A. On neural network-based immersions

For the decoder map 2.3 to be a valid immersion, its differential $\mathrm{d}f$ needs to be injective for all $p \in \mathcal{M}$ as stated in definition 2. The differential of f is represented by its Jacobian matrix $\mathbf{J_f}$ and for it to be injective for all $p \in \mathcal{M}$, it needs to be full rank. This is ensured if for the MLPs representing the decoder $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\sigma}_\psi$ the following are true:

- Each hidden layer in the network has an equal or greater number of units to the previous layer ($n_{L-1} \leq n_L$).
- All weight matrices in the network are full rank.
- The activation functions are at least twice differentiable and strictly monotone.

In our experiments, we opt for the same number of units in each hidden layer of the network and ELU non-linearities. In theory, the ELU activation function could present problems since it has a point of discontinuity at 0, however we did not experience any numerical instability that would arise in such case. All weight matrices are initialized uniformly (He et al., 2015) which practically has zero probability of yielding low rank weight matrices. While theoretically this could change via the gradient updates of the weights, this would once again immediately break experiments because of numerical instabilities, which we did not observe.

B. Geodesic estimation

We estimate geodesic distances by minimizing curve energy. In detail, we represent the geodesic curve with a cubic spline with parameters initialized to form a straight line. These parameters are then optimized via gradient descent by minimizing the curve energy:

$$\begin{aligned}\mathcal{E}(\boldsymbol{\gamma}) &= \frac{1}{2}\int_0^1 ||\dot{\boldsymbol{\gamma}}(t)||_g^2 \,\mathrm{d}t \\ &= \frac{1}{2}\int_0^1 \dot{\boldsymbol{\gamma}}^T(t)\mathbf{G}_{\boldsymbol{\gamma}}\dot{\boldsymbol{\gamma}}(t)\,\mathrm{d}t \end{aligned} \tag{B.1}$$

where $\boldsymbol{\gamma}$ is the geodesic curve, $\dot{\boldsymbol{\gamma}}$ is the first derivative of the curve, i.e. its velocity vector and $\mathbf{G}_{\boldsymbol{\gamma}}$ is the matrix representation of the metric tensor evaluated at the curve points. The integral B.1 is computed by numerical approximation, where the partition of the interval can be chosen as a hyperparameter.

C. Experimental setup

The architectures of all model variants are shown below in Tables 4 and 5. The encoder mean and variance, as well as the decoder mean are modelled by 2-layer MLPs as shown below. The decoder mean mirrors the encoder mean, while the *precision* $\boldsymbol{\beta}$ is estimated by the RBF network. The number of the RBF centers is set to 350 and the bandwidth is set to 0.01 in all cases. For a fair comparison, all models share the same underlying architecture for the encoder and decoder. Tables 4 and 5 summarize the architectures, listing the activation function for each layer with the units corresponding to each layer in parentheses.

Table 4. Encoder network architectures.

Network	Layer 1	Layer 2	Output
$\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$	ELU (300)	ELU (300)	Linear (*dim*($\mathcal{Z}$))
$\boldsymbol{\sigma}^{\mathbf{2}}_{\boldsymbol{\phi}}(\mathbf{x})$	ELU (300)	ELU (300)	Softplus (*dim*($\mathcal{Z}$))

Table 5. Decoder network architectures. * denotes strictly positive weights.

Network	Layer 1	Layer 2	Output
$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z})$	ELU (300)	ELU (300)	Linear (*dim*($\mathcal{X}$))
$\boldsymbol{\beta}_{\boldsymbol{\psi}}(\mathbf{z})$	RBF ($\mathbb{R}^{dim(\mathcal{Z})\times 350}$)	Linear* (*dim*($\mathcal{X}$))	Identity (*dim*($\mathcal{X}$))

C.1. Section 5.1 experiment

Detlefsen et al. (2019) highlighted the importance of optimizing the mean and variance components separately, when training VAEs with Gaussian generative models. Following this paradigm, in all our experiments we first optimize the encoder components ($\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}$) along with the decoder $\boldsymbol{\mu}_{\boldsymbol{\theta}}$. Then, keeping these fixed, we optimize the decoder $\boldsymbol{\sigma}_{\boldsymbol{\psi}}$. All models were trained for 300 epochs. More specifically, the $\mathcal{R}$-VAE was trained as an autoencoder (optimizing only the encoder $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}$ and the decoder $\boldsymbol{\mu}_{\boldsymbol{\theta}}$) for the first 100 epochs and for the remaining 200 epochs the latent prior and the decoder $\boldsymbol{\beta}_{\boldsymbol{\psi}}$ were optimized. Similarly for a VAE, it was deterministically warmed up for 100 epochs and for the remaining 200 epochs, the decoder $\boldsymbol{\beta}_{\boldsymbol{\psi}}$ was optimized. All experiments were run with the Adam optimizer (Kingma & Ba, 2015) with default parameter settings and a fixed learning rate of 10^{-3}. The batch size was 100 for all models.

C.2. Section 5.2 experiment

The classifier used on this section was a single, 100-unit layer MLP with ReLU non-linearities, trained for 100 epochs with the Adam optimizer with default parameter settings and a learning rate of 10^{-3}. The batch size was set at 64. The architectures of the models giving rise to the latent representations are as in the previous section.

C.3. Runtime comparisons

Below is the wall clock time for every model used in the experiments. The statistics were computed without a fixed seed. The latent space dimensions are denoted by d. In VAE-VampPrior, n denotes the number of mixture components in the latent prior.

Table 6. Per epoch training time for each model. Mean and std deviation in seconds, computed over 100 epochs on MNIST.

Model	$d = 2$	$d = 5$	$d = 10$
VAE	$10.64_{\pm.51}$	$11.01_{\pm.77}$	$11.10_{\pm.60}$
VAE-VampPrior ($n = 128$)	$10.66_{\pm.6}$	$11.22_{\pm.9}$	$11.37_{\pm.96}$
VAE-VampPrior ($n = 256$)	$10.72_{\pm.3}$	$11.34_{\pm1.21}$	$11.52_{\pm.77}$
VAE-VampPrior ($n = 512$)	$10.9_{\pm.34}$	$11.38_{\pm.93}$	$12.18_{\pm1.12}$
$\mathcal{R}$-VAE	$55.73_{\pm4.36}$	$59.97_{\pm1.33}$	$60.13_{\pm1.19}$

C.4. Complete results for VAE-VampPrior

Tables 1 & 2 show the results of the best performing VampPrior model variant. In general, we noticed that a VAE with an overly flexible (i.e. many mixture components) VampPrior was prone to overfitting. Here we show the complete results of the VAE-VampPrior in all settings. Below n denotes the number of mixture components in the latent prior, while d denotes the latent space dimensions.

Table 7. MNIST results of VAE with VampPrior for varying latent space dimensions and number of mixture components in the latent prior.

Model	Neg. ELBO	Rec	KL
	$d = 2$		
VAE-VampPrior ($n = 128$)	$-1039.66_{\pm2.56}$	$-1042.13_{\pm2.56}$	$2.46_{\pm.01}$
VAE-VampPrior ($n = 256$)	$-1045.04_{\pm5.20}$	$-1047.34_{\pm5.22}$	$2.30_{\pm.03}$
VAE-VampPrior ($n = 512$)	$-1040.79_{\pm9.23}$	$-1043.24_{\pm9.25}$	$2.45_{\pm.05}$
	$d = 5$		
VAE-VampPrior ($n = 128$)	$-1100.77_{\pm4.98}$	$-1102.46_{\pm4.91}$	$1.69_{\pm.06}$
VAE-VampPrior ($n = 256$)	$-1103.29_{\pm1.85}$	$-1105.04_{\pm1.79}$	$1.75_{\pm.12}$
VAE-VampPrior ($n = 512$)	$-1109.74_{\pm4.87}$	$-1111.63_{\pm4.87}$	$1.88_{\pm.01}$
	$d = 10$		
VAE-VampPrior ($n = 128$)	$-1110.05_{\pm6.10}$	$-1112.23_{\pm5.82}$	$1.84_{\pm.04}$
VAE-VampPrior ($n = 256$)	$-1116.58_{\pm4.23}$	$-1118.28_{\pm4.20}$	$1.69_{\pm.02}$
VAE-VampPrior ($n = 512$)	$-1100.64_{\pm2.93}$	$-1102.42_{\pm2.97}$	$1.78_{\pm.03}$

Table 8. FashionMNIST results of VAE with VampPrior for varying latent space dimensions and number of mixture components in the latent prior.

Model	Neg. ELBO	Rec	KL
	$d = 2$		
VAE-VampPrior ($n = 128$)	-694.63$_{\pm 8.65}$	-697.14$_{\pm 8.65}$	2.50$_{\pm .01}$
VAE-VampPrior ($n = 256$)	-702.67$_{\pm 17.45}$	-705.19$_{\pm 17.44}$	2.52$_{\pm .04}$
VAE-VampPrior ($n = 512$)	-705.90$_{\pm 21.29}$	-708.45$_{\pm 21.29}$	2.54$_{\pm .01}$
	$d = 5$		
VAE-VampPrior ($n = 128$)	-755.80$_{\pm .66}$	-756.58$_{\pm .71}$	0.77$_{\pm .06}$
VAE-VampPrior ($n = 256$)	-767.54$_{\pm 3.22}$	-768.33$_{\pm 3.31}$	0.78$_{\pm .09}$
VAE-VampPrior ($n = 512$)	-769.27$_{\pm 5.0}$	-770.10$_{\pm 5.02}$	0.83$_{\pm .09}$
	$d = 10$		
VAE-VampPrior ($n = 128$)	-754.47$_{\pm 6.78}$	-758.20$_{\pm 6.72}$	3.72$_{\pm .06}$
VAE-VampPrior ($n = 256$)	-756.13$_{\pm 5.40}$	-760.49$_{\pm 5.1}$	3.69$_{\pm .06}$
VAE-VampPrior ($n = 512$)	-774.17$_{\pm 10.83}$	-777.75$_{\pm 10.78}$	3.58$_{\pm .06}$