well suited domain for learning about the visual world due to the natural temporal coherence of image frames within video clips. Rather than relying on costly annotations, this temporal structure provides a type of supervisory signal for free. One method of utilizing the temporal coherence of video data is to learn features that exhibit a range of complex invariances (Goroshin et al., 2015; Gregor and LeCun, 2010a; Jayaraman and Grauman, 2015; Wang and Gupta, 2015; Wiskott and Sejnowski, 2002; Zou et al., 2012). Another approach considers the problem of frame prediction: given a sequence of consecutive video frames, predict the next t frames in the sequence. The importance of accurate prediction abilities for humans and other animals is obvious. More recently this general idea has been pushed further; current cognitive science literature argues that the human brain should be viewed as a prediction machine that is constantly anticipating events at various time scales (Clark, 2013). This motivates prediction as an unsupervised learning objective.

Accurate prediction also underlies many other abilities we might expect from an intelligent agent such as planning, reasoning about and anticipating future events. More concretely, video prediction methods have clear applications in model-based reinforcement learning, planning and control. Just as with image generation, future frame prediction can be viewed as an unsupervised criterion for representation learning or as a method of learning a predictive model that is useful in its own right.

Outline and summary of contributions

This thesis is organized as follows. Chapter 2 provides a summary of several generative modeling frameworks. Chapter 3 outlines prior work related to the methods pre-

sented in this thesis. Chapter 4 introduces a multi-scale image generation model based on Generative Adversarial Networks (Goodfellow et al., 2014). Chapters 5 introduces a representation learning framework that, when applied to video sequences, learns a disentangled representation of video frames. The learned factorized representation facilitates downstream tasks such as predicting future frames of a video sequence or classifying an action sequence. Chapter 6 proposes another video prediction framework that explicitly models the uncertainty in video sequences. Finally, we conclude the thesis in Chapter 7 and discuss future avenues of research.

Chapter 2

Background

In this chapter we review deep generative models. We provide a brief introduction to the problem and then outline two approaches for learning generative models.

2.1 Generative Models

Generative models are a powerful unsupervised learning tool. In contrast to discriminative learning, whereby a distribution over labels y given input $\mathbf{x}$ is of primary concern, generative models aim to represent $p(\mathbf{x})$ or $p(\mathbf{x}, y)$. Throughout the remainder of this chapter we will assume the data domain is that of natural images, however many of these approaches can be applied to audio, text, and other domains. We will also limit our focus in this chapter to a specific class of generative models known as *directed latent variable models*. Chapter 3 gives a high level overview of many alternative generative modeling frameworks for the interested reader.

We assume the data is distributed according to p_{data}, which we have access to through a finite training set $\{x^{(1)}, ..., x^{(N)}\}$ of i.i.d samples. Latent variable models

assume the existence of additional *unobserved* or *latent* variables that explain the underlying factors of variation in the dataset. Intuitively, a latent code $\mathbf{z} \in \mathcal{Z}$ can be viewed as a, typically lower-dimensional, description of an input $\mathbf{x} \in \mathcal{X}$ that summarizes factors that generated the data point. Latent variables may describe high level semantic concepts or details of a particular scene. For example, the factors of variation underlying images of human faces might include lighting condition, face orientation, expression, and hair color.

Directed graphical models express a probability distribution over random variables in terms of conditional probabilities. Using this framework, we can write the joint probability of observed variables $\mathbf{x}$ and latent variables $\mathbf{z}$ as:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2.1}$$

The data likelihood can be obtained by marginalizing over $\mathbf{z}$:

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2.2}$$

The prior distribution $p(\mathbf{z})$ should, at the very least, be easy to sample from (often additional constraints may be required such as a tractable density). $p_\theta(\mathbf{x}|\mathbf{z})$ is commonly referred to as the observation model and is generally chosen to have a tractable likelihood and easy sampling procedure. In the deep generative models considered in this thesis, the observation model will be parameterized by a deep neural network.

This structure gives rise to a simple sampling procedure, known as *ancestral sampling*, where first we sample $\mathbf{z} \sim p(\mathbf{z})$ and subsequently sample $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$. The goal of learning is to find parameters θ such that p_θ is close to p_{data}. The precise way this is quantified for training and evaluation may differ depending on the training framework

and downstream use of the generative model. Broadly speaking, the aim is for samples from the true distribution p_{data} to have high likelihood under p_θ and also for samples from p_θ to resemble samples from p_{data}.

Beyond sampling from, and evaluating likelihood under p_θ, there is another computation we might ask of a generative model. Recall, $p(\mathbf{x}|\mathbf{z})$ describes how to convert a latent representation into an image. In contrast, $p(\mathbf{z}|\mathbf{x})$ describes the latent factors underlying an image $\mathbf{x}$. This distribution is known as the posterior distribution over latent variables $\mathbf{z}$ and uncovering it is frequently known as *inference*. Directed generative models, by design, typically afford easy conditional sampling $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$ but exact inference is intractable. The existence of an efficient inference schemes will be one of the key considerations when comparing generative models in Section 3.1.

2.2 Variational Autoencoders

Autoencoders are a simple and fundamental unsupervised learning framework whereby a neural network is trained to reconstruct the input from a compressed representation. Specifically, an encoder network $f : \mathcal{X} \to \mathcal{Z}$ maps an image to a dense vector representation and a decoder $g : \mathcal{Z} \to \mathcal{X}$ converts back to the input domain. The model is trained to reconstruct the original input by minimizing any pre-defined reconstruction error, e.g. ℓ_2:

$$\mathcal{L}_{AE}(\mathbf{x}) = \|\mathbf{x} - g(f(\mathbf{x}))\|_2^2 \tag{2.3}$$

The latent space $\mathcal{Z}$ is typically of lower dimensionality than $\mathcal{X}$ and may be referred to as a *bottleneck* layer. By compressing the images into a lower dimensional space, the network is encouraged to discard redundant information and encode the most prominant

factors of variation in the dataset, rather than merely copying the input image. Several alternatives to the bottleneck principle have been proposed to prevent the model from learning a trivial mapping. For example, denoising autoencoders reconstruct from a corrupted version of the original image (Vincent et al., 2010), sparse autoencoders induce a sparsity penalty on the latent representation $\mathbf{z}$ (Lee et al., 2007; Makhzani and Frey, 2014), and contractive autoencoders add a regularizing term to the loss function that penalizes the encoder's sensitivity to small changes in the input image (Rifai et al., 2011).

Variational autoencoders (VAEs; Kingma and Welling 2014; Rezende et al. 2014) are a framework for learning directed generative models. As we will see below, VAEs share a strong resemblance with traditional autoencoders. However, the motivation and loss function are derived from a probabilistic modeling perspective. We first review VAEs from the perspective of directed generative models and then relate the framework back to traditional autoencoders.

VAEs are an example of a likelihood-based method. This means that parameters θ are estimated by maximizing the probability of the data under the model p_θ:

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2.4}$$

Optimizing the marginal likelihood of Equation 2.4 directly is intractable. Instead, VAEs optimize a bound on the marginal log-likelihood. Before deriving this bound, we first review the components of a VAE model. Recall the generative process described in

Section 2.1:

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$

The prior, $p(\mathbf{z})$ is typically chosen to be $\mathcal{N}(0, 1)$. The output distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is frequently specified as a conditional Gaussian, with mean given by a neural network μ_θ and fixed diagonal covariance:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \sigma^2 * I) \tag{2.5}$$

VAEs derive a lower bound on the marginal log-likelihood of the data under the model p_θ by introducing a new distribution $q_\phi(\mathbf{z}|\mathbf{x})$ that is an approximation to the true posterior. This *approximate posterior* is typically specified as a conditional Gaussian with mean and diagonal covariance given by neural networks μ_ϕ and σ_ϕ:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma^2_\phi(\mathbf{x}) * I) \tag{2.6}$$

We now derive the bound on the log likelihood of the data:

$$\log p_\theta(\mathbf{x}) = \log \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2.7}$$

$$= \log \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.8}$$

$$= \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.9}$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.10}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \tag{2.11}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{2.12}$$

The loss function optimized by VAEs is given in line 2.12 and is frequently known as the *evidence lower bound* or the *variational lower bound*:

$$\mathcal{L}(x;\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction term}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{prior term}} \tag{2.13}$$

Variational autoencoders are named as such due to their resemblance to traditional autoencoders. In particular, $q_\phi(\mathbf{z}|\mathbf{x})$ can be viewed as an encoder whose input is a data point $\mathbf{x}$ and output is a noisy version of latent representation $\mathbf{z}$. Similarly, $p_\theta(\mathbf{x}|\mathbf{z})$ can be viewed as a decoder that specifies how to convert a latent code $\mathbf{z}$ into an image. In fact, Equation 2.13 can be interpreted as a regularized and stochastic variant of the reconstruction objective used in a standard autoencoder.

The first term in Equation 2.13 is the expected negative log likelihood of the data point $\mathbf{x}$ under latent distribution given by the approximate posterior $q_\theta(\mathbf{x}|\mathbf{z})$. This corresponds to an expected negative reconstruction loss and, in the case of a Gaussian output distribution, reduces to a simple ℓ_2 loss. The second term in Equation 2.13 is

the Kullback-Leibler (KL) divergence of the approximate posterior $q_\theta(\mathbf{z}|\mathbf{x})$ from the prior $p(\mathbf{z})$. When both the prior and posterior distributions are taken to be Gaussian this term can be computed in closed form and easily optimized. Recall, the encoder computes, as a function of $\mathbf{x}$, the mean and (diagonal) covariance of a Gaussian distribution $\mathcal{N}(\mu_\phi(\mathbf{x}), \sigma^2_\phi(\mathbf{x}) * I)$. For a $\mathcal{N}(0, 1)$ prior, the KL divergence term simply pushes $\mu_\phi(\mathbf{x})$ towards a zero vector and $\sigma^2_\phi(\mathbf{x})$ to a vector of ones. The KL divergence loss can then be viewed as a regularizing term that limits the capacity of the encoder. Consequently, the two terms of Equation 2.13 are in opposition with one another and learning involves a delicate balance between accurate reconstruction and fitting the prior.

VAEs are trained by maximizing Equation 2.13, averaged over the training set. As mentioned above, the gradient of the KL-divergence term with respect to ϕ is easily computable for many common choices of posterior and prior. However, optimizing the expected negative reconstruction error requires more care since the expectation is unknown and depends on parameters ϕ. VAEs address this problem using the *reparameterization trick* which re-writes the random variable $\mathbf{z}$ as a deterministic function of another random variable ϵ: $\mathbf{z} = f(\mathbf{x}, \epsilon)$, $\epsilon \sim p(\epsilon)$. For example, in the Gaussian case we have:

$$\mathbf{z} = f_\phi(\mathbf{x}, \epsilon) = \mu_\phi(\mathbf{x}) + \sigma^2_\phi(\mathbf{x}) * \epsilon \qquad \text{, where } \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{2.14}$$

Using this reparameterization, the expectation $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})$ can be re-written as an expectation over ϵ instead of $\mathbf{z}$:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) = \mathbb{E}_{p(\epsilon)} \log p_\theta(\mathbf{x}|f_\phi(\mathbf{x}, \epsilon)) \tag{2.15}$$

The gradient of Equation 2.15 with respect to ϕ can now be obtained by moving the

gradient inside the expectation and using a Monte Carlo estimate of the expectation:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) &= \mathbb{E}_{p(\epsilon)} \nabla_\phi \log p_\theta(\mathbf{x}|f_\phi(\mathbf{x}, \epsilon)) \\ &\simeq \frac{1}{M} \sum_{i=1}^{M} \nabla_\phi \log p_\theta(\mathbf{x}|f_\phi(\mathbf{x}, \epsilon_\mathbf{i})) \qquad \text{, where } \epsilon_\mathbf{i} \sim p(\epsilon).\end{aligned} \tag{2.16}$$

The variational inference tools introduced in this section can also be leveraged to learn generative models of sequential data. Here, there are two potential tasks of interest: (i) estimating the unconditional distribution $p(\mathbf{x}_{1:T})$ and (ii) estimating the conditional distribution $p(\mathbf{x}_{1:c-1}|\mathbf{x}_{c:T})$. The conditional estimation task provides a framework for video prediction, i.e. predicting a distribution over a future sequence of frames, conditioned on past observations, and we will consider it in more depth in Chapter 6.

Bayer and Osendorfer (2014) were the first to utilize the VAE framework for stochastic sequence generation. They incorporated stochastic latent variables into a recurrent neural network to model music and motion capture data. The method is trained analogous to a VAE but with recurrent, rather than feed-forward, encoder and decoder networks. Several additional works have built on this framework to model speech, handwriting, natural language (Bowman et al., 2016; Chung et al., 2015; Fraccaro et al., 2016), perform counterfactual inference (Krishnan et al., 2015) and anomaly detection (Sölch et al., 2016). More recently, these methods have been applied to stochastic video generation (Babaeizadeh et al., 2018; Denton and Fergus, 2018; Lee et al., 2018). These sequential models are all trained analogous to a VAE by optimizing a bound on the data likelihood using an approximate inference network. They differ primarily in the parameterization of the approximate posterior and the choice of prior model.

In summary, VAEs are a method of learning directed generative models by optimizing a bound on the marginal log-likelihood. The method is derived from a probabilistic

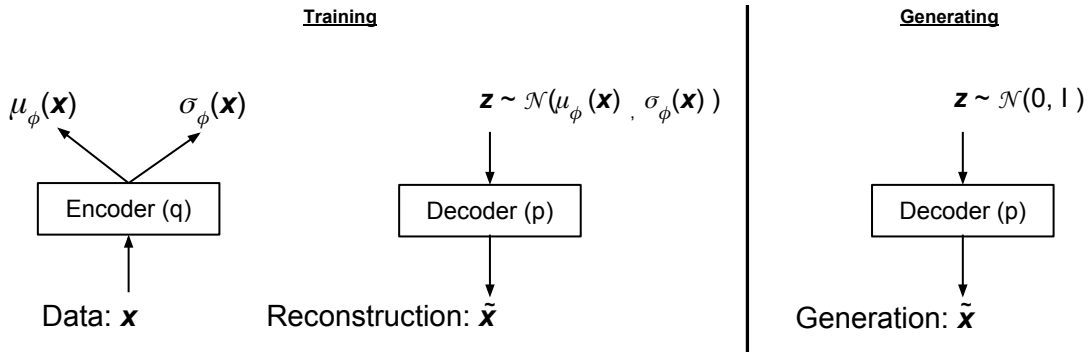

Figure 2.1: Training (left) and generation(right) in a VAE.

modeling perspective but has strong similarities with a regularized autoencoder. VAEs are flexible generative models that have been utilized for semi- and unsupervised learning (Higgins et al., 2017; Kingma et al., 2014) and extended for sequential modeling tasks (Bayer and Osendorfer, 2014; Bowman et al., 2016; Chung et al., 2015; Fraccaro et al., 2016; Krishnan et al., 2015; Sölch et al., 2016). In the next section we introduce an alternative approach for learning directed generative models.

2.3 Generative Adversarial Networks

Generative adversarial networks (GANs) are a framework for learning directed generative models proposed by Goodfellow et al. (2014). GANs are implicit generative models which, in contrast to VAEs, do not explicitly specify a likelihood function. Instead, the method pits two networks against one another: a generative model G_θ captures the data distribution and a discriminative model D_ϕ distinguishes between samples drawn from G_θ and images drawn from the training set. As D_ϕ is trained to discriminate true data from generated samples, G_θ is trained simultaneously to generate samples that fool D_ϕ into thinking they come from the data distribution.

As training progresses, the two networks improve together. As the generator learns

how to fool the discriminator, samples shift from mere noise to more structured outputs. As the generator's samples begin to mirror the training distribution, the discriminator must also improve in order to differentiate the real data from generations. Crucially, the generator does not have direct access to the training data. Rather, it is the discriminator that defines the loss function for the generator.

More formally, G_θ takes as input a latent code vector $\mathbf{z} \in \mathcal{Z}$ sampled from a prior distribution $p(\mathbf{z})$ and outputs an image $\mathbf{x} \in \mathcal{X}$. The GAN generator G_θ is analogous to the decoder of a VAE. However, in contrast to a VAE, the generator is deterministic, i.e. the observation model $p_\theta(\mathbf{x}|\mathbf{z})$ is a Dirac delta function with all the probability mass concentrated on the point $\mathbf{x} = G(\mathbf{z})$. Common choices for the prior $p(\mathbf{z})$ include $\mathcal{N}(0, 1)$ and $\mathcal{U}(-1, 1)$. The discriminator D_ϕ takes as input an image $\mathbf{x} \in \mathcal{X}$ from the training set or sampled from G_θ and outputs a scalar. In the basic GAN formulation (Goodfellow et al., 2014) the discriminator's output is viewed as a probability and is trained to be high if the input was real and low if generated from G_θ. Parameters θ and ϕ are estimated simultaneously by optimizing the minimax objective:

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \log D_\phi(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p_{noise}(\mathbf{z})} \log(1 - D_\phi(G_\theta(\mathbf{z}))) \tag{2.17}$$

Optimizing Equation 2.17 requires finding a saddle point; this can make stability and convergence a challenge. Ideally, the discriminator should be optimized to optimality for each step the generator takes. This is infeasible in practice and Goodfellow et al. (2014) instead propose to alternate between k updates of the discriminator and one update of the generator. The number of discriminator updates steps, k, is frequently taken to be 1.

Optimizing Equation 2.17 can provide poor gradients for the generator early in learning and Goodfellow et al. (2014) also propose an alternate, non-saturating, variant of the

loss that is more frequently used in practice:

$$\max_{\phi} \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \log D_{\phi}(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim p_{noise}(\mathbf{z})} \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}))) \tag{2.18}$$

$$\max_{\theta} \mathbb{E}_{\mathbf{z} \sim p_{noise}(\mathbf{z})} \log D_{\phi}(G_{\theta}(\mathbf{z}))) \tag{2.19}$$

Goodfellow et al. (2014) show that, under ideal conditions, optimizing Equation 2.17 minimizes the Jensen-Shannon (JS) divergence between p_{data} and p_{θ}. The heuristically motivated objective of Equation 2.19 is less theoretically grounded but performs better empirically.

In essence, GANs utilize the power of deep neural networks - which have proved highly effective in discriminative tasks - to define a loss function over images. GANs tend to produce crisper, more visually appealing images than VAE models. This has partially been attributed to the difference in loss functions (Theis et al., 2016). VAEs maximize a bound on the data log likelihood - or equivalently minimize $D_{KL}(p_{data}|p_{\theta})$ - which encourages the model to put probability mass on every data point, at the expense of possibly assigning mass outside the data manifold. In contrast, minimizing the JS divergence encourages the model to only put mass in data regions, at the expense of possibly missing some data regions. However, GANs trained to minimize alternative divergences (Nowozin et al., 2016) tend to exhibit behavior similar to the standard GAN, suggesting the JS divergence minimization is not the primary difference.

GANs have become increasingly popular generative models in recent years. However, training instability and lack of convergence criteria make GANs notoriously difficult to train. One common failure case, known as mode collapse, occurs when the generator produces an extremely restricted set of images. Oscillating behavior is also common, where the generator cycles between a handful of data modes. Balancing the

generator and discriminator can also be challenging. If the discriminator is too good, training can collapse. GANs are also highly sensitive to initial conditions and hyper-parameter settings. They also lack a meaningful loss to track throughout training, making hyper-parameter tuning difficult. Several architectural and optimization improvements have been proposed to address these stability concerns (Metz et al., 2017; Odena et al., 2017; Salimans et al., 2016).

Several variants of the training criterion have also been proposed, which we briefly review. Energy-based GANs (Zhao et al., 2017) cast the discriminator as an energy function trained to assign low energy to true data and high energy to samples from the generator. Least Squares GANs (Mao et al., 2017) replaces the binary cross entropy loss of Equation 2.19 with an ℓ_2 loss. The new formulation corresponds to minimizing a Pearson $\mathcal{X}^2$ divergence. Wasserstein GANs (Arjovsky et al., 2017; Gulrajani et al., 2017) minimize a smooth distance metric known as the Wasserstein distance. In addition to improving training stability, this method provides a clear metric to aid in hyper-parameter searches and test convergence. Boundary Equilibrium GANs (Berthelot et al., 2017) also minimize a Wasserstein distance. However, they consider the loss distribution of an auto-encoder on real and generated data, rather than the image distribution directly. Nowozin et al. (2016) extend the adversarial framework arbitrary f-divergence minimization. The original GAN objective of Equation 2.17, which minimizes the JS-divergence, is a special of the more general method. GANs have also been extended to discrete generation problems (Hjelm et al., 2018) by utilizing policy gradient techniques from reinforcement learning.

Many of these alternative frameworks have claimed improved stability, robustness and generation quality over the original GAN framework. While algorithmic advances have certainly been made, a recent extensive study by Lucic et al. (2017) shows that -

with sufficient hyper-parameter tuning - no single method is consistently superior across a variety of datasets. This study indicates a need for increased rigor when comparing GAN methods.

Despite training difficulties, GANs have demonstrated impressive image generation performance and current state-of-the-art image synthesis models rely on this framework (Karras et al., 2018; Zhang et al., 2018b). They have been utilized for a variety of image processing applications such as super-resolution (Ledig et al., 2016), image-to-image translation (Isola et al., 2015; Kim et al., 2017; Lin et al., 2018; Zhu et al., 2017), text-to-image synthesis (Reed et al., 2016; Zhang et al., 2017), and semantic segmentation (Luc et al., 2016).

GANs and VAEs represent two alternative approaches for learning directed generative models. Each model has a different set of strengths and weaknesses. It is difficult to directly compare generative models in the absence of a particular goal such as image synthesis, density estimation, or representation learning.

Chapter 3

Motivation and Related work

This chapter reviews previous work related to the methods presented in this thesis. Section 3.1 summarizes recent advances in image generation. Section 3.2 reviews representation learning with a focus on disentangled and invariant representations. Finally, Section 3.3 reviews video generation approaches.

3.1 Image generation

Generative image models are well studied, falling into two main approaches: non-parametric and parametric. The former copy patches from training images to perform, for example, texture synthesis (Efros and Leung, 1999) or super-resolution (Freeman et al., 2002). More ambitiously, entire portions of an image can be in-painted, given a sufficiently large training dataset (Hays and Efros, 2007). Early parametric models addressed the easier problem of texture synthesis (De Bonet, 1997; Portilla and Simoncelli, 2000; Zhu et al., 1998), with Portilla and Simoncelli (2000) making use of a steerable pyramid wavelet representation (Simoncelli et al., 1992), similar to our use of a

Laplacian pyramid described in Chapter 4. For image processing tasks, models based on marginal distributions of image gradients are effective (Olshausen and Field, 1997; Roth and Black, 2005), but are only designed for image restoration rather than being true density models (so cannot sample an actual image). Very large Gaussian mixture models (Zoran and Weiss, 2011) and sparse coding models of image patches (Wright et al., 2010) can also be used but suffer the same problem.

More recently, deep learning advances have led to a wide variety of powerful parametric generative models. Deep generative models are of interest for both representation learning and image synthesis. Consequently, as we review recent advances in deep generative models we will consider a given model's ability to (i) synthesize high quality images and (ii) infer generative factors for a particular image.

Deep generative models can broadly be categorized into two groups (Mohamed and Lakshminarayanan, 2016). *Prescribed models* explicitly parameterize a distribution over the data and are trained via maximum likelihood estimation. Restricted Boltzmann machines, variational autoencoders, autoregressive models, and reversible models are all examples of likelihood-based approaches. In contrast, *implicit generative models*, such as generative adversarial networks and moment matching networks, merely define a generative process and do not require an explicit density function.

Restricted Boltzmann machines (Hinton and Salakhutdinov, 2006; Krizhevsky et al., 2010; Osindero and Hinton, 2008; Ranzato et al., 2013), and Deep Boltzmann machines (Eslami et al., 2014; Salakhutdinov, 2015; Salakhutdinov and Hinton, 2009) are likelihood based methods that were the focus of much early work on deep generative models. Restricted Boltzmann machines (RBMs) are undirected graphical models with a bipartite graph structure that facilitates quick and easy inference (i.e., inferring latent features given an image) and conditional generation (i.e. reconstructing an image from a latent

code). They are capable of extracting high-level representations of images and thus are suitable for unsupervised representation learning. Indeed, for many years Restricted Boltzmann machines were utilized for unsupervised layer-wise pretraining of neural networks (Bengio et al., 2006). However, due to the intractability of the marginal distribution, sampling from the model requires running a difficult and expensive Markov chain. Thus, both training and evaluating these models can be challenging in practice.

Recent advances in stochastic variational inference led to the variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014) algorithm. Variational autoencoders (VAEs; see Section 2.2 for detailed overview) are directed generative models with an efficient approximate inference scheme and simple maximum likelihood based learning mechanism. The inference network makes VAEs an appealing unsupervised and semi-supervised representation learning framework (Higgins et al., 2017; Kingma et al., 2014). Many extensions and variants of VAEs have been proposed. For example, the DRAW model of Gregor et al. (2015) builds on the VAE framework, introducing an attentional mechanism with an RNN to generate images via a trajectory of patches. More recently, van den Oord et al. (2017) introduced the Vector Quantised VAE (VQ-VAE) which uses discrete, rather than continuous, latent codes and a learned, rather than fixed, prior. The VQ-VAE has demonstrated impressive image generation and density estimation performance.

Autoregressive models are another class of likelihood-based generative models. They define a tractable density by specifying an ordering on the pixels and modeling the conditional distribution over each pixel given the previous. Early feed-forward autoregressive models show promising results on image generation at low resolutions (Larochelle and Murray, 2011; Uria et al., 2013). More recently, recurrent and CNN based autoregressive models have shown impressive state-of-the-art performance on large scale

datasets such as ImageNet (Reed et al., 2017a; Salimans et al., 2017; Theis and Bethge, 2015; van den Oord et al., 2016a,c). Autoregressive models are currently state-of-the-art density estimators. However, they lack a latent representation limiting their applicability to unsupervised learning and image manipulation applications. The iterative and computationally intensive nature of generation also makes real-time generation a challenge.

Another class of maximum likelihood models are known as *flow-based* or *reversible generative models* (Dinh et al., 2014, 2017; Kingma and Dhariwal, 2018). This approach optimizes the exact log-likelihood by defining a generative process as a sequence of *invertible* transformations. Flow-based generative models allow for exact inference and have a simple and efficient ancestral sampling procedure. The recent flow-based generative model of Kingma and Dhariwal (2018) introduces invertible 1×1 convolutions and demonstrates impressive high resolution image synthesis.

In contrast to the likelihood-based models described above, generative adversarial networks (GANs; Goodfellow et al. 2014; see Section 2.3 for detailed overview) do not explicitly define a density. Rather, a generative network is learned directly by pairing it with a discriminator network in an adversarial game. Goodfellow et al. (2014) demonstrated fully connected GANs were effective at modeling small low-resolution images. However, difficulties in training stability inhibited their immediate application to large natural images. Our work on multi-scale image generation (see Chapter 4) and subsequent work on improved architectures (Radford et al., 2016) demonstrated state-of-the-art image synthesis results, prompting significant interest in GANs. Several architectural and optimization improvements have been proposed to improve the stability of GAN training (Metz et al., 2017; Odena et al., 2017; Salimans et al., 2016). Additionally, many variants of the training criterion have been proposed in recent years (Arjovsky et al., 2017; Berthelot et al., 2017; Gulrajani et al., 2017; Hjelm et al., 2018;

Mao et al., 2017; Nowozin et al., 2016; Poole et al., 2016; Zhao et al., 2017).

The standard GAN setup lacks an inference mechanism, potentially limiting their application to settings where a meaningful latent representation is required. However, several works have proposed a method of learning an inference network jointly with the generator through a combined adversarial training procedure (Belghazi et al., 2018; Donahue et al., 2017; Dumoulin et al., 2017). Another set of methods forgo an encoder entirely and instead utilize the representation learned by the discriminator network (Denton et al., 2015; Odena, 2016; Salimans et al., 2016; Springenberg, 2015).

Bojanowski et al. (2018) propose an framework that utilizes the power of a convolutional generator but avoids the training instability inherent in the adversarial training procedure. This method, known as Generative Latent Optimization (GLO), directly learns a mapping from latent codes to images via a reconstruction loss. This simple method learns a semantically meaningful mapping from noise vectors to images and produces visually appealing samples.

Another approach to learning implicit generative models is based on a statistical hypothesis testing tool called Maximum Mean Discrepancy (MMD) that describes the difference between two probability distributions (Gretton et al., 2007). The MMD criterion can be used to specify an objective function for learning generative models based on minimizing the difference between the data distribution and the model distribution (Dziugaite et al., 2015; Li et al., 2015). Moment matching networks are an appealing alternative to GANs due to the principled nature of the MMD criterion and the stability of training. However, these models have primarily been applied to small resolution toy datasets. Methods that leverage both GANs and the MMD criterion have shown better image generation performance. For example, Li et al. (2017) propose a framework for combining the discriminative power of neural networks with the MMD criterion by

adversarially learning the MMD kernels.

Many of the generative models described in this section lend themselves naturally to unsupervised representation learning. In the next section we focus more specifically on methods for learning representations of the visual world.

3.2 Disentangled representations

A fundamental characteristics of human perception is the ability to decompose experiences into semantically meaningful chunks. For example, visual scenes consist of multiple objects and objects in turn can be described by properties such as pose, texture and class identity. Humans are easily able to characterize these so-called *factors of variation*. This section explores methods of learning disentangled representations with little or no supervision.

Much early work on unsupervised learning with neural networks involves the autoencoding framework (Bengio et al., 2006; Masci et al., 2011; Ranzato et al., 2006; Vincent et al., 2010). A range of feature learning methods have been proposed that aim to learn features invariant to local image transformations, e.g. spatial shifts (Gregor and LeCun, 2010b; Kavukcuoglu et al., 2010; Ranzato and LeCun, 2007). Another set of methods aims to preserve the information discarded by invariant feature learning methods and instead explicitly seperate *what* and *where* representations (Cadieu and Olshausen, 2009; Gregor and LeCun, 2010a; Ranzato et al., 2007; Zhao et al., 2016). The what-where autoencoder of Zhao et al. (2016) also has similarities to ladder networks (Rasmus et al., 2015) which have been used for unsupervised and semi-supervised learning. The capsule model of Hinton et al. (2011) also performs this what and where separation via an explicit autoencoder structure and more recently the idea of capsules has

been extended and applied in a discriminative setting (Hinton et al., 2017; Sabour et al., 2017).

Another natural way of decomposing images is based on the concept of *content* and *style*. Content is typically specified by class identity (e.g. *chair*) and style refers to characteristics that vary within a given class (e.g. *wooden, narrow*, etc.). Several recent content and style separation methods have been proposed that rely on weak supervision in the form of class identities. (Cheung et al., 2014) propose an autoencoder with an additional regularizing term to disentangle factors of variation. Mathieu et al. (2016b) combine VAE and GAN tools to train a conditional generative model with a factorized latent code. Siddharth et al. (2017) extend the VAE framework by adding additional latent variables for which supervision is available. Bouchacourt et al. (2018) propose a related VAE model that relies on group level supervision rather than class identies. InfoGAN (Chen et al., 2016) is an unsupervised disentangling approach based on maximizing the mutual information between the images and a subset of latent variables. **?** disentangle content and style in an unsupervised manner by imposing a prior over latent codes of an autoencoder with a GAN.

Another set of approaches aims to further disentangle the individual factors summarized by *style* characteristics, such as pose, lighting conditions, etc. One group of methods learns these factors by predicting transformed verion of image from original with a gated RBM (Memisevic and Hinton, 2010; Reed et al., 2014; Susskind et al., 2011). Kulkarni et al. (2015) show how explicit graphics code can be learned from datasets with systematic dimensions of variation. Whitney et al. (2016) use a gating principle to encourage each dimension of the latent representation to capture a distinct mode of variation.

Another approach to disentangling factors of variation involves encouraging *inde-*

pendence between components of a latent code. Independent components analysis (Hyvarinen and Oja, 2004) is a classic approach for extracting statistically independent components of a dataset. Several variants of the VAE framework have been proposed to learn independent latent codes from unlabeled data (Higgins et al., 2017; Kumar et al., 2018). Several approaches has utilized adversarial training to induce independence. Schmidhuber (1992) proposed an early method of learning discrete independent representations via an adversarial learning procedure. More recently, several adversarial approaches have been proposed that encourage independence of latent codes within a VAE (Brakel and Bengio, 2017; Kim and Mnih, 2018).

Beyond disentangling factors of variation, another desirable property of a learned representation is *invariance* to certain transformations. For example, in the context of classification a chair should be recognizable as such regardless of its pose, size, lighting conditions, etc. Several recent deep learning methods have utilized video data to learn invariances of this sort. Wang and Gupta (2015) learn an invariant embedding for patches taken from object video tracks. Zou et al. (2012) use similar principles to learn features that exhibit a range of complex invariances. Criterion related to slow feature analysis (Wiskott and Sejnowski, 2002) have been proposed such as linearity of representations (Goroshin et al., 2015) and equivariance to ego-motion (Jayaraman and Grauman, 2015). Gregor and LeCun (2010a) propose a bilinear model that seperates slow varying features from location features.

Invariant feature learning methods also have great relevance to the problem of domain adaptation. Here, the task is to learn a representation in a source domain that transfers well to a different, but related, target domain. Several methods utilize an adversarial loss to learn *domain-invariant features* (Ganin and Lempitsky, 2015; Tzeng et al., 2015, 2017). This framework involves training a discriminator network to differ-

entiate between features in the source and target domains while simultaneously training the features adversarially to confuse the discriminator.

Adversarial methods have also been adapted to learn *fair representations*. Rather than invariance to source/target domain, fair representations seek invariance to certain protected demographic groups. These methods train a discriminator network to predict a sensitive attribute (e.g., gender, race, etc.) from an intermediate representation of a classifier. The classifier features are trained adversarially to confuse the discriminator. Several adversarial objectives have been proposed to learn representations that achieve different notions of fairness (Beutel et al., 2017; Edwards and Storkey, 2016; Madras et al., 2018; Zhang et al., 2018a).

In Chapter 5 we introduce a method that, when applied to video data, learns to factorize *content* and *pose*. Here, content refers to any information that does not change across a video clip (e.g. background features, lighting conditions, etc.) and pose refers to frame-specific features (e.g. location of objects). This factorization relates to slow feature analysis (Wiskott and Sejnowski, 2002) methods in that the content portion of the representation can be understood as maintaining invariance to local transformations observed in the video clip. However, instead of throwing this information away, our method encodes it in the pose portion of the representation. The pose representation is also trained, via an adversarial loss, to be invariant to the content of the clip. The use of this adversarial loss induces a clean disentanglement of content/pose. Since high-level scene semantics tend to change slowly across a video, short video clips provide grouped data - without the need for manual annotation - that naturally lends itself to this type of content/pose factorization. However, the method can be applied anywhere group level supervision is available to disentangle properties constant within a group from those that vary within a group.

In this section we reviewed a set of techniques for learning disentangled, independent and invariant representations from weakly or unlabelled data. In the next section we consider the task of video prediction, where the goal is to predict a future sequence of frames conditioned on past observations. Video prediction is a challenging and important problem with applications ranging from unsupervised representation learning to planning and control.

3.3 Video prediction

The ability to accurately predict the future is fundamental to interacting with and reasoning about the world. One way of approaching this problem is through video prediction where the goal is to generate a plausible sequence of future frames given previously observed frames.

Accurate prediction of future frames requires sophisticated scene understanding as well as in-depth knowledge of physical and causal rules that govern how objects, lighting conditions, etc. change over time. This property, plus the abundance of video data, makes video prediction a promising framework for unsupervised representation learning. Indeed, many video prediction methods produce feature representations effective for downstream tasks such as human action recognition (Denton and Birodkar, 2017; Srivastava et al., 2015; Vondrick et al., 2016a), face recognition (Lotter et al., 2016) and other discriminative tasks.

Video prediction methods also have important applications in reinforcement learning. The vast majority of recent reinforcement learning successes have been in model-free settings (Lillicrap et al., 2016; Mnih et al., 2013; Schulman et al., 2015) where learning is known to be incredibly sample inefficient, requiring vastly more experience

than humans to achieve comparable performance. One method of improving the sample efficiency is to utilize a forward model of the environment that specifies how the state of the world changes in response to different actions. Given an accurate forward model, model-based approaches can be used that rely more heavily on planning, e.g. through Monte-Carlo tree search (Browne et al., 2012), and less on trial-and-error (Guo et al., 2014; Hamrick et al., 2017; Sutton, 1990). Environment dynamics are commonly unknown and challenging to model, pointing to the need for better methods of learning action-conditional forward models. Several recent works have proposed model-based reinforcement learning techniques that utilize learned forward dynamics models (Fragkiadaki et al., 2016; Gu et al., 2016; Leibfried et al., 2017; Levine et al., 2016; Schmidhuber, 1990; Weber et al., 2017). Learned forward models can also be used to guide exploration which in turn can help speed up learning (Bellemare et al., 2016; Chiappa et al., 2017; Oh et al., 2015; Pathak et al., 2016; Stadie et al., 2015). Forward dynamics models can also be used for planning and control outside the context of reinforcement learning (Henaff et al., 2018; Watter et al., 2015).

Building accurate and robust video prediction models is a challenging and well-studied problem. A crucial design choice when building generative video models is the choice of a loss function. A natural choice is to use a mean squared error (MSE) loss, essentially views frame prediction as a multi-dimensional regression problem. Many early video prediction methods adopted this approach (Michalski et al., 2014; Srivastava et al., 2015; Sutskever et al., 2009). Unfortunately, a naive application of MSE for video prediction can lead to poor results. For example, low prediction errors can be achieved by a simple copying or blurring of the previous frame. Another challenge arises from the inherent multi-modality of video data: given a past sequence of frames, multiple future outcomes may be possible. Unless this multi-modality is explicitly addressed with latent

variables, MSE-based models tend to predict an average of multiple possible outcomes.

An alternative approach is to transform frame prediction into a classification, rather than regression, problem. Ranzato et al. (2014) adopted this approach, proposing one of the first video prediction methods to scale beyond image patches and synthetic datasets. Inspired by the effectiveness of recurrent neural networks at modeling sequences of *discrete* tokens, Ranzato et al. (2014) quantized image patches using k-means and trained a recurrent network with a cross-entropy loss to predict a $2 - d$ array of centroids.

Video Pixel Networks (VPNs; Kalchbrenner et al. 2016) are another approach to generation that avoids the blurriness of MSE losses. VPNs discretize pixels and model the conditional distribution over each pixel given the previous (similar to autoregressive image generation models of van den Oord et al. (2016b) and Salimans et al. (2017)). VPNs produce impressive generations but, as with autoregressive models of images, they are expensive to train and evaluate.

Mathieu et al. (2016a) explore several novel loss functions to improve the crispness of predicted video frames. By combining MSE, an adversarial loss, and a penalty on gradient differences between the ground truth and predicted future frame, they demonstrate improved prediction results beyond MSE alone. Many subsequent models have adopted the GAN framework and train either entirely with an adversarial loss (Vondrick et al., 2016b; Vondrick and Torralba, 2017) or with a combined MSE and adversarial loss (Lee et al., 2018; Villegas et al., 2017a).

Orthogonal to the choice of loss function is the challenge of architecture design. A wide variety of generation models have been proposed in recent years, many of which utilize the natural structure and regularity present in video sequences. A key observation underlying several recent models is that object motion tends to be smooth and continuous. Thus, motion can be modeled via transformations of groups of pixels, rather than

generating new future pixels from scratch. One set of approaches predicts optical flows fields which can be used to extrapolate motion beyond the current frame (Liu, 2009; Walker et al., 2015; Xue et al., 2016). Finn et al. (2016) use an LSTM framework to generate convolutional kernels. The kernels are applied to the previous frame and the resulting images are combined via a masked addition to produce the prediction. Vondrick and Torralba (2017) propose a related approach to transform neighborhoods of pixels but explicitly generate a different transformation for each spatial location. Video prediction architectures commonly add skip connections between an encoder and decoder (Denton and Birodkar, 2017; Denton and Fergus, 2018; Finn et al., 2016; Villegas et al., 2017b). This facilitates direct copying of parts of the previous frame, allowing the rest of the model to focus on changes.

Another group of approaches factorize the video into static and dynamic components. Jojic and Frey (2001) propose a method of inferring masks of moving objects from unlabelled video sequences. Vondrick et al. (2016b) propose a two-stream convolutional model that separately generates a static background and a dynamic foreground. Villegas et al. (2017a) propose an LSTM that separates out motion and content in video sequences. In Chapter 5 we introduce a method that decomposes video frames into content and pose representations.

Wang et al. (2017) propose a recurrent model with spatial and temporal memory units in order to better predict far into the future. Wang et al. (2018) extend this model using the recurrent highway approach of (Zilly et al., 2017) to improve gradient flow.

Another perspective on video prediction involves forgoing pixel-level generation entirely and instead learning a prediction model in an abstract latent space. Predicting future pixels is a natural objective when no additional annotations or high-level information is available. However, accurate pixel-level prediction is often less important

than predicting the high-level semantics of future frames. Unfortunately, a high-level semantic representation is not always available, necessitating pixel-level prediction frameworks. Several methods have proposed high-level representations appropriate for video prediction. One group of methods predicts future flow fields (Liu, 2009; Walker et al., 2015; Xue et al., 2016). Another set of methods predict semantic segmentations (Luc et al., 2017) or instance level segmentations (Luc et al., 2018) of future frames. Other approaches predict high-level abstractions such as object motion trajectories (Walker et al., 2014) or visual representations extracted from a discriminative convolutional network (Srivastava et al., 2015; Vondrick et al., 2016a).

Another approach involves predicting high-level structure of future frames and then decoding the predicted abstractions into pixel space. Villegas et al. (2017b) adopt this hierarchical approach, using $2-d$ human poses as the high-level structure. This method is able to successfully generate complex scenes but requires annotated pose information at training time. More recently, Wichers et al. (2018) extended this work to an unsupervised setting, showing impressive long range generations. Rather than training end-to-end, another approach involves first discovering high-level features suitable for video prediction and subsequently training a predictor in the *fixed* latent space. Following this approach, Chapter 5 introduces a method for learning high-level features that facilitate video prediction. Our method disentangles static and dynamic aspects of a video, allowing for stable and coherent long-range prediction. van den Oord et al. (2017) train a prediction model on discrete codes learned with the VQ-VAE procedure, generating high quality future predictions on synthetic game environments.

When available, actions can be integrated into prediction models to improve performance. Chiappa et al. (2017) and Oh et al. (2015) focus on action-conditional prediction in video game environments, showing high-quality long rage predictions.

One of the most fundamental challenges of video prediction stems from the inherent uncertainty in the dynamics of the world. For example, consider an unlabeled video dataset of cars navigating city streets. Upon reaching an intersection a car may drive straight, turn left or turn right. Typically, a dataset will not contain all possible future trajectories for a *particular* past sequence of frames, although collectively all the possible patterns of motion may be represented. A deterministic model trained with MSE will penalize a reasonable future prediction (e.g. the car continues straight) if the ground truth future for that particular training example contained a different outcome (e.g. the car turns right). Consequently, the model will learn to average multiple possible outcomes, resulting in blurry predictions. The multi-modal nature must be explicitly taken into consideration if we want a prediction model that can predict crisp frames and sample different possible outcomes given an input sequence.

Several more sophisticated approaches have been proposed that focus on addressing the multi-modality of future prediction. One approach involves finding better methods of quantifying the loss of a generated sequence of frames so that predictions that deviates from the ground truth, but are reasonable given the past, are not penalized. To this end, the GAN framework has been adopted to define an error metric over images. The idea is to train a conditional GAN (Mirza and Osindero, 2014a) where the discriminator takes a sequence of predicted or ground truth frames in addition to the previous frames provided to the generator. Thus, rather than comparing the predicted and ground truth future frames, the discriminator learns a distribution over possible future outcomes (conditioned on the past) and assesses the generated future based on this. Mathieu et al. (2016a) first utilized this type of adversarial loss in the context of video prediction. However, they combined the adversarial loss with MSE, with the goal of producing crisp images, rather than modeling multi-modal outcomes. Several subsequent approaches

have utilized adversarial losses on their own or in combination with other pixel-wise objectives (Lee et al., 2018; Villegas et al., 2017a; Vondrick et al., 2016b; Vondrick and Torralba, 2017). Despite impressive image generation performance, GANs are not yet a satisfactory method of modeling stochastic videos since training difficulties and mode collapse often mean the full distribution is not captured well.

Other approaches address uncertainty in predicting the future by introducing latent variables into the prediction model. Henaff et al. (2017) disentangle deterministic and stochastic components of a video by encoding prediction errors of a deterministic model in a low dimensional latent variable. Babaeizadeh et al. (2018) propose a variational video generation model from which stochastic videos can be sampled. In Chapter 6 we propose a related variational model that significantly improves upon Babaeizadeh et al. (2018) both in term of prediction quality and ease of training.

Chapter 4

Multi-scale Image Generation using a Laplacian Pyramid of Adversarial Networks

In this chapter we introduce a multi-scale image generation model based on GANs. At the time of publication this work marked a significant advance in deep generative models of images, producing samples of higher resolution and superior fidelity when compared with existing approaches. This work was also the first to illustrate the potential of the GAN framework for generative modeling of high resolution natural images. Since publication, there has been a surge of interest in GANs. Significant advances have been made and current GAN models are capable of generating high quality, crisp images at resolutions up to 1024×1024 (Karras et al., 2018; Zhang et al., 2018b).

4.1 Introduction

Building a good generative model of natural images is a fundamental problem within computer vision. However, images are complex and high dimensional, making them hard to model well, despite extensive efforts. Given the difficulties of modeling entire scenes at high-resolution, we propose an approach that breaks the original problem into a sequence of more manageable stages. To do this, we exploit the multi-scale structure of natural images, building a series of generative models, each of which captures image structure at a particular scale of a Laplacian pyramid (Burt et al., 1983). At each scale we train a convolutional network-based generative model using the Generative Adversarial Networks (GAN) approach of Goodfellow et al. (2014). Samples are drawn in a coarse-to-fine fashion, commencing with a low-frequency residual image. The second stage samples the band-pass structure at the next level, conditioned on the sampled residual. Subsequent levels continue this process, always conditioning on the output from the previous scale, until the final level is reached. Thus, drawing samples is an efficient and straightforward procedure: taking random vectors as input and running forward through a cascade of deep convolutional networks to produce an image.

4.2 Approach

The basic building block of our approach is the generative adversarial network (GAN) of Goodfellow et al. (2014). After reviewing this, we introduce our LAPGAN model which integrates a conditional form of GAN model into the framework of a Laplacian pyramid.

4.2.1 Generative Adversarial Networks

The GAN approach (Goodfellow et al., 2014) is a framework for training generative models, which we briefly explain in the context of image data (see 2.3 for detailed overview). The method pits two networks against one another: a generative model G that captures the data distribution and a discriminative model D that distinguishes between samples drawn from G and images drawn from the training data. In our approach, both G and D are deep convolutional networks. The former takes as input a noise vector $\mathbf{z}$ drawn from a distribution $p_{\text{Noise}}(\mathbf{z})$ and outputs an image $\tilde{\mathbf{x}}$. Following Goodfellow et al. (2014) we used a $\mathcal{U}(0, 1)$ noise distribution. The discriminative network D takes an image as input stochastically chosen (with equal probability) to be either $\tilde{\mathbf{x}}$ – as generated from G, or $\mathbf{x}$ – a real image drawn from the training data $p_{\text{Data}}(\mathbf{x})$. D outputs a scalar probability, which is trained to be high if the input was real and low if generated from G. A minimax objective is used to train both models together:

$$\max_{G} \mathbb{E}_{\mathbf{z} \sim p_{noise}(\mathbf{z})} \log D(G(\mathbf{z})) \tag{4.1}$$

$$\max_{D} \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim p_{noise}(\mathbf{z})} \log(1 - D(G(\mathbf{z}))) \tag{4.2}$$

This encourages G to fit $p_{\text{Data}}(\mathbf{x})$ so as to fool D with its generated samples $\tilde{\mathbf{x}}$. Both G and D are trained by backpropagating the loss in Equation 4.1 through their respective models to update the parameters.

The conditional generative adversarial net (CGAN) is an extension of the GAN where both networks G and D receive an additional vector of information $\mathbf{h}$ as input. This might contain, say, information about the class of the training example $\mathbf{x}$. The loss

function thus becomes:

$$\max_G \mathbb{E}_{\mathbf{z}\sim p_{noise}(\mathbf{z}),\mathbf{h}\sim p_h(\mathbf{h})} \log D(G(\mathbf{z},\mathbf{h}),\mathbf{h}) \tag{4.3}$$

$$\max_D \mathbb{E}_{\mathbf{x},\mathbf{h}\sim p_{data}(\mathbf{x},\mathbf{h})} \log D(\mathbf{x},\mathbf{h}) + \mathbb{E}_{\mathbf{z}\sim p_{noise}(\mathbf{z}),\mathbf{h}\sim p_h(\mathbf{h})} \log(1 - D(G(\mathbf{z},\mathbf{h}),\mathbf{h})) \tag{4.4}$$

where $p_h(\mathbf{h})$ is the prior distribution over $\mathbf{h}$. Note that both D and G receive the conditioning variable $\mathbf{h}$. This allows the output of the generative model to be controlled by $\mathbf{h}$ as the discriminator D learns a join distribution over $\mathbf{x}$ and $\mathbf{h}$. Mirza and Osindero (2014b) and Gauthier (2014) both explore this model with experiments on MNIST and faces, using $\mathbf{h}$ as a class indicator. In our approach, $\mathbf{h}$ will be another image, generated from another CGAN model.

4.2.2 Laplacian Pyramid

The Laplacian pyramid (Burt et al., 1983) is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual. Formally, let $d(.)$ be a downsampling operation which blurs and decimates a $j \times j$ image $\mathbf{x}$, so that $d(\mathbf{x})$ is a new image of size $j/2 \times j/2$. Also, let $u(.)$ be an upsampling operator which smooths and expands $\mathbf{x}$ to be twice the size, so $u(\mathbf{x})$ is a new image of size $2j \times 2j$. We first build a Gaussian pyramid $\mathcal{G}(\mathbf{x}) = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_K]$, where $\mathbf{x}_0 = \mathbf{x}$ and $\mathbf{x}_k$ is k repeated applications[1] of $d(.)$ to $\mathbf{x}$. K is the number of levels in the pyramid, selected so that the final level has very small spatial extent ($\leq 8 \times 8$ pixels).

The coefficients $\mathbf{h}_k$ at each level k of the Laplacian pyramid $\mathcal{L}(\mathbf{x})$ are constructed by taking the difference between adjacent levels in the Gaussian pyramid, upsampling

[1] i.e. $\mathbf{x}_2 = d(d(\mathbf{x}))$.

the smaller one with $u(.)$ so that the sizes are compatible:

$$\mathbf{h}_k = \mathcal{L}_k(\mathbf{x}) = \mathcal{G}_k(\mathbf{x}) - u(\mathcal{G}_{k+1}(\mathbf{x})) = \mathbf{x}_k - u(\mathbf{x}_{k+1}) \tag{4.5}$$

Intuitively, each level captures image structure present at a particular scale. The final level of the Laplacian pyramid $\mathbf{h}_K$ is not a difference image, but a low-frequency residual equal to the final Gaussian pyramid level, i.e. $\mathbf{h}_K = \mathbf{x}_K$. Reconstruction from a Laplacian pyramid coefficients $[\mathbf{h}_1, \ldots, \mathbf{h}_K]$ is performed using the backward recurrence:

$$\mathbf{x}_k = u(\mathbf{x}_{k+1}) + \mathbf{h}_k \tag{4.6}$$

which is started with $\mathbf{x}_K = \mathbf{h}_K$ and the reconstructed image being $\mathbf{x} = \mathbf{x}_0$. In other words, starting at the coarsest level, we repeatedly upsample and add the difference image $\mathbf{h}$ at the next finer level until we get back to the full resolution image.

4.2.3 Laplacian Generative Adversarial Networks (LAPGAN)

Our proposed approach combines the conditional GAN model with a Laplacian pyramid representation. The model is best explained by first considering the sampling procedure. Following training (explained below), we have a set of generative convolutional networks $\{G_0, \ldots, G_K\}$, each of which captures the distribution of coefficients $\mathbf{h}_k$ for natural images at a different level of the Laplacian pyramid. Sampling an image is akin to the reconstruction procedure in Equation 4.6, except that the generative models are used to produce the $\mathbf{h}_k$'s:

$$\tilde{\mathbf{x}}_k = u(\tilde{\mathbf{x}}_{k+1}) + \tilde{h}_k = u(\tilde{\mathbf{x}}_{k+1}) + G_k(\mathbf{z}_k, u(\tilde{\mathbf{x}}_{k+1})) \tag{4.7}$$

The recurrence starts by setting $\tilde{\mathbf{x}}_{K+1} = 0$ and using the model at the final level G_K to generate a residual image $\tilde{\mathbf{x}}_K$ using noise vector $\mathbf{z}_K$: $\tilde{\mathbf{x}}_K = G_K(z_K)$. Note that models at all levels except the final are conditional generative models that take an upsampled version of the current image $\tilde{\mathbf{x}}_{k+1}$ as a condtioning variable, in addition to the noise vector $\mathbf{z}_k$. Figure 4.1 shows this procedure in action for a pyramid with $K = 3$ using 4 generative models to sample a 64×64 image.

The generative models $\{G_0, \ldots, G_K\}$ are trained using the CGAN approach at each level of the pyramid. Specifically, we construct a Laplacian pyramid from each training image $\mathbf{x}$. At each level we make a stochastic choice (with equal probability) to either (i) construct the coefficients $\mathbf{h}_k$ either using the standard procedure from Equation 4.5, or (ii) generate them using G_k:

$$\tilde{\mathbf{h}}_k = G_k(z_k, u(\mathbf{x}_{k+1})) \tag{4.8}$$

Note that G_k is a convolutional network which uses a coarse scale version of the image $\mathbf{l}_k = u(\mathbf{x}_{k+1})$ as conditioning input, as well as noise vector $\mathbf{z}_k$. Discriminator D_k takes as input $\mathbf{h}_k$ or $\tilde{\mathbf{h}}_k$, along with the low-pass image $\mathbf{l}_k$) (which is explicitly added

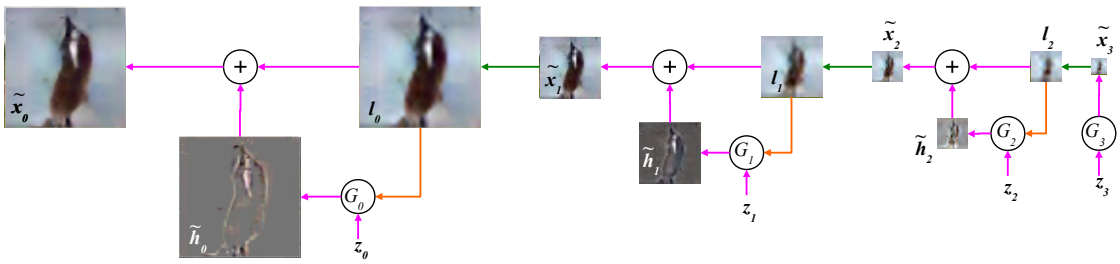

Figure 4.1: The sampling procedure for our LAPGAN model. We start with a noise sample $\mathbf{z}_3$ (right side) and use a generative model G_3 to generate $\tilde{\mathbf{x}}_3$. This is upsampled (green arrow) and then used as the conditioning variable (orange arrow) $\mathbf{l}_2 = u()$ for the generative model at the next level, G_2. Together with another noise sample $\mathbf{z}_2$, G_2 generates a difference image $\tilde{h}_2$ which is added to l_2 to create $\tilde{\mathbf{x}}_2$. This process repeats across two subsequent levels to yield a final full resolution sample $\mathbf{x}_0$.

to $\mathbf{h}_k$ or $\tilde{\mathbf{h}}_k$ before the first convolution layer), and predicts if the image was real or generated. At the final scale of the pyramid, the low frequency residual is sufficiently small that it can be directly modeled with a standard GAN: $\tilde{\mathbf{h}}_K = G_K(\mathbf{z}_K)$ and D_K only has $\mathbf{h}_K$ or $\tilde{h}_K$ as input. The framework is illustrated in Figure 4.2.

Breaking the generation into successive refinements is the key idea in this work. Note that we give up any "global" notion of fidelity; we never make any attempt to train a network to discriminate between the output of a cascade and a real image and instead focus on making each step plausible. Furthermore, the independent training of each pyramid level has the advantage that it is far more difficult for the model to memorize training examples – a hazard when high capacity deep networks are used.

As described, our model is trained in an unsupervised manner. However, we also explore variants that utilize class labels. This is done by adding a 1-hot vector, indicating class identity, as another conditioning variable for G_k and D_k.

4.3 Model Architecture & Training

We apply our approach to three datasets: (i) **CIFAR10** – 32×32 pixel color images of 10 different classes, training samples with tight crops of objects; (ii) **STL** – 96×96 pixel color images of 10 different classes, 100k training samples (we use the unlabeled portion of data); and (iii) **LSUN** (Zhang et al., 2015b) – ∼10M images of 10 different natural scene types, downsampled to 64×64 pixels.

For each dataset, we explored a variety of architectures for $\{G_k, D_k\}$. Model selection was performed using a combination of visual inspection and a heuristic based on ℓ_2 error in pixel space. The heuristic computes the error for a given validation image at level k in the pyramid as $L_k(\mathbf{x}_k) = \min_{\{\mathbf{z}_j\}} ||G_k(\mathbf{z}_j, u(\mathbf{x}_{k+1})) - \mathbf{h}_k||_2$ where $\{\mathbf{z}_j\}$ is a

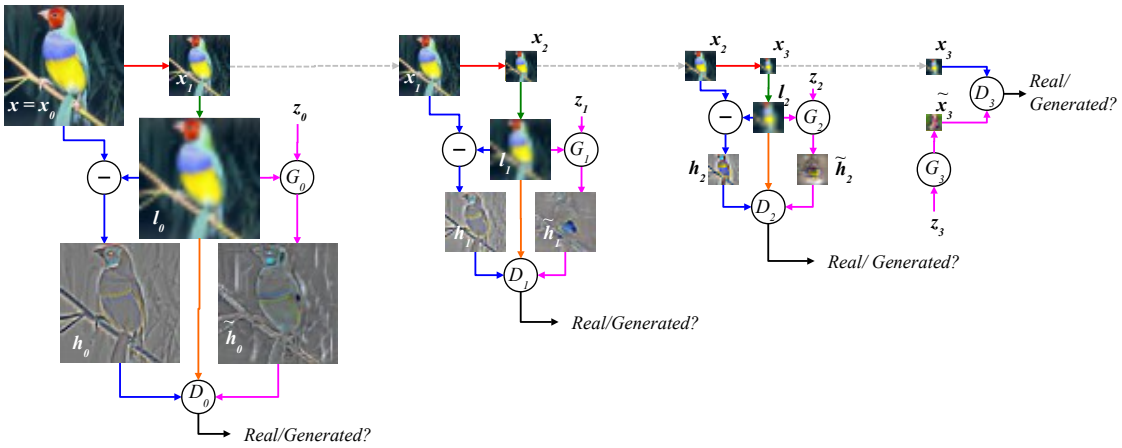

Figure 4.2: The training procedure for our LAPGAN model. Starting with a 64x64 input image $\mathbf{x}$ from our training set (top left): (i) we take $\mathbf{x}_0 = \mathbf{x}$ and blur and downsample it by a factor of two (red arrow) to produce $\mathbf{x}_1$; (ii) we upsample $\mathbf{x}_1$ by a factor of two (green arrow), giving a low-pass version $\mathbf{l}_0$ of $\mathbf{x}_0$; (iii) with equal probability we use l_0 to create *either* a real *or* a generated example for the discriminative model D_0. In the real case (blue arrows), we compute high-pass $h_0 = I_0 - l_0$ which is input to D_0 that computes the probability of it being real vs generated. In the generated case (magenta arrows), the generative network G_0 receives as input a random noise vector z_0 and l_0. It outputs a generated high-pass image $\tilde{h}_0 = G_0(z_0, l_0)$, which is input to D_0. In both the real/generated cases, D_0 also receives l_0 (orange arrow). Optimizing Equation 4.3, G_0 thus learns to generate realistic high-frequency structure $\tilde{h}_0$ consistent with the low-pass image l_0. The same procedure is repeated at scales 1 and 2, using I_1 and I_2. Note that the models at each level are trained independently. At level 3, I_3 is an 8×8 image, simple enough to be modeled directly with a standard GANs G_3 & D_3.

large set of noise vectors, drawn from $p_{noise}(\mathbf{z})$. In other words, the heuristic is asking, *are any of the generated residual images close to the ground truth?* For all models, the noise vector $\mathbf{z}_k$ is drawn from a $\mathcal{U}(-1, 1)$ distribution.

4.3.1 CIFAR10 and STL

Initial scale: This operates at 8×8 resolution, using densely connected nets for both G_K and D_K with 2 hidden layers and ReLU non-linearities. D_K uses Dropout and has 600 units/layer vs 1200 for G_K. $\mathbf{z}_K$ is a 100-d vector.

Subsequent scales: For CIFAR10, we boost the training set size by taking four 28×28 crops from the original images. Thus the two subsequent levels of the pyramid are $8 \rightarrow 14$ and $14 \rightarrow 28$. For STL, we have 4 levels going from $8 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 96$. For both datasets, G_k and D_k are convolutional networks with 3 and 2 layers, respectively. The noise input $\mathbf{z}_k$ to G_k is presented as a 4th "color plane to low-pass $\mathbf{l}_k$, hence its dimensionality varies with the pyramid level. For CIFAR10, we also explore a class conditional version of the model, where a 1-hot vector encodes the label. This is integrated into G_k & D_k by passing it through a linear layer whose output is reshaped into a single plane feature map which is then concatenated with the 1st layer maps.

The loss in Equation 4.3 is trained using SGD with an initial learning rate of 0.02, decreased by a factor of $1e \times 10^{-5}$) at each epoch. Momentum starts at 0.5, increasing by 0.0008 at epoch up to a maximum of 0.8. During training, we monitor log-likelihood using a Parzen-window estimator and retain the best performing model.

4.3.2 LSUN

The larger size of this dataset allows us to train a separate LAPGAN model for each the 10 different scene classes.

Initial scale: This operates at 4×4 resolution, using densely connected nets for both G_K & D_K with 2 hidden layers and ReLU non-linearities. D_K uses Dropout and has 600 units/layer vs 1200 for G_K. $\mathbf{z}_K$ is a 100-d vector.

Subsequent scales: The four subsequent scales $4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 64$ use a common architecture for G_k & D_k at each level. G_k is a 5-layer convolutional networks with $\{64, 368, 128, 224\}$ feature maps and a linear output layer. 7×7 filters, ReLUs, batch normalization (Ioffe and Szegedy, 2015) and Dropout are used at each hidden layer. D_k has 3 hidden layers with $\{48, 448, 416\}$ maps plus a sigmoid output. Note that G_k and D_k are substantially larger than those used for CIFAR10 and STL, as afforded by the larger training set.

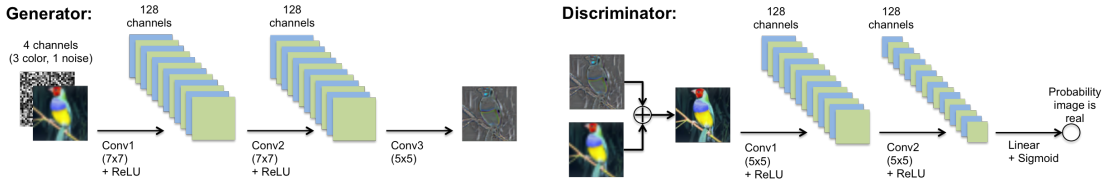

Figure 4.3: LAPGAN architecture of CIFAR-10 and STL-10 models.

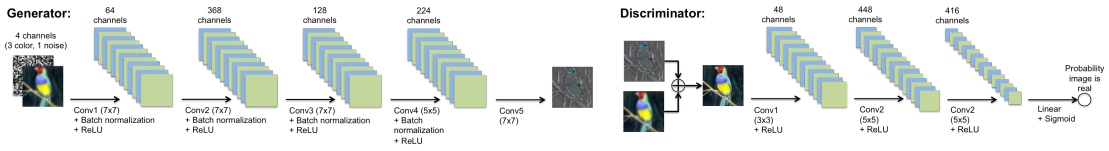

Figure 4.4: LAPGAN architecture of LSUN models.

4.4 Experiments

We evaluate our approach using 3 different methods: (i) computation of log-likelihood on a held out image set; (ii) drawing sample images from the model and (iii) a human subject experiment that compares (a) our samples, (b) those of baseline methods and (c) real images.

4.4.1 Evaluation of Log-Likelihood

Like Goodfellow et al. (2014), we are compelled to use a Gaussian Parzen window estimator to compute log-likelihood, since there is no direct way of computing it using our model. Table 4.1 compares the log-likelihood on a validation set for our LAPGAN model and a standard GAN using 50k samples for each model (the Gaussian width σ was also tuned on the validation set). Our approach shows a marginal gain over a GAN. However, we can improve the underlying estimation technique by leveraging the multi-scale structure of the LAPGAN model. This new approach computes a probability at each scale of the Laplacian pyramid and combines them to give an overall image probability.

To describe the log-likelihood computation in our model, let us consider a two scale pyramid for the moment. Given a (vectorized) $j \times j$ image $\mathbf{x}$, denote by $\mathbf{l} = d(\mathbf{x})$ the coarsened image, and $\mathbf{h} = \mathbf{x} - u(d(\mathbf{x}))$ to be the high pass. To simplify the computations, we use a slightly different u operator than the one described in section4.2.2. Namely, here we take $d(\mathbf{x})$ to be the mean over each disjoint block of 2×2 pixels, and take u to be the operator that removes the mean from each 2×2 block. Since u has rank $3j^2/4$, we write $\mathbf{h}$ in an orthonormal basis of the range of u, then the (linear) mapping

from $\mathbf{x}$ to $(\mathbf{l}, \mathbf{h})$ is unitary. We now build a probability density p on $\mathbb{R}^{j^2}$ by:

$$p(\mathbf{x}) = q_0(\mathbf{l}, \mathbf{h}) q_1(\mathbf{l}) = q_0(d(\mathbf{x}), h(\mathbf{x})) q_1(d(\mathbf{x})) \tag{4.9}$$

In a moment we will carefully define the functions q_i. For now, suppose that $q_i \geq 0$, $\int q_1(\mathbf{l})\, dl = 1$, and for each fixed $\mathbf{l}$, $\int q_0(\mathbf{l}, \mathbf{h})\, dh = 1$. Then we can check that p has unit integral:

$$\int p\, dx = \int q_0(d(\mathbf{x}), h(\mathbf{x})) q_1(d(\mathbf{x})) dx = \int \int q_0(\mathbf{l}, \mathbf{h}) q_1(\mathbf{l})\, dl\, dh = 1 \tag{4.10}$$

Now we define the q_i with Parzen window approximations to the densities of each of the scales. For q_1, we take a set of coarsened training samples $\mathbf{l}_1, ..., \mathbf{l}_{N_0}$, and construct the density function $q_1(\mathbf{l}) \sim \sum_{i=1}^{N_1} e^{||\mathbf{l}-\mathbf{l}_i||^2/\sigma_1}$. We fix $\mathbf{l} = d(\mathbf{x})$, and using this fixed $\mathbf{l}$, we sample N_0 points $\tilde{\mathbf{h}}_1, ..., \tilde{\mathbf{h}}_{N_1}$ from the generative model, and define

$$q_0(\mathbf{x}) = q_0(\mathbf{l}, \mathbf{h}) \sim \sum_{i=1}^{N_0} e^{||\mathbf{h}-\tilde{\mathbf{h}}_i||^2/\sigma_0} \tag{4.11}$$

Note that when defined this way, it is not obvious that q_0 is a measurable function, as the choice of $\mathbf{h}_i$ by the up-sampling model is different for every $\mathbf{l}$ (and in fact depends on the random seed we used to sample). However, because the mapping from fixed "noise variable" and coarse image to refinement is the forward of a convolutional net, and so is continuous, if we use the same random seeds for each $\mathbf{x}$, q_1 is measurable. For pyramids with more levels, we continue in the same way for each of the finer scales. Note we always use the true low pass at each scale, and measure the true high pass against the high pass samples generated from the model. Thus for a pyramid with K levels, the final log likelihood will be: $\log(q_K(\mathbf{l}_K)) + \sum_{k=0}^{K-1} \log(q_k(\mathbf{l}_k, \mathbf{h}_k))$.

Model	CIFAR10 (@32×32)	STL10 (@32×32)
GAN (Parzen window estimate)	-3617 ± 353	-3661 ± 347
LAPGAN (Parzen window estimate)	-3572 ± 345	-3563 ± 311
LAPGAN (multi-scale Parzen window estimate)	-1799 ± 826	-2906 ± 728

Table 4.1: Log-likelihood estimates for a standard GAN and our proposed LAPGAN model on CIFAR10 and STL10 datasets. The mean and std. dev. are given in units of nats/image. Rows 1 and 2 use a Parzen-window approach at full-resolution, while row 3 uses our multi-scale Parzen-window estimator.

We use this procedure to compute the log-likelihoods for CIFAR-10 and STL-10 images (both at 32×32 resolution). The parameter σ (controlling the Parzen window size) was chosen using the validation set. Our multi-scale Parzen estimate, shown in Table 4.1, produces a big gain over the traditional estimator.

The shortcomings of both estimators are readily apparent when compared to a simple Gaussian, fit to the CIFAR-10 training set. Even with added noise, the resulting model can obtain a far higher log-likelihood than either the GAN or LAPGAN models, or other published models. More generally, log-likelihood is problematic as a performance measure due to its sensitivity to the exact representation used. Small variations in the scaling, noise and resolution of the image (much less changing from RGB to YUV, or more substantive changes in input representation) results in wildly different scores, making fair comparisons to other methods difficult. Since publication of this work, Theis et al. (2016) have further outlined problems with applying the Parzen window estimate to high dimensional image data.

4.4.2 Model Samples

The Parzen window approach is appealing due to the objective nature of the metric. However, there is little correspondence between the score and perceptual fidelity of generated images. To this end, we also also show samples from models trained on

CIFAR-10, STL-10 and LSUN datasets.

Figure 4.5 shows samples from our models trained on CIFAR10. Samples from the class conditional LAPGAN are organized by class. Our reimplementation of the fully connected GAN (Goodfellow et al., 2014) produces slightly sharper images than those shown in the original paper. We attribute this improvement to the introduction of data augmentation. The LAPGAN samples improve upon the standard GAN samples. They appear more object-like and have more clearly defined edges. Conditioning on a class label improves the generations as evidenced by the clear object structure in the conditional LAPGAN samples. The quality of these samples compares favorably with other generative models of the time (Gregor et al., 2015; Sohl-Dickstein et al., 2015).

Figure 4.8(a) shows samples from our LAPGAN model trained on STL-10. Here, we lose clear object shape but the samples remain sharp. Figure 4.8(b) shows the generation chain for random STL-10 samples.

Figure 4.6 shows 64×64 samples from LAPGAN models trained on three LSUN categories (tower, bedroom, church front). Collectively, these show the models capturing complex structure within the scenes, being able to recompose scene elements into credible looking images. These samples reflect the first instance of a generative model producing samples of this complexity and quality. The substantial gain in quality over the CIFAR-10 and STL-10 samples is likely due to the much larger training LSUN training set which allows us to train bigger and deeper models.

Figure 4.7 shows 128×128 samples from LAPGAN models trained on the same three LSUN categories. Here, we see some detailed structure added, beyond that already present at 64×64. However, many texture artifacts are also evident at this high resolution.

Figure 4.5: CIFAR-10 samples: our class conditional CC-LAPGAN model, our LAPGAN model and the standard GAN model of (Goodfellow et al., 2014).

Figure 4.6: 64×64 samples from three different LSUN LAPGAN models (top: tower, middle: bedroom, bottom: church front)

Figure 4.7: 128×128 samples from three different LSUN LAPGAN models (top: tower, middle: bedroom, bottom: church front)

(a) (b)

Figure 4.8: STL10 samples: **(a)** Random 96x96 samples from our LAPGAN model. **(b)** Coarse-to-fine generation chain.

4.4.3 Human Evaluation of Samples

To obtain a quantitative measure of sample quality, we asked 15 volunteers to participate in an experiment to see if they could distinguish our samples from real images. The subjects were presented with the user interface shown in Figure 4.9(right) and shown at random four different types of image: samples drawn from three different GAN models trained on CIFAR-10 ((i) LAPGAN, (ii) class conditional LAPGAN and (iii) standard GAN (Goodfellow et al., 2014)) and also real CIFAR-10 images. After being presented with the image, the subject clicked the appropriate button to indicate if they believed the image was real or generated. Since accuracy is a function of viewing time, we also randomly pick the presentation time from one of 11 durations ranging from 50ms to 2000ms, after which a gray mask image is displayed. Before the experiment commenced, they were shown examples of real images from CIFAR-10. After collecting $\sim$10k samples from the volunteers, we plot in Figure 4.9 the fraction of images believed to be real for the four different data sources, as a function of presentation time. The curves show our models produce samples that are far more realistic than those from standard GAN (Goodfellow et al., 2014).

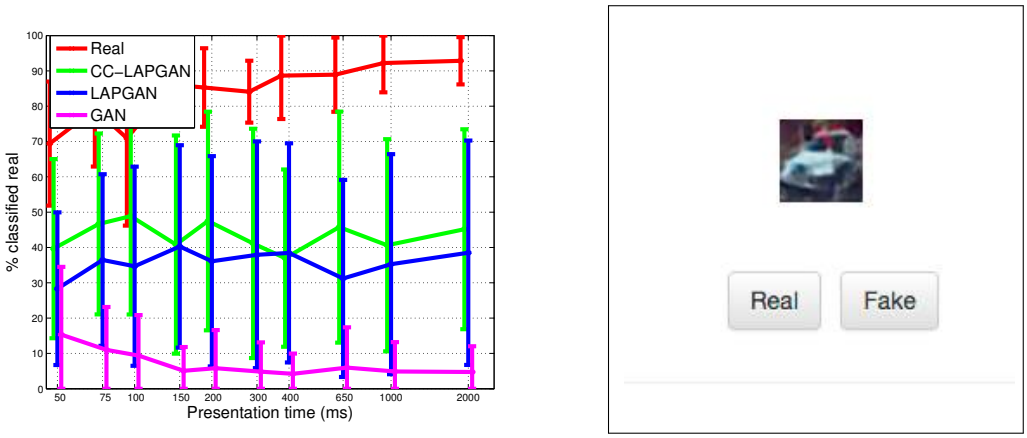

Figure 4.9: Left: Human evaluation of real CIFAR-10 images (red) and samples from Goodfellow et al. (2014) (magenta), our LAPGAN (blue) and a class conditional LAPGAN (green). The error bars show $\pm 1\sigma$ of the inter-subject variability. Around 40% of the samples generated by our class conditional LAPGAN model are realistic enough to fool a human into thinking they are real images. This compares with $\leq 10\%$ of images from the standard GAN model (Goodfellow et al., 2014), but is still a lot lower than the $> 90\%$ rate for real images. Right: The user-interface presented to the subjects.

4.4.4 Sample Variability and Overfitting

Evaluating perceptual fidelity of samples is a crucial, but incomplete, method of assessing generative models. For example, a model that simply retrieves images from a fixed dataset will produce visually appealing samples and yet remain unsatisfying as a generative model. Extra care must be taken to ensure a model has truly learned to synthesize new examples, rather than merely memorizing the training set. One common approach to assessing over-fitting is to compare generated samples with their nearest neighbor in the training set. Figure 4.10 shows nearest neighbors in the training set, using $\ell 2$ distance in pixel space, of generated CIFAR-10 samples. Figure 4.11 shows nearest neighbors using L2 distance in feature space of a state-of-the-art convnet model[2], of generated CIFAR-10 samples. These figure show that the model is not simply memorizing the training examples.

[2]Using this Network in Network model: https://gist.github.com/mavenlin/e56253735ef32c3c296d

Figure 4.12, Figure 4.13 and Figure 4.14 show samples drawn using the same 4×4 initial image (shown in leftmost column). Specifically, after generating from the 1st level GAN, the image is fixed and 8 different samples are then drawn, each using a different set of random noise vectors. These samples show that models produce plausible variations that cannot be the result of trivial copying of the training examples.

We can also condition the generation process on different coarse resolution images while keeping the noise vectors at each level fixed. Figure 4.15(a), Figure 4.15(b) and Figure 4.15(c) show samples drawn from our LSUN tower, bedroom and church models respectively. The coarsest image (leftmost column) in the top and bottom rows of each figure were sampled from our 4x4 GAN. The intermediate coarse images were constructed by linearly interpolating between these two images. Each column shows a sample from a different level of the pyramid conditioned on the coarser image in the previous column. The same noise vectors were used for each row so that the only source of variation comes from the 4x4 images. An indication of overfitting would be the presence of sharp transitions in the generated images, despite the smoothly varying coarse input, as the model snaps between training examples. But this is not observed: the generations at each scale smoothly transition. Furthermore, each high resolution image looks like a plausible natural image, rather than a linear blend between two images. This indicates our model is moving along the manifold of natural images, rather than on a line between the start and end images.

4.5 Discussion

Natural images are highly complex and modeling detailed structure at high resolutions is difficult. To address this problem, we have proposed a conceptually simple

multi-scale generative model that builds upon the GAN framework of Goodfellow et al. (2014). This approach, known as the Laplacian pyramid of adversarial networks (LAPGAN) decomposes the generation process across multiple image scales. A different conditional GAN is trained to capture the distribution of natural images at one level of the Laplacian pyramid, conditioned on coarser resolution images from the level above. A key point in our work is giving up any "global" notion of fidelity, and instead breaking the generation into plausible successive refinements. Images can be sampled from the model in a simple coarse-to-fine fashion.

LAPGAN produces high-quality image samples that, at the time of publication, were qualitatively superior to alternative deep generative modeling approaches. This work was the first to demonstrate the effectiveness of convolutional GANs for image generation. Subsequent work proposed improved convolutional architectures (Radford et al., 2016) and tricks to improve training stability (Salimans et al., 2016). These early works demonstrated the potential of the adversarial framework and research has progressed rapidly in recent years. GAN based models are currently capable of generating crisp, high quality, high resolution (e.g. 1024×1024) images (Karras et al., 2018; Zhang et al., 2018b). Many of the state-of-the-art GAN generation models utilize a multi-scale approach similar to our LAPGAN approach (e.g. Karras et al. 2018; Zhang et al. 2017, 2018c).

Figure 4.10: Samples drawn from our class conditional CIFAR-10 model, with nearest neighbors in L2 pixel space shown in adjacent columns (orange).

Figure 4.11: Samples drawn from our class conditional CIFAR-10 model, with nearest neighbors in feature space shown in adjacent columns (orange).

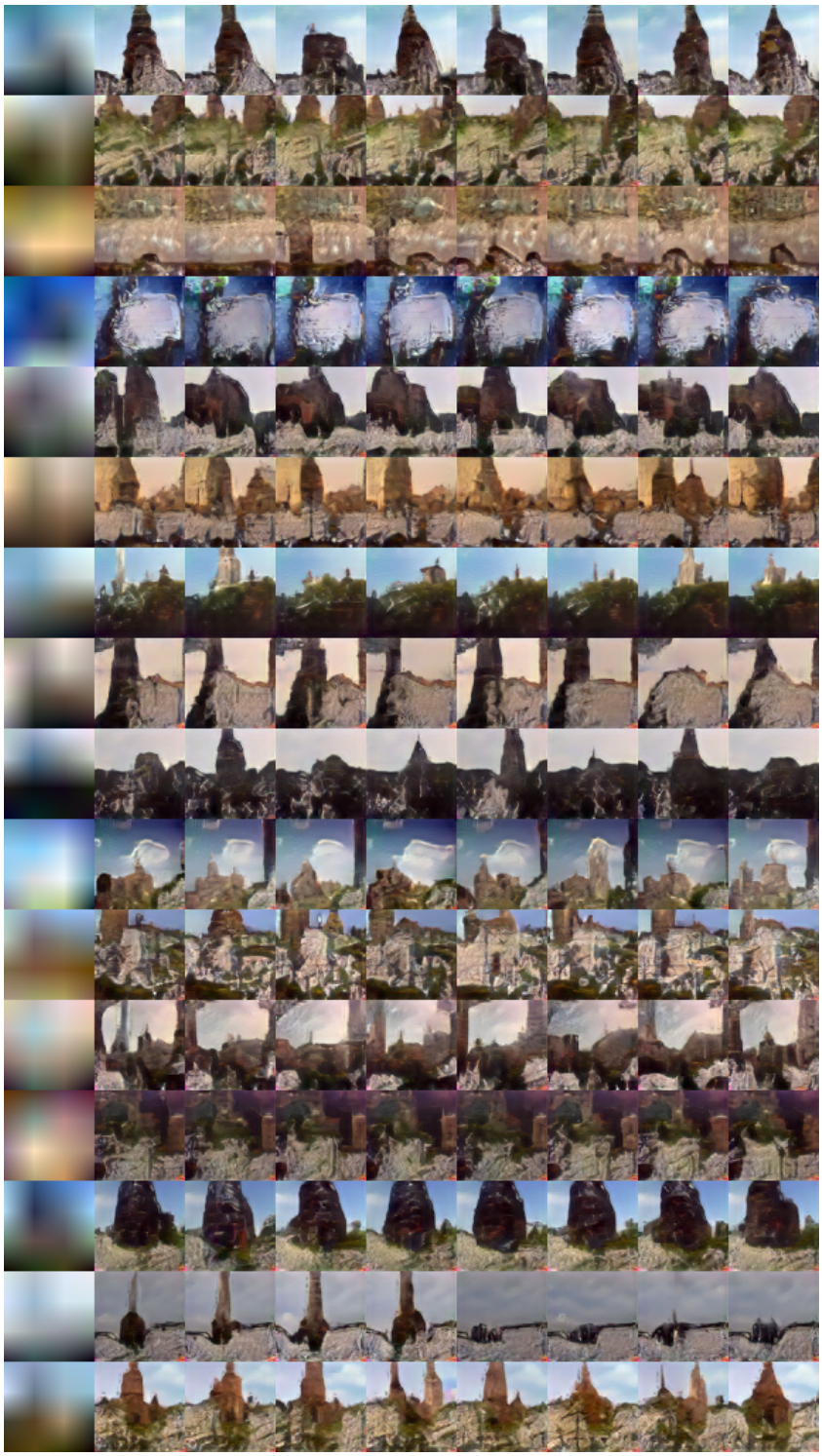

Figure 4.12: LSUN sample from class conditional LAPGAN model (tower) , seeded with generated 4×4 images (1st columns), with other columns showing different draws from the model.

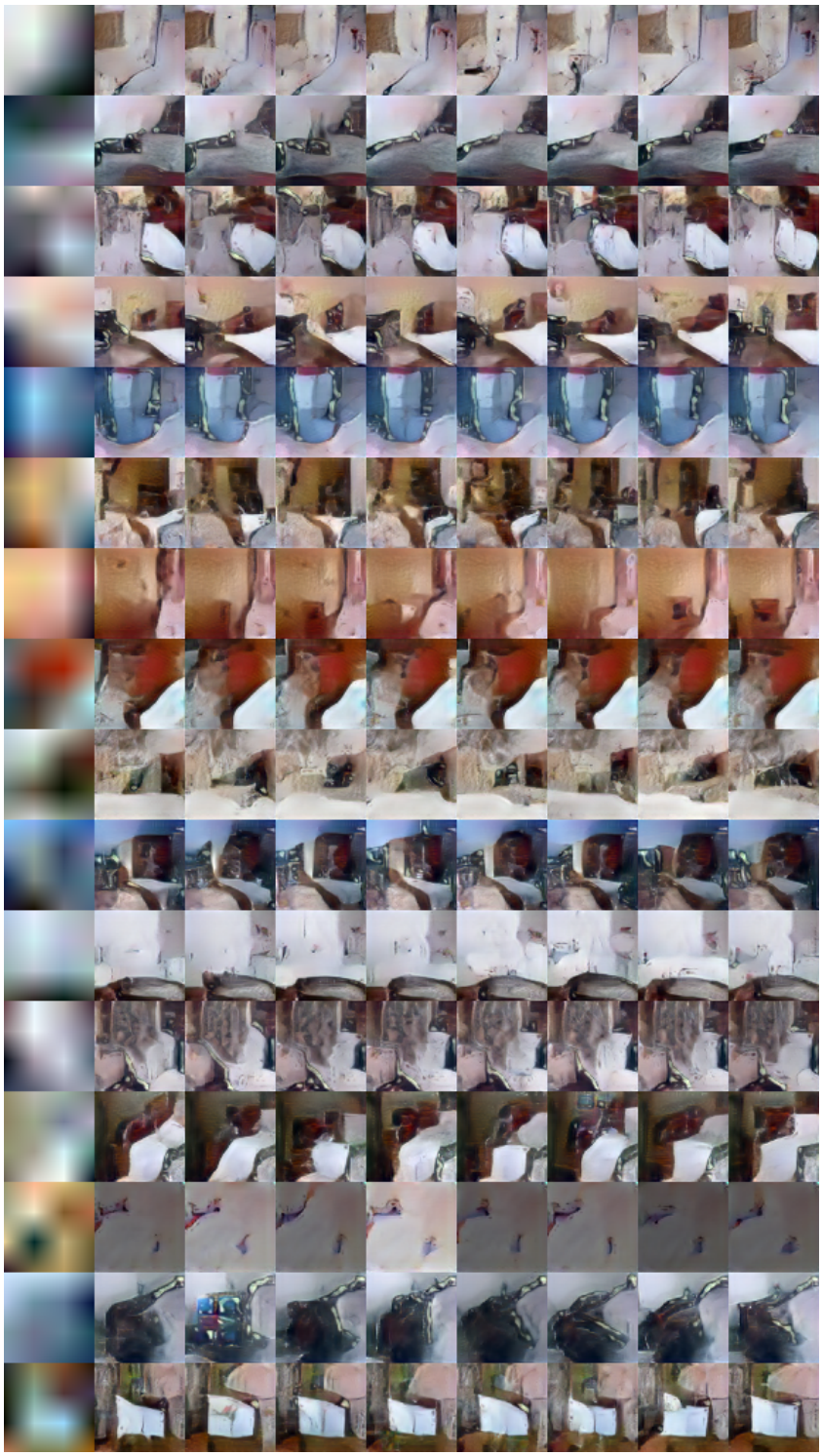

Figure 4.13: LSUN sample from class conditional LAPGAN model (bedroom) , seeded with generated 4×4 images (1st columns), with other columns showing different draws from the model.

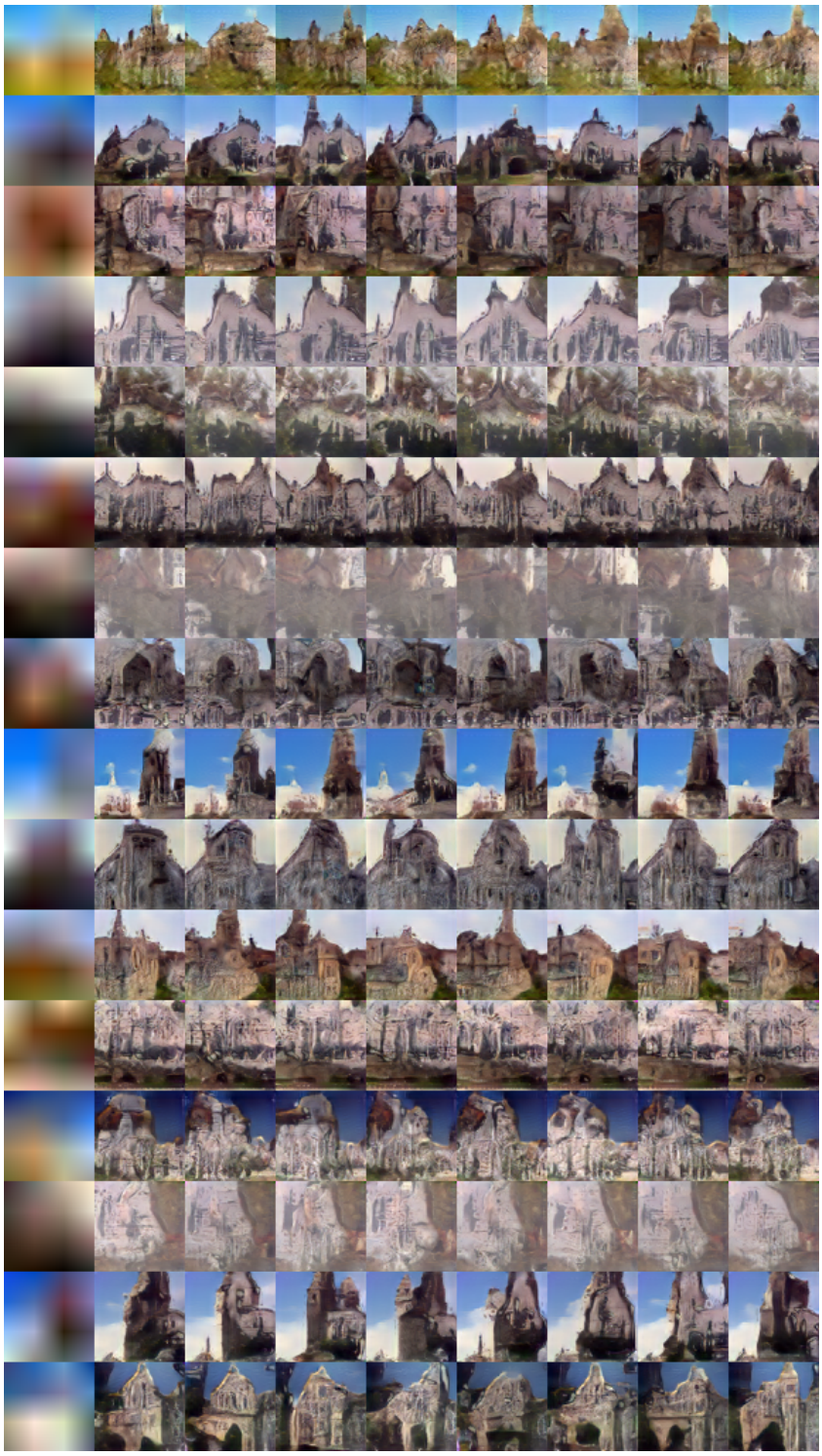

Figure 4.14: LSUN sample from class conditional LAPGAN model (church) , seeded with generated 4×4 images (1st columns), with other columns showing different draws from the model.

(a) (b) (c)

Figure 4.15: Effect of varying the coarsest input, with fixed noise at subsequent layers, on **(a)** tower model, **(b)** bedroom model and **(c)** church model.

Chapter 5

Disentangling Content and Pose for Video Prediction

Video prediction is frequently cast at the pixel level where the goal is to predict the *pixels* of future frames, conditioned on past observations. Predicting future pixels is a natural objective, especially when additional annotations (e.g. object trajectories) are unavailable. An alternative approach involves first constructing a high level representation of individual frames and subsequently training a predictive model in this latent space. Here, the core challenge involves finding a representation effective for the downstream prediction task. One obvious solution is to use features from a pre-trained discriminative network (Srivastava et al., 2015; Vondrick et al., 2016a). However, this requires access to labeled data in the video domain or a close enough correspondence between the video data and a large labeled dataset. Additionally, without a decoder this method is not ideal for settings in which pixel level predictions are required.

In this chapter we introduce a method of learning latent representations suitable for video prediction via an encoder-decoder framework. Our model, Disentangled Rep-

resentation Net (DRNET), factorizes a single frame into two components, one that is roughly constant throughout the video clip and another that captures the dynamic aspects of the sequence. We refer to these components as *content* and *pose* respectively. The learned content/pose representation affords particularly easy video prediction since only the pose representation changes across the clip. We train a recurrent model to predict a future sequence of pose vectors, conditioned on past observations. Content vectors can be ignored by the recurrent predictor since, by design, they only encode time-invariant information. The recurrent model is trained entirely in latent pose space but the DRNET decoder is available to convert latent vectors into images at test time.

5.1 Introduction

Unsupervised learning from video is a long-standing problem in computer vision and machine learning. The goal is to learn, without explicit labels, a representation that generalizes effectively to a previously unseen range of tasks, such as semantic classification of the objects present, predicting future frames of the video or classifying the dynamic activity taking place. There are several prevailing paradigms: the first, known as self-supervision, uses domain knowledge to implicitly provide labels (e.g. predicting the relative position of patches on an object (Doersch et al., 2015) or using feature tracks (Wang and Gupta, 2015)). This allows the problem to be posed as a classification task with self-generated labels. The second general approach relies on auxiliary action labels, available in real or simulated robotic environments. These can either be used to train action-conditional predictive models of future frames (Chiappa et al., 2017; Oh et al., 2015) or inverse-kinematics models (Agrawal et al., 2016) which attempt to predict actions from current and future frame pairs. The third and most general approaches

are predictive auto-encoders (e.g.(Hinton and Salakhutdinov, 2006; Kalchbrenner et al., 2016; Mathieu et al., 2016a; Srivastava et al., 2015)) which attempt to predict future frames from current ones. To learn effective representations, some kind of constraint on the latent representation is required.

In this paper, we introduce a form of predictive auto-encoder which uses a novel adversarial loss to factor the latent representation for each video frame into two components, one that is roughly time-independent (i.e. approximately constant throughout the clip) and another that captures the dynamic aspects of the sequence, thus varying over time. We refer to these as *content* and *pose* components, respectively. The adversarial loss relies on the intuition that while the content features should be distinctive of a given clip, individual pose features should not. Thus the loss encourages pose features to carry no information about clip identity. Empirically, we find that training with this loss to be crucial to inducing the desired factorization.

We explore the disentangled representation produced by our model, which we call Disentangled-Representation Net (DRNET), on a variety of tasks. The first of these is predicting future video frames, something that is straightforward to do using our representation. We apply a standard LSTM model to the pose features, conditioning on the content features from the last observed frame. Despite the simplicity of our model relative to other video generation techniques, we are able to generate convincing long-range frame predictions, out to hundreds of time steps in some instances. This is significantly further than existing approaches that use real video data. We also show that DRNET can be used for classification. The content features capture the semantic content of the video thus can be used to predict object identity. Alternately, the pose features can be used for action prediction.

5.2 Approach

In our model, two separate encoders produce distinct feature representations of content and pose for each frame. They are trained by requiring that the content representation of frame $\mathbf{x}^t$ and the pose representation of future frame $\mathbf{x}^{t+k}$ can be combined (via concatenation) and decoded to predict the pixels of future frame $\mathbf{x}^{t+k}$. However, this reconstruction constraint alone is insufficient to induce the desired factorization between the two encoders. We thus introduce a novel adversarial loss on the pose features that prevents them from being discriminable from one video to another, thus ensuring that they cannot contain content information. A further constraint, motivated by the notion that content information should vary slowly over time, encourages temporally close content vectors to be similar to one another.

More formally, let $\mathbf{x}_i = (\mathbf{x}_i^1, ..., \mathbf{x}_i^T)$ denote a sequence of T images from video i. We subsequently drop index i for brevity. Let E_c denote a neural network that maps an image $\mathbf{x}^t$ to the *content* representation $\mathbf{h}_c^t$ which captures structure shared across time. Let E_p denote a neural network that maps an image $\mathbf{x}^t$ to the *pose* representation $\mathbf{h}_p^t$ capturing content that varies over time. Let D denote a decoder network that maps a content representation from a frame, $\mathbf{h}_c^t$, and a pose representation $\mathbf{h}_p^{t+k}$ from future time step $t+k$ to a prediction of the future frame $\tilde{\mathbf{x}}^{t+k}$. Finally, C is the *scene discriminator network* that takes pairs of pose vectors $(\mathbf{h}_1, \mathbf{h}_2)$ and outputs a scalar probability that they came from the same video or not.

The loss function used during training has several terms:

Reconstruction loss: We use a standard per-pixel ℓ_2 loss between the predicted future frame $\tilde{\mathbf{x}}^{t+k}$ and the actual future frame $\mathbf{x}^{t+k}$ for some random frame offset $k \in$

$[0, K]$:

$$\mathcal{L}_{reconstruction}(E_c, E_p, D) = ||D(E_c(\mathbf{x}^t), E_p(\mathbf{x}^{t+k})) - \mathbf{x}^{t+k}||_2^2 \tag{5.1}$$

We find the simple ℓ_2 loss to be highly effective but note that more complex loss functions, such as an adversarial (Goodfellow et al., 2014; Mathieu et al., 2016b) or gradient difference loss (Mathieu et al., 2016b), could be utilized here.

Similarity loss: To ensure the content encoder extracts mostly time-invariant representations, we penalize the squared error between the content features $\mathbf{h}_c^t, \mathbf{h}_c^{t+k}$ of neighboring frames $k \in [0, K]$:

$$\mathcal{L}_{similarity}(E_c) = ||E_c(\mathbf{x}^t) - E_c(\mathbf{x}^{t+k})||_2^2 \tag{5.2}$$

The similarity loss is related to Slow Feature Analysis (Wiskott and Sejnowski, 2002) where the rate of change in time-independent components is penalized.

Adversarial loss: We now introduce a novel adversarial loss that exploits the fact that the objects present do not typically change *within* a video, but they do *between* different videos. Our desired disenanglement would thus have the content features be (roughly) constant within a clip, but distinct between them. This implies that the pose features should not carry any information about the identity of objects within a clip.

We impose this via an adversarial framework between the scene discriminator network C and pose encoder E_p, shown in Figure 6.2. The latter provides pairs of pose vectors, either computed from the same video $(\mathbf{h}_{p,i}^t, \mathbf{h}_{p,i}^{t+k})$ or from different ones $(\mathbf{h}_{p,i}^t, \mathbf{h}_{p,j}^{t+k})$, for some other video j. The discriminator then attempts to classify the pair as being from the same/different video using a cross-entropy loss:

$$-\mathcal{L}_{adversarial}(C) = \log(C(E_p(\mathbf{x}_i^t), E_p(\mathbf{x}_i^{t+k}))) + \log(1 - C(E_p(\mathbf{x}_i^t), E_p(\mathbf{x}_j^{t+k}))) \tag{5.3}$$

The other half of the adversarial framework imposes a loss function on the pose encoder E_p that tries to maximize the uncertainty (entropy) of the discriminator output on pairs of frames from the same clip:

$$-\mathcal{L}_{adversarial}(E_p) = \frac{1}{2}\log(C(E_p(x_i^t), E_p(\mathbf{x}_i^{t+k}))) + \frac{1}{2}\log(1 - C(E_p(\mathbf{x}_i^t), E_p(\mathbf{x}_i^{t+k})))$$
(5.4)

Thus the pose encoder is encouraged to produce features that the discriminator is unable to classify if they come from the same clip or not. In so doing, the pose features cannot carry information about object content, yielding the desired factorization. Note that this does assume that the object's pose is not distinctive to a particular clip. While adversarial training is also used by GANs, our setup purely considers classification; there is no generator network, for example.

Overall training objective:

During training we minimize the sum of the above losses, with respect to E_c, E_p, D and C:

$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, E_p, D) + \alpha\mathcal{L}_{similarity}(E_c) + \beta(\mathcal{L}_{adversarial}(E_p) + \mathcal{L}_{adversarial}(C))$$
(5.5)

where α and β are hyper-parameters. The first three terms can be jointly optimized, but the discriminator C is updated while the other parts of the model (E_c, E_p, D) are held constant. The overall model is shown in Figure 5.1. Details of the training procedure and model architectures for E_c, E_p, D and C are given in Section 5.3.1.

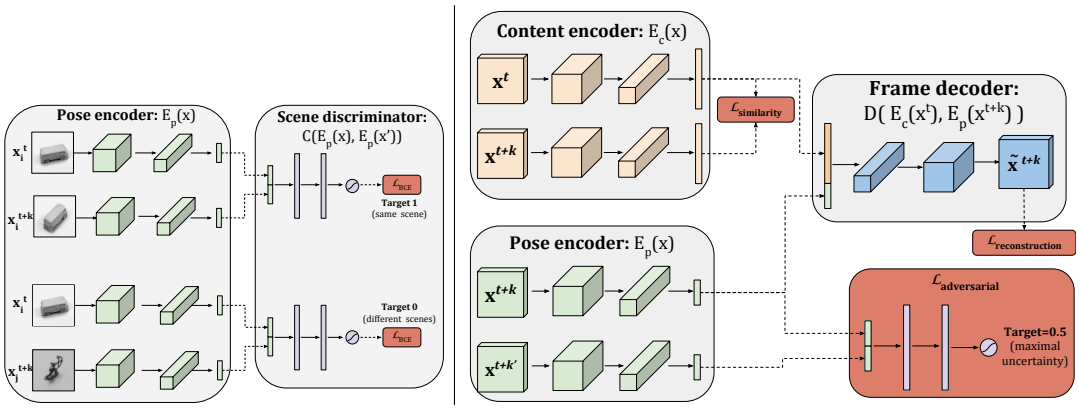

Figure 5.1: Left: The discriminator C is trained with binary cross entropy (BCE) loss to predict if a pair of pose vectors comes from the same (top portion) or different (lower portion) scenes. $\mathbf{x}_i$ and $\mathbf{x}_j$ denote frames from different sequences i and j. The frame offset k is sampled uniformly in the range $[0, K]$. Note that when C is trained, the pose encoder E_p is fixed. Right: The overall model, showing all terms in the loss function. Note that when the pose encoder E_p is updated, the scene discriminator is held fixed.

5.2.1 Forward Prediction

After training, the pose and content encoders E_p and E_c provide a representation which enables video prediction in a straightforward manner. Given a frame $\mathbf{x}^t$, the encoders produce h_p^t and h_c^t respectively. To generate the next frame, we use these as input to an LSTM model to predict the next pose features h_p^{t+1}. These are then passed (along with the content features) to the decoder, which generates a pixel-space prediction $\tilde{\mathbf{x}}^{t+1}$:

$$\tilde{h}_p^{t+1} = LSTM(E_p(\mathbf{x}^t), h_c^t) \qquad \tilde{\mathbf{x}}^{t+1} = D(\tilde{h}_p^{t+1}, h_c^t) \tag{5.6}$$

$$\tilde{h}_p^{t+2} = LSTM(\tilde{h}_p^{t+1}, h_c^t) \qquad \tilde{\mathbf{x}}^{t+2} = D(\tilde{h}_p^{t+2}, h_c^t) \tag{5.7}$$

Note that while pose estimates are generated in a recurrent fashion, the content features h_c^t remain fixed from the last observed real frame. This relies on the nature of $\mathcal{L}_{reconstruction}$ which ensured that content features can be combined with future pose

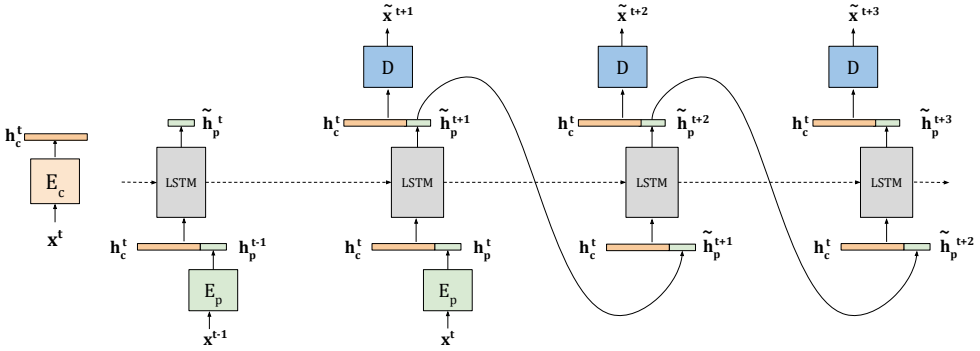

Figure 5.2: Generating future frames by recurrently predicting $\mathbf{h}_p$, the latent pose vector.

vectors to give valid reconstructions.

The LSTM is trained separately from the main model using a standard ℓ_2 loss between $\tilde{h}_p^{t+1}$ and h_p^{t+1}. Note that this generative model is far simpler than many other recent approaches, e.g. Kalchbrenner et al. (2016). This largely due to the forward model being applied within our disentangled representation, rather than directly on raw pixels.

5.2.2 Classification

Another application of our disentangled representation is to use it for classification tasks. Content features, which are trained to be invariant to local temporal changes, can be used to classify the semantic content of an image. Conversely, a sequence of pose features can be used to classify actions in a video sequence. In either case, we train a two layer classifier network S on top of either $\mathbf{h}_c$ or $\mathbf{h}_p$, with its output predicting the class label y.

5.3 Experiments

We evaluate our model on both synthetic (MNIST, NORB, SUNCG) and real (KTH Actions) video datasets. We explore several tasks with our model: (i) the ability to cleanly factorize into content and pose components; (ii) forward prediction of video frames using the approach from Section 5.2.1; (iii) using the pose/content features for classification tasks.

5.3.1 Model details

We explored a variety of convolutional architectures for the content encoder E_c, pose encoder E_p and decoder D. For MNIST, E_c, E_p and D all use a DCGAN architecture (Radford et al., 2016) with $|h_p| = 5$ and $|h_c| = 128$. The encoders consist of 5 convolutional layers with subsampling. Batch normalization and Leaky ReLU's follow each convolutional layer except the final layer which normalizes the pose/content vectors to have unit norm. The decoder is a mirrored version of the encoder with 5 deconvolutional layers and a sigmoid output layer.

For both NORB and SUNCG, D is a DCGAN architecture while E_c and E_p use a ResNet-18 architecture (He et al., 2016) up until the final pooling layer with $|h_p| = 10$ and $|h_c| = 128$.

For KTH, E_p uses a ResNet-18 architecture with $|h_p| = 5$. E_c uses the same architecture as VGG16 (Simonyan and Zisserman, 2015) up until the final pooling layer with $|h_c| = 128$. The decoder is a mirrored version of the content encoder with pooling layers replaced with spatial up-sampling. In the style of U-Net (Ronneberger et al., 2015), we add skip connections from the content encoder to the decoder, enabling the model to easily generate static background features.

In all experiments the scene discriminator C is a fully connected neural network with 2 hidden layers of 100 units. We trained all our models with the ADAM optimizer (Kingma and Ba, 2015) and learning rate $\eta = 0.002$. We used $\beta = 0.1$ for MNIST, NORB and SUNCG and $\beta = 0.0001$ for KTH experiments. We used $\alpha = 1$ for all datasets.

For future prediction experiments we train a two layer LSTM with 256 cells using the ADAM optimizer. On MNIST, we train the model by observing 5 frames and predicting 10 frames. On KTH, we train the model by observing 10 frames and predicting 10 frames.

5.3.2 Synthetic datasets

MNIST: We start with a toy dataset consisting of two MNIST digits bouncing around a 64x64 image. Each video sequence consists of a different pair of digits with independent trajectories. Figure 5.3(left) shows how the content vector from one frame and the pose vector from another generate new examples that transfer the content and pose from the original frames. This demonstrates the clean disentanglement produced by our model. Interestingly, for this data we found it to be necessary to use a different color for the two digits. Our adversarial term is so aggressive that it prevents the pose vector from capturing any content information, thus without a color cue the model is unable to determine which pose information to associate with which digit. In Figure 5.3(right) we perform forward modeling using our representation, demonstrating the ability to generate crisp digits 500 time steps into the future.

NORB: We apply our model to the NORB dataset (LeCun et al., 2004), converted into videos by taking sequences of different azimuths, while holding object identity, lighting and elevation constant. Figure 5.4(left) shows that our model is able to factor

content and pose cleanly on held out data. In Figure 5.4(center) we train a version of our model without the adversarial loss term, which results in a significant degradation in the model and the pose vectors are no longer isolated from content. For comparison, we also show the factorizations produced by Mathieu et al. (2016b), which are less clean, both in terms of disentanglement and generation quality than our approach.

We also evaluate the learned content and pose representation in a classification task. We used a two layer fully connected network with 256 hidden units as the classifier. Leaky ReLUs, batch normalization and dropout were used in every layer. We trained with ADAM as used early stopping on a validation set to prevent over fitting. Table 1 shows classification results on NORB, following the training of a classifier on pose features and also content features. A disentangled content/pose representation should aid in classification by ensuring the content features are invariant to the pose of the object. Indeed, we see that when the adversarial term is used ($\beta = 0.1$) the content features perform well. Without the term, content features become less effective for

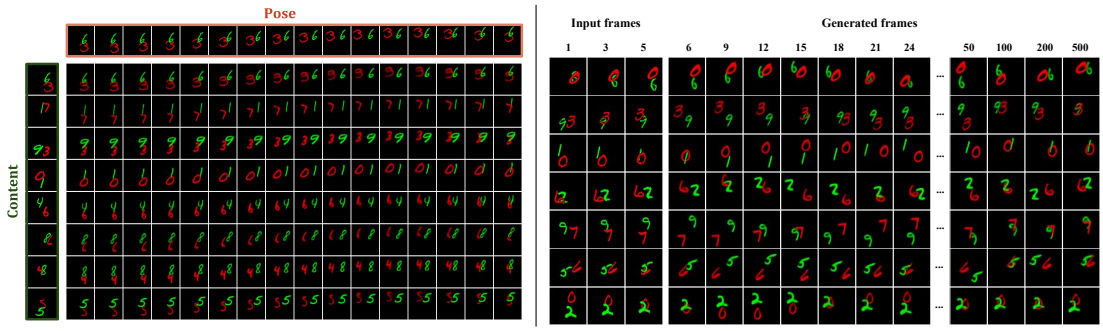

Figure 5.3: Left: Demonstration of content/pose factorization on held out MNIST examples. Each image in the grid is generated using the pose and content vectors h_p and h_c taken from the corresponding images in the top row and first column respectively. The model has clearly learned to disentangle content and pose. Right: Each row shows forward modeling up to 500 time steps into the future, given 5 initial frames. For each generation, note that only the pose part of the representation is being predicted from the previous time step (using an LSTM), with the content vector being fixed from the 5th frame. The generations remain crisp despite the long-range nature of the predictions.

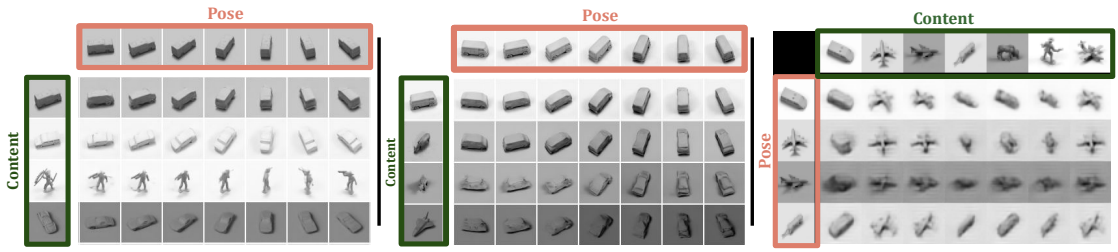

Figure 5.4: Left: Factorization examples using our DRNET model on held out NORB images. Each image in the grid is generated using the pose and content vectors h_p and h_c taken from the corresponding images in the top row and first column respectively. Further examples can be found in the supplemental material. Center: Examples where DRNET was trained without the adversarial loss term. Note how content and pose are no longer factorized cleanly: the pose vector now contains content information which ends up dominating the generation. Right: factorization examples from Mathieu et al. (2016b).

classification.

SUNCG: We use the rendering engine from the SUNCG dataset (Song et al., 2017) to generate sequences where the camera rotates around a range of 3D chair models. DRNET is able to generate high quality examples of this data, as shown in Figure 5.5.

5.3.3 KTH Action Dataset

Finally, we apply DRNET to the KTH dataset (Schuldt et al., 2004). This is a simple dataset of real-world videos of people performing one of six actions (walking, jogging, running, boxing, handwaving, hand-clapping) against fairly uniform backgrounds. In Figure 5.6 we show forward generations of different held out examples, comparing against two baselines: (i) the MCNet of Villegas et al. (2017a) which, at the time of publication, produced the best quality generations of on real-world video and (ii) a baseline auto-encoder LSTM model (AE-LSTM). This is essentially the same as ours, but with a single encoder whose features thus combine content and pose (as opposed to factoring them in DRNET). It is also similar to (Srivastava et al., 2015).

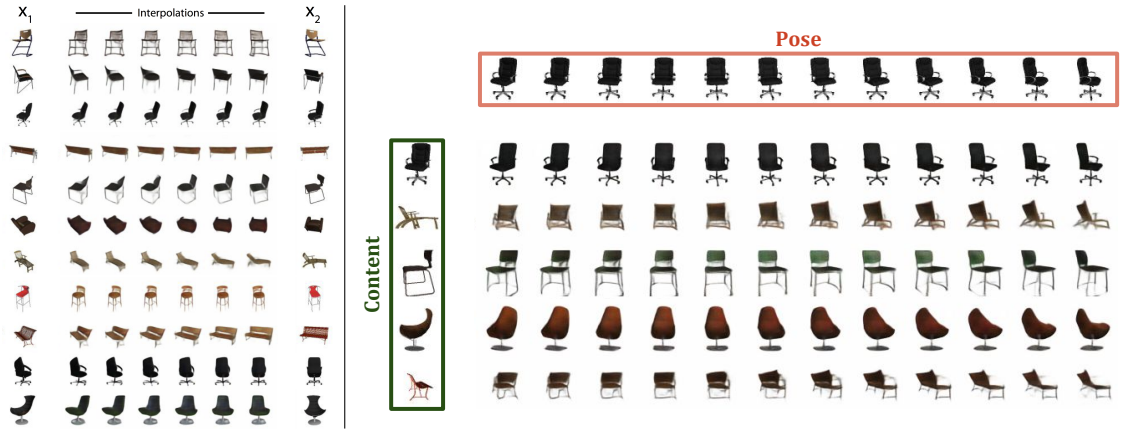

Figure 5.5: Left: Examples of linear interpolation in pose space between the examples x_1 and x_2. Right: Factorization examples on held out images from the SUNCG dataset. Each image in the grid is generated using the pose and content vectors h_p and h_c taken from the corresponding images in the top row and first column respectively. Note how, even for complex objects, the model is able to rotate them accurately.

Figure 5.7 shows more examples, with generations out to 100 time steps. Generations in movie form are viewable at `https://sites.google.com/view/drnet-paper/`. We see artifact emerging frequently in the baseline MCNet (Villegas et al., 2017a), whereas our generations tend to stick to the image manifold. We hypothesize this is due to the fact that our predictions occur in a low dimensional latent space, rather than pixel space, so small prediction errors in the image do not get amplified as generations are fed back into the model.

Evaluating samples from generative models is generally problematic. Pixel-wise measures like PNSR and SSIM (Wang et al., 2004) are appropriate when objects are well aligned, but for long-range generations this is unlikely to be the case. Instead, we adopt an approach based on the Inception score (Salimans et al., 2016) as an alternative to quantify the fidelity of the generations.

The Inception Score of Salimans et al. (2016) utilized a pre-trained Inception network (Szegedy et al., 2016) to evaluate the quality of generative models of images. The

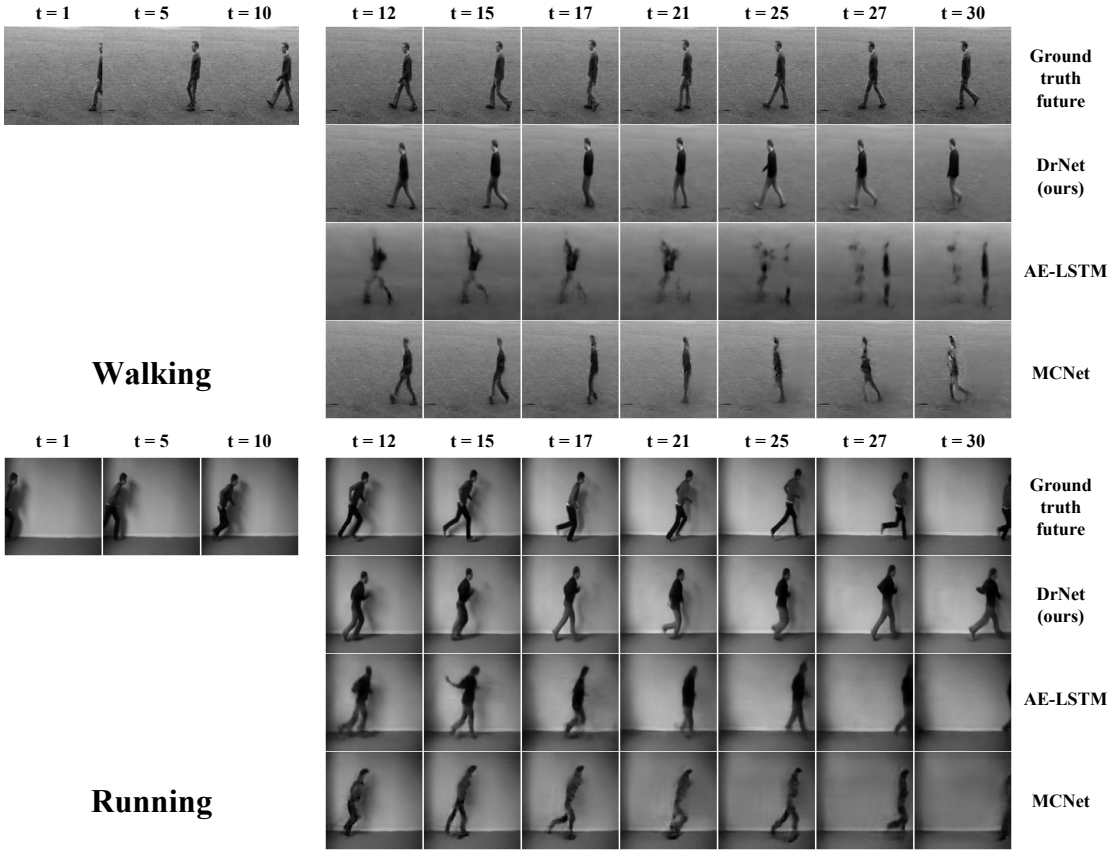

Figure 5.6: Qualitative comparison between our DRNET model, MCNet (Villegas et al., 2017a) and the AE-LSTM baseline. All models are conditioned on the first 10 video frames and generate 20 frames. We display predictions of every 3^{rd} frame. Video sequences are taken from held out examples of the KTH dataset for the classes of walking (top) and running (bottom).

Inception network is a deep convolutional architecture, designed for large scale image classification, that predicts a class label y given input image $\mathbf{x}$. The Inception Score evaluates a generative models by passing a large set of synthesized images through the network and assessing the predicted distribution over labels. A good generative model should produce a highly peaked conditional label distribution $p(y|\mathbf{x})$, i.e. given a particular image the class identity should be certain , and have a marginal distribution equal to $p_{data}(y)$. For KTH, $p_{data}(y)$ is uniform, i.e. all classes should be equally represented

in the samples.

We adopt the basic approach of evaluating generations with a pre-trained discriminative convolutional network. However, in our evaluations we tailor the pre-trained network to our data domain, rather than using a generic off-the-shelf Inception model trained on Imagenet. More specifically, we first train a classifier network to accurately predict the action class of a video from a sequence of 10 frames. We employ a convolutional network classification architecture where the video frames are concatenated as input to the first layer. Once the classifier is trained, we evaluate the samples generated by DRNET and MCNet by considering the label distribution predicted by the classifier for generated sequences. As with the original Inception Score, we expect a good generative model to produce videos with a highly peaked conditional label distribution $p(y|\mathbf{x})$ and a uniform marginal label distribution $p(y)$. Formally, the Inception Score is computed by first sampling N videos form the model, $\mathbf{x}_1, ..., \mathbf{x}_N$. The empirical marginal class distribution is given by

$$\hat{p}(y) = \frac{1}{N} \sum_{i=1}^{N} p(y|\mathbf{x}_i) \tag{5.8}$$

The final score is then given by:

$$IS(\mathbf{x}_1, ..., \mathbf{x}_N) = \exp(\frac{1}{N} \sum_{i=1}^{N} \mathrm{KL}(p(y|\mathbf{x}_i)||\hat{p}(y))) \tag{5.9}$$

Figure 5.9 plots the mean Inception Score for generated sequences from our DRNET model and MCnet (Villegas et al., 2017a). The x-axis indicates the offset, from the final frame of the conditioned input, of the first generated frame used for the Inception Score. The curves show the mean scores of our generations decaying more gracefully than MCNet.

A natural concern with high capacity models is that they might be memorizing the training examples. We probe this in Figure 5.10, where we show the nearest neighbors to our generated frames from the training set. We compute nearest neighbors in feature space using either the pose representation alone, or the combined pose and content representation. The nearest neighbors in pose space show similar body positions but unrelated background, clothing etc. This indicates DRNET has effectively disentangled content and pose. In contrast, the nearest neighbors in combined content and pose space capture the full range of variation. Crucially though, these nearest neighbors are not simple copies of the generations indicating that the model has not memorized the training data.

Figure 5.8 uses the pose representation produced by DRNET to train an action classifier from very few examples. We extract pose vectors from video sequences of length 24 and train a fully connected classifier on these vectors to predict the action class. We compare against an autoencoder baseline, which is the same as ours but with a single encoder whose features thus combine content and pose. We find the factorization significantly boosts performance.

Model		Accuracy (%)
DRNET $_{\beta=0.1}$	h_c	**93.3**
	h_p	60.9
DRNET $_{\beta=0}$	h_c	72.6
	h_p	80.8
Mathieu et al. (2016b)		86.5

Table 5.1: Classification results on NORB dataset, with/without adversarial loss ($\beta = 0.1/0$) using content or pose representations (h_c, h_p respectively). The adversarial term is crucial for forcing semantic information into the content vectors – without it performance drops significantly.

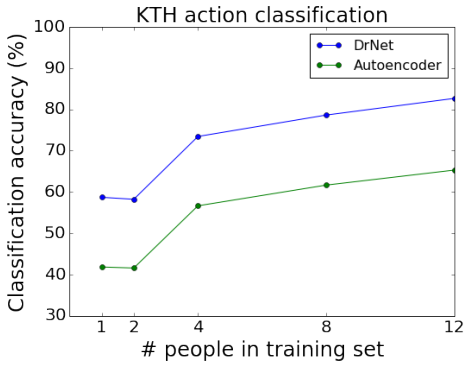

Figure 5.8: Classification of KTH actions from pose vectors with few labeled examples, with autoencoder baseline. N.B. SOA (fully supervised) is 93.9% (Le et al., 2011).

5.4 Discussion

In this chapter we introduced a model based on a pair of encoders that factor video into content and pose. This separation is achieved during training through novel adversarial loss term. The resulting representation is versatile, in particular allowing for stable and coherent long-range prediction through nothing more than a standard LSTM. Our generations compare favorably with leading approaches, despite being a simple model.

The content and pose factorization was motivated by the end goal of video prediction. However, DRNET can be applied to any dataset where group level supervision is available. Specifically, a dataset should be arranged into groups such that items within a group share some common factor(s) of variation. For example, a group may consist of images of the same person with different hairstyle, lighting conditions and facial expressions. Then, the DRNET model can be directly applied to disentangle the shared component amongst the grouped elements (i.e. the content) from the factors that differ within a group (i.e. the pose). Video data provides this group level supervision for free. Since scenes tend to change smoothly over time, short video clips will naturally share many underlying factors of variation. These slow changing elements define the content learned by DRNET . The remaining, fast changing, aspects of the scene define the pose components.

DRNET can also be leveraged to learn features suitable for discriminative tasks. For example, when trained on video data, DRNET learns a content representation that is invariant to natural local transformations. We explored this in a limited setting in our NORB and KTH classification experiments but leave further exploration as a future avenue of research.

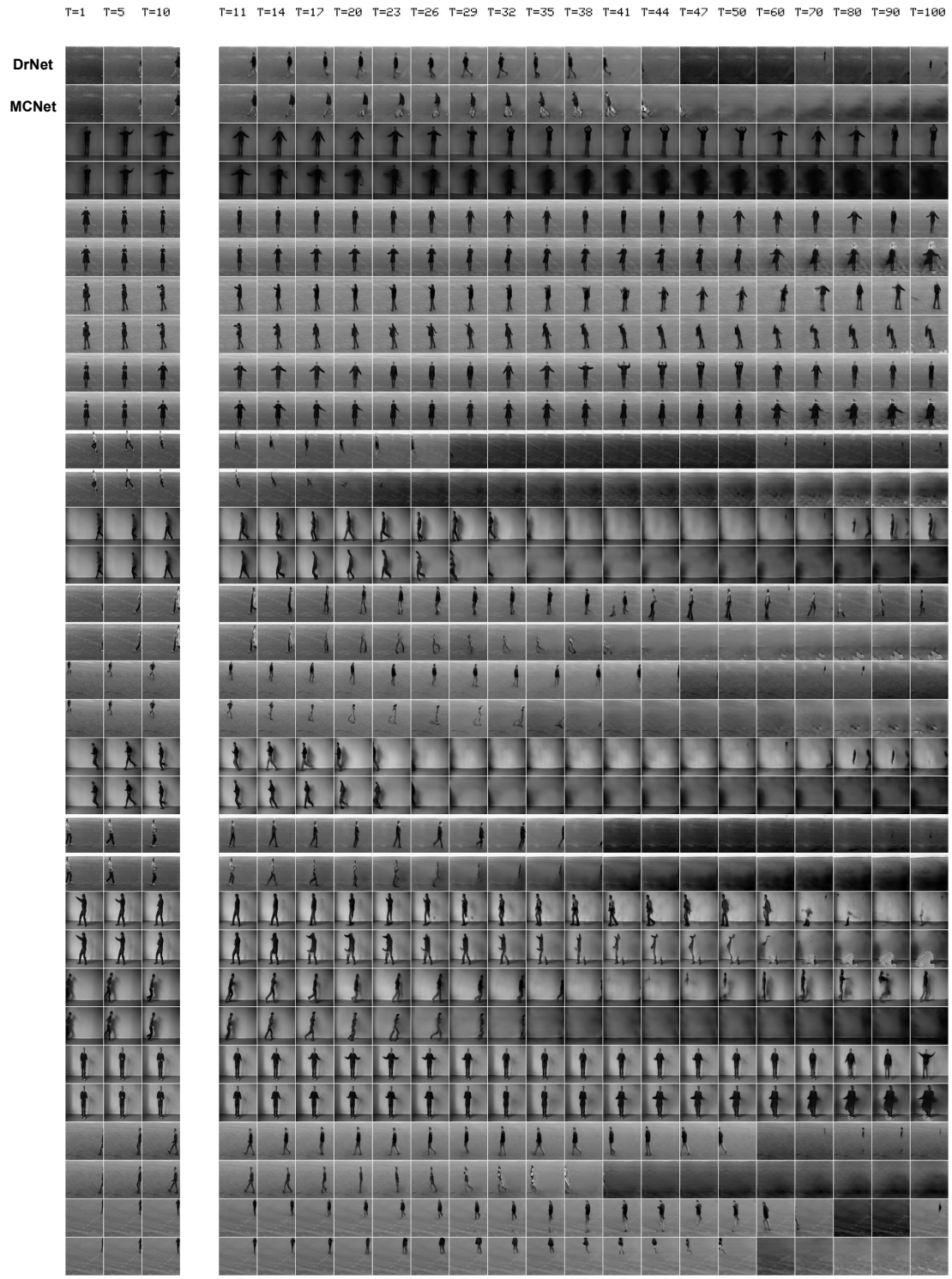

Figure 5.7: Four additional examples of generations on held out examples of the KTH dataset, rolled out to 100 timesteps.

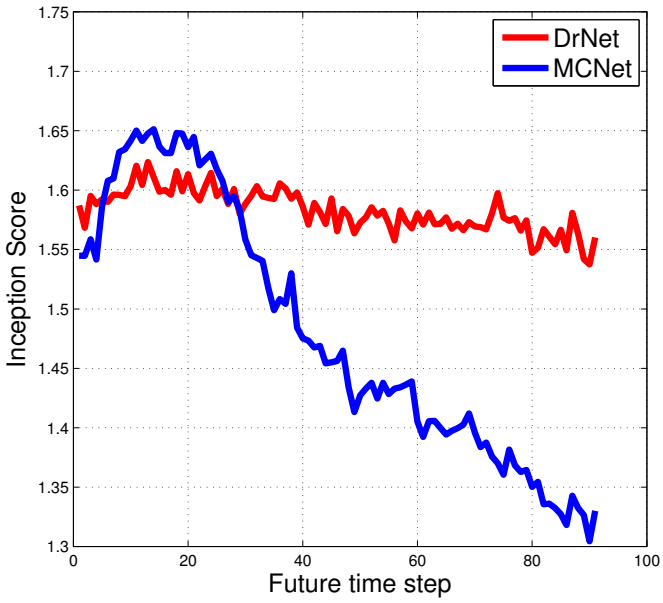

Figure 5.9: Comparison of KTH video generation quality using Inception score. X-axis indicated how far from conditioned input the start of the generated sequence is.

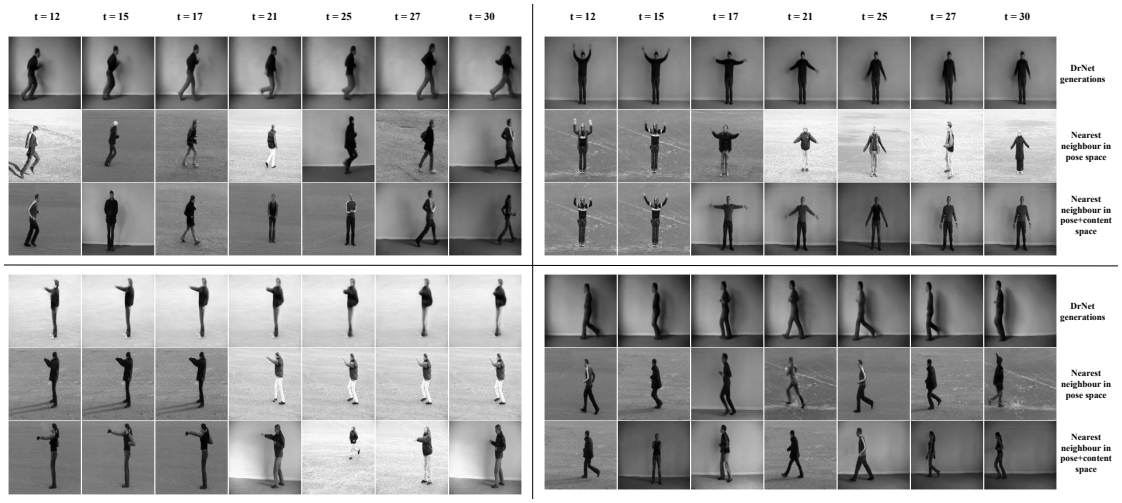

Figure 5.10: For each frame generated by DRNET (top row in each set), we show nearest-neighbor images from the training set, based on pose vectors (middle row) and both content and pose vectors (bottom row). It is evident that our model is not simply copying examples from the training data. Furthermore, the middle row shows that the pose vector generalizes well, and is independent of background and clothing.

Chapter 6

Stochastic Video Generation with a Learned Prior

In the previous chapter we introduced a method of learning a latent representation of images that facilitated the downstream task of video prediction. We demonstrated the utility of the representation by training a simple recurrent neural network to predict future latent vectors given past observations. This method proved highly effective at modeling simple video datasets with *deterministic* motion.

In this chapter we introduce a *stochastic* video generation (SVG) model. The deterministic LSTM network utilized in Chapter 5 produces a single prediction of future frames, given the past. In contrast, the method presented in this chapter predicts a *distribution* over possible future frames. We do so by incorporating stochastic latent variables into a recurrent frame predictor. The latent variables capture non-deterministic elements of a video sequence that the frame predictor alone cannot handle. Once the model is trained, different samples from the latent distribution can be understood as corresponding to different possible futures. We utilize stochastic variational inference techniques

(see Section 2.2 for details) to train the model.

6.1 Introduction

Learning to generate future frames of a video sequence is a challenging research problem with great relevance to reinforcement learning, planning and robotics. Although impressive generative models of still images have been demonstrated (e.g. Karras et al. (2018); Reed et al. (2017b)), these techniques do not extend to video sequences. A key challenge of video prediction is the inherent uncertainty in the dynamics of the world. For example, when a bouncing ball hits the ground unknown effects, such surface imperfections or ball spin, ensure that its future trajectory is inherently random.

Consequently, pixel-level frame predictions of such an event degrade when a deterministic model is trained with MSE, e.g. with the ball itself blurring to accommodate multiple possible futures. Recently, loss functions that impose a distribution instead have been explored. One such approach are adversarial losses (Goodfellow et al., 2014), but training difficulties and mode collapse often mean the full distribution is not captured well.

We propose a new stochastic video generation (SVG) model that combines a deterministic frame predictor with time-dependent stochastic latent variables. We propose two variants of our model: one with a fixed prior over the latent variables (SVG-FP) and another with a learned prior (SVG-LP). The key insight we leverage for the learned-prior model is that for the majority of the ball's trajectory, a deterministic model suffices. Only at the point of contact does the modeling of uncertainty become important. The learned prior can can be interpreted as a *a predictive model of uncertainty*. For most of the trajectory the prior will predict low uncertainty, making the frame estimates deter-

ministic. However, at the instant the ball hits the ground it will predict a high variance event, causing frame samples to differ significantly.

We train our model by introducing a recurrent inference network to estimate the latent distribution for each time step. This novel recurrent inference architecture facilitates end-to-end training of SVG-FP and SVG-LP. We evaluate SVG-FP and SVG-LP on two real world datasets and a stochastic variant of the Moving MNIST dataset. Sample generations are both varied and sharp, even many frames into the future.

6.2 Approach

Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ denote a video sequence composed of T frames. Our task is to predict frames $\mathbf{x}_{c:T}$ given past frames $\mathbf{x}_{1:c-1}$.

We start by explaining how our model generates new video frames, before detailing the training procedure. Our model has two distinct components: (i) a prediction model p_θ that generates the next frame $\hat{\mathbf{x}}_t$, based on previous ones in the sequence $\mathbf{x}_{1:t-1}$ and a latent variable $\mathbf{z}_t$ and (ii) a prior distribution $p(\mathbf{z})$ from which $\mathbf{z}_t$ is sampled at *at each time step* . The prior distribution can be fixed (SVG-FP) or learned (SVG-LP). Intuitively, the latent variable $\mathbf{z}_t$ carries all the stochastic information about the next frame that the deterministic prediction model cannot capture. After conditioning on real frames $x_{1:c-1}$, the model can generate multiple frames into the future by passing generated frames back into the input of the prediction model and, in the case of the SVG-LP model, the prior also.

The model is trained with the aid of a separate inference model (not used a test time). This takes as input the frame $\mathbf{x}_t$, i.e. the target of the prediction model, and previous frames $\mathbf{x}_{1:t-1}$. From this it computes a distribution $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$ from which we

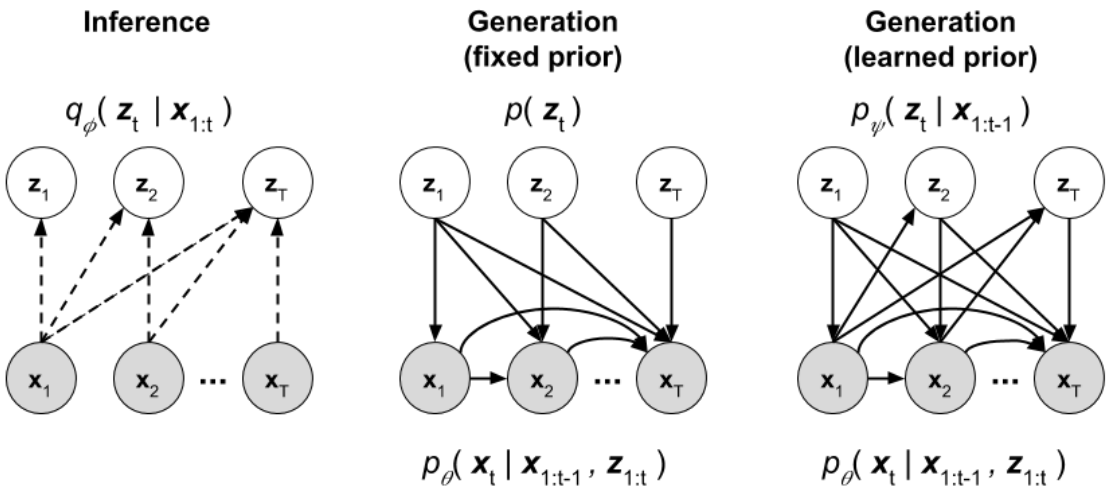

Figure 6.1: Inference (left) and generation in the SVG-FP (middle) and SVG-LP models (right).

sample $\mathbf{z}_t$. To prevent $\mathbf{z}_t$ just copying $\mathbf{x}_t$, we force $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$ to be close to the prior distribution $p(\mathbf{z})$ using a KL-divergence term. This constrains the information that $\mathbf{z}_t$ can carry, forcing it to capture new information not present in previous frames. A second term in the loss penalizes the reconstruction error between $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t$. Figure 6.1 shows the graphical model defined by this set-up. Figure 6.1a shows the inference procedure for both SVG-FP and SVG-LP. The generation procedure for SVG-FP and SVG-LP are shown in Figure 6.1b and Figure 6.1c respectively.

To further explain our model we adopt the formalism of variational auto-encoders. Our recurrent frame predictor $p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ is specified by a fixed-variance conditional Gaussian distribution $\mathcal{N}(\mu_\theta(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}), \sigma^2)$. In practice, we set $\hat{\mathbf{x}}_t = \mu_\theta(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$, i.e. the mean of the distribution, rather than sampling. Note that at time step t the frame predictor only receives $\mathbf{x}_{t-1}$ and $\mathbf{z}_t$ as input. The dependencies on all previous $\mathbf{x}_{1:t-2}$ and $\mathbf{z}_{1:t-1}$ stem from the recurrent nature of the model.

Since the true posterior distribution over latent variables $\mathbf{z}_t$ is intractable, we rely on a time-dependent inference network $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$ that approximates it with a conditional

Gaussian distribution $\mathcal{N}(\mu_\phi(\mathbf{x}_{1:t}), \sigma_\phi(\mathbf{x}_{1:t}))$.

The model is trained by maximizing a variant of the variational lower bound which was presented in Section 2.2. We first review the variational lower bound, as it applies to our sequential data:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}_{c:T}; \mathbf{x}_{1:c-1}) = \mathbb{E}_{q_\phi(\mathbf{z}_{c:T}|\mathbf{x})} \log p_\theta(\mathbf{x}_{c:T}|\mathbf{x}_{1:c-1}, \mathbf{z}_{c:T}) - D_{KL}(q_\phi(\mathbf{z}_{c:T}|\mathbf{x})||p(\mathbf{z}_{c:T})) \tag{6.1}$$

Here, $x_{1:c-1}$ denote the context frames upon which future frame generation is conditioned.

Analogous to a VAE, the first term maximizes the log-likelihood of a sequence of frames $\mathbf{x}_c, \ldots, \mathbf{x}_T$ given the inferred sequence of latent variables $\mathbf{z}_c, \ldots, \mathbf{z}_T$ and past frames $\mathbf{x}_1, \ldots, \mathbf{x}_{c-1}$. The second term minimizes the KL divergence between the approximate posterior $q_\phi(\mathbf{z}_{c:T}|\mathbf{x})$ and the prior $p(\mathbf{z}_{c:T})$.

Both terms can be simplified and decomposed across time steps. Recall that the SVG frame predictor is parameterized by a recurrent neural network. At each time step the model takes as input $\mathbf{x}_{t-1}$ and $\mathbf{z}_t$ and through the recurrence the model also depends on $\mathbf{x}_{1:t-2}$ and $\mathbf{z}_{1:t-1}$. Due to the recurrent nature of our model, the likelihood at time t does not depend on future latent variables $\mathbf{z}_{t+1:T}$. Then, we can further simplify the bound with:

$$\begin{aligned}
\log p_\theta(\mathbf{x}_{c:T}|\mathbf{x}_{1:c-1}, \mathbf{z}) &= \log \prod_{t=c}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:T}) \\
&= \sum_{t=c}^{T} \log p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \cancel{\mathbf{z}_{t+1:T}}) \\
&= \sum_{t=c}^{T} \log p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})
\end{aligned} \tag{6.2}$$

The inference network used by SVG-FP and SVG-LP is also parameterized by a recurrent neural network that outputs a different distribution $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$ for every time step t. Due to the independence across time and LSTM structure of q_ϕ, we have

$$\begin{aligned} q_\phi(\mathbf{z}_{c:T}|\mathbf{x}) &= \prod_{t=c}^{T} q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}, \cancel{\mathbf{x}_{t+1:T}}) \\ &= \prod_{t=c}^{T} q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}) \end{aligned}$$

Here, we note that the approximate posterior at time t does not depend on future frames $x_{t+1:T}$ due to the recurrent nature of the model. The independence of $\mathbf{z}_c, \ldots, \mathbf{z}_T$ allows the D_{KL} term of the loss to be decomposed into individual time steps:

$$\begin{aligned} &D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &= \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} d\mathbf{z} \\ &= \int_{\mathbf{z}_c} \cdots \int_{\mathbf{z}_T} q_\phi(\mathbf{z}_c|\mathbf{x}_{1:c}) \cdots q_\phi(\mathbf{z}_T|\mathbf{x}_{1:T}) \log \frac{q_\phi(\mathbf{z}_c|\mathbf{x}_{1:c}) \cdots q_\phi(\mathbf{z}_T|\mathbf{x}_{1:T})}{p(\mathbf{z}_c) \cdots p(\mathbf{z}_T)} d\mathbf{z}_c \cdots d\mathbf{z}_T \\ &= \int_{\mathbf{z}_c} \cdots \int_{\mathbf{z}_T} q_\phi(\mathbf{z}_c|\mathbf{x}_{1:c}) \cdots q_\phi(\mathbf{z}_T|\mathbf{x}_{1:T}) \sum_{t=c}^{T} \log \frac{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})}{p(\mathbf{z}_t)} d\mathbf{z}_c \cdots d\mathbf{z}_T \\ &= \sum_{t=c}^{T} \int_{\mathbf{z}_c} \cdots \int_{\mathbf{z}_T} q_\phi(\mathbf{z}_c|\mathbf{x}_{1:c}) \cdots q_\phi(\mathbf{z}_T|\mathbf{x}_{1:T}) \log \frac{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})}{p(\mathbf{z}_t)} d\mathbf{z}_c \cdots d\mathbf{z}_T \end{aligned}$$

And because $\int_x p(x)dx = 1$ this simplifies to:

$$\begin{aligned} &= \sum_{t=c}^{T} \int_{\mathbf{z}_t} q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}) \log \frac{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})}{p(\mathbf{z}_t)} d\mathbf{z}_t \\ &= \sum_{t=c}^{T} D_{KL}(q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})||p(\mathbf{z}_t)) \end{aligned} \tag{6.3}$$

Equation 6.1 can now be decomposed across time using using the simplified likeli-

hood and KL divergence of Equation 6.2 and Equation 6.3 respectively:

$$\begin{aligned}\log p_\theta(\mathbf{x}_{c:T}|\mathbf{x}_{1:c-1}) &\geq \mathcal{L}_{\theta,\phi}(\mathbf{x}_{1:T}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &= \sum_t \big[\mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})} \log p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \\ &\qquad - D_{KL}(q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})||p(\mathbf{z}_t))\big] \end{aligned} \tag{6.4}$$

Our final objective function is a re-weighted version of Equation 6.4:

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(\mathbf{x}_{c:T}; \mathbf{x}_{1:c-1}, \beta) = \sum_{t=c}^{T} \big[&\mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})} \log p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \\ &- \beta D_{KL}(q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})||p(\mathbf{z}_t))\big]\end{aligned} \tag{6.5}$$

Given the form of p_θ the likelihood term reduces to an ℓ_2 penalty between $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t$. We train the model using the re-parameterization trick (Kingma and Welling, 2014) and by estimating the expectation over $q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})$ with a single sample.

The hyper-parameter β represents the trade-off between minimizing frame prediction error and fitting the prior. A smaller β increases the capacity of the inference network. If β is too small the inference network may learn to simply copy the target frame $\mathbf{x}_t$, resulting in low prediction error during training. However, test time performance, i.e. when samples are drawn from the prior, due to the mismatch between the posterior $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$ and the prior $p(\mathbf{z}_t)$. If β is too large, the model may under-utilize or completely ignore latent variables $\mathbf{z}_t$ and reduce to a deterministic predictor. In practice, we found β easy to tune, particularly for the learned-prior variant we discuss below. For a discussion of hyperparameter β in the context of VAEs see Higgins et al. (2017).

Fixed prior: The simplest choice for $p(\mathbf{z}_t)$ is a fixed Gaussian $\mathcal{N}(0, \mathbf{I})$, as is typ-

ically used in variational autoencoder models. We refer to this as the SVG-FP model, as shown in Figure 6.2a. A drawback is that samples at each time step will be drawn randomly, thus ignore temporal dependencies present between frames.

Learned prior: A more sophisticated approach is to learn a prior that varies across time, being a function of all past frames up to *but not including* the frame being predicted $p_\psi(\mathbf{z}_t|\mathbf{x}_{1:t-1})$. Specifically, at time t a prior network observes frames $\mathbf{x}_{1:t-1}$ and outputs the parameters of a conditional Gaussian distribution $\mathcal{N}(\mu_\psi(\mathbf{x}_{1:t-1}), \sigma_\psi(\mathbf{x}_{1:t-1}))$. The prior network is trained jointly with the rest of the model by maximizing:

$$\begin{aligned}\mathcal{L}_{\theta,\phi,\psi}(\mathbf{x}_{1:T}) = \sum_{t=c}^{T} \big[&\mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})} \log p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \\ &- \beta D_{KL}(q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})||p_\psi(\mathbf{z}_t|\mathbf{x}_{1:t-1}))\big]\end{aligned} \tag{6.6}$$

We refer to this model as SVG-LP and illustrate the training procedure in Figure 6.2b.

At test time, a frame at time t is generated by first sampling $\mathbf{z}_t$ from the prior. In SVG-FP we draw $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$ and in SVG-LP we draw $\mathbf{z}_t \sim p_\psi(\mathbf{z}_t|\mathbf{x}_{1:t-1})$. Then, a frame is generated by $\hat{\mathbf{x}}_t = \mu_\theta(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$. After conditioning on a short series of real frames, the model begins to pass generated frames $\hat{\mathbf{x}}_t$ back into the input of the prediction model and, in the case of the SVG-LP model, the prior. The sampling procedure for SVG-LP is illustrated in Figure 6.2c.

Architectures: We use a generic convolutional LSTM for p_θ, q_ϕ and p_ψ. Frames are input to the LSTMs via a feed-forward convolutional network, shared across all three parts of the model. A convolutional frame decoder maps the output of the frame predictor's recurrent network back to pixel space.

For a time step t during training, the generation is as follows, where the LSTM

recurrence is omitted for brevity:

$$\begin{aligned}
\mu_\phi(t), \sigma_\phi(t) &= LSTM_\phi(h_t) \,, & h_t &= Enc(\mathbf{x}_t) \,, \\
\mathbf{z}_t &\sim \mathcal{N}(\mu_\phi(t), \sigma_\phi(t)) \,, \\
g_t &= LSTM_\theta(h_{t-1}, \mathbf{z}_t) \,, & h_{t-1} &= Enc(\mathbf{x}_{t-1}) \,, \\
\mu_\theta(t) &= Dec(g_t) \,.
\end{aligned}$$

During training, the parameters of the encoder Enc and decoder Dec are also learned, along with the rest of the model, in an end-to-end fashion (we omit their parameters from the loss functions above for brevity).

In the learned-prior model (SVG-LP), the parameters of the prior distribution at time t are generated as follows, where the LSTM recurrence is omitted for brevity:

$$\begin{aligned}
h_{t-1} &= Enc(\mathbf{x}_{t-1}) \,, \\
\mu_\psi(t), \sigma_\psi(t) &= LSTM_\psi(h_{t-1}) \,, .
\end{aligned}$$

6.2.1 Discussion of related models

Stochastic temporal models have also been explored outside the domain of video generation. Bayer and Osendorfer (2014) introduce stochastic latent variables into a recurrent network in order to model music and motion capture data. This method utilizes a recurrent inference network similar to our approach and the same time-independent Gaussian prior as our fixed-prior model. Several additional works train stochastic recurrent neural networks to model speech, handwriting, natural language (Bowman et al.,

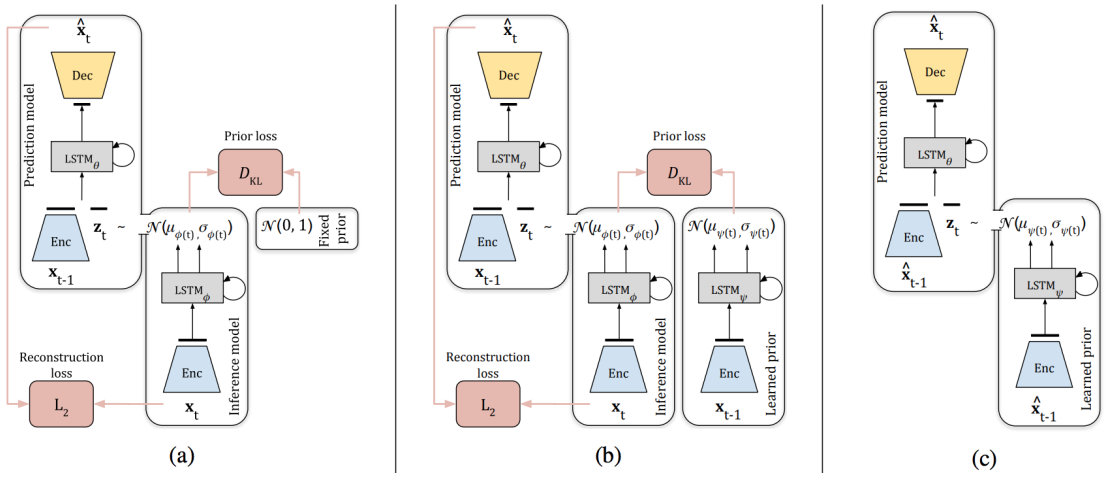

Figure 6.2: Our proposed video generation model. (a) Training with a fixed prior (SVG-FP); (b) Training with learned prior (SVG-LP); (c) Generation with the learned prior model. The red boxes show the loss functions used during training. See text for details.

2016; Chung et al., 2015; Fraccaro et al., 2016), perform counterfactual inference (Krishnan et al., 2015) and anomaly detection (Sölch et al., 2016). As in our work, these methods all optimize a bound on the data likelihood using an approximate inference network. They differ primarily in the parameterization of the approximate posterior and the choice of prior model.

Our model is related to a recent stochastic variational video prediction model of Babaeizadeh et al. (2018). Although their variational framework is broadly similar, a key difference between this work and ours is the way in which the latent variables $\mathbf{z}_t$ are estimated during training and sampled at test time.

The inference network of Babaeizadeh et al. (2018) encodes the *entire video sequence* via a feed forward convolutional network to estimate $q_\theta(\mathbf{z}|\mathbf{x}_{1:T})$. They propose two different models that use this distribution. In the time-invariant version, a single $\mathbf{z}$ is sampled for the entire video sequence. In the time-variant model, a different $\mathbf{z}_t \sim q_\theta(\mathbf{z}|\mathbf{x}_{1:T})$ is sampled for every time step, all samples coming from the *same distribution*.

In contrast, both our fixed-prior and learned-prior models utilize a more flexible inference network that outputs a different posterior distribution for every time step given by $q_\theta(\mathbf{z}_t|\mathbf{x}_{1:t})$ (note $\mathbf{x}_{1:t}$, not $\mathbf{x}_{1:T}$ as above).

At test time, our fixed-prior model and the time-variant model of Babaeizadeh et al. (2018) sample $\mathbf{z}_t$ from a fixed Gaussian prior at every time step. By contrast, our learned-prior model draws samples from the time-varying distribution: $p_\psi(\mathbf{z}_t|\mathbf{x}_{1:t-1})$, whose parameters ψ were estimated during training.

These differences manifest themselves in two ways. First, the generated frames are significantly sharper with both our models (see direct comparisons to Babaeizadeh et al. (2018) in Figure 6.11). Second, training our model is much easier. Despite the same *prior* distribution being used for both our fixed-prior model and Babaeizadeh et al. (2018), the time variant *posterior* distribution introduced in our model appears crucial for successfully training the model. Indeed, Babaeizadeh et al. (2018) report difficulties training their model by naively optimizing the variational lower bound, noting that the model simply ignores the latent variables. Instead, they propose a scheduled three phase training procedure whereby first the deterministic element of the model is trained, then latent variables are introduced but the KL loss is turned off and in the final stage the model is trained with the full loss. In contrast, both our fixed-prior and learned-prior models are easily trainable end-to-end in a single phase using a unified loss function.

6.3 Experiments

We evaluate our SVG-FP and SVG-LP model on one synthetic video dataset (Stochastic Moving MNIST) and two real ones (KTH actions (Schuldt et al., 2004) and BAIR robot (Ebert et al., 2017)). We show quantitative comparisons by computing structural

similarity (SSIM) and Peak Signal-to-Noise Ratio (PSNR) scores between ground truth and generated video sequences. Since neither of these metrics fully captures perceptual fidelity of generated sequences we also make a qualitative comparison between samples from our model and current state-of-the-art methods. We encourage the reader to view additional generated videos at: `https://sites.google.com/view/svglp/`.

6.3.1 Model architectures

$LSTM_\theta$ is a two layer LSTMs with 256 cells in each layer. $LSTM_\phi$ and $LSTM_\psi$ are both single layer LSTMs with 256 cells in each layer. Each network has a linear embedding layer and a fully connected output layer. The output of $LSTM_\theta$ is passed through a tanh nonlinearity before going into the frame decoder.

For Stochastic Moving MNIST, the frame encoder has a DCGAN discriminator architecture (Radford et al., 2016) with output dimensionality $|h|$ = 128. Similarly, the decoder uses a DCGAN generator architecture and a sigmoid output layer. The output dimensionalities of the LSTM networks are $|g| = 128, |\mu_\phi| = |\mu_\psi| = 10$.

For KTH and BAIR datasets, the frame encoder uses the same architecture as VGG16 (Simonyan and Zisserman, 2015) up until the final pooling layer with output dimensionality $|h| = 128$. The decoder is a mirrored version of the encoder with pooling layers replaced with spatial up-sampling and a sigmoid output layer. The output dimensionalities of the LSTM networks are $|g| = 128, |\mu_\phi| = |\mu_\psi| = 32$ for KTH and $|g| = 128, |\mu_\phi| = |\mu_\psi| = 64$ for BAIR.

For all datasets we add skip connections from the encoder at the last ground truth frame to the decoder at t, enabling the model to easily generate static background features.

We also train a deterministic baseline with the same encoder, decoder and LSTM

architecture as our frame predictor p_θ but with the latent variables omitted.

We train all the models with the ADAM optimizer (Kingma and Ba, 2014) and learning rate $\eta = 0.002$. We set $\beta =$ 1e-4 for KTH and BAIR and $\beta =$ 1e-6 for KTH. Source code and trained models are available at `https://github.com/edenton/svg`.

6.3.2 Stochastic Moving MNIST

We introduce the Stochastic Moving MNIST (SM-MNIST) dataset which consists of sequences of frames of size 64×64, containing one or two MNIST digits moving and bouncing off edge of the frame (walls). In the original Moving MNIST dataset (Srivastava et al., 2015) the digits move with constant velocity and bounce off the walls in a deterministic manner. By contrast, SM-MNIST digits move with a constant velocity along a trajectory until they hit at wall at which point they bounce off with a random speed and direction. This dataset thus contains segments of deterministic motion interspersed with moments of uncertainty, i.e. each time a digit hits a wall.

Training sequences were generated on the fly by sampling two different MNIST digits from the training set (60k total digits) and two distinct trajectories. Trajectories were constructed by uniformly sampling (x, y) starting locations and initial velocity vectors $(\Delta x, \Delta y) \in [-4, 4] \times [-4, 4]$. Every time a digit hits a wall a new velocity vector is sampled.

We trained our SVG models and a deterministic baseline on SM-MNIST by conditioning on 5 frames and training the model to predict the next 10 frames in the sequence. We compute SSIM for SVG-FP and SVG-LP by drawing 100 samples from the model for each test sequence and picking the one with the best score with respect to the ground truth. Figure 6.8(left) plots average SSIM on unseen test videos. Both SVG-FP and SVG-LP outperform the deterministic baseline and SVG-LP performs best overall, par-

ticularly in later time steps. Figure 6.3 shows sample generations from the deterministic model and SVG-LP. Generations from the deterministic model are sharp for several time steps, but the model rapidly degrades after a digit collides with the wall, since the subsequent trajectory is uncertain.

We hypothesize that the improvement of SVG-LP over the SVG-FP model is due to the mix of deterministic and stochastic movement in the dataset. In SVG-FP, the frame predictor must determine how and if the latent variables for a given time step should be integrated into the prediction. In SVG-LP , the burden of predicting points of high uncertainty can be offloaded to the prior network.

Empirically, we measure this in Figure 6.4. Five hundred different video sequences were constructed, each with different test digits, but whose trajectories were synchronized. The plot shows the mean of $\sigma_\psi(\mathbf{x}_{1:t})$, i.e., the variance of the distribution over $\mathbf{z}_t$ predicted by the learned prior over 100 time steps. Superimposed in red and blue are the time instants when the the respective digits hit a wall. We see that the learned prior is able to accurately predict these collisions that result in significant randomness in the trajectory.

One major challenge when evaluating generative video models is assessing how accurately they capture the full distribution of possible outcomes, mainly due to the high

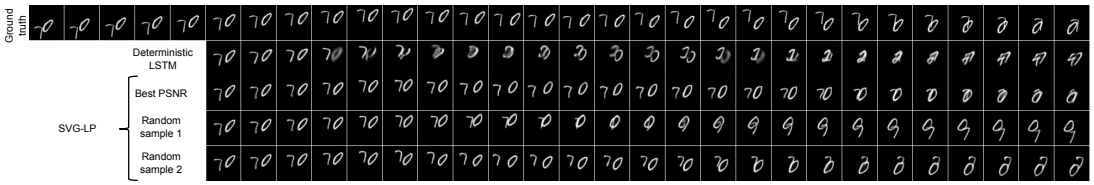

Figure 6.3: Qualitative comparison between SVG-LP and a purely deterministic baseline. The deterministic model produces sharp predictions until ones of the digits collides with a wall, at which point the prediction blurs to account for the many possible futures. In contrast, samples from SVG-LP show the digit bouncing off in different plausible directions.

dimensionality of the space in which samples are drawn. However, the synthetic nature of single digit SM-MNIST allows us to investigate this in a principled way. A key point to note is that with each sequence, the digit appearance remains constant with the only randomness coming from its trajectory once it hits the image boundary. Thus for a sequence generated from our model, we can establish the digit trajectory by taking a pair of frames at any time step and cross-correlating them with the digit used in the initial conditioning frames. Maxima in each frame reveal the location of the digit, and the difference between the two gives us the velocity vector at that time. By taking an expectation over many samples from our model (also using the same trajectory but different digits), we can compute the empirical distribution of trajectories produced by our model. We can then perform the same operation on a validation set of ground truth sequences, to produce the true distribution of digit trajectories and compare it to the one produced by our model.

Figure 6.5 shows SVG-LP (trained on *single* digit SM-MNIST) accurately capturing the distribution of MNIST digit trajectories for many time steps. The digit trajectory is

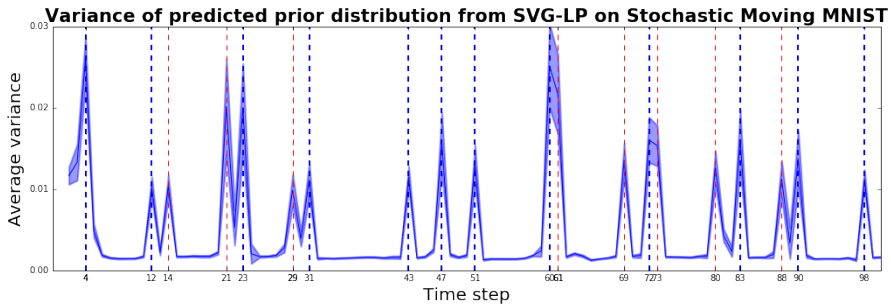

Figure 6.4: Learned prior of SVG-LP accurately predicts collision points in SM-MNIST. Five hundred test video sequences with different MNIST test digits but synchronized motion were fed into the learned prior. The mean ($\pm$ one standard deviation) of $\sigma_\psi(\mathbf{x}_{1:t-1})$ is plotted for $t = 1, ..., 100$. The true points of uncertainty in the video sequences, i.e. when a digits hits a wall, are marked by vertical lines, colored red and blue for each digit respectively.

deterministic before a collision. This is accurately reflected by the highly peaked distribution of velocity vectors from SVG-LP in the time steps leading up to a collision. Following a collision, the distribution broadens to approximately uniform before being reshaped by subsequent collisions. Crucially, SVG-LP accurately captures this complex behavior for many time steps. The temporally varying nature of the true trajectory distributions further supports the need for a learned prior $p_\psi(\mathbf{z}_t|\mathbf{x}_{1:t-1})$.

We also ran this experiment on a more challenging, non-uniform distribution of digit trajectories. Figure 6.6 plots the distribution of Δx and Δy from which velocity vectors are initially sampled at the start of a video sequence. All subsequent velocity vectors are sampled from a modified variant of this distribution where invalid directions are given zero probability and the remaining probabilities are re-normalized. Note that depending which wall the digit hits, a different subset of velocity vectors will be valid (e.g. if the digit hits the right wall, $\Delta x > 0$ would be invalid) and so the distribution is dependent on the precise location the digits hits the wall.

We trained SVG-LP on this non-uniform SM-MNIST dataset and assessed the model's ability to capture the digit trajectory using the same technique described above. Figure 6.7 shows SVG-LP accurately capturing the distribution of MNIST digit trajectories for many time steps. The digit trajectory is deterministic before a collision. This is accurately reflected by the highly peaked distribution of velocity vectors from SVG-LP in the time steps leading up to a collision. Following a collision, the distribution broadens and effectively captures the complex trajectory distribution for many time steps.

6.3.3 KTH Action Dataset

The KTH Action dataset (Schuldt et al., 2004) consists of real-world videos of people performing one of six actions (walking, jogging, running, boxing, handwaving,

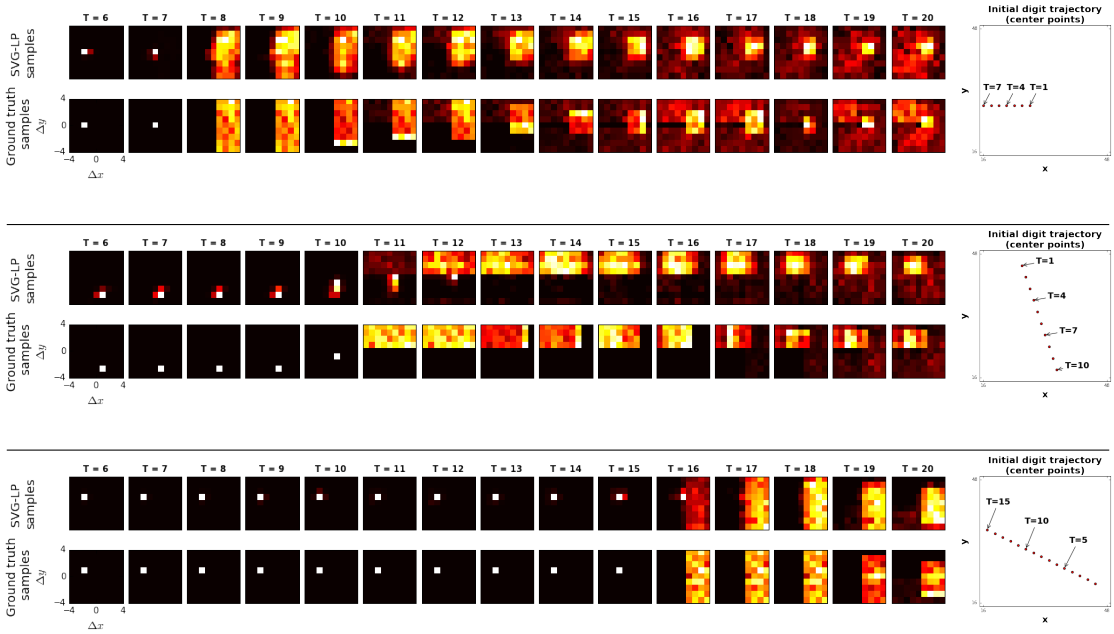

Figure 6.5: Three examples of our SVG-LP model accurately capturing the distribution of MNIST digit trajectories following collision with a wall. On the right we show the trajectory of a digit prior to the collision. In the ground truth sequence, the angle and speed immediately after impact are drawn from at random from uniform distributions. Each of the sub-plots shows the *distribution* of $\Delta x, \Delta y$ at each time step. In the lower ground truth sequence, the trajectory is deterministic before the collision (occurring between $t = 7$ and $t = 8$ in the first example), corresponding to a delta-function. Following the collision, the distribution broadens out to an approximate uniform distribution (e.g. $t = 8$), before being reshaped by subsequent collisions. The upper row shows the distribution estimated by our SVG-LP model (after conditioning on ground-truth frames from $t = 1 \ldots 5$). Note how our model accurately captures the correct distribution many time steps into the future, despite its complex shape. The distribution was computed by drawing many samples from the model, as well as averaging over different digits sharing the same trajectory. The 2nd and 3rd examples show different trajectories with correspondingly different impact times ($t = 11$ and $t = 16$ respectively).

hand-clapping) against fairly uniform backgrounds. The human motion in the video sequences is fairly regular, however there is still uncertainty regarding the precise locations of the person's joints at subsequent time steps. We trained SVG-FP, SVG-LP and the deterministic baseline on 64×64 video sequences by conditioning on 10 frames and training the model to predict the next 10 frames in the sequence.

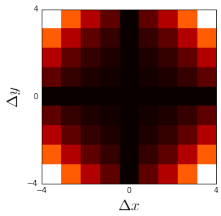

Figure 6.6: Initial distribution of Δx / Δy in non-uniform experiments.

We compute SSIM for SVG-FP and SVG-LP by drawing 100 samples from the model for each test sequence and picking the one with the best score with respect to the ground truth. Figure 6.8(right) plots average SSIM on unseen test videos. SVG-FP and SVG-LP perform comparably on this dataset and both outperform the deterministic baseline. Figure 6.10 shows generations from the deterministic baseline and SVG-FP. The deterministic model predicts plausible future frames but, due to the inherent uncertainty in precise limb locations, often deviates from the ground truth. In contrast, different samples from the stochastic model reflect the variability in future frames indicating the latent variables are being utilized even on this simple dataset.

6.3.4 BAIR robot pushing dataset

The BAIR robot pushing dataset (Ebert et al., 2017) contains videos of a Sawyer robotic arm pushing a variety of objects around a table top. The movements of the arm are highly stochastic, providing a good test for our model. Although the dataset does contain actions given to the arm, we discard them during training and make frame predictions based solely on the video input.

Following Babaeizadeh et al. (2018), we train SVG-FP, SVG-LP and the deterministic baseline by conditioning on the first two frames of a sequence and predicting the subsequent 10 frames. We compute SSIM for SVG-FP and SVG-LP by drawing 100

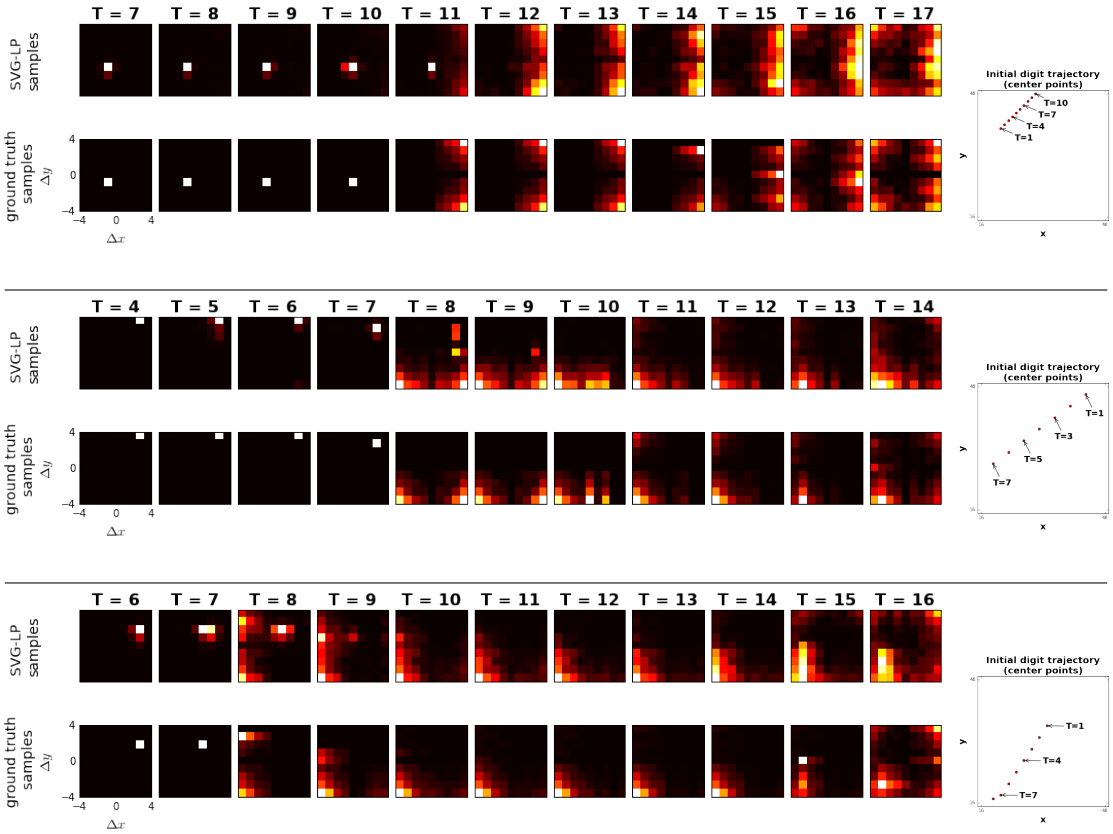

Figure 6.7: Four examples of our SVG-LP model accurately capturing the distribution of MNIST digit trajectories following collision with a wall. Digit trajectory velocity vectors are sampled from a *non-uniform* distribution with higher probability given to greater speeds. On the right we show the trajectory of a digit prior to the collision. Each of the sub-plots shows the *distribution* of $\Delta x, \Delta y$ at each time step. In the lower ground truth sequence, the trajectory is deterministic before the collision (occurring between $t = 8$ and $t = 9$ in the first example), corresponding to a delta-function. Following the collision, the distribution broadens out and is eventually reshaped by subsequent collisions. The upper row shows the distribution estimated by our SVG-LP model (after conditioning on ground-truth frames from $t = 1 \ldots 5$). Note how our model accurately captures the correct distribution many time steps into the future, despite its complex shape. The distribution was computed by drawing many samples from the model, as well as averaging over different digits sharing the same trajectory. The remaining examples show different trajectories with correspondingly different impact times

samples from the model for each test sequence and picking the one with the best score with respect to the ground truth. Figure 6.9 plots average SSIM and PSNR scores on 256

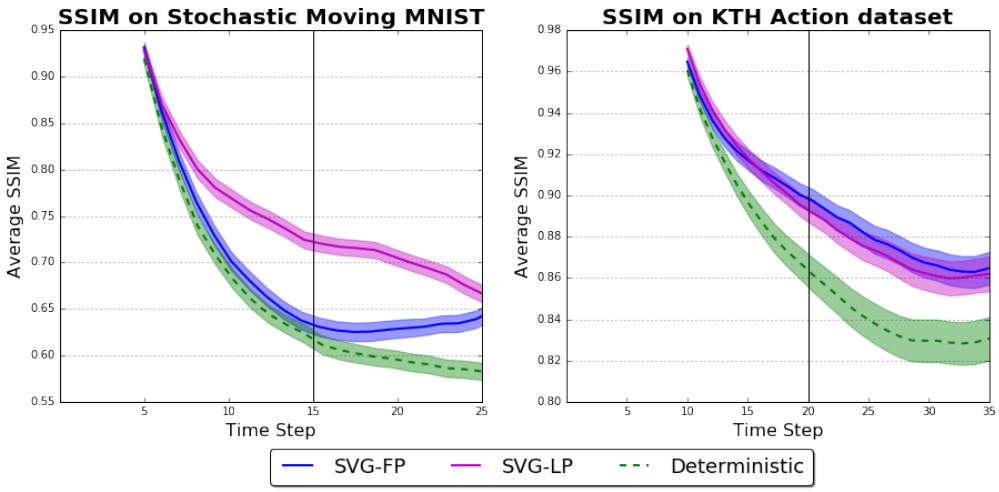

Figure 6.8: Quantitative evaluation of SVG-FP and SVG-LP video generation quality on SM-MNIST (left) and KTH (right). The models are conditioned on the first 5 frames for SM-MNIST and 10 frames for KTH. The vertical bar indicates the frame number the models were trained to predict up to; further generations indicate generalization ability. Mean SSIM over test videos is plotted with 95% confidence interval shaded.

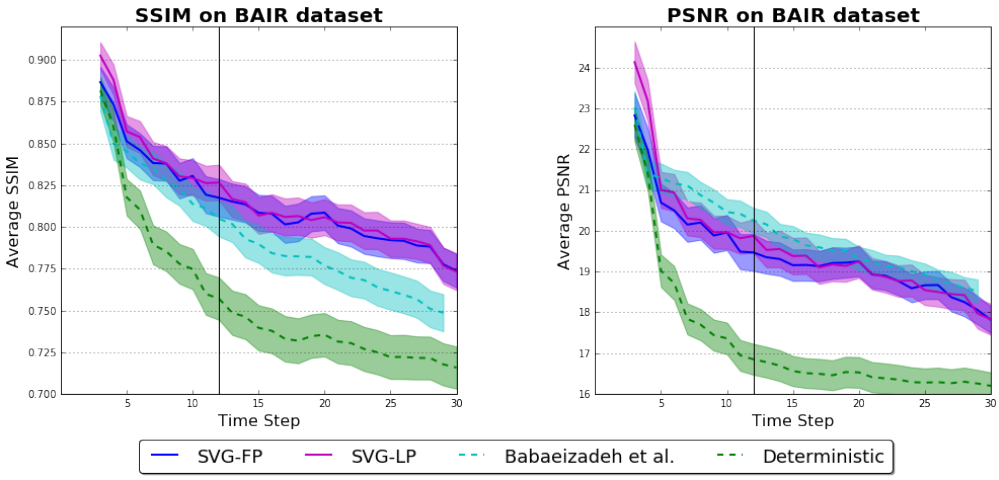

Figure 6.9: Quantitative comparison between our SVG models and Babaeizadeh et al. (2018) on the BAIR robot dataset. All models are conditioned on the first two frames and generate the subsequent 28 frames. The models were trained to predict up 10 frames in the future, indicated by the vertical bar; further generations indicate generalization ability. Mean SSIM and PSNR over test videos is plotted with 95% confidence interval shaded.

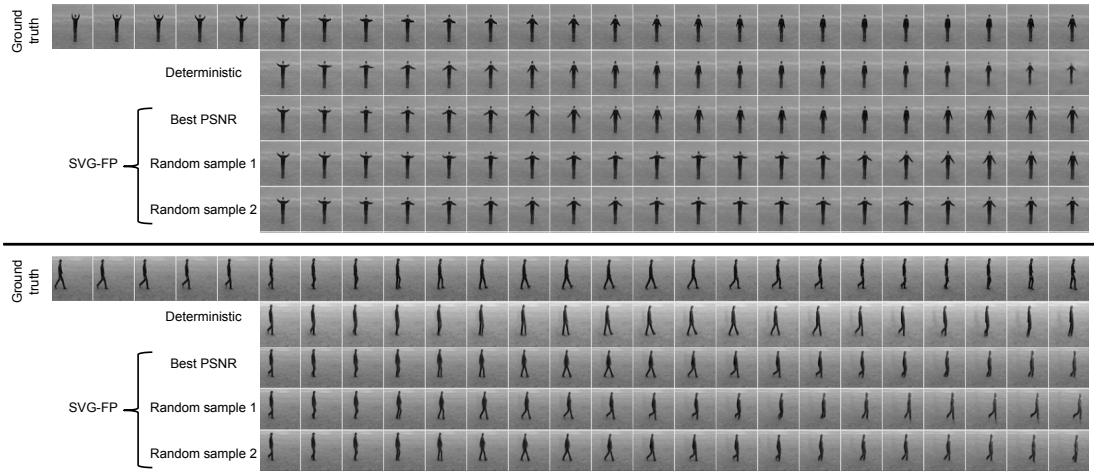

Figure 6.10: Qualitative comparison between SVG-LP and a purely deterministic baseline. Both models were conditioned on the first 10 frames (the final 5 are shown in the figure) of test sequences. The deterministic model produces plausible predictions for the future frames but frequently mispredicts precise limb locations. In contrast, different samples from SVG-FP reflect the variability on the persons pose in future frames. By picking the sample with the best PSNR, SVG-FP closely matches the ground truth sequence.

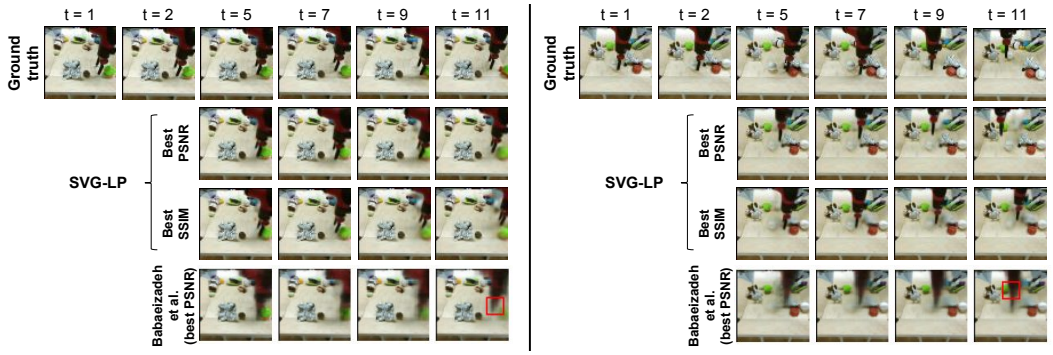

Figure 6.11: Qualitative comparison between our SVG-LP model and Babaeizadeh et al. (2018). All models are conditioned on the first two frames of unseen test videos. SVG-LP generates crisper images and predicts plausible movement of the robot arm.

held out test sequences, comparing to the state-of-the-art approach of Babaeizadeh et al. (2018). This evaluation consists of conditioning on 2 frames and generating 28 subsequent ones, i.e. longer than at train time, demonstrating the generalization capability of SVG-FP and SVG-LP. Both SVG-FP and SVG-LP outperform Babaeizadeh et al. (2018)

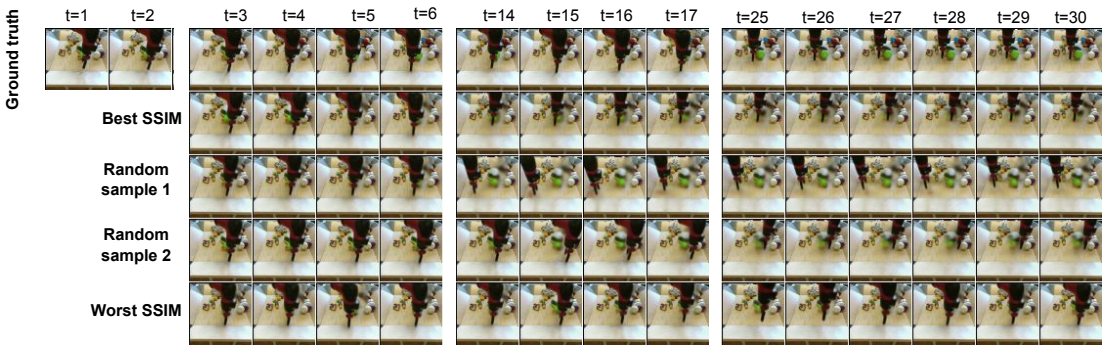

Figure 6.12: Additional examples of generations from SVG-LP showing crisp and varied predictions. A large segment of the background is occluded in conditioning frames, preventing SVG-LP from directly copying these background pixels into generated frames. In addition to crisp robot arm movement, SVG-LP generates plausible background objects in the space occluded by the robot arm in initial frames.

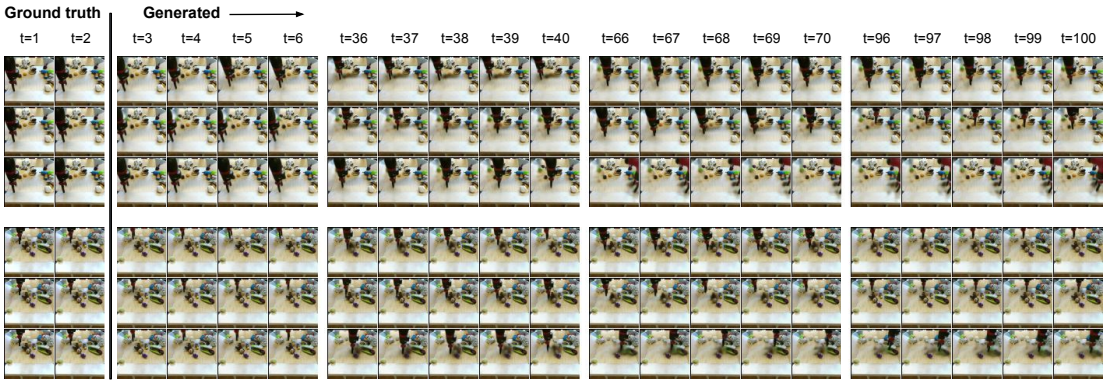

Figure 6.13: Long range generations from SVG-LP. The robot arm remains crisp up to 100 time steps and object motion can be seen in the generated video frames. Additional videos can be viewed at: `https://sites.google.com/view/svglp/`.

in terms of SSIM. SVG-LP outperforms the remaining models in terms of PSNR for the first few steps, after which Babaeizadeh et al. (2018) is marginally better. Qualitatively, SVG-FP and SVG-LP produce significantly sharper generations than Babaeizadeh et al. (2018), as illustrated in Figure 6.11. PSNR is biased towards overly smooth (i.e. blurry) results which might explain the slightly better PSNR scores obtained by Babaeizadeh et al. (2018) for later time steps.

SVG-FP and SVG-LP produce crisp generations many time steps into the future.

Figure 6.12 shows sample generations up to 30 time steps alongside the ground truth video frames. We also ran SVG-LP forward for 100 time steps and continue to see crisp motion of the robot arm (see Figure 6.13).

6.4 Discussion

We have introduced a novel video prediction model that combines a deterministic prediction of the next frame with stochastic latent variables, drawn from a time-varying distribution learned from training sequences. Our recurrent inference network estimates the latent distribution for each time step allowing easy end-to-end training. Evaluating the model on real-world sequences, we demonstrate high quality generations that are comparable to, or better than, existing approaches. On synthetic data where it is possible to characterize the distribution of samples, we see that is able to match complex distributions of futures. The framework is sufficiently general that it can readily be applied to more complex datasets, given appropriate encoder and decoder modules.

Chapter 7

Conclusion

A crucial step towards building intelligent agents is the development of methods for learning about the rich structure in the visual world, without heavy reliance on labeled data. In this context, generative models of image and video are of fundamental importance. Accurate image generation requires an understanding of high level causal factors that describe the underlying structure of a dataset. Similarly, accurate video prediction relies on an understanding of object parts and relations, physics, and high level causal relationships in the world. This makes image generation and video prediction natural frameworks for learning visual representations in an unsupervised manner.

Accurate video prediction models also have broad applications in reinforcement learning, planning, and control. Action-conditional environment models can endow reinforcement learning agents with the ability to predict the outcome of its actions. This can significantly improve the sample efficiency of deep reinforcement learning by facilitating planning (Henaff et al., 2018; Weber et al., 2017) and exploration (Chiappa et al., 2017; Pathak et al., 2017; Stadie et al., 2015). Action-conditional models have also been used for real-world robot control tasks (Agrawal et al., 2016; Ebert et al., 2017; Finn and

Levine, 2017).

Motivated by these considerations, this thesis explored deep learning approaches for building generative and predictive models of the visual world. In Chapter 4 we introduced the Laplacian Pyramid of adversarial networks (LAPGAN), a multi-scale image generation method based on the GAN framework (Goodfellow et al., 2014). This model was the first parametric generative model to demonstrate convincing generation results on complex natural image datasets. LAPGAN, followed shortly after by DCGAN (Radford et al., 2016), showed the generative modeling potential of convolutional GANs. In the following years, GANs have revolutionized the field of generative modeling. Rapid progress has been made and current GAN methods demonstrate extremely high quality image synthesis results. Several recent state-of-the-art methods also utilize a multi-scale approach (Karras et al., 2018; Zhang et al., 2017, 2018c), similar to LAPGAN.

In Chapter 5 we introduced Disentangled Representation Net (DRNET), a framework for learning representations suitable for video prediction. By leveraging the temporal structure of video data as group-level supervision, DRNET learns a disentangled content and pose representation. Here, we defined content as anything which is constant within a short video clip, and pose as anything which varies across the clip. We showed the clean disentanglement learned by DRNET allows for image synthesis by analogy, i.e. combining content and pose of different images, and facilitates video prediction. Video prediction becomes straightforward using our representation since only the pose features change across the clip. We train a recurrent model to predict a future sequence of pose vectors, conditioned on past observations. Content features, by design, remain fixed. Despite the simplicity, this model generates convincing long-range frame predictions.

In Chapter 6 we introduced a stochastic video generation model that combines a deterministic frame predictor with time-dependent stochastic latent variables. Our model

learns a prior over latent variables at each time step. We show empirically that the learned prior can be interpreted as a predictive model of uncertainty. Specifically, the prior predicts high variance distributions for time steps of high uncertainty and low variance distributions for points of low uncertainty. We train the model using techniques from approximate variational inference and show high quality stochastic generation results. We find the model is easy to train and produces significantly sharper results that alternative approaches.

The video prediction models presented in this thesis each address a different challenge. Our DRNET model focuses on learning an abstract representation of images that facilitates video predication, so as to avoid pixel-level objectives; our stochastic video generation model addresses the problem of modeling uncertainty in future predictions. These approaches can also be combined by, for example, using the content/pose representation of DRNET as the input to the stochastic generation model.

Despite rapid progress in recent years, video prediction is far from solved. One critical challenge is developing better methods of evaluation. Current evaluation methods tend to rely on pixel-level metrics such as PSNR and SSIM. However, these metrics do a poor job of quantifying perceptual fidelity of predicted frames. To this end, we utilized a metric based on the Inception score (Salimans et al., 2016) in our evaluation in Chapter 5. This metric has its own suite of problems (Barratt and Sharma, 2018) and only measures a specific aspect of any prediction model.

Ultimately, we advocate for video prediction methods to be evaluated on a suite of metrics. Depending on the downstream application of a model, certain metrics are more relevant that others. For example, for image editing applications, crisp high quality generations may be more important than accurately predicting the content of a future frame. For long term planning applications, precise details may be less important that

capturing the high level semantic content of a future scene. Alternatively, many control applications require prediction of precise and accurate object trajectories. This points to the importance of, when appropriate, evaluating video prediction models in light of the end goal.

Another major challenge is the development of a standard suite of video datasets upon which to benchmark models. One promising avenue of future research is the development of a suite of video datasets, each reflecting a particular challenge. Such challenges might include modeling long term dependencies, multiple object interactions, causal relationships, physics, etc. Generating the datasets in a virtual environment could provide annotations (e.g. object centroids) useful for evaluation metrics. Video generation models could then be evaluated against such a suite to determine particular strengths and weaknesses.

Bibliography

Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. *Advances in Neural Information Processing Systems (NIPS)*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv 1701.07875*.

Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. (2018). Stochastic variational video prediction. *International Conference on Learning Representations (ICLR)*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Barratt, S. and Sharma, R. (2018). A note on the inception score. In *arXiv:1801.01973*.

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

Belghazi, M. I., Rajeswar, S., Mastropietro, O., Rostamzadeh, N., Mitrovic, J., and Courville, A. (2018). Hierarchical adversarially learned inference. *arXiv:1802.01071*.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Berthelot, D., Schumm, T., and Metz, L. (2017). Began: boundary equilibrium generative adversarial networks. In *arXiv:1703.10717*.

Beutel, A., Chen, J., Zhao, Z., and Chi, E. H. (2017). Data decisions and theoretical implications when adversarially learning fair representations. In *arXiv:1707.00075*.

Bojanowski, P., Joulin, A., Lopez-Pas, D., and Szlam, A. (2018). Optimizing the latent space of generative networks. In *International Conference on Machine Learning (ICML)*.

Bouchacourt, D., Tomioka, R., and Nowozin, S. (2018). Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI Conference on Artificial Intelligence*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of The SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Brakel, P. and Bengio, Y. (2017). Learning independent features with adversarial nets for non-linear ica. *Workshop on Implicit Models, ICML*.

Brock, A., Lim, T., Ritchie, J., and Weston, N. (2017). Neural photo editing with introspective adversarial networks. In *International Conference on Learning Representations (ICLR)*.

Browne, C., Whitehouse, D., Lucas, S., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games, 4(1)*.

Burt, P. J., Edward, and Adelson, E. H. (1983). The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540.

Cadieu, C. F. and Olshausen, B. A. (2009). Learning transformational invariants from natural movies. In *Advances in Neural Information Processing Systems (NIPS)*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.

Cheung, B., Livezey, J. A., Bansal, A. K., and Olshausen, B. A. (2014). Discovering hidden factors of variation in deep networks. In *arXiv:1412.6583*.

Chiappa, S., Racaniere, S., Wierstra, D., and Mohamed, S. (2017). Recurrent environment simulators. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems (NIPS)*.

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. In *Behavioural and Brain Sciences, 36(3)*.

De Bonet, J. S. (1997). Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368. ACM Press/Addison-Wesley Publishing Co.

Denton, E. and Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems (NIPS)*.

Denton, E., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Denton, E. and Fergus, R. (2018). Stochastic video generation with a learned prior. In *International Conference on Machine Learning (ICML)*.

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: non-linear independent components estimation. *arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real nvp. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430.

Dolhansky, B. and Ferrer, C. C. (2018). Eye in-painting with exemplar generative ad-

versarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Donahue, J., Krhenbhl, P., and Darrell, T. (2017). Adversarial feature learning. *International Conference on Learning Representations (ICLR)*.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2017). Adversarially learned inference. *International Conference on Learning Representations (ICLR)*.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. In *Uncertainty in Artificial Intelligence*.

Ebert, F., Finn, C., Lee, A. X., and Levine, S. (2017). Self-supervised visual planning with temporal skip connections. In *Conference on Robot Learning (CoRL)*.

Edwards, H. and Storkey, A. (2016). Censoring representations with an adversary. *International Conference on Learning Representations (ICLR)*.

Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, volume 2, pages 1033–1038. IEEE.

Eslami, S. A., Heess, N., Williams, C. K., and Winn, J. (2014). The shape boltzmann machine: a strong model of object shape. *International Journal of Computer Vision*, 107(2):155–176.

Finn, C., Goodfellow, I., and Levine, S. (2016). Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems (NIPS)*.

Finn, C. and Levine, S. (2017). Deep visual foresight for planning robot motion. *International Conference on Robotics and Automation (ICRA)*.

Fraccaro, M., Snderby, S. K., Paquet, U., and Winther, O. (2016). Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Freeman, W. T., Jones, T. R., and Pasztor, E. C. (2002). Example-based super-resolution. *Computer Graphics and Applications, IEEE*, 22(2):56–65.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*.

Gauthier, J. (2014). Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.

Goroshin, R., Mathieu, M., and LeCun, Y. (2015). Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems (NIPS)*.

Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623.

Gregor, K. and LeCun, Y. (2010a). Emergence of complex-like cells in a temporal product network with local receptive fields. In *arXiv:1006.0448*.

Gregor, K. and LeCun, Y. (2010b). Learning fast approximations of sparse coding. In *International Conference on Machine Learning (ICML)*.

Gretton, A., Borgwardt, K., Rasch, M. J., Scholkopf, B., and Smola, A. J. (2007). A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems (NIPS)*.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning (ICML)*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*.

Guo, X., Singh, S., Lee, H., Lewis, R., and Wang, X. (2014). Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in Neural Information Processing Systems (NIPS)*.

Hamrick, J. B., Ballard, A. J., Pascanu, R., Vinyals, O., Heess, N., and Battaglia, P. W. (2017). Metacontrol for adaptive imagination-based optimization. In *International Conference on Learning Representations (ICLR)*.

Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask r-cnn. In *International Conference on Computer Vision (ICCV)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Henaff, M., Whitney, W., and LeCun, Y. (2018). Model-based planning with discrete and continuous actions. *arXiv:1705.07177*.

Henaff, M., Zhao, J., and LeCun, Y. (2017). Prediction under uncertainty with error-encoding networks. *arXiv:1711.04994*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). Early visual concept learning with unsupervised deep learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hinton, G., Sabour, S., and Frosst, N. (2017). Matrix capsules with em routing. In *International Conference on Learning Representations (ICLR)*.

Hinton, G. E., Krizhevsky, A., and Wang, S. (2011). Transforming auto-encoders. In *ICANN*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hjelm, R. D., Jacob, A. P., Trischler, A., Che, G., Cho, K., and Bengio, Y. (2018). Boundary-seeking generative adversarial networks. In *International Conference on Learning Representations (ICLR)*.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hyvarinen, A. and Oja, E. (2004). Independent component analysis: Algorithms and applications. *Neural Networks, 13(4-5)*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167v3*.

Isola, P., Zhu, J., Zhou, T., and Efros, A. (2015). Image-to-image translation with conditional adversarial networks. *arXiv:1611.070041*.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*.

Jayaraman, D. and Grauman, K. (2015). Learning image represen- tations tied to ego-motion. In *International Conference on Computer Vision (ICCV)*.

Jojic, N. and Frey, B. J. (2001). Learning flexible sprites in video layers. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv:1806.10293*.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2016). Video pixel networks. *arXiv:1610.00527*.

Karpathy, A. and Li, F.-F. (2015). Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition*.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kavukcuoglu, K., Sermanet, P., Boureau, Y.-L., Gregor, K., Mathieu, M., and LeCun, Y. (2010). Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS)*.

Kim, H. and Mnih, A. (2018). Disentangling by factorising. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. (2017). Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning (ICML)*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D., Rezende, D., Mohamed, S., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 11 convolutions. *arXiv:1807.03039*.

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014). Multimodal neural language models. In *International Conference on Machine Learning*.

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

Krizhevsky, A., Hinton, G. E., et al. (2010). Factored 3-way restricted boltzmann machines for modeling natural images. In *AISTATS*, pages 621–628.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114.

Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2539–2547.

Kumar, A., Sattigeri, P., and Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations (ICLR)*.

Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., and Ranzato, M. (2017). Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems (NIPS)*.

Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. In *AISTATS*.

Le, Q. V., Zou, W. Y., Yeung, S. Y., and Ng, A. Y. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*.

LeCun, Y., Huang, F., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2016). Photo-realistic single image super-resolution using a generative adversarial network. *https://arxiv.org/abs/1609.04802*.

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

Lee, H., Ekanadham, C., and Ng, A. (2007). Sparse deep belief net model for visual area v2. In *Advances in Neural Information Processing Systems (NIPS)*.

Leibfried, F., Kushman, N., and Hofmann, K. (2017). A deep learning approach for joint video frame and reward prediction in atari games. In *Workshop on Principled Approaches to Deep Learning, ICML*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Pczos, B. (2017). Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems (NIPS)*.

Li, Y., Swersky, K., and Zemel, R. S. (2015). Generative moment matching networks. *arXiv 1502.02761*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Lin, J., Xia, Y., Qin, T., Chen, Z., and Liu, T.-Y. (2018). Conditional image-to-image translation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, C. (2009). Beyond pixels: exploring new representations and applications for motion analysis. *PhD thesis, Massachusetts Institute of Technology*.

Lotter, W., Kreiman, G., and Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv:1605.08104*.

Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic segmentation using adversarial networks. In *NIPS Worshop on Adversarial training*.

Luc, P., Couprie, C., Lecun, Y., and Verbeek, J. (2018). Predicting future instance segmentations by forecasting convolutional features. In *arXiv:1803.11496*.

Luc, P., Neverova, N., Couprie, C., Verbeek, J., and LeCun, Y. (2017). Predicting deeper into the future of semantic segmentation. In *International Conference on Computer Vision (ICCV)*.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2017). Are gans created equal? a large-scale study. In *arXiv:1711.10337*.

Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., and Gool, L. V. (2017). Pose guided person image generation. In *Advances in Neural Information Processing Systems (NIPS)*.

Madras, D., Creager, E., Pitassi, T., and Zemel, R. (2018). Learning adversarially fair and transferable representations. In *International Conference on Machine Learning (ICML)*.

Makhzani, A. and Frey, B. (2014). k-sparse autoencoders. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mao, X., Li, Q., Xie, H., Lau, R. Y., and Wang, Z. (2017). Least squares generative adversarial networks. *arXiv:1611.04076*.

Masci, J., Meier, U., Ciresan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning*.

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

Mathieu, M., Junbo Zhao, P. S., Ramesh, A., and LeCun, Y. (2016b). Disentangling factors of variation in deep representations using adversarial training. In *Advances in Neural Information Processing Systems (NIPS)*.

Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Michalski, V., Memisevic, R., and Konda, K. (2014). Modeling deep temporal dependencies with recurrent grammar cells. In *Advances in Neural Information Processing Systems (NIPS)*.

Mikolov, T. (2012). Statistical language models based on neural networks. *PhD thesis, Brno University of Technology*.

Mirowski, P., Grimes, M. K., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., Simonyan, K., Kavukcuoglu, K., Zisserman, A., and Hadsell, R. (2018). Learning to navigate in cities without a map. In *arXiv:1804.00168*.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., and Hadsell, R. (2017). Learning to navigate in complex environments. In *International Conference on Learning Representations (ICLR)*.

Mirza, M. and Osindero, S. (2014a). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Mirza, M. and Osindero, S. (2014b). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *Deep Learning Workshop, NIPS*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv:1610.03483*.

Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., and Yosinski, J. (2017). Plug & play generative networks: Conditional iterative generation of images in latent space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NIPS)*.

Odena, A. (2016). Semi-supervised learning with generative adversarial networks. In *ICML Workshop on Data-Efficient Machine Learning*.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. (2015). Action-conditional video prediction using deep networks in Atari games. In *Advances in Neural Information Processing Systems (NIPS)*.

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325.

Osindero, S. and Hinton, G. E. (2008). Modeling image patches with a directed hierarchy of markov random fields. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems (NIPS)*.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition*.

Perarnau, G., van de Weijer, J., Raducanu, B., and lvarez, J. M. (2016). Invertible conditional gans for image editing. In *Workshop on Adversarial Training, NIPS*.

Poole, B., Alemi, A. A., Sohl-Dickstein, J., and Angelova, A. (2016). Improved generator objectives for gans. In *Workshop on Adversarial Training, NIPS*.

Portilla, J. and Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ranzato, M. and LeCun, Y. (2007). A sparse and locally shift invariant feature extractor applied to document images. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Ranzato, M., Mnih, V., Susskind, J. M., and Hinton, G. E. (2013). Modeling natural images using gated MRFs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):2206–2222.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder network. In *Advances in Neural Information Processing Systems (NIPS)*.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016). Generative adversarial text to image synthesis. In *International Conference on Machine Learning (ICML)*.

Reed, S., Sohn, K., Zhang, Y., and Lee, H. (2014). Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning (ICML)*.

Reed, S., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Belov,

D., and de Freitas, N. (2017a). Parallel multiscale autoregressive density estimation. *arXiv 1703.03664*.

Reed, S., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Belov, D., and de Freitas, N. (2017b). Parallel multiscale autoregressive density estimation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and variational inference in deep latent gaussian models. *arXiv preprint arXiv:1401.4082*.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning (ICML)*.

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

Roth, S. and Black, M. J. (2005). Fields of experts: A framework for learning image priors. In *In Computer Vision and Pattern Recognition*, pages 860–867.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NIPS)*.

Salakhutdinov, R. (2015). Learning deep generative models. *Annual Review of Statistics and Its Application*, 2:361–385.

Salakhutdinov, R. and Hinton, G. E. (2009). Deep boltzmann machines. In *AISTATS*, pages 448–455.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advance in Neural Information Processing Systems (NIPS)*.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv:1701.05517*.

Schmidhuber, J. (1990). An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *International Joint Conference on Neural Networks (IJCNN)*.

Schmidhuber, J. (1992). Learning factorial codes by predictability minimization. *Neural Computation, 4(6)*.

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local svm approach. In *Proceedings of the International Conference on Pattern Recognition*.

Schulman, J., Levine, S., Abbeel, P., and Jordan, M. I. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*.

Siddharth, N., Paige, B., van de Meent, J.-W., Desmaison, A., Goodman, N. D., Kohli, P., Wood, F., and Torr, P. H. (2017). Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Simoncelli, E. P., Freeman, W. T., Adelson, E. H., and Heeger, D. J. (1992). Shiftable multiscale transforms. *Information Theory, IEEE Transactions on*, 38(2):587–607.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585.

Sölch, M., Bayer, J., Ludersdorfer, M., and van der Smagt, P. (2016). Variational inference for on-line anomaly detection in high-dimensional time series. *arXiv:1602.07109*.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. (2017). Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv 1511.06390*.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using LSTMs. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. In *Workshop on Deep Reinforcement Learning, NIPS*.

Susskind, J., Memisevic, R., Hinton, G., and Pollefeys, M. (2011). Modeling the joint density of two images under a variety of transformations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sutskever, I., Hinton, G., and Taylor, G. (2009). The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems (NIPS)*.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *International Conference on Machine Learning (ICML)*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Theis, L. and Bethge, M. (2015). Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems (NIPS)*.

Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. (2015). Simultaneous deep transfer across domains and tasks. In *International Conference on Computer Vision (ICCV)*.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discrimina-

tive domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive densityestimator. In *Advances in Neural Information Processing Systems (NIPS)*.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016a). Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016c). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems (NIPS)*.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. (2017a). Decomposing motion and content for natural video sequence prediction. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. (2017b). Learning to generate long-term future via hierarchical prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*.

Vondrick, C., Pirsiavash, H., and Torralba, A. (2016a). Anticipating visual representations with unlabeled video. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vondrick, C., Pirsiavash, H., and Torralba, A. (2016b). Generating videos with scene dynamics. In *Advance in Neural Information Processing Systems (NIPS)*.

Vondrick, C. and Torralba, A. (2017). Generating the future with adversarial transformers. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*.

Walker, J., Gupta, A., and Hebert, M. (2014). Patch to the future: Unsupervised visual prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Walker, J., Gupta, A., and Hebert, M. (2015). Dense optical flow prediction from a static image. In *International Conference on Computer Vision (ICCV)*.

Wang, X. and Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2794–2802.

Wang, Y., Gao, Z., Long, M., Wang, J., and Yu, P. S. (2018). Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *arXiv:1804.06300*.

Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P. S. (2017). Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *Advances in Neural Information Processing Systems (NIPS)*.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv:1506.07365*.

Weber, T., Racanire, S., Reichert, D. P., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., and Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Whitney, W. F., Chang, M., Kulkarni, T., and Tenenbaum, J. B. (2016). Understanding visual concepts with continuation learning. *arXiv:1502.04623*.

Wichers, N., Villegas, R., Erhan, D., and Lee, H. (2018). Long-term video prediction without supervision. In *International Conference on Machine Learning (ICML)*.

Wiskott, L. and Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariance. *Neural Computation*, 14(4):715–770.

Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T. S., and Yan, S. (2010). Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044.

Xue, T., Wu, J., Bouman, K. L., and Freeman, W. T. (2016). Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhang, B. H., Lemoine, B., and Mitchell, M. (2018a). Mitigating unwanted biases with adversarial learning. In *Conference on Artificial Intelligence, Ethics and Society (AEIS)*.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2018b). Self-attention generative adversarial networks. *arXiv:1805.08318*.

Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., and Metaxas, D. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *International Conference on Computer Vision (ICCV)*.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. (2018c). Stackgan++: Realistic image synthesis with stacked generative adversarial networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*.

Zhang, X., Zhao, J., and LeCun, Y. (2015a). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhang, Y., Yu, F., Song, S., Xu, P., Seff, A., and Xiao, J. (2015b). Large-scale scene understanding challenge. In *Computer Vision and Pattern Recognition Workshop*.

Zhao, J., Mathieu, M., Goroshin, R., and LeCun, Y. (2016). Stacked what-where auto-encoders. In *International Conference on Learning Representations (ICLR)*.

Zhao, J., Mathieu, M., and LeCun, Y. (2017). Energy based generative adversarial net-

works. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Zhu, S. C., Wu, Y., and Mumford, D. (1998). Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126.

Zilly, J. G., Srivastava, R. K., Koutnk, J., and Schmidhuber, J. (2017). Recurrent highway networks. In *International Conference on Machine Learning (ICML)*.

Zoran, D. and Weiss, Y. (2011). From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision*.

Zou, W. Y., Zhu, S., Ng, A. Y., , and Yu., K. (2012). Deep learning of invariant features via simulated fixations in video. In *Advances in Neural Information Processing Systems (NIPS)*.