Modeling Sequential Data with VAEs: Disentanglement and Connections to Variational Inference

Timothy Leplae-Arthur [1] **Eshaan Nichani** [2]

Abstract

In this project we consider the problem of modeling sequential data, such as a video, and producing a disentangled representation. Such a representation should separate out the time-dependent and time-independent components. To accomplish this we use a Sequential Autoencoder, which relies on a probabilistic graphical model to encourage disentanglement. We run experiments on synthetic data to test the effectiveness of the sequential autoencoder on modeling data, and analyze disentanglement by running our model on a data set of image sequences, showing that it ultimately is able to produce a disentangled representation.

1. Introduction

An important problem in machine learning is finding a low-dimensional representation of some high-dimensional data, such as an image or video. One particularly property that we may want of such a representation is that it is *disentangled*, i.e. that each of its features are roughly independent and represent distinct, semantically meaningful components.

Rather than thinking about disentanglement in an abstract sense, it is easier to restrict our focus to sequential data, such as videos or speech, in which we have a sequence of data over various points in time. One specific example of disentanglement for this scenario is separating the time-independent, global features of the sequence from the time-dependent, dynamical features of the sequence. For example, for a video of a person, we may wish to produce a set of latent variables which describes the overall content of the scene such as the person and the setting, as well as a set of latent variables which describes how the person moves. Such an encoding may have applications in video compression or style transfer, among others.

Recent work in producing disentangled representations involves the use of Variational Autoencoders (VAE) (Kingma & Welling, 2014). A VAE uses ideas from Variational Inference to formulate the loss function of a neural network. A VAE consists of a stochastic encoder and decoder which are neural networks, outputting parameters to a probability distribution rather than a deterministic value; a diagram of one is shown in Figure 1[1]. Many papers experiment with modifying the objective function in order to achieve disentanglement (Higgins et al., 2017; Burgess et al., 2017). We, however, follow the model of (Li & Mandt, 2018), where disentanglement is achieved through the design of a probabilistic graphical model with a special structure, which they term a Sequential Autoencoder.

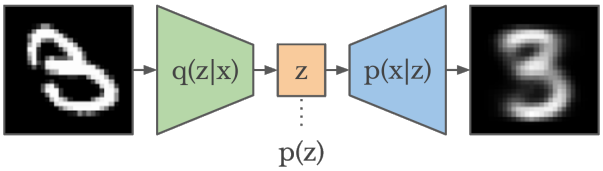

Figure 1. A diagram of a Variational Autoencoder with input x and latent variables z. This VAE is trained on the MNIST dataset.

However, variational inference literature has shown that variational inference can struggle when applied to time-series data (Turner & Sahani, 2011). Since the VAE can be interpreted in terms of variational Expectation-Maximization, we may expect the sequential autoencoder to suffer from similar problems. We investigate whether problems may arise when applying the sequential autoencoder to time-series data.

Our paper is organized as follows. In Section 2 we define the Sequential Autoencoder by describing our encoder and decoder as generative models, and formulate our loss function. In Section 3 we present an interesting interpretation of VAEs as variational EM, and replicate the experiments of (Turner & Sahani, 2011) to assess whether the Sequential Autoencoder suffers from bias. In Section 4 we discuss metrics for measuring disentanglement, and in Section 5 we replicate the experiments run in (Li & Mandt, 2018) on video data. Finally, in Section 6 we modify the objective function via the β-VAE (Higgins et al., 2017), interpret it in

[1]tkleplae@mit.edu [2]eshnich@mit.edu.

[1]https://danijar.com/building-variational-auto-encoders-in-tensorflow/

the context of Bayesian inference, and rerun our video data experiments.

2. Model and Objective

We are given a sequence of data $\mathbf{x} = (x_1, \ldots, x_T)$. This data is temporal, say some video, where each x_i is a frame. We want to find a latent representation of our data which consists of some time-independent component f, as well as some time-dependent components $\mathbf{z} = (z_1, \ldots, z_T)$. To accomplish this we use the Sequential Autoencoder defined in (Li & Mandt, 2018). The sequential autoencoder defines a generative model and inference model which structurally forces f to govern the time-independent components and z the time dependent ones.

The decoder is the generative model with joint distribution

$$p_\theta(f, z_{1:T}, x_{1:T}) = p(f) \prod_{t=1}^{T} p_\theta(z_t \mid \mathbf{z}_{<t}) p_\theta(x_t \mid z_t, f).$$

Here, we let $p(f)$ be the standard normal distribution, with the prior dynamics $p_\theta(z_t \mid \mathbf{z}_{<t})$, a Gaussian distribution with means and variances learned by a recurrent neural network (RNN) with inputs $\mathbf{z}_{<t}$. We call the model for $p_\theta(x_t \mid z_t, f)$ our decoder, and let it be a unit-variance Gaussian with means learned by a neural network with input (z_t, f).

Our encoder, $q_\phi(\mathbf{z}, f \mid \mathbf{x})$, should approximate our true posterior. To facilitate training and encourage disentanglement, we restrict the possible q to be those which factorize as:

$$q_\phi(\mathbf{z}, f \mid \mathbf{x}) = q_\phi(f \mid \mathbf{x}) \prod_{t=1}^{T} q_\phi(z_t \mid x_t).$$

Here, the encoder for f, $q_\phi(f \mid \mathbf{x})$, is a Gaussian with means and variances learned by an RNN with inputs $\mathbf{x}$. The encoder for z_t, $q_\phi(z_t \mid x_t)$, is a Gaussian with means and variances outputted by a neural network with input x_t. Note that we use the same network for $q_\phi(z_t \mid x_t)$ regardless of the value of t, and also the same network for $p_\theta(x_t \mid z_t, f)$ independent of t. This is because we want the encoding of a frame to be independent of its place in the sequence of frames. Graphical models of our encoder and decoder are shown in Figure 2, which is taken from (Li & Mandt, 2018).

The objective of a VAE is to maximize the ELBO with respect to the neural network parameters ϕ, θ, averaged over all our data points. This quantity is

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}, f \mid \mathbf{x}^{(i)})} \left[\log \frac{p_\theta(\mathbf{x}^{(i)}, \mathbf{z}, f)}{q_\phi(\mathbf{z}, f \mid \mathbf{x}^{(i)})} \right].$$

We simplify this expression further than was done in (Li & Mandt, 2018) to facilitate our implementation. Defining our loss to be the negative of the ELBO, we get that the loss for a single sequence of data is

$$\begin{aligned} L(\theta, \phi; \mathbf{x}) &= -\mathbb{E}_{q_\phi(\mathbf{z}, f \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z}, f)] \\ &\quad + KL(q_\phi(\mathbf{z}, f \mid \mathbf{x}) \| p(f) p_\theta(\mathbf{z})) \\ &= -\mathbb{E}_{q_\phi(\mathbf{z}, f \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z}, f)] \\ &\quad + KL(q_\phi(f \mid \mathbf{x}) \| p(f)) \\ &\quad + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t} \mid \mathbf{x})}[KL(q_\phi(z_t \mid x_t) \| p_\theta(z_t \mid \mathbf{z}_{<t}))] \end{aligned}$$

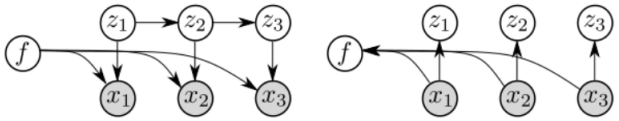

Figure 2. Probabilistic graphical models showing the decoder p (left) and encoder q (right) for our model

Given our data point $\mathbf{x}$, we can run the encoder and sample $\mathbf{z}, f$ from $q_\phi(\mathbf{z}, f \mid \mathbf{x})$. We then run the decoder to get $\mathbf{x}_{recon}$, the mean of the generative distribution $p_\theta(\mathbf{x} \mid \mathbf{z}, f)$. We can use this to obtain a Monte Carlo estimate of the first term, as $\log p_\theta(\mathbf{x} \mid \mathbf{z}, f)$ is just the mean-squared error between $\mathbf{x}_{recon}$ and $\mathbf{x}$. Since we know both the means and variances of the parameters of $q_\phi(f \mid \mathbf{x})$, and $p(f)$ is a unit Gaussian, we can compute their KL divergence exactly to get the second term. Finally, we use the sampled $\mathbf{z}_{1:t-1}$ to compute the parameters for the distribution $p_\theta(z_t | \mathbf{z}_{<t})$. Since this distribution, along with $q_\phi(z_t \mid x_t)$ are both Gaussian, we can compute their KL divergence exactly, giving us a Monte Carlo estimate of the last term.

Though based off the methodology of (Li & Mandt, 2018), our models were coded up from scratch in pytorch and can be found in our github[2].

3. Are Sequential Autoencoders Biased?

Before we explore the effectiveness of a sequential autoencoder in disentangling data, it is useful to ask ourselves whether it makes sense to model sequential data with a VAE to begin with. To answer this question we can analyze the VAE in the context of Bayesian inference, as variational Expectation-Maximization.

3.1. VAE is vEM

Consider a set of observations x which depend on latent variables z and parameters θ. Our goal is to find the set of parameters θ which maximize the log-likelihood $\log p(x \mid$

[2] https://github.com/eshnich/6.435_project

θ). Oftentimes, this problem is intractable; instead, we can consider the evidence lower bound (ELBO), defined as

$$\begin{aligned}\mathcal{F}(q,\theta) &= \mathbb{E}_{q(z)}\left[\log \frac{p(x,z \mid \theta)}{q(z)}\right] \\ &\leq \log p(x \mid \theta).\end{aligned}$$

This quantity is optimized when the distribution $q(z)$ is equivalent to the posterior over the latents $p(z \mid x, \theta)$. Calculating such a posterior, however, is intractable, and we instead restrict q to a smaller family of distributions characterized by ϕ, which we call the variational parameters. In this case, we can write out the ELBO in terms of ϕ and θ as $\mathcal{F}(\phi, \theta)$. Variational Expectation-Maximization (vEM) is a process which leverages this variational approximation (Dempster et al., 1977), by alternating between maximizing $\mathcal{F}$ with respect to ϕ and θ. This gives a local optimum of our approximation, which we hope is close to the θ yielding the maximum likelihood.

Variational Autoencoders also seek to maximize the ELBO

$$\max_{\phi,\theta} \mathcal{F}(\phi,\theta) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right].$$

In fact, this objective function and its parameters can be interpreted precisely in terms of vEM. The generative model for our observations is a neural network with the parameters θ. We seek to find the set of parameters which maximizes the likelihood, i.e. train our neural network to best match our data. Our variational approximation is restricted to the set of distributions which can be learned by a neural network with parameters ϕ; optimizing our objective in terms of ϕ is then equivalent to training this neural network. Rather than alternating maximizing ϕ and θ in vEM, VAEs use gradient descent to simultaneously make updates to both ϕ and θ. Therefore a VAE accomplishes precisely the same goal as vEM: finding the parameter values θ to our neural network which approximately maximize the likelihood of the data.

3.2. Testing the Sequential Autoencoder for Bias

Now that we have drawn this parallel, we can use our knowledge of vEM to hypothesize how a VAE fares in modeling data. We have seen that vEM may struggle to perform maximum likelihood estimation when applied to sequential data. In (Turner & Sahani, 2011), the authors found that vEM produces incorrect estimates of θ not only due to the variational approximation being loose for different values of θ, but more importantly as a result of not being equally loose everywhere. This ultimately causes θ to be biased to regions where the variational approximation is tighter, not necessarily with the greatest likelihood. Motivated by these results, we run experiments on synthetic data to test for whether such phenomena occurs in a sequential autoencoder.

We define a simple time-series model for observations x depending on time independent f and time dependent z. A simple model allows us to exactly calculate the log likelihood, as well as train our model and calculate the ELBO for a variety of parameter settings. Our model has a single latent $f \in \mathbb{R}^2$, as well as time-dependent latents $z_1, z_2, z_3 \in \mathbb{R}^2$ with the following priors:

$$\begin{aligned}p(f) &\sim \mathcal{N}(0, \sigma_f^2 I) \\ p(z_1) &\sim \mathcal{N}(0, \sigma_z^2 I) \\ p(z_2 \mid z_1) &\sim \mathcal{N}(z_1, \sigma_z^2 I) \\ p(z_3 \mid z_2) &\sim \mathcal{N}(z_2, \sigma_z^2 I)\end{aligned}$$

Our observations $x_1, x_2, x_3 \in \mathbb{R}^2$ have the likelihood

$$p(x_t \mid z_t, f) \sim \mathcal{N}(f + z_t, \sigma_x^2 I).$$

In this experiment our parameters are $\theta = (\sigma_f, \sigma_z, \sigma_x)$. As in (Turner & Sahani, 2011), we fix the value of all but one of the parameters to their true value, and vary the last parameter. For each setting of θ, we find the optimal variational approximation $q_\phi(\mathbf{z} \mid \mathbf{x})$, and plot both the ELBO and log-likelihood as a function of our held-out parameter. To test our sequential autoencoder on this data, we keep the decoder fixed based on θ and train our encoder q using gradient descent.

Our experiment differs from (Turner & Sahani, 2011) in a couple minor ways. There, they calculate the optimal E-step, and derive the optimal ELBO for each parameter setting. In contrast, our sequential autoencoder must be trained from data, so we must sample data points x from our generative model and plot a Monte-Carlo estimate of the ELBO after the encoder has converged. Furthermore, we plot the likelihood $p(\mathbf{x})$ of the observed data rather than its expectation. This is acceptable since our model is chosen to be consistent, i.e. the parameter maximizing the likelihood is the true parameter in the limit of infinite data.

Results for learning the observation noise σ_x are shown in figure 3. Here, we see that the optimum parameter found by maximizing the ELBO (the dashed blue line) is biased to be slightly greater than the true parameter maximizing the log likelihood (the dashed red line). This is because our variational approximation is extremely poor when σ_x^2 is small - a low observation noise implies that any set of observations which are far from expected is extremely unlikely. Our algorithm is thus biased towards regions where the approximation is tighter.

Results for learning the dynamical noise σ_z are shown in figure 4. Here, our variational approximation for small σ_z isn't poor, as variation in the x_t can be explained by the observation noise. This causes our algorithm to underestimate the true parameter. In the second example, we see that our ELBO is roughly constant and is no longer monotonic, demonstrating the stochastic nature of our autoencoder.

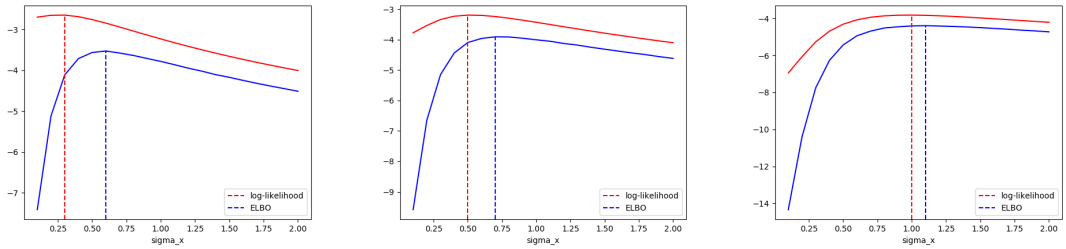

Figure 3. Plots of the log-likelihood and ELBO as a function of σ_x for various true values of σ_x: 0.3 (top), 0.5 (middle), and 1.0 (bottom). We see that our sequential autoencoder systematically overestimates the value of σ_x by biasing towards regions where the variational approximation is tighter.

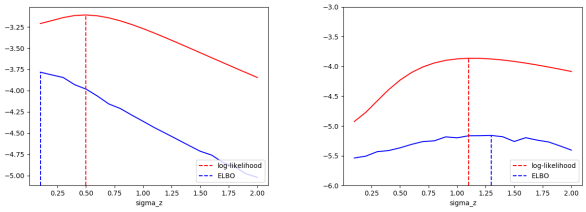

Figure 4. Plots of the log-likelihood and ELBO as a function of σ_y for various true values of σ_y: 0.5 (top) and 1.0 (bottom).

These graphs demonstrate that, although there is some bias, the sequential autoencoder does a reasonable job at modeling time-series data. This suggests that it is a good candidate for the task of disentangling time-series data.

4. Measuring Disentanglement

Much of the work in this field has consisted of answering the more fundamental problem of defining a measure for disentanglement. Rather than just learning independent factors, we are curious of how well one can capture the true factors of variation that are interpretable from the data. Prior work has relied significantly on qualitative analysis by visualizing the latent representation. While such an analysis is informative, it doesn't give us an idea of how much better one representation is than another.

One definition is given by (Eastwood & Williams, 2018) who defines disentanglement to be the extent to which a latent variable $d \in D$ in a representation predicts some true generative factor $k \in K$ such that each latent variable captures at most one generative factor. This definition implicitly assumes that $D \geq K$, since otherwise the latent variables are not able to explain the true generative factors. In practice D is much smaller than K as we learn a low-dimensional abstraction of our data. As a result, such simple representations cannot be found for complex data sets like the ones we consider here. The authors of (Chen et al., 2018) suggest looking at the empirical mutual information between the latent variable z and a ground truth factor v which can be estimated using the joint distribution. The higher the mutual information implies that z contains a lot of information about v. In general, the mutual information is maximal if there exists a deterministic, invertible relationship between z and v. Our algorithm tries to maximize this mutual information, but in order to allow for interpretable results, we use the measurement introduced in (Li & Mandt, 2018).

We first evaluate the persistence of time-invariant features from our representation. For some random sample $\mathbf{x}$, we sample $f \sim p(f|\mathbf{x})$ and $\mathbf{z}_1 \sim p(\mathbf{z}|\mathbf{x})$. We randomly sample another $\mathbf{z}_2$ from a standard normal distribution. Given these latent assignments pairs, $(f, \mathbf{z}_1)$ and $(f, \mathbf{z}_2)$, we generate two samples from the likelihood distribution

$$\mathbf{x}_1 \sim p(\mathbf{x}|\mathbf{z}_1, f)$$
$$\mathbf{x}_2 \sim p(\mathbf{x}|\mathbf{z}_2, f)$$

We expect that if some time-invariant ground-truth v taking on one of a finite number of classes is fully represented by f, then our confidence of v given $\mathbf{x}_1$ (defined by the probability distribution $\mathbf{p}_{data}$) should be equal to our confidence of v given $\mathbf{x}_2$ ($\mathbf{p}_{recon}$). We measure this confidence by training a classifier over the training set to predict the ground truth features, and use it to make predictions on the two generated samples. If v is preserved over time, then the probability distributions $\mathbf{p}_{recon}$ and $\mathbf{p}_{data}$ should be identical over the possible values of v. To train this classifier, we require the labels of the factor v on all training samples.

We choose to measure the difference of these vectors in terms of predictive disagreement and in terms of KL-divergence. Given N test samples, we define the predictive disagreement as

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\text{argmax}\mathbf{p}_{recon}^{(i)} \neq \text{argmax}\mathbf{p}_{data}^{(i)}\right]$$

which counts the number of disagreeing predictions of the classifier on the two generated samples. We also measure

Figure 5. Each input is composed of 8 frames of a character performing a single action. In the examples above, the character is walking left.

the disagreement of the probability vectors in terms of the average KL-divergence,

$$\frac{1}{N}\sum_{i=1}^{N} KL(\mathbf{p}_{recon}^{(i)}||\mathbf{p}_{data}^{(i)})$$

5. Experiments: Video Data

To assess the performance of disentanglement, we required a data set whose time-dependent and time-independent features are relatively obvious. The Sprites data set is a perfect example. The data, from the Liberated Pixel Cup,[3] consists of animated cartoon characters whose hairstyle, skin color, and clothing can be interchanged. In addition, a given character can perform a sequence of actions that form a short video. In the context of this data set, the time-independent features of a video are the attributes of the character and the time-dependent features of the video are the actions in each frame. At the conclusion of our experiment, we see that actions are preserved even after changes to the character attributes.

5.1. Data set

We took a small sample from the original Sprites data set. A character is characterized by 4 attribute categories: skin color, top, pants, and hair. In these experiments, we only use 6 variants of each attribute for a total of 1296 combinations. A given character can perform 3 different actions facing 3 directions. For each action, character, and direction, we form a video composed of 8 frames, where in each frame, the character is performing the action in the specified direction. Figure 5 shows one such training sample. We use the first 1000 characters as our training samples and the remaining 296 characters as our test samples. While our model is completely unsupervised, we also generate labels for the attributes and action of each sample which will be used to train our classifier to quantify disentanglement discussed in Section 5.4.

5.2. Model Architecture

We implemented all of our models and evaluation suites in pytorch. Inputs to our model consist of tensors of size (`num_frames`, `height`, `width`, `num_channels`) = (8, 64, 64, 3).

[3] http://lpc.opengameart.org/

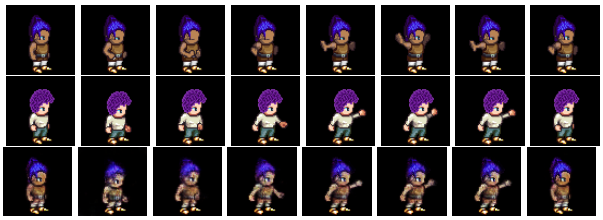

Figure 6. The first sample (first row) is used to sample f and the second sample (second row) is used to sample $\mathbf{z}$. The third row, shows the sample generated from the likelihood $p_\theta(\mathbf{x}|\mathbf{z}, f)$. Visually, we see that the attributes of the generated sample match the first sample and the actions match the second sample.

Our encoder $q_\phi(z_t|x_t)$ is a convolutional neural network with 4 convolutional layers, going to 256 channels, followed by a 1-hidden layer MLP with 512 hidden units. The output is a vector with length $2 * dim(z)$, where the first $dim(z_t)$ elements are the means and the second $dim(z_t)$ elements are the logvariances of the output Gaussian. $q_\phi(f|\mathbf{x}_{1:T})$ encodes x_t using the same convolutional layers as the encoder for the z_t; it then passes these features into a 1-hidden layer RNN with hidden layer size fixed to 512.

Our decoder $p(x_t \mid z_t, f)$ is a deconvolutional neural network. We map (z_t, f) to a 256-channel 2-by-2 image with a fully connected layer, and then apply 4 deconvolutional layers followed by a sigmoid to output our $64 \times 64 \times 3$ image x_t.

We fix $dim(z_t)$ to be 32 and experiment with the dimension of f being 32 and 256. By making z_t low dimensional we hope that it will capture only the time-dependent features and none of the time-independent ones. We train our VAE using the Adam optimizer (Kingma & Ba, 2015).

5.3. Qualitative Performance

For our qualitative assessment, we look at the images generated by our decoder after performing attribute swapping and random sampling. In figure 6 we show an output from feature mixing (last row). For two randomly selected samples, $\mathbf{x}_1, \mathbf{x}_2$, this output is generated by sampling $f \sim q_\phi(f|\mathbf{x}_1)$ and sampling $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}_2)$. Finally, we sample from our decoder $\mathbf{x}_3 \sim p_\theta(\mathbf{x}|\mathbf{z}, f)$. If our representation is correct, then f represents the attributes of the character, so we should expect the generated sample's attributes to match those of the first sample. Since $\mathbf{z}$ represents the actions, we should expect the generated sample's actions to match those of the second sample. This appears to be the case for the generated sample.

Before running our experiments, we had anticipated that the direction, in addition to the action, of the character would be represented in $\mathbf{z}$. However the results shown in figure 7 suggest that the direction of the character is actually encoded

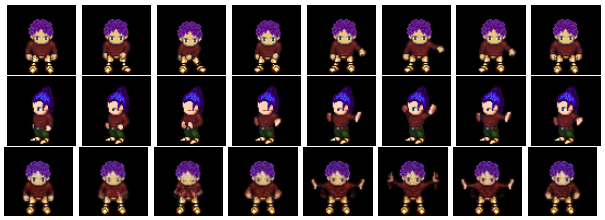

Figure 7. The direction of the character is inherited from the video with sampled f. Again, the sequence in the third row is decoded from sampling f from the first row and $\mathbf{z}$ from the second row

Figure 8. The first sample is used to sample $\mathbf{z}$ at each frame. We sample f according to a standard normal Gaussian. Given f and $\mathbf{x}$ we generate a sample from $p_\theta(\mathbf{x}|\mathbf{z}, f)$

in f. After further thought, this should be expected since the character in each sample faces the same direction in each frame, thus becoming a time-invariant feature. The generated video in this example inherits the actions of the first sample from which we sampled $\mathbf{z}$ (shown in the first row); however, it inherits the attributes and the direction of the second sample from which we sampled f (shown in the second row).

We also evaluate our representation for a randomly generated f and a $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ for some randomly selected sample $\mathbf{x}$. In figure 8 we show an example output. The generated sample (shown in the second row) is sampled from our decoder $p_\theta(\mathbf{x}|f, \mathbf{z})$. Visually, we see that the attributes of the characters are noticeably different in every category; however, the actions performed in each frame are largely preserved. Assuming a disentangled representation, this is exactly what we would expect from the above sampling procedure.

We also test our representation without any conditioning by sampling $f \sim \mathcal{N}(0, I)$ and $\mathbf{z} \sim p_\theta(\mathbf{z})$ and then generating a sample $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}, f)$. In figure 9, we show an generated sample. Compared to the previous examples where we sample f conditioned on auxiliary data, the quality of images are noticeably lower. Still, though, we see that the attributes and the direction of the character is preserved across all frames.

Since we sample z_t conditioned on $z_{1:t-1}$, if we sample a single poor z_t it is likely that the remaining sampled $z_{t+1}, \ldots, z_T$ will off-rail and form non-sense values of z_i that cannot be explained by our training set. While this event is rare, we were able to generate samples that showed

Figure 9. We sample f from a standard Gaussian and $\mathbf{z}$ from its learned prior $p_\theta(z)$, and generate the above sample according to $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}, f)$

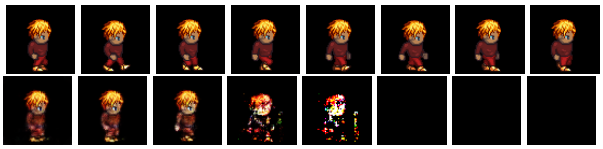

Figure 10. By conditioning z_t on $\mathbf{z}_{1:t-1}$, one poorly sampled z_i from the prior leads to subsequent poorly sampled $\mathbf{z}_{i+1:T}$. Here, we sample f from the top image and $\mathbf{z}$ from the prior.

evidence of this when we experimented with randomly sampled $\mathbf{z}$. Figure 10 shows an example of such a generated sample.

Many of these methods were motivated by (Li & Mandt, 2018). After simulating many of the experiments we observed some strange behaviour in generation when sampling random f. In particular, the authors choose to fix the size of the dimension of f to 256, which we suspected to be large relative to the 1000 unique attributes of the training samples. Because of this very high dimension, by sampling $f \sim \mathcal{N}(0, I)$ we may select an f that is very far away from the distributions of any of the training samples. Indeed, some of the images produced from randomly sampled f were deformed as shown in figure 11 which we believe was related to this choice. We improved this deformation by reducing this dimension from 256 to 32.

5.4. Quantitative Performance

To perform the quantitative analysis, we train a separate classifier on each of the 4 attributes (skin color, pants, tops, hair) and one classifier for all actions. There are 6 classes for each attribute and 9 classes for the action classifier. In all cases we pass in a sequence of 8 frames; our classifier applies a convolutional neural network on each of the frames to produce 8 feature vectors, and applies an RNN to this sequence of feature vectors to produce a probability distribution over class labels.

We first test how well f encodes the time-independent fea-

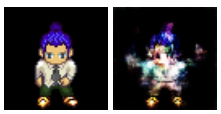

Figure 11. The randomly generated f gives us a deformed image

tures. For an image $\mathbf{x}$ in our test set we sample $f \sim q_\phi(f \mid \mathbf{x})$ from our encoder. We then sample $\mathbf{z}$ from our prior, and decode to get an $\mathbf{x}_{recon}$. For each attribute, we run the corresponding classifier on both $\mathbf{x}$ and $\mathbf{x}_{recon}$ and compare the outputted probability distributions over classes. We report both the disagreement (i.e. for what fraction of test samples does the maximum probability class label differ) as well as the average KL-divergence. Results for this are reported in Table 1.

We see that the latent variable f does a good job at encoding the hairstyle and top of each sprite, but struggles to properly capture the skin color and pant. This may be due to the similarity between skin colors of sprites, and also that the pant is involved in the actions (which are primary walking) and may be encoded within $\mathbf{z}$. There is still evidence of disentanglement for these features. If the generated image had a truly random attribute we would expect a disagreement of 83%, far greater than the observed values. We can also compare the measured $KL(\mathbf{p}_{recon} \| \mathbf{p}_{data})$ to the KL if all the class labels were random, $KL(\mathbf{p}_{rand} \| \mathbf{p}_{data})$, where $\mathbf{p}_{rand} = [1/N, \ldots, 1/N]$ if there are N attributes. In all 4 cases the reconstruction KL is much lower than the random KL, although again we see disentanglement is significantly better for the top and hairstyle. We should also note that these results are worse than those in (Li & Mandt, 2018), which is likely a result of our comparative lack of experience in training CNNs. We do, however, see a similar trend of our model better disentangling hair and top than skin color and pant.

The last row is the result of our experiment of sampling $z_t \sim q_\phi(z_t|x_t)$ from the encoder for each t, and sampling f from its prior $p(f)$, before producing $\mathbf{x}_{recon}$. This aims to see how well the encoded z_t do at modeling the dynamics of the sequence. We would expect the action sequence for x and x_{recon} to be the same, and thus use the predicted action sequence based on an action classifier with $3 \times 3 = 9$ categories of action-direction pairs.

We see that the latent z_t do slightly worse at encoding the action sequence, although this does show signs of pretty significant disentanglement, as the action label is kept unchanged 57% of the time, far greater than the 1 in 9 expected by random guessing. As seen in the qualitative section, encoding the action from an image is much harder as it is more difficult to tell what the action is to begin with, and oftentimes information about the direction is encoded in f.

6. Changes to the Objective: β-VAE

Other work in finding disentangled representations relies on modifying the variational objective. The most common approach is the β-VAE (Higgins et al., 2017), which multiplies the KL term in our objective by a tunable hyperparameter

Sequential Autoencoder			
attributes	disagreement	KL-recon	KL-random
skin color	24.8%	3.006	25.148
pant	29.2%	3.959	21.487
top	5.4%	0.695	21.456
hair	2.0%	0.268	17.367
action	43.0%	7.769	18.612

Table 1. Average disagreement and KL-divergence between test data and reconstructed images produced by our model.

$\beta > 1$, yielding the following loss function per data point.

$$\begin{aligned} L(\theta, \phi; x) = &- \mathbb{E}_{q_\phi(z,f|x)}[\log p_\theta(x \mid z, f)] \\ &+ \beta \cdot KL(q_\phi(f \mid x) \| p(f)) \\ &+ \beta \sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|x)}[KL(q_\phi(z_t \mid x_t) \| p_\theta(z_t|z_{<t}))] \end{aligned}$$

By more harshly penalizing the KL-terms, we may expect our encoders $q_\phi(f \mid x)$ and $q_\phi(z_t \mid x_t)$ to more closely match their respective priors. This may encourage independence between z_t and f, which could cause them to encode different features of a scene and thus produce a disentangled representation.

The β-VAE hasn't been studied much in the context of sequential data, posing the question of whether encouraging independence will separate the global and dynamical features. Furthermore, since we model the prior dynamics $p_\theta(z_t \mid z_{<t})$ with a trained neural network, we may expect this to better model the prior dynamics of the latent variables, thus generating images with more realistic dynamics.

6.1. β-VAE in the Context of Bayesian Inference

One potential drawback is that the β-VAE is difficult to interpret within variational inference. Consider variational inference with the ELBO augmented with this β term. The ELBO expands to

$$\begin{aligned} ELBO &= \mathbb{E}_q[\log p(x \mid z)] - \beta \cdot KL(q(z) \| p(z \mid x) \\ &= \log p(x) - \beta \cdot \mathbb{E}_q[\log \frac{p(z \mid x)^{\frac{1}{\beta}} p(z)^{\frac{\beta-1}{\beta}}}{q(z)}]. \end{aligned}$$

If $\pi(z)$ is the weighted geometric mean between the prior and the posterior, $\pi(z) \propto p(z \mid x)^{\frac{1}{\beta}} p(z)^{\frac{\beta-1}{\beta}}$, the expectation is just the KL between $q(z)$ and $\pi(z)$, shifted by a constant, and thus is optimized when $q(z) = \pi(z)$. There lacks motivation for why q should be a geometric mean between the prior and the posterior; this may be interpreted as throwing away data by not updating all the way towards the posterior. This makes the idea of a β-VAE somewhat dubious.

Figure 12. $\beta = 150$ was too large, forcing $q_\phi(\mathbf{z})$ and $q_\phi(f)$ to their respective priors

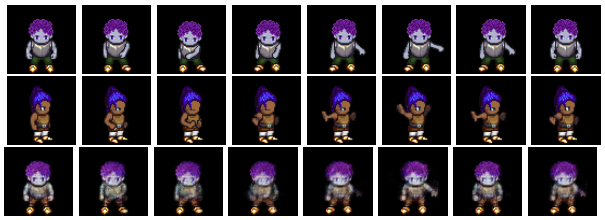

Figure 13. The first sample (first row) is used to sample f and the second sample (second row) is used to sample $\mathbf{z}$. The third row, shows the sample generated from the likelihood $p_\theta(\mathbf{x}|\mathbf{z}, f)$. These results are shown for $\beta = 6$.

6.2. Qualitative Results

As in the $\beta = 1$ case, we look at the images generated by our decoder after performing attribute swapping and random sampling. Because training is computationally expensive, we only train a few models. We experiment with $\beta = 150$ as in (Higgins et al., 2017; Burgess et al., 2017) as well as the much smaller $\beta = 6$.

We discovered that our loss was extremely sensitive to β after looking at the results from the $\beta = 150$ trained model. $q_\phi(\mathbf{z}|\mathbf{x})$ and $q_\phi(f|\mathbf{x})$ were both pushed to their corresponding priors so that they were effectively independent of $\mathbf{x}$. As a result, all of the reconstructed images looked as in figure 12. We later concluded, after performing more tests, that $\beta = 6$ was the appropriate choice.

We re-performed the experiment in the $\beta = 1$ case by picking two random samples $\mathbf{x}_1$ and $\mathbf{x}_2$, and then sampling $\mathbf{z}_1 \sim q_\phi(\mathbf{z}|\mathbf{x}_1)$ and $f \sim q_\phi(f|\mathbf{x}_2)$. We sampled $\mathbf{x}_3 \sim p_\theta(\mathbf{x}|f, \mathbf{z})$ shown in figure 13. The reconstructed images were lower quality than the $\beta = 1$ test with no visual evidence of improved disentanglement. In fact, the quality of disentanglement seems to be visually worse indicated by the incorrect actions in the generated frames. In figure 14 we re-perform the experiment but sample f from its prior. Again, there is no visual evidence of improved disentanglement, but noticeably worse reconstruction.

6.3. Quantitative Results

Quantitative results measuring the disagreement and KL-divergence between the classifier-outputted probability distributions on test images and reconstructed images are shown in Table 2. The β-VAE performs much worse on these metrics; while disentanglement for hairstyle is roughly the same, we observe much worse disagreement rates for skin color and pants. The results get significantly worse when we encode $\mathbf{z}$ and sample f from its prior, and classify the action sequence, as our model is not much better than random guessing.

Figure 14. The first sample is used to sample $\mathbf{z}$ at each frame. We sample f according to a standard normal Gaussian. Given f and $\mathbf{x}$ we generate an sample from $p_\theta(\mathbf{x}|\mathbf{z}, f)$. Here, $\beta = 6$.

β-VAE			
attributes	disagreement	KL-recon	KL-random
skin color	41.4%	5.325	25.148
pant	51.4%	7.851	21.487
top	13.2%	1.511	21.456
hair	2.2%	0.0.325	17.367
action	73.6%	15.633	18.612

Table 2. Average disagreement and KL-divergence between test data and reconstructed images produced by the β-VAE, for $\beta = 6$.

One explanation for this poor performance is that the β penalty deprioritizes having a good reconstruction. Instead, it may cause clusters of attribute classes to be unseparated in the latent space, forcing much overlap between the latent distributions for each data point, thus making it likely that a reconstructed image loses its attribute class. Since the sequential autoencoder already achieves disentanglement through its graphical model, achieving good reconstructions is much more important than adding the β penalty.

7. Conclusion

We analyzed the performance of the Sequential Autoencoder presented in (Li & Mandt, 2018) as a method to disentangle time-independent and time-dependent features of sequential data. We showed that sequential autoencoders estimate the maximum likelihood parameters with small bias, and thus are good candidates for modeling time-series data. We tested our methods on the Sprites data set of short videos, where we sought to disentangle the attributes of a character from the actions performed in each frame. Qualitatively, we observed disentanglement in our ability to carefully control the attributes of the character while almost completely retaining the original actions. We quantitatively backed this claim after establishing an interpretable measurement of disentanglement. Finally, we analyzed the β-VAE, which performed much worse on our data set.

Acknowledgements

We'd like to thank Tamara and Brian for their invaluable feedback on our project over the course of the class!

Division of Work

Eshaan and Tim worked jointly on most of the components of this project. Eshaan focused more on running the sequential autoencoder on synthetic data and producing the log likelihood plots. Tim worked more on running the sequential autoencoder on the Sprite data and generating the reconstructed images. We both contributed to the evaluation section, with Tim working more on the qualitative metrics and Eshaan more on the quantitative metrics, and we jointly analyzed the β-VAE.

References

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in Beta-VAE. *Neural Information Processing Systems*, 2017.

Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Eastwood, C. and Williams, C. K. A framework for the quantitative evaluation of disentangled representations. 2018.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. Beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Li, Y. and Mandt, S. Disentangled sequential autoencoder. *International Conference on Machine Learning*, 2018.

Turner, R. E. and Sahani, M. Two problems with variational expectation maximisation for time-series models. In Barber, D., Cemgil, T., and Chiappa, S. (eds.), *Bayesian Time series models*, chapter 5, pp. 109–130. Cambridge University Press, 2011.