A Experimental details

Here we report the details and parameter settings for our experiments to foster reproducibility.

A.1 Moving ball experiment

For the moving ball experiment described in Section 4.1, we use the same neural networks architectures and training setting as in Pearce (2020).

Table A.1: Parameter settings for the moving ball experiment.

Parameter	Value
Nr. of feedforward layers in inference network	2
Nr. of feedforward layers in generative network	2
Width of a hidden feedforward layer	500
Dimensionality of latent space (L)	2
Activation function	*tanh*
Learning rate	0.001
Optimizer	Adam
Nr. of epochs	25000
Nr. of frames in each video (N)	30
Dimension of each frame	32×32

A squared-exponential GP kernel with length scale $l = 2$ was used. For the exact data generation procedure, we refer to Pearce (2020). During training, 35 videos were generated in each epoch. The test MSE is reported on a held-out set of 350 videos. For the Adam optimizer (Kingma and Ba, 2014), the default Tensforflow parameters are used.

A.2 MNIST experiment

For the rotated MNIST experiment described in Section 4.2, we used the same neural networks architectures as in Casale et al. (2018): three convolutional layers followed by a fully connected layer in the inference network and vice-versa in the generative network.

Table A.2: Neural networks architectures for the MNIST experiment.

Parameter	Value
Nr. of CNN layers in inference network	3
Nr. of CNN layers in generative network	3
Nr. of filters per CNN layer	8
Filter size	3×3
Nr. of feedforward layers in inference network	1
Nr. of feedforward layers in generative network	1
Activation function in CNN layers	ELU
Dimensionality of latent space (L)	16

The SVGP-VAE model is trained for 1000 epochs with a batch size of 256. The Adam optimizer (Kingma and Ba, 2014) is used with its default parameters and a learning rate of 0.001. Moreover, the GECO algorithm (Rezende and Viola, 2018) was used for training our SVGP-VAE model in this experiment. The reconstruction parameter in GECO was set to $\kappa = 0.020$ in all reported experiments.

For the GP-VAE model from Casale et al. (2018), we used the same training procedure as reported in Casale et al. (2018). We have observed in our reimplementation that a joint optimization at the end does not improve performance. Hence, we report results for the regime where the VAE parameters are optimized for the first 100 epochs, followed by 100 epochs during which the GP parameters are optimized. Moreover, we could not get their

proposed low-memory modified forward pass to work, so in our reimplementation the entire dataset is loaded into the memory at one point during the forward pass. Our reimplementation of the GP-VAE model from Casale et al. (2018) is publicly available at `https://github.com/ratschlab/SVGP-VAE`.

For both models, the GP kernel proposed in Casale et al. (2018) is used. For more details on the kernel, we refer to Appendix B.3. Note that the auxiliary data $\mathbf{X}$ is only partially observed in this experiment — for both models we use a GP-LVM to learn the missing parts of $\mathbf{X}$. For both models, we use Principal Component Analysis (PCA) to initialize the GP-LVM vectors, as it was observed to lead to a slight increase in performance. PCA is also used in SVGP-VAE to initialize the inducing points. For more details see Appendix C.1.

A.3 SPRITES experiment

For the SPRITES experiment described in Section 4.3, we used similar neural networks architectures as for the rotated MNIST experiment. Details are provided in Table A.3.

Table A.3: Neural networks architectures for the SPRITES experiment.

Parameter	Value
Nr. of CNN layers in inference network	6
Nr. of CNN layers in generative network	6
Nr. of filters per CNN layer	16
Filter size	3×3
Nr. of feedforward layers in inference network	1
Nr. of feedforward layers in generative network	1
Activation function in CNN layers	ELU
Dimensionality of latent space (L)	64

The SVGP-VAE model is trained for 50 epochs with a batch size of 500. The Adam optimizer (Kingma and Ba, 2014) is used with its default parameters and a learning rate of 0.001. Moreover, the GECO algorithm (Rezende and Viola, 2018) was used for training our SVGP-VAE model in this experiment. The reconstruction parameter in GECO was set to $\kappa = 0.0075$.

The auxiliary data $\mathbf{X}$ is fully unobserved in this experiment. Recall that in SPRITES, the auxiliary data has two parts $\mathbf{X} = [\mathbf{X}_s,\, \mathbf{X}_a]$, with $\mathbf{X}_s \in \mathbb{R}^{N \times p_1}$ containing information about the character **s**tyle and $\mathbf{X}_a \in \mathbb{R}^{N \times p_2}$ containing information about the specific **a**ction/pose. Let $\mathbf{x}_i = [\mathbf{x}_{s,i}\, \mathbf{x}_{a,i}]$ denote auxiliary data for the i-th image (corresponding to the i-th row of the $\mathbf{X}$ matrix). A product kernel between two linear kernels is used:[3]

$$k_\theta(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_{s,i}^T \mathbf{x}_{s,j}}{\|\mathbf{x}_{s,i}\| \|\mathbf{x}_{s,j}\|} \cdot \frac{\mathbf{x}_{a,i}^T \mathbf{x}_{a,j}}{\|\mathbf{x}_{a,i}\| \|\mathbf{x}_{a,j}\|} + \sigma^2 \cdot \delta_{ij} \,.$$

The kernel normalization and the addition of the diagonal noise are used to improve the numerical stability of kernel matrices.

To learn the action part of the auxiliary data $\mathbf{X}_a$, we rely on a GP-LVM (Lawrence, 2004), that is, we try to directly learn the matrix $\boldsymbol{A} \in \mathbb{R}^{72 \times p_2}$ consisting of GP-LVM vectors that each represent a specific action/pose. Since we want to extrapolate to new characters during the test phase[4], the GP-LVM approach can not be used to learn the part of the auxiliary data that captures the character style information $\mathbf{X}_s$. This would require rerunning the optimization at test time to obtain a corresponding GP-LVM vector for the new, previously unseen style. To get around this, we introduce the *representation network* $r_\zeta : \mathbb{R}^K \to \mathbb{R}^{p_1}$, similar to what is done in Eslami et al. (2018b), with which we aim to amortize the learning of the unobserved parts of the auxiliary data. Specifically, the representation for the i-th character style is then

$$\mathbf{s}_i = f\big(r_\zeta(\mathbf{y}_1),\, \ldots,\, r_\zeta(\mathbf{y}_{N_i})\big) \in \mathbb{R}^{p_1} \,,$$

[3] $\delta_{ij} = 1$ if $i = j$ and 0 else.

[4] Note that an easier version of the SPRITES experiment would be to generate actions for characters already seen during the training phase. Such a conditional generation task would closely resemble the one from the face experiment in Casale et al. (2018).

where $\mathbf{Y}_i = [\mathbf{y}_1 \dots \mathbf{y}_{N_i}]^T \in \mathbb{R}^{N_i \times K}$ represents all images of the i-th character, and f is a chosen aggregation function (in our experiment we used the sum function). Instead of the GP-LVM vectors, the parameters of the representation network ζ are jointly optimized with the rest of the SVGP-VAE parameters. During training, we pass all 50 images (50 different actions) for each character through r_ζ to obtain the corresponding style representation. During the test phase, we first pass 36 actions through r_ζ and then use the resulting style representation vector to conditionally generate the remaining 36 actions. To help with the stability of training, we additionally pretrain the representation network on the classification task using the training data. Concretely, we train a classifier on top of the representations of the training data $r_\zeta(\mathbf{y}_i)$, $i = 1, ..., N$. The (pretraining) label for each representation is a given character ID.[5]

The details on the architecture of the representation network are provided in Table A.4 (it is essentially a downsized inference network).

Table A.4: The architecture for the representation network r_ζ and some additional parameters in the SPRITES experiment.

Parameter	Value
Nr. of CNN layers	3
Nr. of filters per CNN layer	16
Filter size	2×2
Nr. of pooling layers	1
Activation function in CNN layers	ELU
Dimensionality of style representation (p_1)	16
Dimensionality of action GP-LVM vectors (p_2)	8
Nr. of epochs for pretraining of r_ζ	400

A.4 On the training of GP-VAE models (a practitioner's perspective)

While working on implementations of different GP-VAE models, we have noticed that balancing the absolute magnitudes of the reconstruction and the KL-term is critical for achieving optimal results, even more so than in standard VAE models. In Fortuin et al. (2020), this was tackled by introducing a weighting β parameter, whereas in Casale et al. (2018) a CV search on the noise parameter σ_y^2 of the likelihood $p_\psi(\mathbf{y}_i|\mathbf{z}_i)$ is performed. One downside of both solutions is that they introduce (yet) another training hyperparameter that needs to be manually tuned for every new dataset/model architecture considered.

To get around this, we instead used the GECO algorithm (Rezende and Viola, 2018) to train our SVGP-VAE in the rotated MNIST experiment. Compared to the original GECO algorithm in Rezende and Viola (2018), where the maximization objective is the KL divergence between a standard Gaussian prior and the variational distribution, the GECO maximization objective in the SVGP-VAE is composed of a cross-entropy term $\mathbb{E}_{q_S}[\log \tilde{q}_\phi(\cdot)]$ and a sparse GP ELBO $\mathcal{L}_H(\cdot)$. We have observed that GECO greatly simplifies training of GP-VAE models as it eliminates the need to manually tune the different magnitudes of the ELBO terms. Based on this, we would make a general recommendation for GECO to be used for training such models.

[5]Recall that there are 1000 different characters in our training dataset, i.e., the pretraining task is a 1000-class classification problem.

B Supporting derivations

B.1 Vanishing of the inference network in GP-VAE with ELBO from Hensman et al. (2013)

In this section we show that working with the sparse GP approach presented in Hensman et al. (2013) leads to vanishing of the inference network parameters ϕ in the GP-VAE model from Pearce (2020). Recall that the sparse GP posterior from Hensman et al. (2013) for the l-th latent channel has the form

$$q_S^l(\mathbf{z}_{1:N}^l|\cdot) = \mathcal{N}\big(\mathbf{z}_{1:N}^l|\mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\boldsymbol{\mu}^l,\, \mathbf{K}_{NN} - \mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mN} + \mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\mathbf{A}^l\mathbf{K}_{mm}^{-1}\mathbf{K}_{mN}\big)$$

with $\boldsymbol{\mu}^l \in \mathbb{R}^m, \mathbf{A}^l \in \mathbb{R}^{m\times m}$ as free variational parameters, while the sparse GP ELBO for the l-th latent channel is given as

$$\mathcal{L}_H^l(\mathbf{U}, \boldsymbol{\mu}^l, \mathbf{A}^l, \phi, \theta) = \sum_{i=1}^N \left\{ \log \mathcal{N}\big(\tilde{y}_{l,i} \mid \boldsymbol{k}_i^T\mathbf{K}_{mm}^{-1}\boldsymbol{\mu}^l,\, \tilde{\sigma}_{l,i}^{-2}\big) - \frac{1}{2\tilde{\sigma}_{l,i}^{-2}}\big(\tilde{k}_{ii} + Tr(\mathbf{A}^l\,\Lambda_i)\big)\right\} - KL\big(q_S^l(\mathbf{f}_m|\cdot)\,||\,p_\theta(\mathbf{f}_m|\cdot)\big)$$

with $q_S^l(\mathbf{f}_m|\cdot) = \mathcal{N}(\mathbf{f}_m|\boldsymbol{\mu}^{\boldsymbol{l}}, \boldsymbol{A}^{\boldsymbol{l}})$ and $p_\theta(\mathbf{f}_m|\cdot) = \mathcal{N}(\mathbf{f}_m|\mathbf{0},\, \mathbf{K}_{mm})$. $\boldsymbol{k}_i$ represents the i-th column of $\mathbf{K}_{mN}$, $\Lambda_i := \mathbf{K}_{mm}^{-1}\boldsymbol{k}_i\boldsymbol{k}_i^T\mathbf{K}_{mm}^{-1}$ and $\tilde{k}_{ii}$ is the i-th diagonal element of $\mathbf{K}_{NN} - \mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mN}$. As mentioned in Section 3, $\mathcal{L}_H^l$ depends on the inference network parameters ϕ through the (amortized) l-th latent dataset $\tilde{\mathbf{y}}_l = \mu_\phi^l(\mathbf{Y})$, $\tilde{\boldsymbol{\sigma}}_l = \sigma_\phi^l(\mathbf{Y})$.

Note that the full sparse GP posterior equals $q_S(\mathbf{Z}) = \prod_{l=1}^L q_S^l(\mathbf{z}_{1:N}^l|\cdot)$. Similarly, the full sparse GP ELBO is $\mathcal{L}_H = \sum_{l=1}^L \mathcal{L}_H^l(\mathbf{U}, \boldsymbol{\mu}^l, \mathbf{A}^l, \phi, \theta)$.

Proposition B.1 *For the l-th latent channel in the GP-VAE model with the bound from Hensman et al. (2013), the following relation holds:*

$$\mathbb{E}_{q_S^l}\big[\log \tilde{q}_\phi(\boldsymbol{z}_{1:N}^l|\,\boldsymbol{Y})\big] = \sum_{i=1}^N \left\{ \log \mathcal{N}\big(\tilde{y}_{l,i} \mid \boldsymbol{k}_i^T\boldsymbol{K}_{mm}^{-1}\boldsymbol{\mu}^l,\, \tilde{\sigma}_{l,i}^{-2}\big) - \frac{1}{2\tilde{\sigma}_{l,i}^{-2}}\big(\tilde{k}_{ii} + Tr(\boldsymbol{A}^l\,\Lambda_i)\big)\right\}.$$

Proof For notational convenience, define $\tilde{D}_l := \mathrm{diag}(\tilde{\boldsymbol{\sigma}}_l^2)$ and $\boldsymbol{B} := \mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}$. Also recall that $\tilde{q}_\phi(\mathbf{z}_{1:N}^l|\mathbf{Y}) = \mathcal{N}\big(\mathbf{z}_{1:N}^l|\tilde{\mathbf{y}}_l,\, \tilde{D}_l\big)$. Using the formula for the cross-entropy between two multivariate Gaussian distributions, we proceed as

$$\begin{aligned}\mathbb{E}_{q_S^l}\big[\log \tilde{q}_\phi(\mathbf{z}_{1:N}^l|\mathbf{Y})\big] &= -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\tilde{D}_l| - \frac{1}{2}\big(\tilde{\mathbf{y}}_l - \boldsymbol{B}\boldsymbol{\mu}^l\big)^T\tilde{D}_l^{-1}\big(\tilde{\mathbf{y}}_l - \boldsymbol{B}\boldsymbol{\mu}^l\big) - \frac{1}{2}Tr\big(\tilde{D}_l^{-1}(\tilde{\mathbf{K}} + \boldsymbol{B}\boldsymbol{A}^l\boldsymbol{B}^T)\big)\\ &= \log\mathcal{N}\big(\tilde{\mathbf{y}}_l|\boldsymbol{B}\boldsymbol{\mu}^{\boldsymbol{l}},\, \tilde{D}_l\big) - \frac{1}{2}Tr(\tilde{D}_l^{-1}\tilde{\mathbf{K}}) - \frac{1}{2}Tr(\tilde{D}_l^{-1}\boldsymbol{B}\boldsymbol{A}^l\boldsymbol{B}^T)\,.\end{aligned}$$

It remains to show that the last trace term equals $\sum_{i=1}^N \tilde{\sigma}_{l,i}^{-2}Tr(\mathbf{A}^l\,\Lambda_i)$, which follows from

$$Tr(\tilde{D}_l^{-1}\boldsymbol{B}\boldsymbol{A}^l\boldsymbol{B}^T) = Tr(\boldsymbol{A}^l\boldsymbol{B}^T\tilde{D}_l^{-1}\boldsymbol{B}) = Tr\big(\boldsymbol{A}^l\mathbf{K}_{mm}^{-1}\big(\sum_{i=1}^N \tilde{\sigma}_{l,i}^{-2}\boldsymbol{k}_i\boldsymbol{k}_i^T\big)\mathbf{K}_{mm}^{-1}\big) = \sum_{i=1}^N \tilde{\sigma}_{l,i}^{-2}Tr(\boldsymbol{A}^l\Lambda_i)\,. \qquad \square$$

Proposition B.2 *The GP-VAE ELBO with the bound from Hensman et al. (2013) reduces to*

$$\mathcal{L}_{PH}(\boldsymbol{U}, \psi, \theta, \boldsymbol{\mu}^{1:L}, \boldsymbol{A}^{1:L}) = \sum_{i=1}^N \mathbb{E}_{q_S}\Big[\log p_\psi(\boldsymbol{y}_i|\boldsymbol{z}_i)\Big] - \sum_{l=1}^L KL\big(q_S^l(\boldsymbol{f}_m|\cdot)\,||\,p_\theta^l(\boldsymbol{f}_m|\cdot)\big)$$

Proof Using the above proposition, we have

$$\begin{aligned}
&\mathbb{E}_{q_S}\Big[\sum_{i=1}^{N}\log p_\psi(\mathbf{y}_i|\mathbf{z}_i) - \log\tilde{q}_\phi(\mathbf{z}_i|\mathbf{y}_i)\Big] + \sum_{l=1}^{L}\mathcal{L}_H^l \\
&= \mathbb{E}_{q_S}\Big[\sum_{i=1}^{N}\log p_\psi(\mathbf{y}_i|\mathbf{z}_i)\Big] - \mathbb{E}_{q_S}\Big[\sum_{l=1}^{L}\log\tilde{q}_\phi(\mathbf{z}_{1:N}^l|\mathbf{Y})\Big] + \sum_{l=1}^{L}\mathcal{L}_H^l \\
&= \mathbb{E}_{q_S}\Big[\sum_{i=1}^{N}\log p_\psi(\mathbf{y}_i|\mathbf{z}_i)\Big] - \sum_{l=1}^{L}\Big(\mathbb{E}_{q_S^l}\big[\log\tilde{q}_\phi(\mathbf{z}_{1:N}^l|\mathbf{Y})\big] - \mathcal{L}_H^l\Big) \\
&= \mathbb{E}_{q_S}\Big[\sum_{i=1}^{N}\log p_\psi(\mathbf{y}_i|\mathbf{z}_i)\Big] - \sum_{l=1}^{L}KL\big(q_S^l(\mathbf{f}_m|\cdot)\,||\,p_\theta(\mathbf{f}_m|\cdot)\big)\,.
\end{aligned}$$

□

Observe that in $\mathcal{L}_{PH}(\cdot)$ all terms that include $\tilde{\mathbf{y}}_l$ or $\tilde{\boldsymbol{\sigma}}_l$ cancel out, hence such ELBO is independent of the inference network parameters ϕ.

B.2 Monte Carlo estimators in the SVGP-VAE

The idea behind the estimators used in q_S in our SVGP-VAE is based on the work presented in Evans and Nair (2020). The main insight is to rewrite the matrix operations as expectations with respect to the empirical distribution of the training data. Those expectations are then approximated with Monte Carlo estimators.

Recall that the (amortized) latent dataset for the l-th channel is denoted by $\{\mathbf{X}, \tilde{\mathbf{y}}_l, \tilde{\boldsymbol{\sigma}}_l\}$, with $\tilde{\mathbf{y}}_l := \mu_\phi^l(\mathbf{Y})$ and $\tilde{\boldsymbol{\sigma}}_l := \sigma_\phi^l(\mathbf{Y})$. For notational convenience, additionally denote $\tilde{D}_l := \mathrm{diag}(\tilde{\boldsymbol{\sigma}}_l^2)$. First, observe that the matrix product $\mathbf{K}_{mN}\tilde{D}_l^{-1}\mathbf{K}_{Nm}$ in $\boldsymbol{\Sigma}^l$ can be rewritten as a sum over data points $\sum_{i=1}^{N} B_i(\mathbf{x}_i, \mathbf{y}_i)$ with

$$B_i(\mathbf{x}_i, \mathbf{y}_i) := \frac{1}{\tilde{\sigma}_{l,i}^2}\begin{bmatrix} k_\theta(\mathbf{u}_1,\mathbf{x}_i)k_\theta(\mathbf{u}_1,\mathbf{x}_i) & \dots & k_\theta(\mathbf{u}_1,\mathbf{x}_i)k_\theta(\mathbf{u}_m,\mathbf{x}_i) \\ \vdots & \ddots & \vdots \\ k_\theta(\mathbf{u}_m,\mathbf{x}_i)k_\theta(\mathbf{u}_1,\mathbf{x}_i) & \dots & k_\theta(\mathbf{u}_m,\mathbf{x}_i)k_\theta(\mathbf{u}_m,\mathbf{x}_i) \end{bmatrix}.$$

Let $\bar{b}$ represent a set of indices of data points in the current batch with size b. Moreover, define $\mathbf{K}_{bm} \in \mathbb{R}^{b\times m}$, $\tilde{D}_{l,b} \in \mathbb{R}^{b\times b}$, $\tilde{\mathbf{y}}_b^l \in \mathbb{R}^b$ as the sub-sampled versions of $\mathbf{K}_{Nm} \in \mathbb{R}^{N\times m}$, $\tilde{D}_l \in \mathbb{R}^{N\times N}$ and $\tilde{\mathbf{y}}_l \in \mathbb{R}^N$, respectively, consisting only of data points in $\bar{b}$. An (unbiased) Monte Carlo estimator for $\boldsymbol{\Sigma}^l$ is then derived as follows

$$\begin{aligned}
\boldsymbol{\Sigma}^l &= \mathbf{K}_{mm} + \mathbf{K}_{mN}\tilde{D}_l^{-1}\mathbf{K}_{Nm} = \mathbf{K}_{mm} + N\sum_{i=1}^{N}\frac{1}{N}B_i(\mathbf{x}_i,\mathbf{y}_i) = \mathbf{K}_{mm} + N\cdot\mathbb{E}_{i\sim\{1,\dots,N\}}\big[B_i(\mathbf{x}_i,\mathbf{y}_i)\big] \\
&\approx \mathbf{K}_{mm} + \frac{N}{b}\sum_{i\in\bar{b}}B_i(\mathbf{x}_i,\mathbf{y}_i) = \mathbf{K}_{mm} + \frac{N}{b}\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\mathbf{K}_{bm} =: \boldsymbol{\Sigma}_b^l\,.
\end{aligned}$$

Additionally, define $\boldsymbol{c}_l := \mathbf{K}_{mN}\tilde{D}_l^{-1}\tilde{\mathbf{y}}_l$ and proceed similarly as above

$$\boldsymbol{c}_l = \sum_{i=1}^{n} b_i(\mathbf{x}_i,\mathbf{y}_i) = N\cdot\mathbb{E}_{i\sim\{1,\dots,N\}}[b_i(\mathbf{x}_i,\mathbf{y}_i)] \approx \frac{N}{b}\sum_{i\in\bar{b}}b_i(\mathbf{x}_i,\mathbf{y}_i) = \frac{N}{b}\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\tilde{\mathbf{y}}_b^l =: \boldsymbol{c}_b^l\,,$$

where

$$b_i(\mathbf{x}_i,\mathbf{y}_i) := \frac{\tilde{y}_{l,i}}{\tilde{\sigma}_{l,i}^2}\begin{bmatrix} k_\theta(\mathbf{u}_1,\mathbf{x}_i) \\ \vdots \\ k(\mathbf{u}_m,\mathbf{x}_i) \end{bmatrix}.$$

The estimators for $\boldsymbol{\mu}_T^l$ and $\boldsymbol{A}_T^l$ are then obtained using a plug-in approach,

$$\boldsymbol{\mu}_T^l = \mathbf{K}_{mm}(\boldsymbol{\Sigma}^l)^{-1}\boldsymbol{c}_l \approx \mathbf{K}_{mm}(\boldsymbol{\Sigma}_b^l)^{-1}\boldsymbol{c}_b^l =: \boldsymbol{\mu}_b^l \,,$$
$$\boldsymbol{A}_T^l = \mathbf{K}_{mm}(\boldsymbol{\Sigma}^l)^{-1}\mathbf{K}_{mm} \approx \mathbf{K}_{mm}(\boldsymbol{\Sigma}_b^l)^{-1}\mathbf{K}_{mm} =: \boldsymbol{A}_b^l \,.$$

Note that neither of the above estimators is unbiased, since both depend on the inverse $(\boldsymbol{\Sigma}_b^l)^{-1}$. For the empirical investigation of the magnitude of the bias, see Appendix C.4. However, $\boldsymbol{A}_b^l$ can be shown to be approximately (up to the first order Taylor approximation) unbiased.

Proposition B.3 *For the estimator $\boldsymbol{A}_b^l$ in SVGP-VAE, it holds that*

$$\mathbb{E}[\boldsymbol{A}_b^l] - \boldsymbol{A}_T^l \approx 0 \,.$$

Proof Note that expectation here is taken with respect to the empirical distribution of the training data, that is, $\mathbb{E}_{i\sim\{1,\dots,N\}}$. Using the definitions of $\boldsymbol{A}_b^l$ and $\boldsymbol{A}_T^l$, we get

$$\mathbb{E}[\boldsymbol{A}_b^l] - \boldsymbol{A}_T^l = \mathbf{K}_{mm}\big(\mathbb{E}\big[(\boldsymbol{\Sigma}_b^l)^{-1}\big] - (\boldsymbol{\Sigma}^l)^{-1}\big)\mathbf{K}_{mm} \,,$$

so it remains to show that $\mathbb{E}[(\boldsymbol{\Sigma}_b^l)^{-1}] - (\boldsymbol{\Sigma}^l)^{-1} \approx 0$. To this end, we exploit the positive definiteness of the kernel matrix $\mathbf{K}_{mm}$ and we approximate both inverse terms with the first order Taylor expansion:

$$\begin{aligned}(\boldsymbol{\Sigma}_b^l)^{-1} &= \big(\mathbf{K}_{mm} + \frac{N}{b}\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\mathbf{K}_{bm}\big)^{-1} = \mathbf{K}_{mm}^{-\frac{1}{2}}\big(\mathbf{I} + \frac{N}{b}\mathbf{K}_{mm}^{-\frac{1}{2}}\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\mathbf{K}_{bm}\mathbf{K}_{mm}^{-\frac{1}{2}}\big)^{-1}\mathbf{K}_{mm}^{-\frac{1}{2}} \\ &\approx \mathbf{K}_{mm}^{-\frac{1}{2}}\big(\mathbf{I} - \frac{N}{b}\mathbf{K}_{mm}^{-\frac{1}{2}}\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\mathbf{K}_{bm}\mathbf{K}_{mm}^{-\frac{1}{2}}\big)\mathbf{K}_{mm}^{-\frac{1}{2}} = \mathbf{K}_{mm}^{-1} - \frac{N}{b}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\mathbf{K}_{bm}\mathbf{K}_{mm}^{-1} \,.\end{aligned}$$

Similarly, we have $(\boldsymbol{\Sigma}^l)^{-1} \approx \mathbf{K}_{mm}^{-1} - \mathbf{K}_{mm}^{-1}\mathbf{K}_{mN}\tilde{D}_l^{-1}\mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}$. Using this, we proceed as

$$\begin{aligned}\mathbb{E}[(\boldsymbol{\Sigma}_b^l)^{-1}] - (\boldsymbol{\Sigma}^l)^{-1} &\approx -\frac{N}{b}\mathbf{K}_{mm}^{-1}\mathbb{E}\big[\mathbf{K}_{mb}\tilde{D}_{l,b}^{-1}\mathbf{K}_{bm}\big]\mathbf{K}_{mm}^{-1} + \mathbf{K}_{mm}^{-1}\mathbf{K}_{mN}\tilde{D}_l^{-1}\mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1} \\ &= -\frac{N}{b}\mathbf{K}_{mm}^{-1}\mathbb{E}\Big[\sum_{i\in b} B_i(\mathbf{x}_i, \mathbf{y}_i)\Big]\mathbf{K}_{mm}^{-1} + \mathbf{K}_{mm}^{-1}\Big(\sum_{i=1}^{N} B_i(\mathbf{x}_i, \mathbf{y}_i)\Big)\mathbf{K}_{mm}^{-1} \\ &= -N\mathbf{K}_{mm}^{-1}\mathbb{E}\big[B_i(\mathbf{x}_i, \mathbf{y}_i)\big]\mathbf{K}_{mm}^{-1} + N\mathbf{K}_{mm}^{-1}\mathbb{E}\big[B_i(\mathbf{x}_i, \mathbf{y}_i)\big]\mathbf{K}_{mm}^{-1} = 0\end{aligned}$$

□

Note that a similar proof technique unfortunately cannot be used to show that $\boldsymbol{\mu}_b^l$ is approximately unbiased for $\boldsymbol{\mu}_T^l$, due to the product of two plug-in estimators that both depend on the data in the same batch.

B.3 Low-rank kernel matrix in Casale et al. (2018)

In the following, we present an approach from Casale et al. (2018) to reduce the cubic GP complexity in their GP-VAE model. Note that the exact approach is not given in Casale et al. (2018) and the derivation shown here is our best attempt at recreating the results.

In Casale et al. (2018), datasets composed of P unique objects observed in Q unique views are considered, for instance, images of faces captured from different angles. In total, this amounts to $N = P \cdot Q$ images. The auxiliary data consist of two sets of features $\mathbf{X} = \big[\mathbf{X}_o\ \mathbf{X}_v\big]$, with $\mathbf{X}_o \in \mathbb{R}^{N\times p_1}$ containing information about objects (e.g., drawing style of the digit or characteristics of the face) and $\mathbf{X}_v \in \mathbb{R}^{N\times p_2}$ containing information about views (e.g., an angle or position in space). Let $\mathbf{x}_i = [\mathbf{x}_{o,i}\ \mathbf{x}_{v,i}]$ denote auxiliary data for the i-th image (corresponding to the i-th row of the $\mathbf{X}$ matrix). Additionally, denote by $\mathbf{P} \in \mathbb{R}^{P\times p_1}$ and $\mathbf{Q} \in \mathbb{R}^{Q\times p_2}$ matrices consisting of all unique object and view representations, respectively. A product kernel between a linear kernel for object information and a periodic kernel for view information is used:

$$k_\theta(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\Big(-\frac{2\sin^2\big(\|\mathbf{x}_{v,i} - \mathbf{x}_{v,j}\|\big)}{l^2}\Big)\cdot \mathbf{x}_{o,i}^T\mathbf{x}_{o,j} \,,\ \theta = \{\sigma^2, l\} \,.$$

Exploiting the product and (partial) linear structure of the kernel and using properties of the Kronecker product, $\mathbf{K}_{NN}$ can be written in a low-rank form as

$$\mathbf{K}_{NN}(\mathbf{X},\mathbf{X}) = \mathbf{P}\mathbf{P}^T \otimes \mathbf{K}(\mathbf{Q}) = \mathbf{P}\mathbf{P}^T \otimes \mathbf{L}\mathbf{L}^T = \big(\mathbf{P}\otimes\mathbf{L}\big)\big(\mathbf{P}^T\otimes\mathbf{L}^T\big) = \big(\mathbf{P}\otimes\mathbf{L}\big)\big(\mathbf{P}\otimes\mathbf{L}\big)^T =: \mathbf{V}\mathbf{V}^T,$$

where $\mathbf{K}(\mathbf{Q}) \in \mathbb{R}^{Q\times Q}$ is a kernel matrix of all unique view vectors based on the periodic kernel, $\mathbf{L}$ is its Cholesky decomposition and $\mathbf{V} \in \mathbb{R}^{N\times H}$, $H = Q\cdot p_1$, is the obtained low-rank matrix ($H \ll N$ due to the assumption that the number of unique views Q is not large). For such matrices, the inverse and log-determinant can be computed in $O(NH^2)$ using a matrix inversion lemma (Henderson and Searle, 1981) and a matrix determinant lemma (Harville, 1998), respectively.

While the above approach elegantly reduces the GP complexity for a given dataset (for auxiliary data $\mathbf{X}$ with a product structure), it is not readily extensible for other types of datasets (e.g. time series). In contrast, our SVGP-VAE makes no assumptions on neither the data nor the GP kernel used. Therefore, it is a more general solution to scale GP-VAE models.

B.4 Sparse GP-VAE based on Titsias (2009)

Using the sparse GP posterior q_S (Equation 3) and ELBO $\mathcal{L}_T$ (Equation 6) from Titsias (2009) gives rise to the following sparse GP-VAE ELBO:

$$\mathcal{L}_{PT}\big(\mathbf{U},\psi,\phi,\theta\big) := \sum_{l=1}^{L}\mathcal{L}_T^l(\mathbf{U},\,\phi,\,\theta) + \sum_{i=1}^{N}\mathbb{E}_{q_S}\Big[\log p_\psi(\mathbf{y}_i|\mathbf{z}_i) - \log\tilde{q}_\phi(\mathbf{z}_i|\mathbf{y}_i)\Big].$$

In Section 3.3, we have outlined how to obtain the above sparse ELBO from the GP-VAE ELBO proposed in Pearce (2020). Alternatively, $\mathcal{L}_{PT}$ can be derived in the standard way by directly considering the KL divergence between the sparse GP posterior and the (intractable) true posterior for the latent variables $\mathrm{KL}\big(q_S(\mathbf{Z}|\cdot)||p_{\psi,\theta}(\mathbf{Z}|\mathbf{Y},\mathbf{X})\big)$.

Following Titsias (2009), we consider the joint distribution of observed and *augmented* latent variables $p_{\psi,\theta}(\mathbf{Z},\mathbf{F}_m,\mathbf{Y}|\mathbf{X})$ where $\mathbf{F}_m := \big[\mathbf{f}^1,\ldots,\mathbf{f}^L\big]$, $\mathbf{f}^l := f^l(\mathbf{U}) \in \mathbb{R}^m$. The sparse GP posterior decomposes as $q_S(\mathbf{Z},\mathbf{F}_m|\cdot) = p_\theta(\mathbf{Z}|\mathbf{F}_m)p_S(\mathbf{F}_m)$, where $p_S(\mathbf{F}_m) := \prod_{l=1}^{L}\mathcal{N}(\mathbf{f}_m^l|\boldsymbol{\mu}^l,\mathbf{A}^l)$ is a free variational distribution and $p_\theta(\mathbf{Z}|\mathbf{F}_m)$ is a (standard) conditional GP prior. The problem of minimizing the KL divergence is then equivalently posed as a maximization of a lower bound of the model evidence as follows, where in the first steps we introduce $\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})$ and $q_S(\mathbf{Z},\mathbf{F}_m|\cdot)$ and apply Jensen's inequality:

$$\begin{aligned}
\log p(\mathbf{Y}|\mathbf{X}) &= \log\int p_{\psi,\theta}(\mathbf{Z},\mathbf{F}_m,\mathbf{Y}|\mathbf{X})\frac{q_S(\mathbf{Z},\mathbf{F}_m|\cdot)}{q_S(\mathbf{Z},\mathbf{F}_m|\cdot)}\frac{\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})}{\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})}d\mathbf{Z}d\mathbf{F}_m \\
&\geq \int q_S(\mathbf{Z},\mathbf{F}_m|\cdot)\log\frac{p_{\psi,\theta}(\mathbf{Z},\mathbf{F}_m,\mathbf{Y}|\mathbf{X})}{q_S(\mathbf{Z},\mathbf{F}_m|\cdot)}\frac{\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})}{\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})}d\mathbf{Z}d\mathbf{F}_m \\
&= \int q_S(\mathbf{Z},\mathbf{F}_m|\cdot)\log\frac{\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})p_\psi(\mathbf{Y}|\mathbf{Z})p_\theta(\mathbf{Z}|\mathbf{F}_m)p_\theta(\mathbf{F}_m|\mathbf{X})}{\tilde{q}_\phi(\mathbf{Z}|\mathbf{Y})p_\theta(\mathbf{Z}|\mathbf{F}_m)p_S(\mathbf{F}_m)}d\mathbf{Z}d\mathbf{F}_m \\
&= \sum_{l=1}^{L}\int q_S(\mathbf{z}^l,\mathbf{f}_m^l|\cdot)\log\frac{\tilde{q}_\phi(\mathbf{z}^l|\mathbf{Y})p_\theta(\mathbf{f}_m^l|\mathbf{X})}{p_S(\mathbf{f}_m^l)}d\mathbf{z}^l d\mathbf{f}_m^l + \sum_{i=1}^{N}\int q_S(\mathbf{z}_i|\cdot)\Big(\log p_\psi(\mathbf{y}_i|\mathbf{z}_i) - \log\tilde{q}_\phi(\mathbf{z}_i|\mathbf{y}_i)\Big)d\mathbf{z}_i \\
&= \sum_{l=1}^{L}\mathcal{L}_T(\mathbf{U},\phi,\theta,\boldsymbol{\mu}^l,\mathbf{A}^l) + \sum_{i=1}^{N}\mathbb{E}_{q_S}\big[\log p_\psi(\mathbf{y}_i|\mathbf{z}_i) - \log\tilde{q}_\phi(\mathbf{z}_i|\mathbf{y}_i)\big]
\end{aligned}$$

Recall the symmetry of the Gaussian distribution $\tilde{q}_\phi(\mathbf{z}_i^l|\mathbf{y}_i) = \mathcal{N}(\mathbf{z}_i^l|\boldsymbol{\mu}^l(\mathbf{y}_i),\sigma^l(\mathbf{y}_i)) = \mathcal{N}(\boldsymbol{\mu}^l(\mathbf{y}_i)|\mathbf{z}_i^l,\sigma^l(\mathbf{y}_i))$. Hence, the first term of the penultimate expression is a sum over sparse Gaussian processes, one for each latent channel, and each term is precisely Equation 8 of Titsias (2009) for sparse Gaussian process regression. Therefore we write $\mathcal{L}_T^l$ and let $\boldsymbol{\mu}^l = \boldsymbol{\mu}_T^l$ and $\mathbf{A}^l = \mathbf{A}_T^l$. For further derivation steps see Titsias (2009).

C Additional experiments

C.1 PCA initialization of GP-LVM vectors and inducing points

In this section, we describe how Principal Component Analysis (PCA) is used to initialize the GP-LVM digit representations as well as the inducing points in the rotated MNIST experiment. Note that both the GP-VAE (Casale et al., 2018) and the SVGP-VAE depend on GP-LVM vectors, with the SVGP-VAE additionally relying on inducing points.

To obtain a continuous digit representations for each digit instance, we start with the data matrix $\mathbf{X} \in \mathbb{R}^{P \times K}$ that consists of unrotated MNIST images. PCA is then performed on $\mathbf{X}$, yielding a matrix $\boldsymbol{D} \in \mathbb{R}^{P \times M}$ whose rows $\boldsymbol{d}_i$ are used as initial values for the GP-LVM vectors. M represents the number of principal components kept.

For initialization of the inducing points, we sample n GP-LVM vectors from the empirical distribution based on the PCA matrix $\boldsymbol{D}$ for each of the Q angles. This results in a matrix $\mathbf{U}_{init} \in \mathbb{R}^{m \times (1+M)}$ with $m = n \cdot Q$ representing the number of inducing points. The exact procedure is given in Algorithm 1. Results from the ablation study on the PCA initialization described here are presented in Table C.1.

Algorithm 1: Initialization of inducing points in the SVGP-VAE (rotated MNIST experiment)

input : PCA matrix $\boldsymbol{D}$, number of inducing points per angle n, set of angles $\{\frac{2\pi k}{Q} \mid k = 1, ..., Q\}$

$\mathbf{U}_{init} = [\,]$
\# sample $m = n \cdot Q$ points from empirical distribution of each principle component
for $i = 1, ..., M$ **do**
  $\mathbf{U}_{init} = \big[\mathbf{U}_{init},\ sample\big(\boldsymbol{D}[:, i],\ nr\_samples = n\big)\big]$
end
\# add column with angle information
$\boldsymbol{a} = \big[\underbrace{2\pi/Q, ..., 2\pi/Q}_{n\times},\ ...\ , \underbrace{2\pi, ..., 2\pi}_{n\times}\big]^T \in \mathbb{R}^m$
$\mathbf{U}_{init} = \big[\boldsymbol{a},\ \mathbf{U}_{init}\big]$
return $\boldsymbol{U}_{init}$

	PCA init	**random init**
GP-VAE Casale et al. (2018)	0.0370 ± 0.0012	0.0374 ± 0.0009
SVGP-VAE	0.0251 ± 0.0005	0.0272 ± 0.0006

Table C.1: A comparison of different initialization regimes for GP-LVM vectors and inducing points in the rotated MNIST experiment. For random initialization, a Gaussian distribution with mean 0 and standard deviation 1.5 was used.

C.2 SVGP-VAE latent space visualization

In Figure C.1, we depict two-dimensional t-SNE (Maaten and Hinton, 2008) embeddings of SVGP-VAE latent vectors ($L = 16$). Visualized here are latent vectors for training data of the *five-digit* version of the rotated MNIST dataset ($N = 20250$). As expected, the model clusters images based on the digit identity. More interestingly, SVGP-VAE also seems to order images within each digit cluster with respect to angles. For example, looking at the cluster of the digit 3 (the blue cluster in the middle of the lower plot), we observe that embeddings of rotated images are ordered continuously from 0 to 2π as we move in clockwise direction around the circular shape of the cluster.

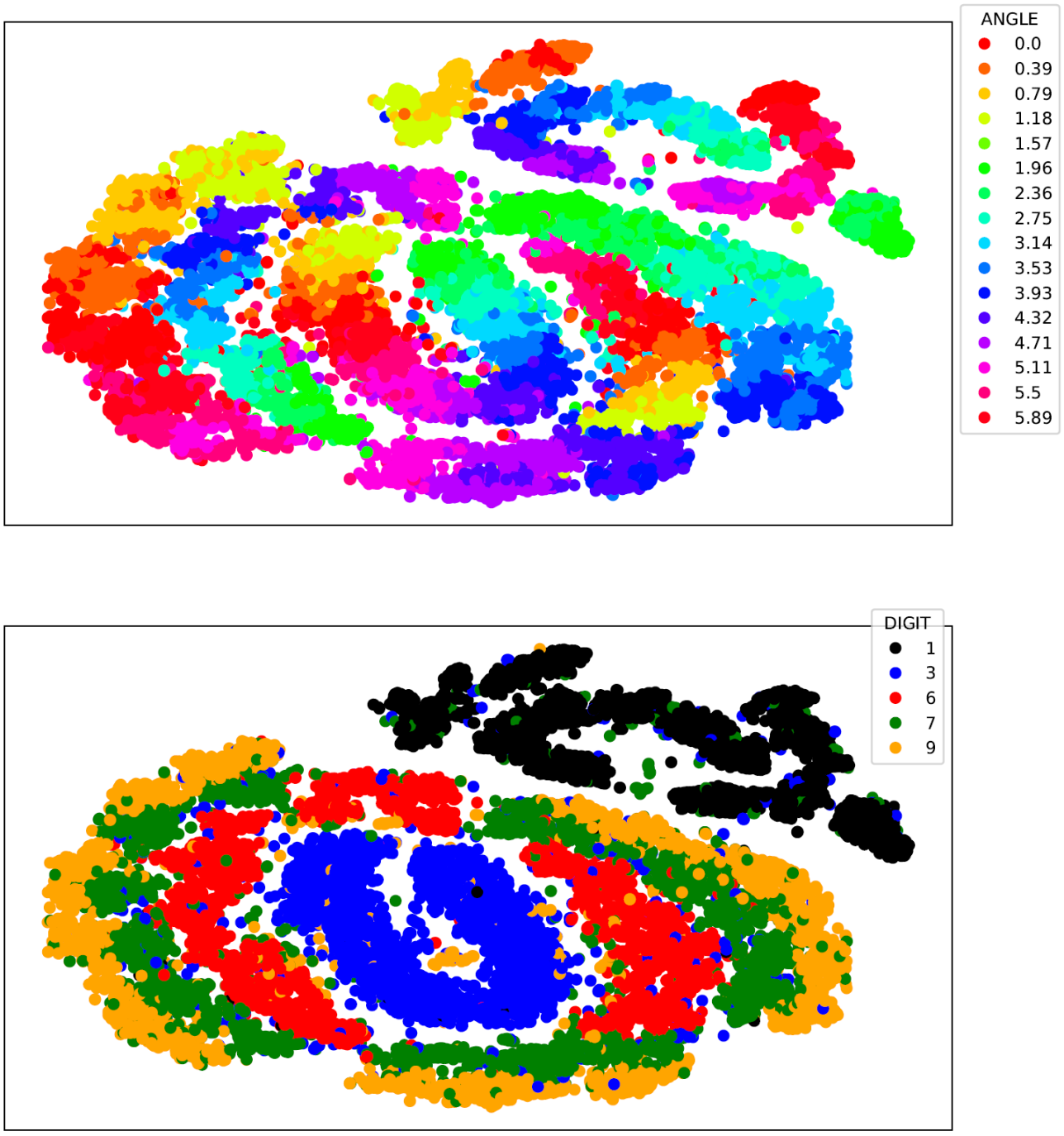

Figure C.1: t-SNE embeddings of SVGP-VAE latent vectors on the training data for rotated MNIST. On the upper scatter plot, each image embedding is colored with respect to its associated angle. On the lower scatter plot, each image embedding is colored with respect to its associated digit. The t-SNE perplexity parameter was set to 50.

C.3 Rotated MNIST: generated images

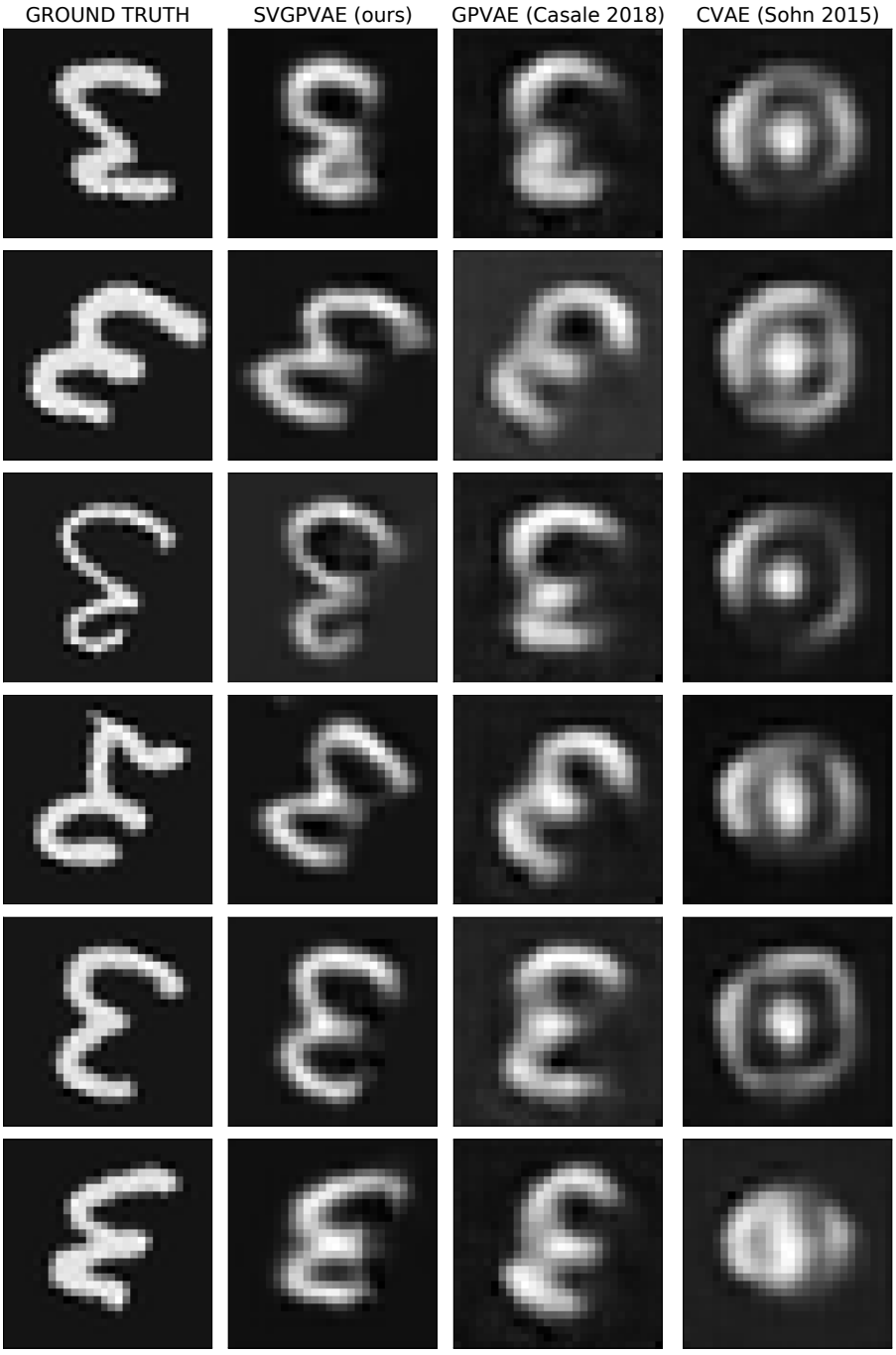

Figure C.2: Generated test images in the rotated MNIST experiment for all considered models.

C.4 Bias analysis of MC estimators in SVGP-VAE

Here we look at some additional experiments that were conducted to get a better understanding of the SVGP-VAE model. Depicted in Figure C.3 are the results when varying the batch size and the number of inducing points. We first notice that the SVGP-VAE performance improves as the batch size is increased. As pointed out in Section 3.4, this is a consequence of the Monte Carlo estimators from (9) used in q_S whose quality depends on the batch size. While the dependence on the batch size can surely be seen as one limitation of the model, it

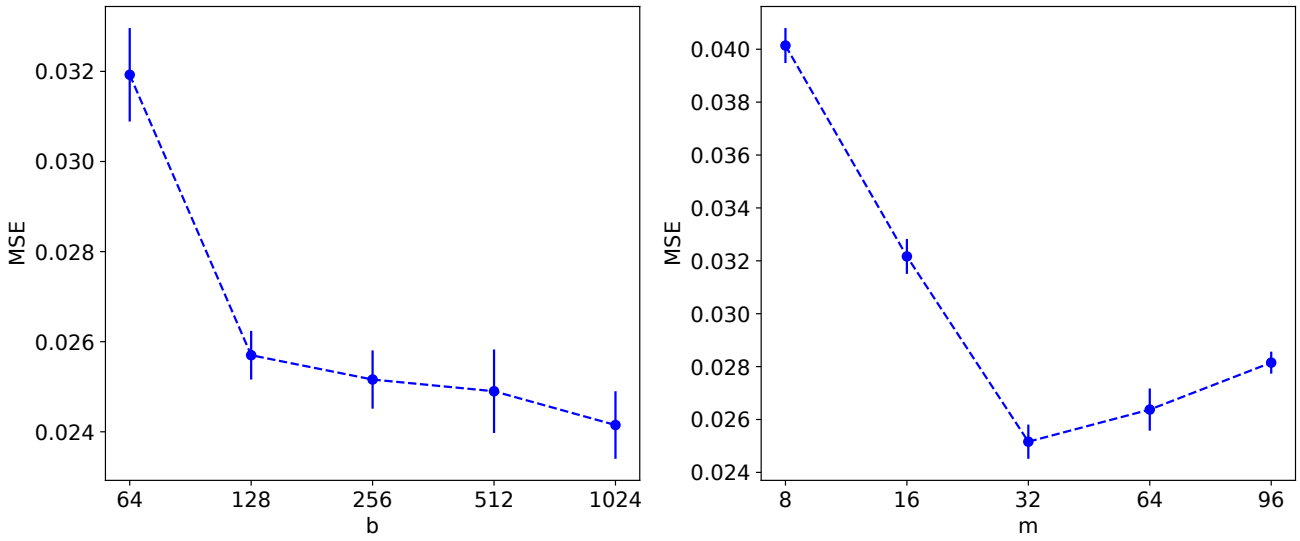

Figure C.3: SVGP-VAE results on the rotated MNIST dataset (digit 3) for varying batch size (left) and number of inducing points (right). For the batch size experiment, m was set to 32. For the inducing points experiment, b was set to 256. For each configuration, a mean MSE together with a standard deviation based on 5 runs is shown.

is encouraging to see that the model achieves good performance already for a reasonably small batch size (e.g., $b = 128$). Moreover, the batch size parameter in the SVGP-VAE offers a simple and intuitive way to navigate a trade-off between performance and computational demands. If one is more concerned regarding the performance, a higher batch size should be used. On the other hand, if one only has limited computational resources at disposal, a lower batch size can be utilized resulting in a faster and less memory-demanding model.

Looking at the plot with the varying number of inducing points next, we observe that the model achieves a solid performance with as little as 16 inducing points on the rotated MNIST data. However, increasing the number of inducing points m starts to have a negative impact on the performance after a certain point. This can be partly attributed to numerical issues that arise during training — the higher the m, the more numerically unstable the inducing point kernel matrix $\mathbf{K}_{mm}$ becomes. Moreover, since the number of inducing points equals the dimension of the Monte Carlo estimators in (9), increasing m results in a larger dimension of the space, potentially increasing the complexity of the estimation problem.

To better understand the effect of the number of inducing points m on the quality of estimation in our proposed MC estimators, we investigate here the trajectory of the bias throughout training. To this end, for each epoch i an estimator $\boldsymbol{\mu}^l_{j,i}$ is calculated for each latent channel l and for each batch j. Additionally, the true value $\boldsymbol{\mu}^l_{T,i}$ is obtained (based on the entire dataset) for every epoch and every latent channel using model weights from the end of the epoch. The bias for the l-th latent channel and i-th epoch is then computed as

$$\boldsymbol{b}^l_i := \frac{1}{B}\sum_{j=1}^{B} \boldsymbol{\mu}^l_{j,i} - \boldsymbol{\mu}^l_{T,i}$$

where $B := \lceil \frac{N}{b} \rceil$ represents the number of batches in a single epoch. Finally, for each epoch i the $L1$ norms of the bias vectors for each latent channel are averaged $\boldsymbol{b}_i = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{b}^l_i$.

Moving averages of the resulting bias trajectories are depicted in Figure C.4. For comparison purposes, each trajectory is normalized by the number of inducing points used. Notice how for smaller m, the bias trajectories display the expected behavior and converge (or stay close) to 0. Conversely, for larger numbers of inducing points ($m = 64$ and $m = 96$), the bias is larger and does not decline as the training progresses. This suggests that the proposed estimation might get worse in larger dimensions.

However, despite seemingly deteriorating approximation in higher dimensions, it is also evident that the approximation does not completely break down — the model still achieves a solid performance even for a larger number

of inducing points. Nevertheless, we note that getting a better theoretical grasp of the quality of estimation or reparameterizing the SVGP-VAE ELBO in a way such that these estimators are no longer needed could be a fruitful area of future work.

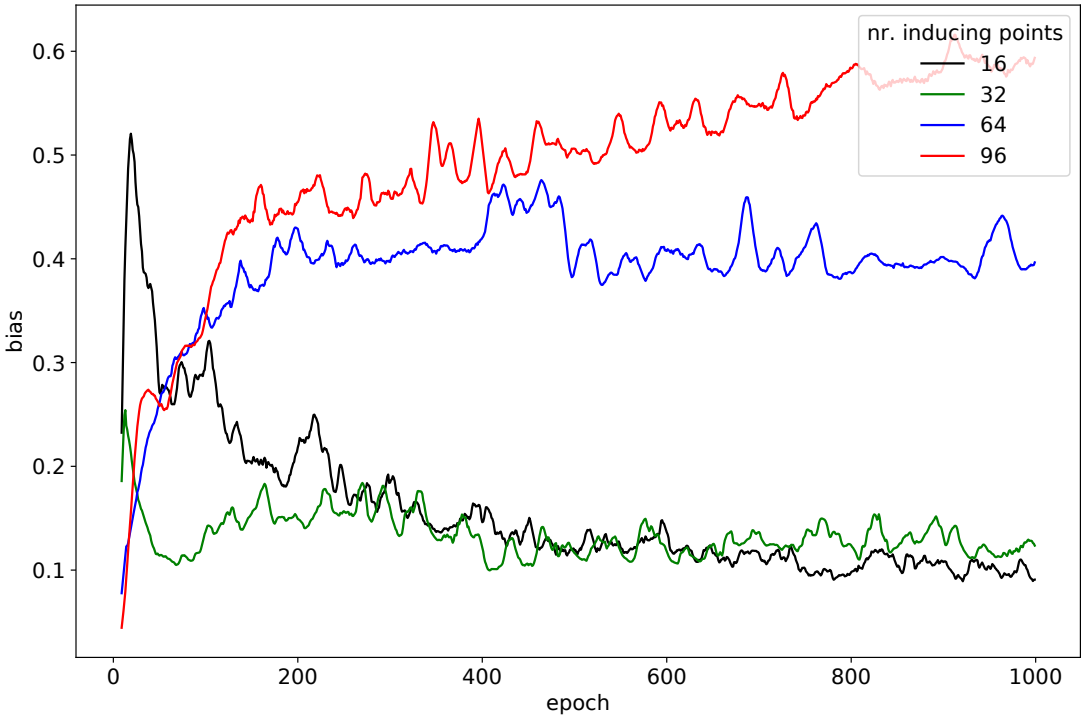

Figure C.4: Bias trajectories in the SVGP-VAE model for a varying number of inducing points. For all runs, the batch size was set to 256.

C.5 Deep sparse GP from Hensman et al. (2013) for conditional generation

In Section 3.3, we demonstrate that a sparse GP approach from Hensman et al. (2013) cannot be used in the GP-VAE framework as it does not lend itself to amortization. In Section B.1, we then provide a detailed derivation of this phenomenon. Here, we leave out the amortization completely and consider directly the sparse GP from Hensman et al. (2013). To this end, we modify the ELBO in eq. (4) in Hensman et al. (2013). To model our high-dimensional data $\mathbf{y}_i \in \mathbb{R}^K$, we utilize a deep likelihood parameterized by a neural network $\psi : \mathbb{R}^L \to \mathbb{R}^K$ (instead of a simple Gaussian likelihood). Moreover, we replicate a GP regression L times (across all latent channels), which yields the following objective function

$$\mathcal{L}(\mathbf{U}, \psi, \theta, \boldsymbol{\mu}^{1:L}, \mathbf{A}^{1:L}, \sigma) = \sum_{i=1}^{N} \left\{ \log \mathcal{N}\big(\mathbf{y}_i \,|\, \psi(\boldsymbol{m}_i),\, \sigma^2 \mathbf{I}\big) - \frac{1}{2\sigma^2} \sum_{l=1}^{L} \left(\tilde{k}_{ii} + Tr(\mathbf{A}^l\, \Lambda_i)\right) \right\} - \sum_{l=1}^{L} \mathrm{KL}\big(q_S^l(\mathbf{f}_m|\cdot) \,||\, p_\theta(\mathbf{f}_m|\cdot)\big)$$

where $\boldsymbol{m}_i := [\boldsymbol{k}_i \mathbf{K}_{mm}^{-1} \boldsymbol{\mu}^1,\ \dots\ , \boldsymbol{k}_i \mathbf{K}_{mm}^{-1} \boldsymbol{\mu}^L]^T \in \mathbb{R}^L$. Also recall that $q_S^l(\mathbf{f}_m|\cdot) = \mathcal{N}(\mathbf{f}_m|\boldsymbol{\mu}^{\boldsymbol{l}}, \boldsymbol{A}^{\boldsymbol{l}})$, $p_\theta(\mathbf{f}_m|\cdot) = \mathcal{N}(\mathbf{f}_m|\mathbf{0},\, \mathbf{K}_{mm})$, $\Lambda_i := \mathbf{K}_{mm}^{-1} \boldsymbol{k}_i \boldsymbol{k}_i^T \mathbf{K}_{mm}^{-1}$ and $\tilde{k}_{ii}$ is the i-th diagonal element of $\mathbf{K}_{NN} - \mathbf{K}_{Nm} \mathbf{K}_{mm}^{-1} \mathbf{K}_{mN}$.

For a test point $\boldsymbol{x_*}$, we first obtain $\boldsymbol{m}_* = [\boldsymbol{k}_* \mathbf{K}_{mm}^{-1} \boldsymbol{\mu}^1, \ ..., \boldsymbol{k}_* \mathbf{K}_{mm}^{-1} \boldsymbol{\mu}^L]^T$, $\boldsymbol{k_*} = [k_\theta(\mathbf{x}_*, \mathbf{u}_1), ..., k_\theta(\mathbf{x}_*, \mathbf{u}_m)]^T \in \mathbb{R}^m$, and then pass it through the network ψ to generate $\mathbf{y}_*$.

For comparison purposes, the same number of latent channels ($L = 16$) and the same architecture for the network ψ as in our SVGP-VAE is used. We train this baseline model for 2000 epochs using the Adam optimizer and a batch size of 256.

The strong performance (see Table 1) of this baseline provides interesting new insights into the role of amortization in GP-VAE models. For the task of conditional generation, where a single GP prior is placed over the entire dataset, the amortization is not necessary, and one can modify existing sparse GP approaches (Hensman et al., 2013) to achieve good results in a computationally efficient way. Note that this is not the case for tasks like learning interpretable low-dimensional embeddings (Pearce, 2020) or time-series imputation (Fortuin et al., 2020). For such tasks, the inference network is needed in order to be able to quickly obtain predictions for new test points without rerunning the optimization.

More thorough investigation of this baseline, its interpretation, and its comparison to the existing work on deep Gaussian Processes (Damianou and Lawrence, 2013; Wilson et al., 2016) is left for future work.