2nd Symposium on Advances in Approximate Bayesian Inference, 2019 1–12

The Gaussian Process Prior VAE for Interpretable Latent Dynamics from Pixels

Michael Pearce SCRAMBLEDPIE@GMAIL.COM
Complexity Science, University of Warwick, UK

1. Introduction

We consider the problem of unsupervised learning of a low dimensional, interpretable, latent state of a video containing a moving object. The problem of distilling interpretable dynamics from pixels has been extensively considered through the lens of graphical/state space models (Fraccaro et al., 2017; Lin et al., 2018; Pearce et al., 2018; Chiappa and Paquet, 2019) that exploit Markov structure for cheap computation and structured priors for enforcing interpretability on latent representations. We take a step towards extending these approaches by discarding the Markov structure; inspired by Gaussian process dynamical models (Wang et al., 2006), we instead repurpose the recently proposed Gaussian Process Prior Variational Autoencoder (Casale et al., 2018) for learning interpretable latent dynamics. We describe the model and perform experiments on a synthetic dataset and see that the model reliably reconstructs smooth dynamics exhibiting U-turns and loops. We also observe that this model may be trained *without* any β annealing or freeze-thaw of training parameters in contrast to previous works, albeit for slightly different use cases, where application specific training tricks are often required.

2. The Gaussian Process Prior VAE

Assume we are given a dataset of images and each image has a corresponding feature vector, e.g. images of faces and features are pose angle, lighting intensity etc. The Gaussian Process Prior Variational Autoencoder (GPP-VAE) may be used to learn intermediate latent representations from features (input) to images (output) where the input-output relationship is supervised but the latent representation is unsupervised. In the use case we consider, we assume that we have a set of videos of a single object moving around on a pixel display. Each video consists of a set of T images: $v_1, ..., v_T \in [0, 1]^{32\times32}$ which are binary arrays and the corresponding feature of an image is its time stamp t. The intermediate state we aim to learn is an *interpretable* 2D latent time-series of $x_{1:T}$, $y_{1:T}$ coordindates for the object for each frame. We have no ground truth data of object position hence this is unsupervised. We place a Gaussian process prior over time on the functions $x, y : [1, T] \to \mathbb{R}$. A Gaussian process may be viewed as a more general linear Gaussian state space model, this has the benefit of being more flexible, and smoothing (aggregating information across all time) is performed by default. However sacrificing Markov structure typically increases computational complexity from linear to cubic. A Gaussian process is fully specified by its mean and covariance functions. Evaluating these functions at a discretization of the input domain, $\underline{T} = 1 : T \subset [1, T]$, yields the mean vector and covariance matrix of a multivariate Gaussian distribution. Hence without loss of generality, we adopt the notation of scalars x_t, y_t and vectors $x_{1:T}, y_{1:T}$ instead of functions $x(t), y(t)$. The generative model is given by

$$\begin{aligned} \mathbb{P}[v_{1:T}, x_{1:T}, y_{1:T}] &= \prod_{t=1}^{T} \mathbb{P}[v_t|x_t, y_t]\ \mathbb{P}[x_{1:T}, y_{1:T}] && (1) \\ &= \underbrace{\prod_{t=1}^{T} \mathcal{B}(v_t|p_\theta(x_t, y_t))}_{\text{Bernoulli over pixels}} \underbrace{\mathcal{N}\big(x_{1:T}|\underline{0}, k_x(\underline{T}, \underline{T})\big)\mathcal{N}\big(y_{1:T}|\underline{0}, k_y(\underline{T}, \underline{T})\big)}_{=\mathcal{GP}(x_{1:T})\mathcal{GP}(y_{1:T})}, && (2) \end{aligned}$$

where $\mathcal{B}(v_t|p_\theta(x_t, y_t))$ is a product of 32×32 independent Bernoulli distributions over pixels parameterised by a neural network $p_\theta(x_t, y_t)$ with parameters θ. The functions $k_x, k_y : [1, T] \times [1, T] \to \mathbb{R}$ are positive semi-definite kernels with hyperparameters that we assume are known in this work for simplicity and $\mathcal{N}(x|\underline{\mu}, \Sigma)$ is the multivariate Gaussian density. Given a video $v_{1:T}$ we aim to learn $x_{1:T}$ and $y_{1:T}$ which ideally would be inferred by Bayes rule, $\mathbb{P}[x_{1:T}, y_{1:T}|v_{1:T}]$. However due to the $\mathcal{B}(v_t|p_\theta(x_t, y_t))$ terms, the true posterior has no normalized analytic form.

We desire a variational approximation with two properties. Firstly for fast inference at test time we require amortization; we aim to learn a function from observed variables $v_{1:T}$ directly to approximate posterior $q(x_{1:T}, y_{1:T}|v_{1:T})$. Secondly, at test time, video data is streaming and frames are accumulated over time thus we also require an approximate posterior that can handle variable length videos. Satisfying both of the properties and exploiting the structure of the generative model, we propose the following variational approximation

$$\begin{aligned} q(x_{1:T}, y_{1:T}|v_{1:T}) &= \frac{1}{Z(v_{1:T})} \prod_{t=1}^{T} q^*_\phi(x_t, y_t|v_t)\ \mathbb{P}[x_{1:T}, y_{1:T}] && (3) \\ &= \frac{1}{Z(v_{1:T})} \prod_{t=1}^{T} \underbrace{\mathcal{N}(x_t|\mu^*_{x\phi}(v_t), {\sigma^*_{x\phi}}^2(v_t))\mathcal{N}(y_t|\mu^*_{y\phi}(v_t), {\sigma^*_{y\phi}}^2(v_t))}_{q^*_\phi(x_t,y_t|v_t)\text{ approximating }\mathcal{B}(v_t|p_\theta(x_t,y_t))} \mathcal{GP}\left(x_{1:T}, y_{1:T}\right) \end{aligned}$$

which is simply the true generative model with only the troublesome $\mathcal{B}(v_t|p_\theta(x_t, y_t))$ terms replaced by Gaussian densities denoted $q^*_\phi(x_t, y_t|v_t)$. These new Gaussian factors are conjugate to the GP prior thereby enabling normalization. Each factor is parameterized by the output of a recognition network $\mu^*_{x\phi}(v_t)$, ${\sigma^*_{x\phi}}^2(v_t)$ (similarly for y_t) with parameters ϕ. By noting the symmetry of the Gaussian distribution, $\mathcal{N}(x|\mu, \sigma^2) = \mathcal{N}(\mu|x, \sigma^2)$, the denominator is exactly the form of the unnormalized *likelihood* × *prior* of standard GP regression. The $x_{1:T}$ are latent function values and $\{(t, \mu^*_{\phi x}(v_t)\}_1^T$ are a set of observations (or pseudo-points) each with noise ${\sigma^*_{\phi x}}^2(v_t)$. These points condition the generative prior GP yielding an (analytic) posterior GP that approximates the (non analytic) true posterior $\mathbb{P}[x_{1:T}, y_{1:T}|v_{1:T}]$. Thus the standard GP equations (see Appendix B) yield means, $\mu_x(t)$, $\mu_y(t)$, and variances plotted in Figure 1.

Since, for simplicity, we assume $x_{1:T}$ and $y_{1:T}$ are independent, this may be viewed as two standard 1-D Gaussian process regression models and the term $Z(v_{1:T}) = Z_x(v_{1:T})Z_y(v_{1:T})$

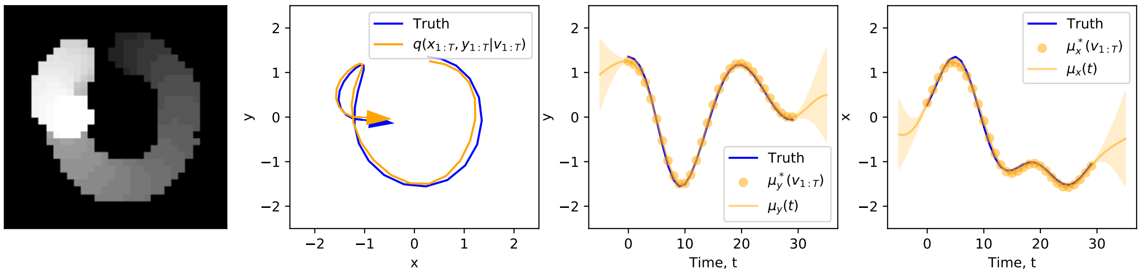

Figure 1: Left: video frames overlayed and shaded by time. Centre: example inferred latent path rotated onto ground truth (not used in training). Right: each latent dimension is a GP over time, using the generative prior GP conditioned on pseudo points from the recognition network.

and $Z_x(v_{1:T})$ is thus precisely the marginal likelihood of the x GP commonly used in GP regression for hyperparameter learning. At test time, if there are missing frames or the video is shorter, $T_s < T$, mathematically, only $q^*(v_{1:T_s})$ terms are included in Equation 3 and we have $Z(v_{1:T_s})$. Intuitively, the approximate posterior simply becomes a GP regression model with fewer points, $\{(t, \mu^*_{\phi x}(v_t)\}_1^{T_s}$.

In this work we assume that the kernels are the popular squared exponential $k_x(t,t') = \exp(-\frac{1}{2}(t-t')^2/l_x^2)$ with hyperparameters $l_x = l_y = 5$ that represent the time scale, or speed, of changes in x and y positions respectively and may be learnt or informed by prior knowledge, i.e. observing the volatility of object movement in videos. This is a stationary kernel and enforces smoothness over latent trajectories and by setting $l_x, l_y \ll 1$ recovers a standard factorised Gaussian prior of the traditional VAE model. In general, any kernel may be used, *quadratic* kernel for parabolic motion, *min* kernel for Brownian motion, *periodic* kernels for oscillatory motion, or any sum or product of these kernels depending on the prior belief about a particular dataset or physical system.

For training the neural network parameters θ, ϕ, we maximize the evidence lower bound,

$$\mathcal{F}(\theta, \phi; v_{1:T}) = \mathbb{E}_q\left[\sum_{t\in\underline{T}} \log \mathcal{B}(v_t|p_\theta(x_t, y_t)) - \log q^*_\phi(x_t, y_t|v_t)\right] + \log Z(v_{1:T}), \tag{4}$$

a full derivation is given in the Appendix. The first (inner) term in Equation 4 is the reconstruction term, evaluated with the reparameterisation trick (Kingma and Welling, 2013; Rezende et al., 2014), and the middle and final terms compose the analytically tractable Kullback-Leibler divergence between the GP prior and the inference model. Alternatively, the first two terms together may be viewed as the "error" between the true posterior and approximate posterior introduced by replacing only the Bernoulli likelihoods with Gaussian approximations, and the final term is a surrogate marginal likelihood.

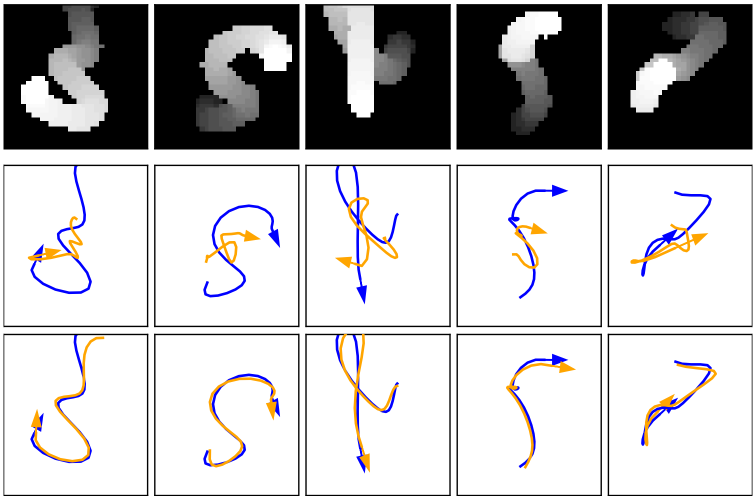

Figure 2: Top: input video with frames over layed and shaded by time. Middle: inferred $x_{1:T}$, $y_{1:T}$ when using a VAE (ignoring time stamp of frames) showing both the ground truth (blue, not used for training) and the inferred posterior mean (orange). The inferred trajectory is linearly transformed onto the ground truth. Bottom: $x_{1:T}$, $y_{1:T}$ inferred using the the GPP-VAE.

3. Experiments

In order to test the ability to learn latent dynamics we synthesize a controlled dataset. We generate videos of length $T = 30$ by first sampling a time series $x_{1:T}, y_{1:T} \sim \mathcal{N}\big(\cdot\,|\underline{0}, k(\underline{T},\underline{T})\big)$. Each pair (x_t, y_t) is rescaled to pixel indeces and rendered as a ball onto a binary canvas. Example videos are shown in Figure 2 and Appendix A, network details and training are given in Appendix B.3. As a naive baseline model, we train a standard Variational Autoencoder using the same inference and rendering networks. For model evaluation, we consider how the latent space compares to the ground truth. In Figure 2 we plot videos and approximate posterior means and see the GPP-VAE largely recovers the ground truth, note this is never used in training. By construction, the generated images lie in a low dimensional subspace of pixel space, and we expect similar images (many overlapping white pixels) to be encoded into similar latents. Thus we consider how a regular pattern of images is encoded into the latent space. In Figure 3 we plot such patterns with the output of the recognition network $(\mu^*_{\phi x}(v), \mu^*_{\phi y}(v))$. In this case, the VAE appears to learn a distorted and discontinous mapping from pixels to latents while the GPP-VAE learns a continuous mapping with mild distortion. Both methods learn near perfect reconstruction of videos shown in Appendix A. Source code is available at https://github.com/scrambledpie/GPVAE/.

4. Conclusion

We present a simple model and show proof-of-concept results that a Gaussian Process Prior within a VAE may be used for learning complex but smooth latent dynamics without any

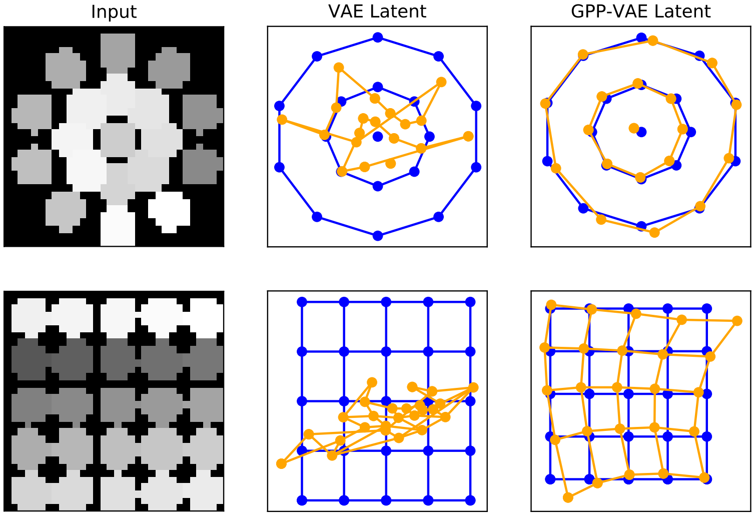

Figure 3: Left: top: 19 images, bottom: 25 images generated with the ball in a regular pattern. Centre: the patterns output from the recognition network $q^*(x, y|v)$ from the trained VAE. Ground truth in blue and recognition network means in orange (rotated onto ground truth). Lines are for visual aid only. Right: the output of $q^*(x, y|v)$ from the trained GPP-VAE (rotated onto ground truth). There is no time correlation in the images hence we do not plot approximate posterior/apply smoothing. The VAE latent space is a highly distorted and discontinuous transformation of the pixel space while the GPP-VAE latent space is much more coherent. For training, see video https://www.youtube.com/watch?v=riVhb6K_iMo.

special training. In this work we consider a toy dataset and the dynamics model generating the data was also used to fit the model removing miss-specification issues. Hence future work is to apply the model to a wider variety of less controlled settings, and comparison with more sophisticated baselines. By comparison, using similar data, the Kalman-Variational Autoencoder learnt dynamics (also including sharp turns, hence non-smooth) using an LSTM and training required freeze-thaw of model parameters and re-weighting of objective terms. Likewise extensions to this model (Chiappa and Paquet, 2019; Pearce et al., 2018) consider multiple objects constrained to parabolic motion and either require β annealing or other training tricks.

Acknowledgments

Thanks go to Ayman Boustati and Janis Klaise for proof reading the paper in the last hours before the deadline and to Tobias Grafke and Warwick University Centre for Complexity Science for providing Nvidia Quadro RTX 6000 graphics cards for experiments.

References

Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 10369–10380, 2018.

Silvia Chiappa and Ulrich Paquet. Unsupervised separation of dynamics from pixels. *METRON*, 77(2):119–135, 2019.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *International Conference on Learning Representations*, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

Wu Lin, Nicolas Hubacher, and Mohammad Emtiyaz Khan. Variational message passing with structured inference networks. *International Conference on Learning Representations*, 2018.

Michael Pearce, Silvia Chiappa, and Ulrich Paquet. Comparing interpretable inference models for videos of physical motion. In *1st Symposium on Advances in Approximate Bayesian Inference*, 2018.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448, 2006.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Appendix A. Further Experimental Results

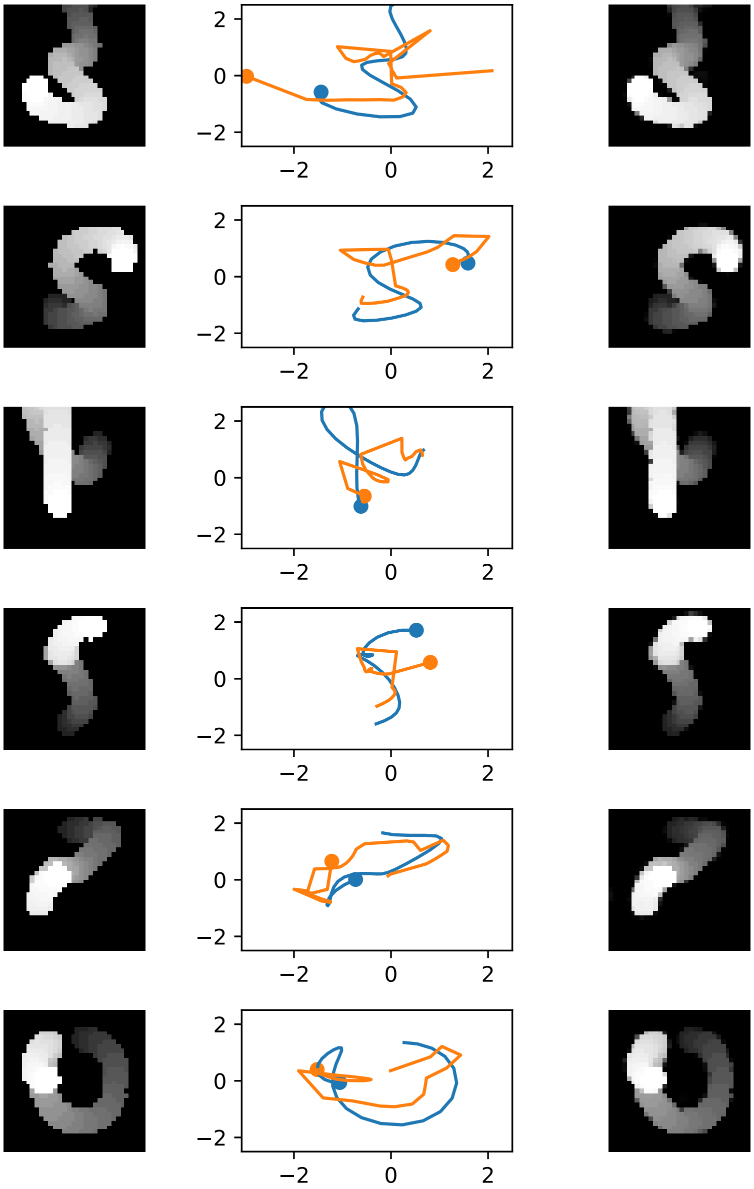

Figure 4: Left: input video $v_{1:T}^{test}$ frames overlayed and shaded by time. Centre: the latent position learnt by a variational autoencoder, VAE. Ground truth $x_{1:T}^{test}, y_{1:T}^{test}$ in blue, and the mean of $q(x_{1:T}, y_{1:T}|v_{1:T}^{test})$ in orange. Right: reconstruction learnt by a VAE. The time stamp of each frame is ignored during training and each frame is encoded in an i.i.d. fashion. Reconstruction is near perfect yet the latent space is a not a physically accurate space.

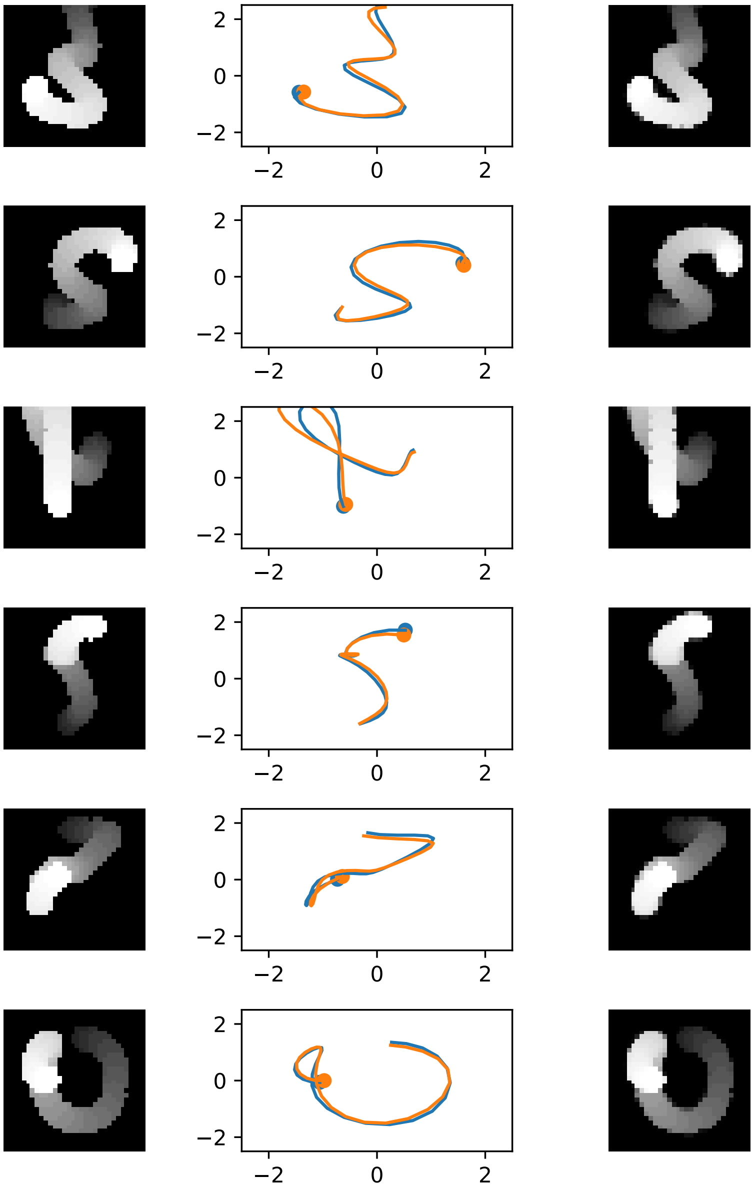

Figure 5: Left: input video $v_{1:T}^{test}$ frames overlayed and shaded by time. Centre: the latent position learnt by GPP-VAE. Right: reconstruction learnt by a GPP-VAE. The latent space, when rotated onto the ground truth, recovers original dynamics.

Appendix B. Mathematical Derivations

B.1. Gaussian Process Regression and The Approximate Posterior

We assume independent factorised Gaussian process priors for the horizontal position $x_{1:T}$ and vertical position $y_{1:T}$. The recognition network returns fully factorised Gaussian densities therefore the approximate posterior is also factorised across vertical and horizontal positions. As a consequence the log marginal likelihood is a sum of individual marginal likelihoods $\log Z(v_{1:T}) = \log Z_x(v_{1:T}) + \log Z_y(v_{1:T})$. We therefore give the expression for a single term. In our case, for a single video, $v_{1:T}$, the input-output pairs for Gaussian process regression is the set of time stamps and means from the recognition network $\{(1, \mu^*_{\phi x}(v_1)), ..., (T, \mu^*_{\phi x}(v_T))\}$ with noise variance for each observation given by $({\sigma^*_{\phi x}}^2(v_1), ..., {\sigma^*_{\phi x}}^2(v_T))$. Denote the column vector of means as $\underline{\mu}^*_x = \mu^*_{\phi x}(v_{1:T}) \in \mathbb{R}^T$, the kernel matrix $K_x = k_x(\underline{T}, \underline{T}) \in \mathbb{R}^{T\times T}$ and the diagonal noise matrix as $\Sigma^*_x = \operatorname{diag}({\sigma^*_{\phi x}}^2(v_{1:T})) \in \mathbb{R}^{T\times T}$. The log marginal likelihood $\log Z_x(v_{1:T}) \in \mathbb{R}$ is given by

$$\log Z_x(v_{1:T}) = -\frac{1}{2}\Big(T\log(2\pi) + \log|K_x + \Sigma^*_x| + {\underline{\mu}^*_x}^\intercal \big(K_x + \Sigma^*_x\big)^{-1}\underline{\mu}^*_x\Big)\,. \tag{5}$$

Due to the matrix inversion, this has cubic cost in the number of observed frames T. The matrix inversion is done via Cholesky decomposition as suggested by Williams and Rasmussen (2006).

Secondly to compute the approximate posterior mean and approximate posterior variance, we may use the standard Gaussian process regression equations,

$$\mu_x(t|v_{1:T}) = k_x(t, \underline{T})\big(K_x + \Sigma^*_x\big)^{-1}\underline{\mu}^*_x \tag{6}$$

$$k_x(t, t'|v_{1:T}) = k_x(t, t') - k_x(t, \underline{T})\big(K_x + \Sigma^*_x\big)^{-1}k_x(\underline{T}, t') \tag{7}$$

where the dependence upon $v_{1:T}$ is embedded in the $\underline{\mu}^*_x$ and Σ^*_x terms. The approximate posterior variance for a single point is given by $\sigma^2_x(t|v_{1:T}) = k_x(t, t|v_{1:T})$ and the approximate posterior distribution for a single position x_t, y_t *conditioned on all frames* is

$$q(x_t, y_t|v_{1:T}) = \mathcal{N}\big(x_t\big|\ \mu_x(t|v_{1:T}), \sigma^2_x(t|v_{1:T})\big)\ \mathcal{N}\big(y_t\big|\ \mu_y(t|v_{1:T}), \sigma^2_y(t|v_{1:T})\big). \tag{8}$$

B.2. Deriving Evidence Lower Bound Objective

The objective function given in the main text is

$$\mathcal{F}(\theta, \phi; v_{1:T}) = \mathbb{E}_q\left[\sum_{t\in\underline{T}} \log\mathcal{B}(v_t|p_\theta(x_t, y_t)) - \log q^*_\phi(x_t, y_t|v_t)\right] + \log Z(v_{1:T}). \tag{9}$$

For the next equations, we shall drop the time indices

$$\begin{aligned}
\log \mathbb{P}[v] &= \log \int_{x,y} \mathbb{P}[v,x,y]dxdy && (10)\\
&= \log \int_{x,y} \frac{q(x,y|v)}{q(x,y|v)} \mathbb{P}[v,x,y]dxdy && (11)\\
&\geq \int_{x,y} q(x,y|v) \log \frac{\mathbb{P}[v,x,y]}{q(x,y|v)} dxdy && (12)\\
&= \int_{x,y} q(x,y|v) \log \mathbb{P}[v|x,y] + \log \frac{\mathbb{P}[x,y]}{q(x,y|v)} dxdy && (13)\\
&= \mathbb{E}_q \left[\log \mathbb{P}[v|x,y]\right] + \mathbb{E}_q \left[\log \frac{\mathbb{P}[x,y]}{q(x,y|v)}\right] && (14)
\end{aligned}$$

where expectations are over x, y. The first term must be evaluated by Monte-Carlo using the reparameterization trick (Kingma and Welling, 2013), a sample x_t^i, y_t^i is generated by taking the approximate posterior mean and standard deviation and sampling white-noise $\epsilon \sim \mathcal{N}(\cdot|0,1)$, $x_t^i = \mu_x(t|v_{1:T}) + \epsilon * \sigma_x(t|v_{1:T})$ which can then be passed to $p_\theta(\cdot)$ and the likelihood of the true image $\log \mathcal{B}(v_t|p_\theta(x_t^i, y_t^i))$ can be computed.

Next we focus on the second term, the KL divergence from the approximate posterior to the prior. Substituting in the from of $q(x,y|v) = \mathbb{P}[x,y]q^*(x,y|v)/Z(v)$,

$$\begin{aligned}
\mathbb{E}_q \left[\log \frac{\mathbb{P}[x,y]}{q(x,y|v)}\right] &= \mathbb{E}_q \left[\log \frac{\cancel{\mathbb{P}[x,y]}}{\cancel{\mathbb{P}[x,y]} q^*(x,y|v) \frac{1}{Z(v)}}\right] && (15)\\
&= \mathbb{E}_q \left[-\log q^*(x,y|v)\right] + \log Z(v) && (16)
\end{aligned}$$

and note that the prior term that is common to both generative and inference model cancels out. Combining terms yields the expression in Equation 9. Recall that $q^*(x,y|v_{1:T})$ is a factorised Gaussian over all variables $x_1, .., x_T, y_1, ..., y_T$. Therefore the first term of Equation 16 term is a sum of univariate cross-entropys of Gaussian distributions, again dropping time indices to minimize cluttering notation, the term for a single time is given by

$$\mathbb{E}_q \left[-\log q^*(x,y|v)\right] = \mathbb{E}_q \left[-\log \mathcal{N}(x|\mu_x^*, \sigma_x^{*2}) - \log \mathcal{N}(y|\mu_y^*, \sigma_y^{*2})\right] \qquad (17)$$

where the expectation is over the approximate posterior given in Equation 8. Let $x \sim \mathcal{N}(x|\mu, \sigma^2)$, then the univariate Guassian cross-entropy is given by

$$\mathbb{E}_x \left[\log \mathcal{N}(x|\mu^*, \sigma^{*2})\right] = -\frac{1}{2}\left(\log 2\pi + 2\log \sigma^* + \frac{\sigma^2 + (\mu - \mu^*)^2}{\sigma^{*2}}\right)$$

The Monte-Carlo integral of the reconstruction term and the analytic expression for the KL divergence can all be implemented in any modern machine learning framework and optimized by gradient ascent. This is discussed below in Section B.3.

B.3. Data generation, Network Architectures and Training

For training data, we generate videos of length $T = 30$ by first sampling two time series $x_{1:T}, y_{1:T} \sim \mathcal{N}\big(\,\cdot\,|\underline{0}, k(\underline{T}, \underline{T})\big)$ where $k(t,t') = \exp(-(t-t)^2/(2 \cdot 5^2))$. Each pair (x_t, y_t) is rescaled to pixel space $(\tilde{x}_t, \tilde{y}_t) = 7 * (x_t, y_t) + (16, 16)$ and rendered as a ball with radius $r = 3$ onto a binary canvas. $v_t \in \{0, 1\}^{32\times32}$ where $[v_t]_{ij} = \mathbb{1}(||(i, j) - (\tilde{x}_t, \tilde{y}_t)||^2 < 3^2)$.

The recognition network $q^* : \{0, 1\}^{1024} \to \mathbb{R}^4$ is a fully connected network that takes as input a $32*32 = 1024$ image flattened to a vector $v_t \in \{0, 1\}^{1024}$. This is followed by a fully connected hidden layer of 500 nodes with the $tanh()$ activation function, and finally the output layer of four nodes returning $\mu^*_{\phi x}(v_t)$, $\log\sigma^*_{\phi x}(v_t)$ and $\mu^*_{\phi y}(v_t)$, $\log\sigma^*_{\phi x}(v_t)$, the network parameters are therefore two weight matrices and two bias vectors $\phi = \{W^1_q, B^1_q, W^2_q, B^2_q\}$.

The decoder, or rendering network, $p_\theta : \mathbb{R}^2 \to [0, 1]^{1024}$, is almost the same architecture in reverse, the input layer has only two nodes, followed by a single fully connected layer of 500 nodes with the $tanh()$ activation and finally 1024 nodes with the $sigmoid()$ activation yielding a unique independent Bernoulli probability between 0 and 1 for each of the 1024 pixels. The parameters are thus $\theta = \{W^1_p, B^1_p, W^2_p, B^2_p\}$.

Training is performed using the Adam optimizer with Tensorflow default parameters $\alpha = 0.001,\ \beta_1 = 0.9,\ \beta_2 = 0.999,\ \epsilon = 1e - 08$ and a batchsize of 35 randomly generated videos. We train the each method for 50,000 iterations.

In preliminary testing we applied β annealing of the prior KL term in the objective as it has been shown to stabilize learning however we found that a value of $\beta > 1$ would lead to the approximate model learning the prior distribution and never recovering, posterior collapse, and for $\beta < 1$ we found that training often became numerically unstable and overflow/underflow errors would cause training to halt. Therefore we apply no β annealing, all results are with the objective unmodified and optimized end-to-end.

B.4. Linear Trajectory Projection

We maintain a held-out test set of latent trajectories $x^{test}_{1:T}, y^{test}_{1:T}$ and their rendered counterparts $v^{test}_{1:T}$. We pass the images into the inference model to yield posterior mean vectors $\mu_x(\underline{T}|v^{test}_{1:T}), \mu_y(\underline{T}|v^{test}_{1:T}) \in \mathbb{R}^T$. We then use linear regression to predict the ground truth trajectories. Specifically we learn a rotation $W \in \mathbb{R}^{2\times2}$ and translation $B \in \mathbb{R}^2$ that minimizes the mean squared error over the whole test set

$$W, B = \arg\min_{W,B} \sum_{test} \sum_{t=1}^{T} \big|\big|W(\mu_x(t|\cdot), \mu_y(t|\cdot)) + B - (x^{test}_t, y^{test}_t)\big|\big|^2. \tag{18}$$

The error minimizing W and B are used to rotate the latent trajectories onto the true trajectories for the figures. The true trajectories are never used during training.

B.5. Comparison with Attentive Neural Processes

As a temporary aside for the interested reader, we may draw parallels between the Gaussian Process Prior Variational Autoencoder and Attentive Neural processes (Kim et al., 2019). A stochastic process is a collection of random variables (outputs) over an index set (inputs), i.e. a random function generator. A video generator may be viewed as realisations of a stochastic process, over the index set of time, random outputs are images in pixel space. At test time, given a set of frames from a video $v_{1:T}$ and a new time stamp t', we may query a model to predict the new frame $\mathbb{P}[v_{t'}|v_{1:T}]$, a simple regression problem, albeit with 1D input and high dimensional output.

Assume we are given the distribution of the stochastic process realisations, the full generative model of outputs given any inputs, and a subset of input-output pairs from a single realisation of a stochastic process. For any new input, we desire statistical predictions of the corresponding output from the same realisation.

The Neural process architecture allows a user to make such new predictions where the generative stochastic process is not known, however instead one has access to a large corpus of input-output pairs from many realisations of the same generative process. Firstly, each element from the set of observed points, (t, v_t), is encoded into a new representation r_t, and the set $r_1, ..., r_T$ is accumulated through summation to get R. Secondly, R is used in a decoder with the new input t' to parameterise a distribution over the output $\mathbb{P}[v_{t'}|R, t']$.

Attentive Neural processes augment this architecture in two ways. First, a self-attention layer (Vaswani et al., 2017) is applied to the set $r_1, ..., r_T$ yielding $\tilde{r}_1, ..., \tilde{r}_T$. Second, for prediction, the new input t' is used as a query to inform attention weights over $\tilde{r}_1, ..., \tilde{r}_T$, such that the aggregated representation $R_{t'}$ is informed by t', essentially augmenting Neural processes with a non-parametric memory.

In our use case, the encoder of (t, v_t) is the identity function for t and the recognition network for v_t and the encoding r_t is the time stamp and means and variances,

$$\text{encoder} : \mathbb{R} \times \{0,1\}^{32\times 32} \to \mathbb{R}^5 \tag{19}$$

$$(t, v_t) \xrightarrow{(\mathrm{Id}(\cdot), q^*(\cdot|v_t))} \big(t,\, \mu^*_{\phi x}(v_t),\, {\sigma^*_{\phi x}}^2(v_t),\, \mu^*_{\phi y}(v_t),\, {\sigma^*_{\phi x}}^2(v_t)\big). \tag{20}$$

Encoding all observed time stamps and images yield the set of encodings $r_1, ..., r_T$. Predicting $\mathbb{P}[v_{t'}|v_{1:T}]$ requires the approximate posterior at time t, the GP mean and variance. Recall the equation for the approximate posterior mean at a new point t',

$$\mu_x(t'|v_{1:T}) = \overbrace{k_x(t', \underline{T})}^{\substack{\text{attention}\\ \text{weights}}} \underbrace{\overbrace{\left(K_x + \Sigma^*_x\right)^{-1}}^{\substack{\text{self-attention}\\ \text{weights}}} \overbrace{\underline{\mu}^*_x}^{\text{values}}}_{\approx \text{ self-attention layer}} \tag{21}$$

The new time stamp t' is a *query*, the time stamps of observed/encoded frames $1 : T$ are *keys* (the first element of each r_t), and the kernel between query and keys are the attention weights. The kernel matrix inversion may be viewed as one layer of self-attention (admittedly with a stretch of the imagination). The self attention weight matrix has keys $1 : T$, queries $1 : T$, (the first elements of each r_t vector) and also incorporates ${\sigma^*_{\phi x}}^2(v_{1:T})$. The values being accumulated are $\underline{\mu}^*_x$. The post self-attention representations are thus

$$\tilde{r}_t = \left(t, \left(K_x + \Sigma^*_x\right)^{-1}_t \underline{\mu}^*_x\right).$$

Therefore, standard Gaussian process regression may be viewed as applying self-attention to the collection of encoded representations, and when making a new prediction, attention is applied over the new representations. For prediction, a weighted average over other encoded frames is used to compute $q(x_t, y_t|v_{1:T})$, this is sampled to get $x^i_{t'}, y^i_{t'}$, and passed through the network $p_\theta(\cdot)$ to parameterize the distribution over pixels $\mathcal{B}(v_t|p_\theta(x^i_{t'}, y^i_{t'}))$, one sample for each pixel yields one predicted output $\mathbb{P}[v'_t|v_{1:T}]$.