

Supplementary Information for Gaussian Process Prior Variational Autoencoders

Contents

1	Supplementary Methods	1
1.1	ELBO derivation	1
1.2	Loss derivation	1
1.3	Fast low-rank computations	2
1.4	Implementation of low-memory stochastic backpropagation	2
1.5	Out of sample predictions.	3
2	Supplementary Figures	4

1 Supplementary Methods

1.1 ELBO derivation

We assume the following posterior distribution on latent variables

$$q_{\psi}(\mathbf{Z} | \mathbf{Y}) = \prod_n \mathcal{N}(\mathbf{z}_n | \boldsymbol{\mu}_{\psi}^z(\mathbf{y}_n), \text{diag}(\boldsymbol{\sigma}_{\psi}^{z^2}(\mathbf{y}_n))), \quad (1)$$

where ψ are variational parameters. The evidence lower bound (ELBO) can be derived as follows:

$$\begin{aligned}
 \log p(\mathbf{Y} | \mathbf{X}, \mathbf{W}, \phi, \sigma_y^2, \theta) &= \log \int \frac{p(\mathbf{Y} | \mathbf{z}, \phi, \sigma_y^2) p(\mathbf{Z} | \mathbf{X}, \mathbf{W}, \theta, \alpha)}{q_{\psi}(\mathbf{Z} | \mathbf{Y})} q_{\psi}(\mathbf{Z} | \mathbf{Y}) d\mathbf{Z} \\
 &\geq \int \log \left(\frac{p(\mathbf{Y} | \mathbf{z}, \phi, \sigma_y^2) p(\mathbf{Z} | \mathbf{X}, \mathbf{W}, \theta, \alpha)}{q_{\psi}(\mathbf{Z} | \mathbf{Y})} \right) q_{\psi}(\mathbf{Z} | \mathbf{Y}) d\mathbf{Z} \\
 &= \mathbb{E}_{\mathbf{Z} \sim q_{\psi}} \left[\sum_n \log \mathcal{N}(\mathbf{y}_n | g_{\phi}(\mathbf{z}_n), \sigma_y^2 \mathbf{I}_K) + \log p(\mathbf{Z} | \mathbf{X}, \mathbf{W}, \theta, \alpha) \right] - \int \log q_{\psi}(\mathbf{Z} | \mathbf{Y}) q_{\psi}(\mathbf{Z} | \mathbf{Y}) d\mathbf{Z} \\
 &= \mathbb{E}_{\mathbf{Z} \sim q_{\psi}} \left[\sum_n \log \mathcal{N}(\mathbf{y}_n | g_{\phi}(\mathbf{z}_n), \sigma_y^2 \mathbf{I}_K) + \log p(\mathbf{Z} | \mathbf{X}, \mathbf{W}, \theta, \alpha) \right] + \frac{1}{2} \sum_{nl} \log(\boldsymbol{\sigma}_{\psi}^{z^2}(\mathbf{y}_n)_l) + \text{const}
 \end{aligned}$$

1.2 Loss derivation

First, we approximate the expectation by sampling. Specifically, we sample a latent representation $\mathbf{Z}_{\psi} = [\mathbf{z}_{\psi_1}, \dots, \mathbf{z}_{\psi_N}] \in \mathbb{R}^{N \times L}$ from the posterior as:

$$\mathbf{z}_{\psi_n} = \boldsymbol{\mu}_{\psi}^z(\mathbf{y}_n) + \boldsymbol{\epsilon}_n \odot \boldsymbol{\sigma}_{\psi}^z(\mathbf{y}_n), \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{L \times L}), \quad n = 1, \dots, N, \quad (2)$$

where we used the reparametrization trick to separate the noisy generation of samples from the model parameters. The resulting approximate ELBO is

$$\text{ELBO} \approx \sum_n \log \mathcal{N}(\mathbf{y}_n | g_\phi(\mathbf{z}_{\psi_n}), \sigma_y^2 \mathbf{I}_K) + \log p(\mathbf{Z}_\psi | \mathbf{X}, \mathbf{W}, \boldsymbol{\theta}, \alpha) + \frac{1}{2} \sum_{nl} \log(\sigma_\psi^{z_\psi^2}(\mathbf{y}_n)_l) + \text{const}$$

Finally, as we optimize σ_y^2 on a validation set (and not on the training set in order to avoid overfitting), maximization of the approximate ELBO on the training set is equivalent to minimizing the following cost:

$$\mathcal{L}(\phi, \psi, \boldsymbol{\theta}, \alpha, \sigma_y^2) = \underbrace{\frac{1}{K} \sum_i (\mathbf{y}_i - g_\phi(\boldsymbol{\omega}_{\psi_n}))^2}_{\text{reconstruction term}} - \frac{\lambda}{L} \left[\underbrace{\log p(\mathbf{Z}_\psi | \mathbf{X}, \mathbf{W}, \boldsymbol{\theta}, \alpha)}_{\text{latent-space GP model}} + \underbrace{\frac{1}{2} \sum_{nl} \log(\sigma_\psi^{z_\psi^2}(\mathbf{y}_n)_l)}_{\text{regularization}} \right], \quad (3)$$

where K is the number of pixels in the image and λ is a trade-off parameter balancing data reconstruction and latent space prior regularization.

Selection of λ . We select the value of λ based on a standard VAE and use the same value for all compared models. Specifically, we optimize the VAE loss for different values of λ and select the value for which the VAE ELBO is maximal on a validation set. In order to compute the VAE validation ELBO we need to estimate optimal value of σ_y^2 for every value of λ in the grid. Given a certain value of $\lambda = \hat{\lambda}$, the optimal value of σ_y^2 can be estimated as

$$\sigma_y^{2(\text{val})} = \frac{1}{N^{(\text{val})}} \sum_{n=1}^{N^{(\text{val})}} \left(\mathbf{y}_n^{(\text{val})} - g_{\phi_{\hat{\lambda}}}(\mathbf{z}_{\psi_{\hat{\lambda}} n}^{(\text{val})}) \right)^2, \quad (4)$$

where $N^{(\text{val})}$ is the number of samples in the validation set and $(\phi_{\hat{\lambda}}, \psi_{\hat{\lambda}})$ are the values of the encoder/decoder parameters after training for $\lambda = \hat{\lambda}$.

1.3 Fast low-rank computations

Parameter inference in Gaussian models scales cubically with the number of observations. To preform fast parameter inference in our Gaussian model, we use the fact that the total covariance is the sum of low-rank matrix and the identity matrix:

$$\mathbf{K} = \mathbf{V}\mathbf{V}^T + \alpha \mathbf{I} \quad (5)$$

where $\mathbf{V} \in \mathbb{R}^{N \times O}$ and $O \ll N$. Using the Woodbury identity [1] and the determinant lemma [2], the linear system $\mathbf{K}^{-1}\mathbf{M}$ with $\mathbf{M} \in \mathbb{R}^{N \times K}$ and the log determinant of \mathbf{K} can be computed as:

$$\mathbf{K}^{-1}\mathbf{M} = \frac{1}{\alpha} \mathbf{I} - \frac{1}{\alpha} \mathbf{V}(\alpha \mathbf{I} + \mathbf{V}^T \mathbf{V})^{-1} \mathbf{V}^T \mathbf{M}, \quad (6)$$

$$\log \det \mathbf{K} = N \log \det \alpha + \log \det (\mathbf{I} + \frac{1}{\alpha} \mathbf{V}^T \mathbf{V}), \quad (7)$$

which have $O(NO^2 + O^3 + ONK)$ and $O(NO^2 + O^3)$ complexities, respectively.

1.4 Implementation of low-memory stochastic backpropagation

For a single latent dimension the GP prior introduces the following term in the ELBO:

$$\log p(\mathbf{z}_\psi | \mathbf{X}, \mathbf{W}, \boldsymbol{\theta}, \alpha) = -\frac{1}{2} \mathbf{z}_\psi^T \mathbf{K}_\theta(\mathbf{X}, \mathbf{V})^{-1} \mathbf{z}_\psi - \frac{1}{2} \log \det \mathbf{K}_\theta(\mathbf{X}, \mathbf{V}) \quad (8)$$

In the equation above, \mathbf{z}_ψ functionally depends on the image space representations and thus a naive computation of full gradient descent would require loading the entire high-dimensional dataset in memory, which would be unfeasible for many high-dimensional image datasets. To overcome this limitation, we recast computations in a form where full-matrix operations only take place in the low dimensional space, while the dependency on the nested derivatives, which involve high memory loads, is linearized and mini-batch operations are therefore allowed. This can be achieved by considering a first-order Taylor expansion of the GP prior term. Specifically, using that $\mathbf{K}_\theta = \mathbf{V}_\theta \mathbf{V}_\theta^T + \alpha \mathbf{I}$, and collecting all parameters in $\boldsymbol{\xi} = \{\psi, \theta, \alpha\}$, we can rewrite the term in Eq. (8) in this functional form $f(\mathbf{z}(\boldsymbol{\xi}), \mathbf{V}(\boldsymbol{\xi}), \alpha(\boldsymbol{\xi}))$. The first-order Taylor expansion of $f(\mathbf{z}(\boldsymbol{\xi}), \mathbf{V}(\boldsymbol{\xi}), \alpha(\boldsymbol{\xi}))$ around $(\mathbf{z}_{\boldsymbol{\xi}_0}, \mathbf{V}_{\boldsymbol{\xi}_0}, \alpha_{\boldsymbol{\xi}_0})$ is:

$$f(\mathbf{z}(\boldsymbol{\xi}), \mathbf{V}(\boldsymbol{\xi}), \alpha(\boldsymbol{\xi})) \approx \mathbf{a}^T \mathbf{z}(\boldsymbol{\xi}) + \text{tr}(\mathbf{B}^T \mathbf{V}(\boldsymbol{\xi})) + c\alpha(\boldsymbol{\xi}) + \text{const.} \quad (9)$$

where

$$\mathbf{a} = \left(\frac{\partial f}{\partial \mathbf{z}} \right)_{\boldsymbol{\xi}_0} = (\mathbf{K}^{-1} \mathbf{z})_{\boldsymbol{\xi}_0} \quad (10)$$

$$\mathbf{B} = \left(\frac{\partial f}{\partial \mathbf{V}} \right)_0 = (-\mathbf{K}^{-1} \mathbf{z} \mathbf{z}^T \mathbf{K}^{-1} \mathbf{V} + \mathbf{K}^{-1} \mathbf{V})_{\boldsymbol{\xi}_0} \quad (11)$$

$$c = \left(\frac{\partial f}{\partial \alpha} \right)_{\boldsymbol{\xi}_0} = \frac{1}{2} (-\mathbf{z}^T \mathbf{K}^{-1} \mathbf{K}^{-1} \mathbf{z} + \text{tr}(\mathbf{K}^{-1}))_{\boldsymbol{\xi}_0} \quad (12)$$

Note that this approximation, applied at every step of gradient descent, locally preserves the gradients. Thus we can engineer the following low-memory four-step procedure for full gradient descent:

- produce and store latent noise realizations ($\boldsymbol{\epsilon}$) used for the reparametrization trick;
- obtain latent variable representations \mathbf{z} , combining the outputs of the encoder and the noise $\boldsymbol{\epsilon}$. We employ mini-batch forward propagation for this step;
- evaluate \mathbf{a} , \mathbf{B} and c across all samples. Note that this step has low-memory requirements as it only involves low-dimensional representations;
- exploit the local Taylor expansion as a proxy for our optimization. As this function is linear in the data, we can accumulate its gradient in a mini-batch fashion, by passing mini-batches of $\boldsymbol{\epsilon}$, \mathbf{Y} , \mathbf{a} and \mathbf{B} ;
- update the parameters $\boldsymbol{\xi}$ using the full gradients as in standard gradient descent.

Finally, we note that in our specific setting \mathbf{a} , \mathbf{B} and c can be computed linearly on the number of sample because of the low rank structure of \mathbf{K} (see previous section).

1.5 Out of sample predictions.

We here derive an approximate predictive posterior for GPPVAE. Specifically, given training images \mathbf{Y} , object representations \mathbf{X} and view representations \mathbf{V} the predictive posterior for image \mathbf{y}_\star representing

object \mathbf{x}_\star in view \mathbf{v}_\star is

$$p(\mathbf{y}_\star | \mathbf{x}_\star, \mathbf{v}_\star, \mathbf{Y}, \mathbf{X}, \mathbf{V}) = \frac{p(\mathbf{y}_\star, \mathbf{Y} | \mathbf{x}_\star, \mathbf{v}_\star, \mathbf{X}, \mathbf{V})}{p(\mathbf{Y} | \mathbf{X}, \mathbf{V})} \quad (13)$$

$$= \frac{1}{p(\mathbf{Y} | \mathbf{X}, \mathbf{V})} \int p(\mathbf{y}_\star | \mathbf{z}_\star) p(\mathbf{Y} | \mathbf{Z}) \underbrace{p(\mathbf{z}_\star, \mathbf{Z} | \mathbf{x}_\star, \mathbf{v}_\star, \mathbf{X}, \mathbf{V})}_{\text{joint distribution}} d\mathbf{z}_\star d\mathbf{Z} \quad (14)$$

$$= \frac{1}{p(\mathbf{Y} | \mathbf{X}, \mathbf{V})} \int p(\mathbf{y}_\star | \mathbf{z}_\star) p(\mathbf{Y} | \mathbf{Z}) \underbrace{p(\mathbf{z}_\star | \mathbf{x}_\star, \mathbf{v}_\star, \mathbf{Z}, \mathbf{X}, \mathbf{V})}_{\text{predictive posterior}} \underbrace{p(\mathbf{Z} | \mathbf{X}, \mathbf{V})}_{\text{GP prior on training}} d\mathbf{z}_\star d\mathbf{Z} \quad (15)$$

$$= \int p(\mathbf{y}_\star | \mathbf{z}_\star) \underbrace{p(\mathbf{z}_\star | \mathbf{x}_\star, \mathbf{v}_\star, \mathbf{Z}, \mathbf{X}, \mathbf{V})}_{\text{predictive posterior}} \underbrace{\frac{p(\mathbf{Y} | \mathbf{Z}) p(\mathbf{Z} | \mathbf{X}, \mathbf{V})}{p(\mathbf{Y} | \mathbf{X}, \mathbf{V})}}_{\text{posterior on } \mathbf{Z}} d\mathbf{z}_\star d\mathbf{Z} \quad (16)$$

$$\approx \int p(\mathbf{y}_\star | \mathbf{z}_\star) \underbrace{p(\mathbf{z}_\star | \mathbf{x}_\star, \mathbf{v}_\star, \mathbf{Z}, \mathbf{X}, \mathbf{V})}_{\text{predictive posterior}} \underbrace{q(\mathbf{Z} | \mathbf{Y})}_{\text{approx. post. on } \mathbf{Z}} d\mathbf{z}_\star d\mathbf{Z} \quad (17)$$

where we dropped dependencies from model parameters to simplify the notation.

2 Supplementary Figures

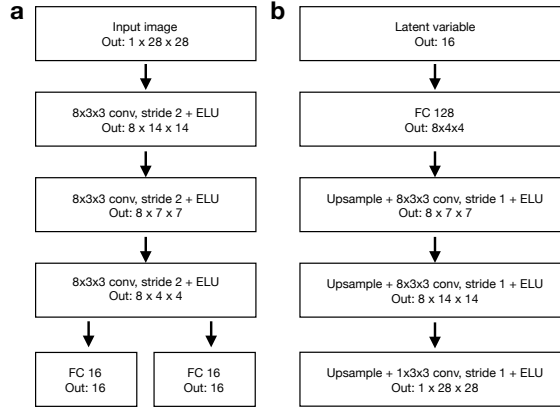


Figure S1 Neural network architecture used in the MNIST experiments. (a) Encoder architecture; (b) Decoder architecture. The same encoder/decoder architectures are used for both VAE and GPPVAE. For CVAE, we still use the same architecture but we provide view representations (rotation angles) as inputs to both the encoder and the decoder. For the encoder, we provide rotation angles to the first layer (by adding additional channels to the input image) and again before the dense layer (by stacking them to the vector produced by the last convolution). Similarly, for the decoder, we provide rotation angles as inputs to the first layer (by stacking them with the 16-dimensional latent variable) and again before the first convolution layer (by adding them as additional channels to the 8x8x4 tensor fed to the first convolution layer). For a fair comparison with GPPVAE, we account for periodicity in the view representation by providing both the angle sine and cosine, i.e. $\mathbf{w} = [\sin\phi, \cos\phi]$, where ϕ is the rotation angle.

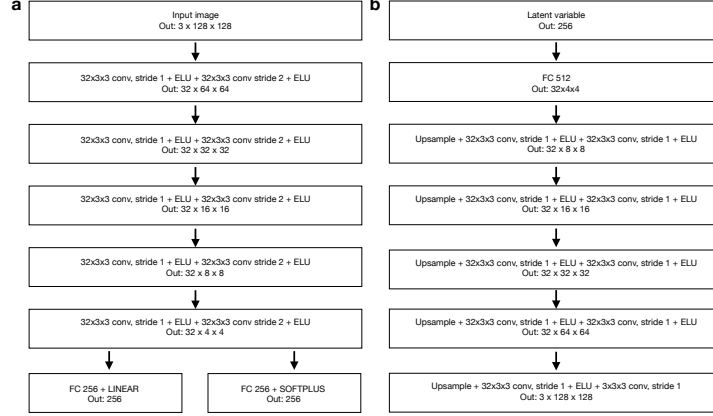


Figure S2 Neural network architecture used in the face dataset experiments. (a) Encoder architecture; (b) Decoder architecture. The same encoder/decoder architectures are used for both VAE and GPPVAE. For CVAE, we still use the same architecture but we provide view information as one hot encoding of the 9 different poses to both the encoder and the decoder networks. For the encoder, we provide view representations to the first layer (by adding additional channels to the input image) and again before the dense layer (by stacking them to the vector produced by the last convolution). Similarly, for the decoder, we provide view representations as inputs to the first layer (by stacking them with the 16-dimensional latent variable) and again before the first convolution layer (by adding them as additional channels to the $8 \times 8 \times 4$ tensor fed to the first convolution layer).

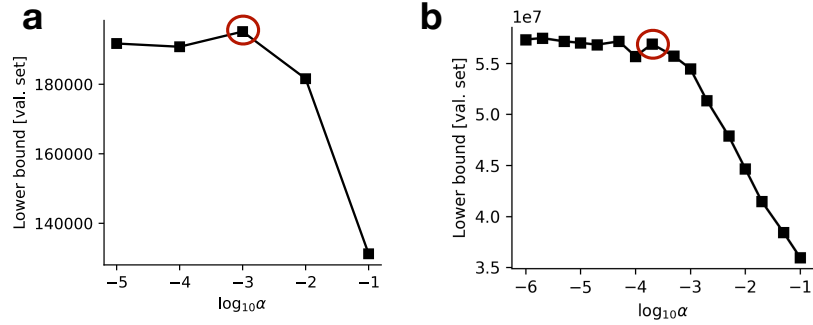


Figure S3 Selection of the trade-off parameter between fit-to-data and latent-space model. The trade-off parameter was selected as to maximize VAE ELBO. We here show the VAE validation lower bound as a function of the trade-off parameter for the MNIST (a) and the face dataset (b).

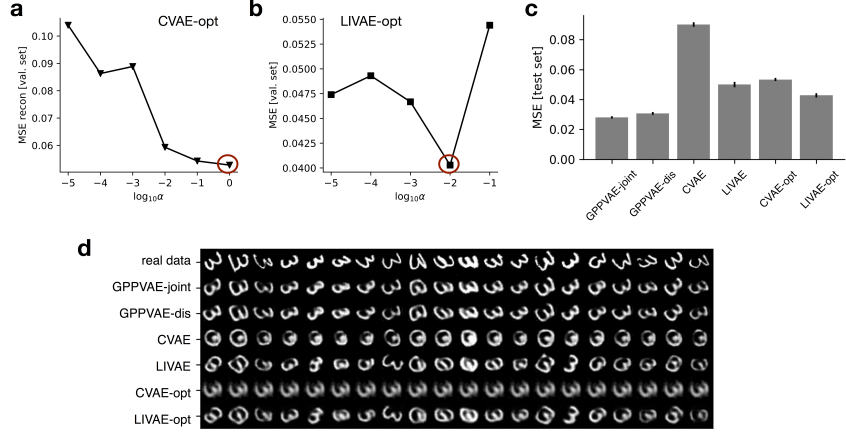


Figure S4 For CVAE and LIVAE, we also considered the alternative strategy of selecting the value of the trade-off parameter λ that maximizes prediction performance on the validation set. We refer to these additional methods as CVAE-opt and LIVAE-opt. We here show the selection of λ for CVAE-opt (a) and LIVAE-opt (b) and their performance (c-d) for the experiments in MNIST.

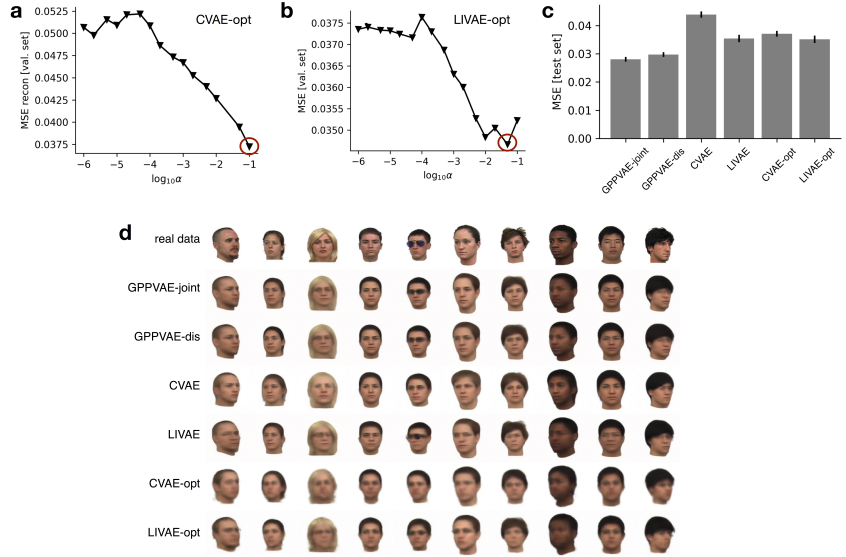


Figure S5 Analogous to Figure S4 but for the face data experiments.

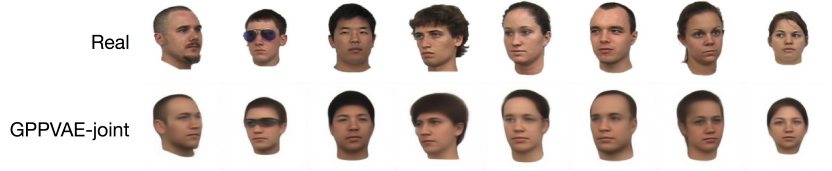


Figure S6 Out-of-sample predictions obtained from GPPVAE-joint when considering the perceptual loss introduced in [3].

References

- [1] Henderson, H. V. & Searle, S. R. On deriving the inverse of a sum of matrices. *Siam Review* **23**, 53–60 (1981).
- [2] Harville, D. A. *Matrix algebra from a statistician's perspective*, vol. 1 (Springer, 1997).
- [3] Hou, X., Shen, L., Sun, K. & Qiu, G. Deep feature consistent variational autoencoder. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, 1133–1141 (IEEE, 2017).