Disentangled Sequential Autoencoder

Yingzhen Li [1]   **Stephan Mandt** [2]

Abstract

We present a VAE architecture for encoding and generating high dimensional sequential data, such as video or audio. Our deep generative model learns a latent representation of the data which is split into a static and dynamic part, allowing us to approximately disentangle latent time-dependent features (dynamics) from features which are preserved over time (content). This architecture gives us partial control over generating content and dynamics by conditioning on either one of these sets of features. In our experiments on artificially generated cartoon video clips and voice recordings, we show that we can convert the content of a given sequence into another one by such content swapping. For audio, this allows us to convert a male speaker into a female speaker and vice versa, while for video we can separately manipulate shapes and dynamics. Furthermore, we give empirical evidence for the hypothesis that stochastic RNNs as latent state models are more efficient at compressing and generating long sequences than deterministic ones, which may be relevant for applications in video compression.

1. Introduction

Representation learning remains an outstanding research problem in machine learning and computer vision. Recently there is a rising interest in disentangled representations, in which each component of learned features refers to a semantically meaningful concept. In the example of video sequence modelling, an ideal disentangled representation would be able to separate time-independent concepts (e.g. the identity of the object in the scene) from dynamical information (e.g. the time-varying position and the orientation or pose of that object). Such disentangled represen-

tations would open new efficient ways of compression and style manipulation, among other applications.

Recent work has investigated disentangled representation learning for images within the framework of variational auto-encoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014). Some of them, e.g. the β-VAE method (Higgins et al., 2016), proposed new objective functions/training techniques that encourage disentanglement. On the other hand, network architecture designs that directly enforce factored representations have also been explored by e.g. Siddharth et al. (2017); Bouchacourt et al. (2017). These two types of approaches are often mixed together, e.g. the infoGAN approach (Chen et al., 2016) partitioned the latent space and proposed adding a mutual information regularisation term to the vanilla GAN loss. Mathieu et al. (2016) also partitioned the encoding space into style and content components, and performed adversarial training to encourage the datapoints from the same class to have similar content representations, but diverse style features.

Less research has been conducted for *unsupervised* learning of disentangled representations of sequences. For video sequence modelling, Villegas et al. (2017) and Denton & Birodkar (2017) utilised different networks to encode the content and dynamics information separately, and trained the auto-encoders with a combination of reconstruction loss and GAN loss. Structured (Johnson et al., 2016) and Factorised VAEs (Deng et al., 2017) used hierarchical priors to learn more interpretable latent variables. Hsu et al. (2017) designed a structured VAE in the context of speech recognition. Their VAE architecture is trained using a combination of the standard variational lower bound and a discriminative regulariser to further encourage disentanglement. More related work is discussed in Section 3.

In this paper, we propose a generative model for unsupervised structured sequence modelling, such as video or audio. We show that, in contrast to previous approaches, a disentangled representation can be achieved by a careful design of the probabilistic graphical model. In the proposed architecture, we explicitly use a latent variable to represent *content*, i.e., information that is invariant through the sequence, and a set of latent variables associated to each frame to represent *dynamical* information, such as pose and position. Com-

---

[1]University of Cambridge [2]Los Angeles, CA, USA. Correspondence to: Yingzhen Li <yl494@cam.ac.uk>, Stephan Mandt <stephan.mandt@gmail.com>.

pared to the mentioned previous models that usually predict future frames conditioned on the observed sequences, we focus on learning the distribution of the video/audio content and dynamics to enable sequence generation without conditioning. Therefore our model can also generalise to unseen sequences, which is confirmed by our experiments. In more detail, our contributions are as follows:

- Controlled generation. Our architecture allows us to approximately control for content and dynamics when generating videos. We can generate random dynamics for fixed content, and random content for fixed dynamics. This gives us a controlled way of manipulating a video/audio sequence, such as swapping the identity of moving objects or the voice of a speaker.

- Efficient encoding. Our representation is more data efficient than encoding a video frame by frame. By factoring out a separate variable that encodes content, our dynamical latent variables can have smaller dimensions. This may be promising when it comes to end-to-end neural video encoding methods.

- We design a new metric that allow us to verify disentanglement of the latent variables, by investigating the stability of an object classifier over time.

- We give empirical evidence, based on video data of a physics simulator, that for long sequences, a stochastic transition model generates more realistic dynamics.

The paper is structured as follows. Section 2 introduces the generative model and the problem setting. Section 3 discusses related work. Section 4 presents three experiments on video and speech data. Finally, Section 5 concludes the paper and discusses future research directions.

2. The model

Let $\boldsymbol{x}_{1:T} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ denote a high dimensional sequence, such as a video with T consecutive frames. Also, assume the data distribution of the training sequences is $p_{\lceil \dashv \sqcup \dashv \rceil}(\boldsymbol{x}_{1:T})$. In this paper, we model the observed data with a latent variable model that separates the representation of time-invariant concepts (e.g. object identities) from those of time-varying concepts (e.g. pose information).

Generative model. Consider the following probabilistic model, which is also visualised in Figure 1:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{f}) = p_{\boldsymbol{\theta}}(\boldsymbol{f}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{<t}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{f}).$$

$$(1)$$

We use the convention that $\boldsymbol{z}_0 = \boldsymbol{0}$. The generation of frame $\boldsymbol{x}_t$ at time t depends on the corresponding latent variables $\boldsymbol{z}_t$ and $\boldsymbol{f}$. θ are model parameters.

Ideally, $\boldsymbol{f}$ will be capable of modelling global aspects of the whole sequence which are time-invariant, while $\boldsymbol{z}_t$ will encode time-varying features. This separation may be achieved when choosing the dimensionality of $\boldsymbol{z}_t$ to be small enough, thus reserving $\boldsymbol{z}_t$ only for time-dependent features while compressing everything else into $\boldsymbol{f}$. In the context of video encodings, $\boldsymbol{z}_t$ would thus encode a "morphing transformation", which encodes how a frame at time t is morphed into a frame at time $t+1$.

Inference models. We use variational inference to learn an approximate posterior over latent variables given data (Jordan et al., 1999). This involves an approximating distribution q. We train the generative model with the VAE algorithm (Kingma & Welling, 2013):

$$\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \quad \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x}_{1:T})} \left[\mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{f})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}, \boldsymbol{f}|\boldsymbol{x}_{1:T})} \right] \right]. \quad (2)$$

To quantify the effect of the architecture of q on the learned generative model, we test with two types of q factorisation structures as follows.

The first architecture constructs a factorised q distribution

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}, \boldsymbol{f}|\boldsymbol{x}_{1:T}) = q_{\boldsymbol{\phi}}(\boldsymbol{f}|\boldsymbol{x}_{1:T}) \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{x}_t) \quad (3)$$

as the amortised variational distribution. We refer to this as "factorised q" in the experiments section. This factorization assumes that content features are approximately independent of motion features. Furthermore, note that the distribution over content features is conditioned on the entire time series, whereas the dynamical features are only conditioned on the individual frames.

The second encoder assumes that the variational posterior of $\boldsymbol{z}_{1:T}$ depends on $\boldsymbol{f}$, and the q distribution has the following architecture:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}, \boldsymbol{f}|\boldsymbol{x}_{1:T}) = q_{\boldsymbol{\phi}}(\boldsymbol{f}|\boldsymbol{x}_{1:T}) q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{f}, \boldsymbol{x}_{1:T}), \quad (4)$$

and the distribution $q(\boldsymbol{z}_{1:T}|\boldsymbol{f}, \boldsymbol{x}_{1:T})$ is conditioned on the entire time series. It can be implemented by e.g. a bi-directional LSTM (Graves & Schmidhuber, 2005) conditioned on $\boldsymbol{f}$, followed by an RNN taking the bi-LSTM hidden states as the inputs. We provide a visualisation of the corresponding computation graph in the appendix. This encoder is referred to as "full q". The idea behind the structured approximation is that content may affect dynamics: in video, the shape of objects may be informative about their motion patterns, thus $\boldsymbol{z}_{1:T}$ is conditionally dependent on $\mathbf{f}$. The architectures of the generative model and both encoders are visualised in Figure 1.

Unconditional generation. After training, one can use the generative model to synthesise video or audio sequences
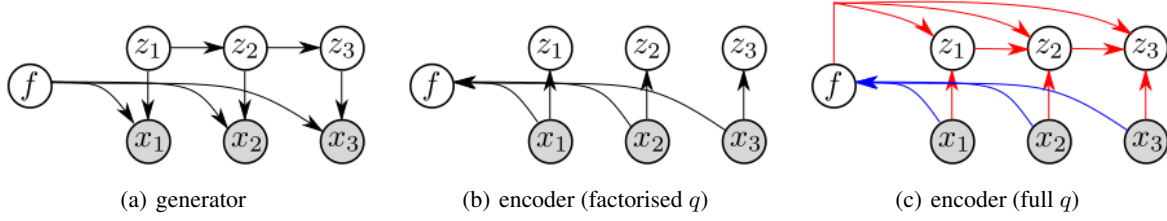
Figure 1. A graphical model visualisation of the generator and the encoder.

by sampling the latent variables from the prior and decoding them. Furthermore, the proposed generative model allows generation of multiple sequences entailing the same global information (e.g. the same object in a video sequence), simply by fixing $\boldsymbol{f} \sim p(\boldsymbol{f})$, sampling different $\boldsymbol{z}_{1:T}^k \sim p(\boldsymbol{z}_{1:T})$, $k = 1, ..., K$, and generating the observations $\boldsymbol{x}_t^k \sim p(\boldsymbol{x}_t|\boldsymbol{z}_t^k, \boldsymbol{f})$. Generating sequences with similar dynamics is done analogously, by fixing $\boldsymbol{z}_{1:T} \sim p(\boldsymbol{z}_{1:T})$ and sampling $\boldsymbol{f}^k, k = 1, ...K$ from the prior.

Conditional generation. Together with the encoder, the model also allows conditional generation of sequences. As an example, given a video sequence $\boldsymbol{x}_{1:T}$ as reference, one can manipulate the latent variables and generate new sequences preserving either the object identity or the pose/movement information. This is done by conditioning on $\boldsymbol{f} \sim q(\boldsymbol{f}|\boldsymbol{x}_{1:T})$ for a given $\boldsymbol{x}_{1:T}$ then randomising $\boldsymbol{z}_{1:T}$ from the prior, or the other way around.

Feature swapping. One might also want to generate a new video sequence with the object identity and pose information encoded from different sequence. Given two sequences $\boldsymbol{x}_{1:T}^a$ and $\boldsymbol{x}_{1:T}^b$, the synthesis process first infers the latent variables $\boldsymbol{f}^a \sim q(\boldsymbol{f}|\boldsymbol{x}_{1:T}^a)$ and $\boldsymbol{z}_{1:T}^b \sim q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}^b)$[1], then produces a new sequence by sampling $\boldsymbol{x}_t^{\text{new}} \sim p(\boldsymbol{x}_t|\boldsymbol{z}_t^b, \boldsymbol{f}^a)$. This allows us to control both the content and the dynamics of the generated sequence, which can be applied to e.g. conversion of voice of the speaker in a speech sequence.

3. Related work

Research on learning disentangled representation has mainly focused on two aspects: the training objective and the generative model architecture. Regarding the loss function design for VAE models, Higgins et al. (2016) propose the β-VAE by scaling up the $\text{KL}[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})]$ term in the variational lower-bound with $\beta > 1$ to encourage learning of independent attributes (as the prior $p(\boldsymbol{z})$ is usually factorised). While the β-VAE has been shown effective in learning better representations for natural images and might be able to further improve the performance of our model, we do not

test this recipe here to demonstrate that disentanglement can be achieved by a careful model design.

For sequence modelling, a number of prior publications have extended VAE to video and speech data (Fabius & van Amersfoort, 2014; Bayer & Osendorfer, 2014; Chung et al., 2015). These models, although being able to generate realistic sequences, do not explicitly disentangle the representation of time-invariant and time-dependent information. Thus it is inconvenient for these models to perform tasks such as controlled generation and feature swapping.

For GAN-like models, both Villegas et al. (2017) and Denton & Birodkar (2017) proposed an auto-encoder architecture for next frame prediction, with two separate encoders responsible for content and pose information at each time step. While in Villegas et al. (2017), the pose information is extracted from the difference between two consecutive frames $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$, Denton & Birodkar (2017) directly encoded $\boldsymbol{x}_t$ for both pose and content, and further designed a training objective to encourage learning of disentangled representations. On the other hand, Vondrick et al. (2016) used a spatio-temporal convolutional architecture to disentangle a video scene's foreground from its background. Although it has successfully achieved disentanglement, we note that the time-invariant information in this model is predefined to represent the background, rather than learned from the data automatically. Also this architecture is suitable for video sequences only, unlike our model which can be applied to any type of sequential data.

Very recent work (Hsu et al., 2017) introduced the *factorised hierarchical variational auto-encoder* (FHVAE) for unsupervised learning of disentangled representation of speech data. Given a speech sequence that has been partitioned into segments $\{\boldsymbol{x}_{1:T}^n\}_{n=1}^N$, FHVAE models the joint distribution of $\{\boldsymbol{x}_{1:T}^n\}_{n=1}^N$ and latent variables as follows:

$$p(\{\boldsymbol{x}_{1:T}^n, \boldsymbol{z}_1^n, \boldsymbol{z}_2^n\}, \boldsymbol{\mu}_2) = p(\boldsymbol{\mu}_2) \prod_{n=1}^N p(\boldsymbol{x}_{1:T}^n, \boldsymbol{z}_1^n, \boldsymbol{z}_2^n|\boldsymbol{\mu}_2),$$

$$p(\boldsymbol{x}_{1:T}^n, \boldsymbol{z}_1^n, \boldsymbol{z}_2^n|\boldsymbol{\mu}_2) = p(\boldsymbol{z}_1^n)p(\boldsymbol{z}_2^n|\boldsymbol{\mu}_2)p(\boldsymbol{x}_{1:T}^n|\boldsymbol{z}_1^n, \boldsymbol{z}_2^n).$$

Here the $\boldsymbol{z}_2^n$ variable has a hierarchical prior $p(\boldsymbol{z}_2^n|\boldsymbol{\mu}_2) = \mathcal{N}(\boldsymbol{\mu}_2, \sigma^2\mathbf{I})$, $p(\boldsymbol{\mu}_2) = \mathcal{N}(\mathbf{0}, \lambda\mathbf{I})$. The authors showed that by having different prior structures for $\boldsymbol{z}_1^n$ and $\boldsymbol{z}_2^n$, it allows the model to encode with $\boldsymbol{z}_2^n$ speech sequence-level

---

[1]For the full q encoder it also requires $\boldsymbol{f}^b \sim q(\boldsymbol{f}|\boldsymbol{x}_{1:T}^b)$.

attributes (e.g. pitch of a speaker), and other residual information with z_1^n. A discriminative training objective (see discussions in Section 4.2) is added to the variational lower-bound, which has been shown to further improve the quality of the disentangled representation. Our model can also benefit from the usage of hierarchical prior distributions, e.g. $\boldsymbol{f}^n \sim p(\boldsymbol{f}|\boldsymbol{\mu}_2), \boldsymbol{\mu}_2 \sim p(\boldsymbol{\mu}_2)$, and we leave the investigation to future work.

4. Experiments

We carried out experiments both on video data (Section 4.1) as well as speech data (Section 4.2). In both setups, we find strong evidence that our model learns an approximately disentangled representation that allows for conditional generation and feature swapping. We further investigated the efficiency for encoding long sequences with a stochastic transition model in Section 4.3. The detailed model architectures of the networks used in each experiment are reported in the appendix.

4.1. Video sequence: Sprites

We present an initial test of the proposed VAE architecture on a dataset of video game "sprites", i.e. animated cartoon characters whose clothing, pose, hairstyle, and skin color we can fully control. This dataset comes from an open-source video game project called Liberated Pixel Cup[2], and has been also considered in Reed et al. (2015); Mathieu et al. (2016) for image processing experiments. Our experiments show that static attributes such as hair color and clothing are well preserved over time for randomly generated videos.

Data and preprocessing. We downloaded and selected the online available sprite sheets[3], and organised them into 4 attribute categories (skin color, tops, pants and hairstyle) and 9 action categories (walking, casting spells and slashing, each with three viewing angles). In order to avoid a combinatorial explosion problem, each of the attribute categories contains 6 possible variants (see Figure 2), therefore it leads to $6^4 = 1296$ unique characters in total. We used 1000 of them for training/validation and the rest of them for testing. The resulting dataset consists of sequences with $T = 8$ frames of dimension 64×64. Note here we did not use the labels for training the generative model. Instead these labels on the data frames are used to train a classifier that is later deployed to produce quantitative evaluations on the VAE, see below.

Qualitative analysis. We start with a qualitative evaluation of our VAE architecture. Figure 3 shows both re-
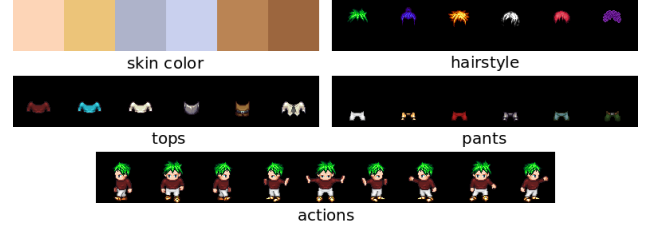

Figure 2. A visualisation of the attributes and actions used to generate the Sprite data set. See main text for details.

constructed as well as generated video sequences from our model. Each panel shows three video sequences with time running from left to right. Panel (a) shows parts of the original data from the test set, and (b) shows its reconstruction.

The sequences visualised in panel (c) are generated using $z_t \sim q(z_t|\boldsymbol{x}_t)$ but $\boldsymbol{f} \sim p(\boldsymbol{f})$. Hence, the dynamics are imposed by the encoder, but the identity is sampled from the prior. We see that panel (c) reveals the same motion patterns as (a), but has different character identities. Conversely, in panel (d) we take the identity from the encoder, but sample the dynamics from the prior. Panel (d) reveals the same characters as (a), but different motion patterns.

Panels (e) and (f) focus on feature swapping. In (e), the frames are constructed by computing $z_t \sim q(z_t|\boldsymbol{x}_t)$ on one input sequence but $\boldsymbol{f}$ encoded on another input sequence. These panels demonstrate that the encoder and the decoder have learned a factored representation for content and pose.

Panels (g) and (h) focus on conditional generation, showing randomly generated sequences that share the same $\boldsymbol{f}$ or $z_{1:T}$ samples from the prior. Thus, in panel (g) we see the same character performing different actions, and in (h) different characters performing the same motion. This again illustrates that the prior model disentangles the representation.

Quantitative analysis. Next we perform quantitative evaluations of the generative model, using a classifier trained on the labelled frames. Empirically, we find that the fully factorized and structured inference networks produce almost identical results here, presumably because in this dataset the object identity and pose information are truly independent. Therefore we only report results on the fully factorised q distribution case.

The first evaluation task considers reconstructing the test sequences with encoded $\boldsymbol{f}$ and randomly sampled z_t (in the same way as to produce panel (d) in Figure 3). Then we compare the classifier outputs on both the original frames and the reconstructed frames. If the character's identity is preserved over time, the classifier should produce identical probability vectors on the data frames and the reconstructed frames (denoted as $\boldsymbol{p}_{data}$ and $\boldsymbol{p}_{recon}$ respectively).

(a) random test data sequences   (b) reconstruction

(c) reconstruction with randomly sampled $\boldsymbol{f}$ (d) reconstruction with randomly sampled $\boldsymbol{z}_{1:T}$

(e) reconstruction with swapped encoding $\boldsymbol{f}$ (f) reconstruction with swapped encoding $\boldsymbol{z}_{1:T}$

(g) generated sequences with fixed $\boldsymbol{f}$ (h) generated sequences with fixed $\boldsymbol{z}_{1:T}$
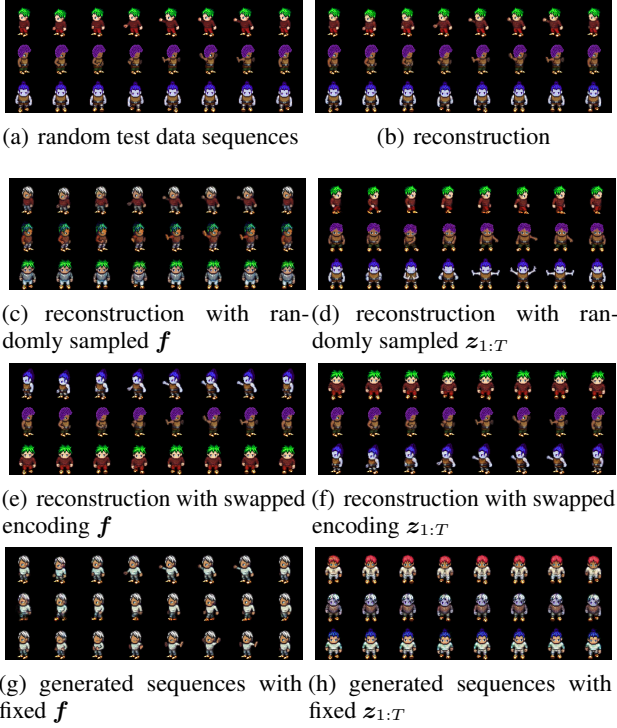
Figure 3. Visualisation of generated and reconstructed video sequences. See main text for discussions.

We evaluate the similarity between the original and reconstructed sequences both in terms of the disagreement of the predicted class labels $\max_i[\boldsymbol{p}_{recon}(i)] \neq \max_i[\boldsymbol{p}_{data}(i)]$ and the KL-divergence $\mathrm{KL}[\boldsymbol{p}_{recon}||\boldsymbol{p}_{data}]$. We also compute the two metrics on the action predictions using reconstructed sequences with randomised $\boldsymbol{f}$ and inferred $\boldsymbol{z}_t$. The results in Table 1 indicate that the learned representation is indeed factorised. For example, in the fix-$\boldsymbol{f}$ generation test, only 4% out of 296×9 data-reconstruction frame pairs contain characters whose generated skin color differs from the rest, where in the case of hairstyle preservation the disagreement rate is only 0.06%. The KL metric is also much smaller than the KL-divergence $\mathrm{KL}[\boldsymbol{p}_{random}||\boldsymbol{p}_{data}]$ where $\boldsymbol{p}_{random} = (1/N_{\text{class}}, ..., 1/N_{\text{class}})$, indicating that our result is significant.

In the second evaluation, we test whether static attributes of generated sequences, such as clothing or hair style, are preserved over time. We sample 200 video sequences from the generator, using the same $\boldsymbol{f}$ but different latent dynamics $\boldsymbol{z}_{1:T}$. We use the trained classifier to predict both the attributes and the action classes for each of the generated frames. Results are shown in Figure 4(a), where we plot the prediction of the classifiers for each frame over time. For example, the trajectory curve in the "skin color" panel in Figure 4(a) corresponds to the skin color attribute classification results for frames $\boldsymbol{x}_{1:T}$ of a generated video sequence. We repeat this process 5 times with different $\boldsymbol{f}$ samples,

Table 1. Averaged classification disagreement and KL similarity measures for our model on Sprite data. Note here KL-recon = $\mathrm{KL}[\boldsymbol{p}_{recon}||\boldsymbol{p}_{data}]$ and KL-random = $\mathrm{KL}[\boldsymbol{p}_{random}||\boldsymbol{p}_{data}]$.

attributes	disagreement	KL-recon	KL-random
skin colour	3.98%	0.7847	8.8859
pants	1.82%	0.3565	8.9293
tops	0.34%	0.0647	8.9173
hairstyle	0.06%	0.0126	8.9566
action	8.11%	0.9027	13.7510

where each $\boldsymbol{f}$ corresponds to one color.

It becomes evident that those lines with the same color are clustered together, confirming that $\boldsymbol{f}$ mainly controls the generation of time-invariant attributes. Also, most character attributes are preserved over time, e.g. for the attribute "tops", the trajectories are mostly straight lines. However, some of the trajectories for the attributes drift away from the majority class. We conjecture that this is due of the mass-covering behaviour of (approximate) maximum likelihood training, which makes the trained model generate characters that do not exist in the dataset. Indeed the middle row of panel (c) in Figure 3 contains a character with an unseen hairstyle, showing that our model is able to generalise beyond the training set. On the other hand, the sampling process returns sequences with diverse actions as depicted in the action panel, meaning that $\boldsymbol{f}$ contains little information regarding the video dynamics.

We performed similar tests on sequence generations with shared latent dynamics $\boldsymbol{z}_{1:T}$ but different $\boldsymbol{f}$, shown in Figure 4(b). The experiment is repeated 5 times as well, and again trajectories with the same color encoding correspond to videos generated with the same $\boldsymbol{z}_{1:T}$ (but different $\boldsymbol{f}$). Here we also observe diverse trajectories for the attribute categories. In contrast, the characters' actions are mostly the same. These two test results again indicate that the model has successfully learned disentangled representations of character identities and actions. Interestingly we observe multi-modalities in the action domain for the generated sequences, e.g. the trajectories in the action panel of Figure 4(b) are jumping between different levels. We also visualise in Figure 5 generated sequences of the "turning" action that is not present in the dataset. It again shows that the trained model generalises to unseen cases.

4.2. Speech data: TIMIT

We also experiment on audio sequence data. Our disentangled representation allows us to convert speaker identities into each other while conditioning on the content of the speech. We also show that our model gives rise to speaker verification, where we outperform a recent probabilistic baseline model.
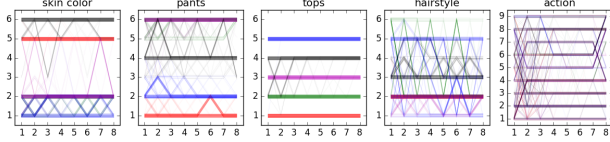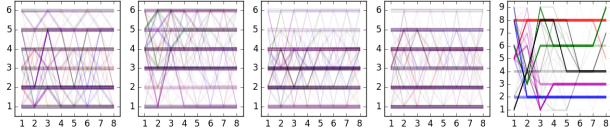
(a) Trajectory plots on the generated sequences with shared $\boldsymbol{f}$.

(b) Trajectory plots on the generated sequences with shared $\boldsymbol{z}_{1:T}$.

Figure 4. Classification test on the generated video sequences with shared $\boldsymbol{f}$ (top) or shared $\boldsymbol{z}_{1:T}$ (bottom), respectively. The experiment is repeated 5 times and depicted by different color coding. The x and y axes are time and the class id of the attributes, respectively.

Figure 5. Visualising multi-modality in action space. In this case the characters turn from left to right, and this action sequence is not observed in data.

Data and preprocessing. The TIMIT data (Garofolo et al., 1993) contains broadband 16kHz recordings of phonetically-balanced read speech. A total of 6300 utterances (5.4 hours) are presented with 10 sentences from each of the 630 speakers (70% male and 30% female). We follow Hsu et al. (2017) for data pre-processing: the raw speech waveforms are first split into sub-sequences of 200ms, and then preprocessed with sparse fast Fourier transform to obtain a 200 dimensional log-magnitude spectrum, computed every 10ms. This implies $T = 20$ for the observation $\boldsymbol{x}_{1:T}$.

Qualitative analysis. We perform voice conversion experiments to demonstrate the disentanglement of the learned representation. The goal here is to convert male voice to female voice (and vice versa) with the speech content being preserved. Assuming that $\boldsymbol{f}$ has learned the representation of speaker's identity, the conversion can be done by first encoding two sequences $\boldsymbol{x}_{1:T}^{\text{male}}$ and $\boldsymbol{x}_{1:T}^{\text{female}}$ with q to obtain representations $\{\boldsymbol{f}^{\text{male}}, z_{1:T}^{\text{male}}\}$ and $\{\boldsymbol{f}^{\text{female}}, \boldsymbol{z}_{1:T}^{\text{female}}\}$, then construct the converted sequence by feeding $\boldsymbol{f}^{\text{female}}$ and $\boldsymbol{z}_{1:T}^{\text{male}}$ to the decoder $p(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{f})$. Figure 6 shows the reconstructed spectrogram after the swapping process of the $\boldsymbol{f}$ features. We also provide the reconstructed speech waveforms using the Griffin-Lim algorithm (Griffin & Lim, 1984) in the appendix.

(a) female speech (original)

(b) female to male

(c) male speech (original)
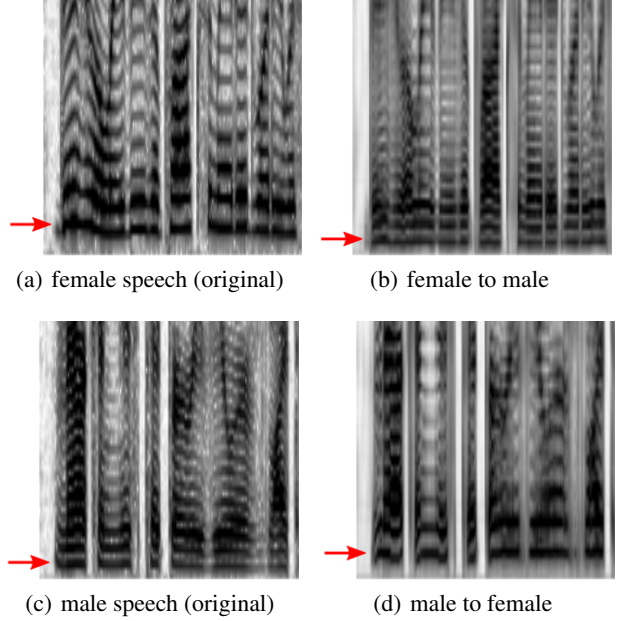

(d) male to female

Figure 6. Visualising the spectrum of the reconstructed speech sequences. Here we show the spectrogram for the first 2000ms, with horizontal axis denoting time and the vertical axis denoting frequencies. The red arrow points to the first harmonics which indicates the fundamental frequency of the speech signal.

The experiments show that the harmonics of the converted speech sequences shifted to higher frequency in the "male to female" test and vice versa. Also the pitch (the red arrow in Figure 6 indicating the fundamental frequency, i.e. the first harmonic) of the converted sequence (b) is close to the pitch of (c), same as for the comparison between (d) and (a). By an informal listening test of the speech sequence pairs (a, d) and (b, c), we confirm that the speech content is preserved. These results show that our model is successfully applied to speech sequences for learning disentangled representations.

Quantitative analysis. We further follow Hsu et al. (2017) to use speaker verification for quantitative evaluation. Speaker verification is the process of verifying the claimed identity of a speaker, usually by comparing the "features" $\boldsymbol{w}^{\text{test}}$ of the test utterance $\boldsymbol{x}_{1:T_1}^{\text{test}}$ with those of the target utterance $\boldsymbol{x}_{1:T_2}^{\text{target}}$ from the claimed identity. The claimed identity is confirmed if the cosine similarity $\cos(\boldsymbol{w}^{\text{test}}, \boldsymbol{w}^{\text{target}})$ is grater than a given threshold ϵ (Dehak et al., 2009). By varying $\epsilon \in [0, 1]$, we report the verification performance in terms of equal error rate (EER), where the false rejection rate equals the false acceptance rate.

The extraction of the "features" is crucial for the performance of this speaker verification system. Given a speech sequence containing N segments $\{\boldsymbol{x}_{1:T}^{(n)}\}_{n=1}^{N}$, we constructed two types of "features", one by computing $\boldsymbol{\mu}_{\boldsymbol{f}}$ as the mean

of $q(\boldsymbol{f}^{(n)}|\boldsymbol{x}_{1:T}^{(n)})$ across the segments, and the other by extracting the mean $\boldsymbol{\mu}_{\boldsymbol{z}_t}$ of $q(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})$ and averaging them across both time T and segments. In formulas,

$$\boldsymbol{\mu}_{\boldsymbol{f}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\mu}_{\boldsymbol{f}^n}, \quad \boldsymbol{\mu}_{\boldsymbol{f}^n} = \mathbb{E}_{q(\boldsymbol{f}^n|\boldsymbol{x}_{1:T}^n)}[\boldsymbol{f}^n],$$

$$\boldsymbol{\mu}_{\boldsymbol{z}} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N}\boldsymbol{\mu}_{\boldsymbol{z}_t^n}, \quad \boldsymbol{\mu}_{\boldsymbol{z}_t^n} = \mathbb{E}_{q(\boldsymbol{z}_t^n|\boldsymbol{x}_{1:T}^n)}[\boldsymbol{z}_t^n].$$

We also include two baseline results from Hsu et al. (2017): one used the i-vector method (Dehak et al., 2011) for feature extraction, and the other one used $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ (analogous to $\boldsymbol{\mu}_{\boldsymbol{z}}$ and $\boldsymbol{\mu}_{\boldsymbol{f}}$ in our case) from a trained FHVAE model on Mel-scale filter bank (FBank) features.

The test data were created from the test set of TIMIT, containing 24 unique speakers and 18,336 pairs for verification. Table 2 presents the EER results of the proposed model and baselines.[4] It is clear that the $\boldsymbol{\mu}_{\boldsymbol{f}}$ feature performs significantly better than the i-vector method, indicating that the $\boldsymbol{f}$ variable has learned to represent a speaker's identity. On the other hand, using $\boldsymbol{\mu}_{\boldsymbol{z}}$ as the features returns considerably worse EER rates compared to the i-vector method and $\boldsymbol{\mu}_{\boldsymbol{f}}$ feature. This is good, as it indicates that the $\boldsymbol{z}$ variables contain less information about the speaker's identity, again validating the success of disentangling time-variant and time-independent information. Note that the EER results for $\boldsymbol{\mu}_{\boldsymbol{z}}$ get worse when using the full q encoder, and in the 64 dimensional feature case the verification performance of $\boldsymbol{\mu}_{\boldsymbol{f}}$ improves slightly. This also shows that for real-world data it is useful to use a structured inference network to further improve the quality of disentangled representation.

Our results are competitive with (or slightly better than) the FHVAE results ($\alpha = 0$) reported in Hsu et al. (2017). The better results for FHVAE ($\alpha = 10$) is obtained by adding a discriminative training objective (scaled by α) to the variational lower-bound. In a nutshell, the time-invariant information in FHVAE is encoded in a latent variable $\boldsymbol{z}_2^n \sim p(\boldsymbol{z}_2^n|\boldsymbol{\mu}_2)$, and the discriminative objective encourages $\boldsymbol{z}_2^n$ encoded from a segment of one sequence to be close to the corresponding $\boldsymbol{\mu}_2$ while far away from $\boldsymbol{\mu}_2$ of other sequences. However, we do not test this idea here because (1) our goal is to demonstrate that the proposed architecture is a minimalistic framework for learning disentangled representations of sequential data; (2) this discriminative objective is specifically designed for hierarchical VAE, and in general the assumption behind it might not always be true (consider encoding two speech sequences coming from the same speaker). Similar ideas for discriminative training have been considered in e.g. Mathieu et al. (2016), but that discriminative objective can only be applied

[4] Hsu et al. (2017) did not provide the EER results for $\alpha = 0$ and $\boldsymbol{\mu}_1$ in the 16 dimension case.

Table 2. Speaker verification errors, comparing the FHVAE with our approach. Static information is encoded in $\boldsymbol{\mu}_2$ / $\boldsymbol{\mu}_{\boldsymbol{f}}$ and dynamic information in $\boldsymbol{\mu}_1$ / $\boldsymbol{\mu}_{\boldsymbol{z}}$ for the FHVAE / our approach, respectively. Large errors are expected when predicting based on $\boldsymbol{\mu}_1$ / $\boldsymbol{\mu}_{\boldsymbol{z}}$, and small errors for $\boldsymbol{\mu}_2$ / $\boldsymbol{\mu}_{\boldsymbol{f}}$, respectively (see main text). Our data-agnostic approach compares favourably.

model	feature	dim	EER
-	i-vector	200	9.82%
FHVAE ($\alpha = 0$)	$\boldsymbol{\mu}_2$	16	5.06%
FHVAE ($\alpha = 10$)	$\boldsymbol{\mu}_2$	32	2.38%
	$\boldsymbol{\mu}_1$	32	22.47%
factorised q	$\boldsymbol{\mu}_{\boldsymbol{f}}$	16	4.78%
	$\boldsymbol{\mu}_{\boldsymbol{z}}$	16	17.84%
factorised q	$\boldsymbol{\mu}_{\boldsymbol{f}}$	64	4.94%
	$\boldsymbol{\mu}_{\boldsymbol{z}}$	64	17.49%
full q	$\boldsymbol{\mu}_{\boldsymbol{f}}$	16	5.64%
	$\boldsymbol{\mu}_{\boldsymbol{z}}$	16	19.20%
full q	$\boldsymbol{\mu}_{\boldsymbol{f}}$	64	4.82%
	$\boldsymbol{\mu}_{\boldsymbol{z}}$	64	18.89%

to two sequences that are known to entail different time-invariant information (e.g. two sequences with different labels), which implicitly uses supervisions. Nevertheless, a better design for the discriminative objective without supervision can further improve the disentanglement of the learned representations, and we leave it to future work.

4.3. Comparing stochastic & deterministic dynamics

Lastly, although not a main focus of the paper, we show that the usage of a *stochastic* transition model for the prior leads to more realistic dynamics of the generated sequence. For comparison, we consider another class of models:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}, \boldsymbol{f}) = p(\boldsymbol{f})p(\boldsymbol{z})\prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{z}, \boldsymbol{f}).$$

The parameters of $p(\boldsymbol{x}_t|\boldsymbol{z}, \boldsymbol{f})$ are defined by a neural network $\text{NN}(\boldsymbol{h}_t, \boldsymbol{f})$, with $\boldsymbol{h}_t$ computed by a *deterministic* RNN conditioned on $\boldsymbol{z}$. We experiment with two types of deterministic dynamics. The first model uses an LSTM with $\boldsymbol{z}$ as the initial state: $\boldsymbol{h}_0 = \boldsymbol{z}, \boldsymbol{h}_t = \text{LSTM}(\boldsymbol{h}_{t-1})$. In later experiments we refer this dynamics as **LSTM-f** as the latent variable $\boldsymbol{z}$ is forward propagated in a deterministic way. The second one deploys an LSTM conditioned on $\boldsymbol{z}$ (i.e. $\boldsymbol{h}_0 = \boldsymbol{0}, \boldsymbol{h}_t = \text{LSTM}(\boldsymbol{h}_{t-1}, \boldsymbol{z})$), therefore we refer it as **LSTM-c**. This is identical to the transition dynamics used in the FHVAE model (Hsu et al., 2017). For comparison, we refer to our model as the 'stochastic' model (Eq. 1).

The LSTM models encodes temporal information in a global latent variable $\boldsymbol{z}$. Therefore, small differences/errors in $\boldsymbol{z}$ will accumulate over time, which may result in unrealistic long-time dynamics. In contrast, the stochastic model (Eq. 1) keeps track of the time-varying aspects of $\boldsymbol{x}_t$ in $\boldsymbol{z}_t$ for every t, making the reconstruction to be time-local and

(a) data for reconstruction   (b) data for prediction

(c) reconstruction (stochastic)   (d) prediction (stochastic)

(e) reconstruction (LSTM-f)   (f) prediction (LSTM-f)

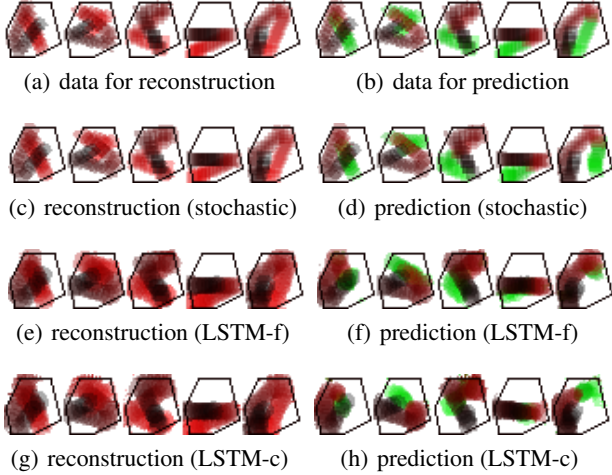(g) reconstruction (LSTM-c)   (h) prediction (LSTM-c)

Figure 7. Predicted and reconstructed video sequences. The videos are shown as single images, with colour intensity (starting from black) representing the incremental sequence index ('stochastic' is proposed). The missing/predicted frames are shown in green.

therefore much easier. Therefore, the stochastic model is better suited if the sequences are long and complex. We give empirical evidence to support this claim.

Data preprocessing & hyper-parameters. We follow Fraccaro et al. (2017) to simulate video sequences of a ball (or a square) bouncing inside an irregular polygon using Pymunk.[5] The irregular shape was chosen because it induces chaotic dynamics, meaning that small deviations from the initial position and velocity of the ball will create exponentially diverging trajectories at long times. This makes memorizing the dynamics of a prototypical sequence challenging. We randomly sampled the initial position and velocity of the ball, but did not apply any force to the ball, except for the fully elastic collisions with the walls. We generated 5,000 sequences in total (1000 for test), each of them containing $T = 30$ frames with a resolution of 32×32. For the deterministic LSTMs, we fix the dimensionality of z_t to 64, and set h_t and the LSTM internal states to be 512 dimensions. The latent variable dimensionality of the stochastic dynamics is $\dim(z_t) = 16$.

Qualitative & quantitative analyses. We consider both reconstruction and missing data imputation tasks for the learned generative models. For the latter and for $T = 30$, the models observe the first $t < T$ frames of a sequence and predict the remaining $T - t$ frames using the prior dynamics. We visualise in Figure 7 the ground truth, reconstructed, and predicted sequences ($t = 20$) from all models.

---

[5]http://www.pymunk.org/en/latest/. For simplicity we disabled rotation of the square when hitting the wall, by setting the inertia to infinity.
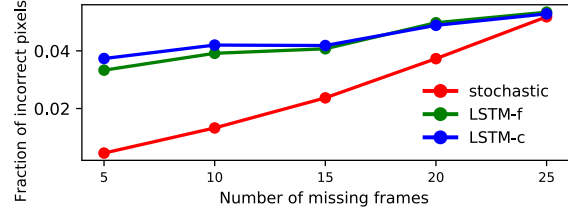

Figure 8. Reconstruction error of different models as a function of the number of consecutive missing frames (see main text). Lower values are better. 'stochastic' refers to the proposed approach.

We further consider average fraction of incorrectly reconstructed/predicted pixels as a quantitative metric, to evaluate how well the ground-truth dynamics is recovered given consecutive missing frames. The result is reported in Figure 8. The stochastic model outperforms the deterministic models both qualitatively and quantitatively. The shape of the ball is better preserved over time, and the trajectories are more physical. This explains the lower errors of the stochastic model, and the advantage is significant when the number of missing frames is small.

Our experiments give evidence that the stochastic model is better suited to modelling long, complex sequences when compared to the deterministic dynamical models. We expect that a better design for the stochastic transition dynamics, e.g. by combining deep neural networks with well-studied linear dynamical systems (Krishnan et al., 2015; Fraccaro et al., 2016; Karl et al., 2016; Johnson et al., 2016; Krishnan et al., 2017; Fraccaro et al., 2017), can further enhance the quality of the learned representations.

5. Conclusions and outlook

We presented a minimalistic generative model for learning disentangled representations of high-dimensional time series. Our model consists of a global latent variable for content features, and a stochastic RNN with time-local latent variables for dynamical features. The model is trained using standard amortized variational inference. We carried out experiments both on video and audio data. Our approach allows us to perform full and conditional generation, as well as feature swapping, such as voice conversion and video content manipulation. We also showed that a stochastic transition model generally outperforms a deterministic one.

Future work may investigate whether a similar model applies to more complex video and audio sequences. Also, disentangling may further be improved by additional cross-entropy terms, or discriminative training. A promising avenue of research is to explore the usage of this architecture for neural compression. An advantage of the model is that it separates dynamical from static features, allowing the latent space for the dynamical part to be low-dimensional.

Acknowledgements

References

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

Bouchacourt, D., Tomioka, R., and Nowozin, S. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. *arXiv preprint arXiv:1705.08841*, 2017.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Dehak, N., Dehak, R., Kenny, P., Brümmer, N., Ouellet, P., and Dumouchel, P. Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification. In *Tenth Annual conference of the international speech communication association*, 2009.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

Deng, Z., Navarathna, R., Carr, P., Mandt, S., Yue, Y., Matthews, I., and Mori, G. Factorized variational autoencoders for modeling audience reactions to movies. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 6014–6023. IEEE, 2017.

Denton, E. and Birodkar, V. Unsupervised learning of disentangled representations from video. *arXiv preprint arXiv:1705.10915*, 2017.

Fabius, O. and van Amersfoort, J. R. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pp. 3604–3613, 2017.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. *Web Download. Philadelphia: Linguistic Data Consortium*, 1993.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Graves, A. and Schmidhuber, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

Griffin, D. and Lim, J. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hsu, W.-N., Zhang, Y., and Glass, J. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pp. 1876–1887, 2017.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pp. 2946–2954, 2016.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

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

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

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. 2017.

Mathieu, M. F., Zhao, J. J., Zhao, J., Ramesh, A., Sprechmann, P., and LeCun, Y. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pp. 5040–5048, 2016.

Reed, S. E., Zhang, Y., Zhang, Y., and Lee, H. Deep visual analogy-making. In *Advances in neural information processing systems*, pp. 1252–1260, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Siddharth, N., Paige, B., de Meent, V., Desmaison, A., Wood, F., Goodman, N. D., Kohli, P., Torr, P. H., et al. Learning disentangled representations with semi-supervised deep generative models. *arXiv preprint arXiv:1706.00400*, 2017.

Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017.

Vondrick, C., Pirsiavash, H., and Torralba, A. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.
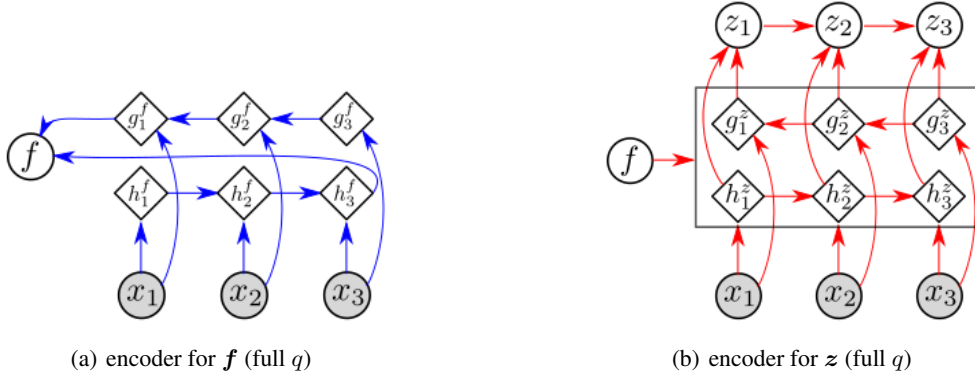
(a) encoder for $\boldsymbol{f}$ (full q)

(b) encoder for $\boldsymbol{z}$ (full q)

Figure 9. A graphical model visualisation of the generator and the encoder.

A. Computation graph for the full q inference network

In Figure 9 we show the computation graph of the full q inference framework. The inference model first computes the mean and variance parameters of $q(\boldsymbol{f}|\boldsymbol{x}_{1:T})$ with a bi-directional LSTM (Graves & Schmidhuber, 2005) and samples $\boldsymbol{f}$ from the corresponding Gaussian distribution (see Figure (a)). Then $\boldsymbol{f}$ and $\boldsymbol{x}_{1:T}$ are fed into another bi-directional LSTM to compute the hidden state representations $\boldsymbol{h}_t^z$ and $\boldsymbol{g}_t^z$ for the $\boldsymbol{z}_t$ variables (see Figure (b)), where at each time-step both LSTMs take $[\boldsymbol{x}_t, \boldsymbol{f}]$ as the input and update their hidden and internal states. Finally the parameters of $q(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}, \boldsymbol{f})$ is computed by a simple RNN with input $[\boldsymbol{h}_t^z, \boldsymbol{g}_t^z]$ at time t.

B. Sound files for the speech conversion test

We provide sound files to demonstrate the conversion of female/male speech sequences at `https://drive.google.com/file/d/1zpiZJNjGWw9pGPYVxgSeoipiZdeqHatY/view?usp=sharing`. Given a spectrum (magnitude information), the sound waveform is reconstructed using the Griffin-Lim algorithm (Griffin & Lim, 1984), which initialises the phase randomly, then iteratively refine the phase information by looping the SFFT/inverse SFFT transformation until convergence or reaching some stopping criterion. We note that the sound quality can be further improved by e.g. conjugate gradient methods. Also we found in general it is more challenging to convert female speech to male speech than the other way around, which is also observed by (Hsu et al., 2017).

We also note here that the phase information is not modelled in our experiments, nor in the FHVAE tests. First, as phase is a circular variable ranging from $[-\pi, \pi]$, Gaussian distribution is inappropriate, and instead a von Mises distribution is required. However, fast computation of the normalising constant of a von Mises distribution – which is a Bessel function – remains a challenging task, let alone differentiation and optimisation of the concentration parameters.

C. Network architecture

Sprite. The prior dynamics $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$ is Gaussian with parameters computed by an LSTM (Hochreiter & Schmidhuber, 1997). Then $\boldsymbol{x}_t$ is generated by a deconvolutional neural network, which first transforms $[\boldsymbol{z}_t, \boldsymbol{f}]$ with a one hidden-layer MLP, then applies 4 deconvolutional layers with 256 channels and up-sampling. We use the ℓ_2 loss for the likelihood term, i.e. $\log p(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{f}) = -\frac{1}{2}||\boldsymbol{x}_t - \text{NN}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \boldsymbol{f})||_2^2 + const.$

For the inference model, we first use a convolutional neural network, with a symmetric architecture to the deconvolutional one, to extract visual features. Then $q(\boldsymbol{f}|\boldsymbol{x}_{1:T})$ is also a Gaussian distribution parametrised by an LSTM and depends on the entire sequence of these visual features. For the factorised q encoder, $q(\boldsymbol{z}_t|\boldsymbol{x}_t)$ is also Gaussian parametrised by a one-hidden layer MLP taking the visual feature of $\boldsymbol{x}_t$ as input. The dimensions of $\boldsymbol{f}$ and $\boldsymbol{z}_t$ are 256 and 32, respectively, and the hidden layer sizes are fixed to 512.

TIMIT. We use almost identical architecture as in the Sprite data experiment, except that the likelihood term $p(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{f})$ is defined as Gaussian with mean and variance determined by a 2-hidden-layer MLP taking both $\boldsymbol{z}_t$ and $\boldsymbol{f}$ as inputs. The dimensions of $\boldsymbol{f}$ and $\boldsymbol{z}_t$ are 64 if not specifically stated, and the hidden layer sizes are fixed to 256.

For the full q inference model we use the architecture visualised in Figure 9. Again the bi-LSTM networks take the features of $\boldsymbol{x}_t$ as inputs, where those features are extracted using a one hidden-layer MLP.

Bouncing ball. We use an RNN (instead of an LSTM as in previous experiments) to parametrise the stochastic prior dynamics of our model, and set the dimensionality of $\boldsymbol{z}_t$ to be 16. For the deterministic models we set $\boldsymbol{z}$ to be 64 dimensional. We use a 64 dimensional $\boldsymbol{f}$ variable and Bernoulli likelihood for all models.

For inference models, we use the full q model for the stochastic dynamics case. For the generative models with deterministic dynamics, we also use bi-LSTMs of the same architecture to infer the parameters of $q(\boldsymbol{f}|\boldsymbol{x}_{1:T})$ and $q(\boldsymbol{z}|\boldsymbol{x}_{1:T})$. Again a convolutional neural network is deployed to compute visual features for LSTM inputs.

All models share the same architecture of the (de-)convolution network components. The deconvolution neural network has 3 deconvolutional layers with 64 channels and up-sampling. The convolutional neural network for the encoder has a symmetric architecture to the deconvolution one. The hidden layer sizes in all networks are fixed to 512.