Extended Abstract Track

Learning Generative Models with Invariance to Symmetries

Editors: List of editors' names

Abstract

While imbuing a model with invariance to symmetries can improve data efficiency and predictive performance, most methods require specialised architectures and thus prior knowledge of the symmetries. Unfortunately, we don't always know what symmetries are present in the data. Recent work has solved this problem by jointly learning the invariance (or the degree of invariance) with the model from the data alone. But, this work has focused on discriminative models. We describe a method for learning invariant *generative* models. We demonstrate that our method can learn a generative model of handwritten digits that is invariant to rotation. We hope this line of work will enable more data-efficient deep generative models.

Keywords: generative models, invariance, symmetry.

1. Introduction

Generative models are often motivated as being more data efficient (Welling, 2019). We expect that since we are encoding our beliefs about how the data are generated (our inductive biases) into the model, it will be able to learn from fewer data. Unfortunately, deep generative models are *more* data hungry and slow to converge than their discriminative counterparts. This should not surprise us as we tend to only incorporate very general inductive biases (smoothness, hierarchies of random variables, etc.) in deep generative models. One class of inductive bias that improves does improve efficiency is invariance to symmetries in the data (van der Ouderaa and van der Wilk, 2022). In some cases, we know apriori which symmetries the model should be invariant to. However, in many cases, we need to learn when (or to what extent) to be invariant. For example, in the case of handwritten digit recognition, we know that we should be invariant to some degree of rotation. However, a model that is invariant to the full range of rotations in the range $[-180°, 180°]$ would be unable to distinguish between '6' and '9'. Thus, the degree of invariance should be learned. Much recent work has focused on solving this problem for discriminative models (Nalisnick and Smyth, 2018; van der Wilk et al., 2018; Benton et al., 2020; Schwöbel et al., 2021; van der Ouderaa and van der Wilk, 2022; Immer et al., 2022). But, the problem has been neglected in the generative case. We investigate the learning of invariance to symmetries in generative models.

We aim to construct a deep generative model $p(\mathbf{x})$ which is invariant to certain transformations of $\mathbf{x}$. That is, $p(\mathbf{x}) = p(\mathcal{T}_{\boldsymbol{\eta}}(\mathbf{x}))$, where $\boldsymbol{\eta}$ is a r.v. which parameterizes the transformation $\mathcal{T}$. For example, $\mathcal{T}$ could be the rotation of an image, and $\boldsymbol{\eta} = \theta$ could be the rotation angle. In a latent-variable generative model, the latent variable $\mathbf{z}$ should contain no information about $\mathcal{T}$. We hope that jointly learning the invariance and the parameters of the model will be 'easier' than learning in a standard model. The degree of invariance learnt should match the true invariance of the data. I.e., unlike Benton et al. (2020) we should not learn a 'maximum data augmentation'.

© 2022 .

2. The Model and Design Choices

We assume the following generative model for the data $\mathbf{x}$, also depicted in Figure 1:

$$\hat{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\hat{\mathbf{x}})\,, \tag{1}$$

$$\boldsymbol{\eta} \sim p_{\boldsymbol{\psi}}(\boldsymbol{\eta}\,|\,\hat{\mathbf{x}})\,, \tag{2}$$

$$\mathbf{x} \sim p(\mathbf{x}\,|\,\boldsymbol{\eta},\,\hat{\mathbf{x}})\,, \tag{3}$$

where the *prototype* $\hat{\mathbf{x}}$ can be considered a standardised or reference example with no transformation applied to it. For example, this could be a '6' at in the upright (90°) rotation[1]. Here, $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$ is some generative model with parameters $\boldsymbol{\theta}$—a VAE or a normalizing flow, for example. Similarly, $p_{\boldsymbol{\psi}}(\boldsymbol{\eta}\,|\,\hat{\mathbf{x}})$ is a NN, with parameters $\boldsymbol{\psi}$, that predicts the distribution over transformations for a given prototype.

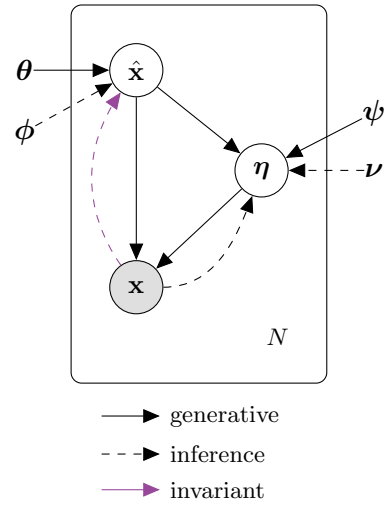


Figure 1: Graphical model.

We propose a VAE-like inference strategy for this model. Given an inference model—depicted in Figure 1—we can derive an ELBO for jointly learning the generative $\{\boldsymbol{\theta},\,\boldsymbol{\psi}\}$ and inference $\{\boldsymbol{\phi},\,\boldsymbol{\nu}\}$ parameters, with gradient descent:

$$\log p(\mathbf{x}) = \log \iint p(\mathbf{x},\,\hat{\mathbf{x}},\,\boldsymbol{\eta})\,d\boldsymbol{\eta}\,d\hat{\mathbf{x}} \tag{4}$$

$$= \log \mathop{\mathbb{E}}_{q_{\boldsymbol{\nu}}(\boldsymbol{\eta}\,|\,\mathbf{x})\,q_{\boldsymbol{\phi}}(\hat{\mathbf{x}}\,|\,\mathbf{x})} \left[\frac{p(\mathbf{x}\,|\,\hat{\mathbf{x}},\,\boldsymbol{\eta})\;p_{\boldsymbol{\psi}}(\boldsymbol{\eta}\,|\,\hat{\mathbf{x}})\;p_{\boldsymbol{\theta}}(\hat{\mathbf{x}})}{q_{\boldsymbol{\nu}}(\boldsymbol{\eta}\,|\,\mathbf{x})\;q_{\boldsymbol{\phi}}(\hat{\mathbf{x}}\,|\,\mathbf{x})}\right] \tag{5}$$

$$\geq \mathop{\mathbb{E}}_{q_{\boldsymbol{\nu}}\,q_{\boldsymbol{\phi}}} \left[\log p(\mathbf{x}\,|\,\hat{\mathbf{x}},\,\boldsymbol{\eta})\right] - \mathop{\mathbb{E}}_{q_{\boldsymbol{\phi}}} \left[D_{\mathrm{KL}}\left(q_{\boldsymbol{\nu}}\,||\,p_{\boldsymbol{\psi}}\right)\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}\,||\,p_{\boldsymbol{\theta}}\right) \equiv -\mathcal{L}\left(\boldsymbol{\theta},\,\boldsymbol{\psi},\,\boldsymbol{\phi},\,\boldsymbol{\nu}\right), \tag{6}$$

where $q_{\boldsymbol{\nu}}(\boldsymbol{\eta}\,|\,\mathbf{x})$ and $q_{\boldsymbol{\phi}}(\hat{\mathbf{x}}\,|\,\mathbf{x})$ are NNs with parameters $\boldsymbol{\nu}$ and $\boldsymbol{\phi}$. By choosing $q_{\boldsymbol{\phi}}(\hat{\mathbf{x}}\,|\,\mathbf{x})$ to be invariant to $\boldsymbol{\eta}$, we encourage $\mathbf{x}$ not to contain any information about the symmetry for $\mathcal{T}_{\boldsymbol{\eta}}$. Perhaps counter-intuitively, and in contrast with previous approaches to learning invariances (van der Wilk et al., 2018; Benton et al., 2020; Schwöbel et al., 2021; van der Ouderaa and van der Wilk, 2022; Immer et al., 2022), we do not make this NN *partially* invariant. Nonetheless, as we will show, we can still learn a degree of invariance.

We now present four informal propositions to explain some of the design choices for the model in Figure 1. We illustrate these propositions by means of examples. In each example we will assume that $\mathcal{T}$ is counter-clockwise rotation, and thus $\boldsymbol{\eta} = \theta$.

Proposition 1: $p_{\boldsymbol{\psi}}(\boldsymbol{\eta}\,|\,\hat{\mathbf{x}})$ must be sufficiently flexible. Consider a dataset of slightly rotated '8's: $\{8,\,\text{ȣ},\,\text{ȣ}\}$. Let us assume that the prototype is '8'. Table 1 shows $p(\theta\,|\,\mathbf{x},\,\hat{\mathbf{x}})$, the true distribution for θ given $\mathbf{x}$ and $\hat{\mathbf{x}}$, under the data generating process. Because '8' is symmetric, $p(\theta\,|\,\mathbf{x},\,\hat{\mathbf{x}})$ is the sum of two deltas. Figure 2 compares the learned $p_{\boldsymbol{\psi}}(\theta\,|\,\hat{\mathbf{x}})$ given a *simple* uni-modal Gaussian family and a more *flexible* bi-modal mixture-of-Gaussian family with the aggregate distribution $p(\theta\,|\,\hat{\mathbf{x}}) = \sum_{\mathbf{x}} p(\theta\,|\,\mathbf{x},\,\hat{\mathbf{x}})\,p(\mathbf{x})$. Here, the simple distribution is not flexible enough, which results in a large amount of probability mass being wasted on angles with low density under the true data generating process. The definition of 'flexible enough' will, of course, vary depending on the problem.

1. Note, as we will see later, the prototypical example is arbitrary. I.e., the angle could take any value.

Extended Abstract Track

Table 1: True distribution for θ given $\mathbf{x}$ and $\hat{\mathbf{x}}$.

$\mathbf{x}$	$\hat{\mathbf{x}}$	$p(\theta \mid \mathbf{x}, \hat{\mathbf{x}})$
8	8	$0.5 \cdot \delta(\theta - 0°) + 0.5 \cdot \delta(\theta - 180°)$
8	8	$0.5 \cdot \delta(\theta - 30°) + 0.5 \cdot \delta(\theta + 150°)$
8	8	$0.5 \cdot \delta(\theta + 30°) + 0.5 \cdot \delta(\theta - 150°)$

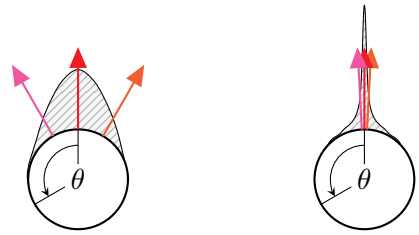
(a) Simple $p_{\boldsymbol{\psi}}(\theta \mid \hat{\mathbf{x}})$ (b) Flexible $p_{\boldsymbol{\psi}}(\theta \mid \hat{\mathbf{x}})$

Figure 2: $p_{\boldsymbol{\psi}}(\theta \mid \hat{\mathbf{x}})$— —and $p(\theta \mid \hat{\mathbf{x}})$— .

Proposition 2: $q_{\boldsymbol{\phi}}(\hat{\mathbf{x}} \mid \mathbf{x})$ must be *fully* invariant w.r.t $\boldsymbol{\eta}$. Consider a dataset of slightly rotated '2's: {2, 2, 2}. Let us assume that the prototype is predicted by a *partially* invariant $q_{\boldsymbol{\phi}}(\hat{\mathbf{x}} \mid \mathbf{x})$—i.e., $q_{\boldsymbol{\phi}}(\hat{\mathbf{x}} \mid \mathbf{x})$ is only similar for '2's which are equivalent under rotations smaller than ρ. Table 2 shows predicted prototypes and the corresponding distributions over θ for two cases[2] $\rho \geq 60°$ and $60° > \rho$. In the first case, each '2' maps to the same prototype. However, in the second case, each '2' can have a unique prototype. This is not desirable since it means that the prototype can contain rotation information, which is invalid under our generative model of the data. Unfortunately, as illustrated by Figure 3, in the second case $p_{\boldsymbol{\psi}}(\theta \mid \hat{\mathbf{x}})$ can become arbitrarily large, which would result in a lower loss. By making $q_{\boldsymbol{\phi}}(\hat{\mathbf{x}} \mid \mathbf{x})$ fully invariant, we avoid this degenerate local optima.

Table 2: True distribution for θ given $\mathbf{x}$ and $\hat{\mathbf{x}}$.

	$\rho \geq 60°$		$60° > \rho$	
$\mathbf{x}$	$\hat{\mathbf{x}}$	$p(\theta \mid \mathbf{x}, \hat{\mathbf{x}})$	$\hat{\mathbf{x}}$	$p(\theta \mid \mathbf{x}, \hat{\mathbf{x}})$
2	2	$\delta(\theta - 0°)$	2	$\delta(\theta - 0°)$
2	2	$\delta(\theta - 30°)$	2	$\delta(\theta - 0°)$
2	2	$\delta(\theta + 30°)$	2	$\delta(\theta - 0°)$



(a) $\rho \geq 60°$ (b) $60° > \rho$

Figure 3: $p_{\boldsymbol{\psi}}(\theta \mid \hat{\mathbf{x}})$— —and $p(\theta \mid \hat{\mathbf{x}})$— .

Proposition 3: $\boldsymbol{\eta}$ must depend on $\hat{\mathbf{x}}$. Consider a dataset of slightly rotated '2's and '8's: {2, 2, 2, 8, 8, 8}, with prototypes '2' and '8'. Table 3 shows $p(\theta \mid \mathbf{x}, \hat{\mathbf{x}})$, the true distribution for θ given $\mathbf{x}$ and $\hat{\mathbf{x}}$, under the data generating process. Figure 4 compares learning distributions over θ and θ given $\hat{\mathbf{x}}$. Note that in the first case, we are able to sample invalid digits such as {2, 2, 2}, which did not appear in the dataset. However, when θ depends on $\hat{\mathbf{x}}$, this does not occur.

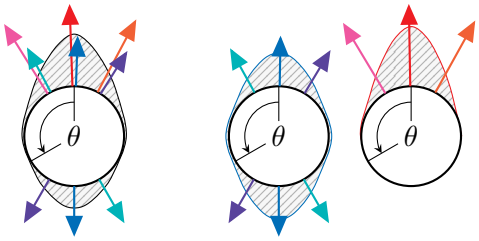
3. Experiments

We validate that our model is able to learn invariance to symmetries by training on a rotated version of the MNIST dataset. We choose $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$ to be a VAE, we implement $p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \mid \hat{\mathbf{x}})$ as a Neural Spline Flow $f_{\mathrm{NN}_{\boldsymbol{\psi}}}$ with base distribution $\mathcal{N}\big(\mu_{\mathrm{NN}_{\boldsymbol{\psi}}}(\hat{\mathbf{x}}), \sigma_{\mathrm{NN}_{\boldsymbol{\psi}}}(\hat{\mathbf{x}})\big)$ to ensure sufficient flexibility, and we approximate $p(\mathbf{x} \mid \hat{\mathbf{x}}, \boldsymbol{\eta}) \approx \mathcal{N}(\mathcal{T}_{\boldsymbol{\eta}}(\hat{\mathbf{x}}), \mathbb{I})$. Further implementation details are in Appendix A. Figure 6 shows that our model successfully (i) learns prototype images that are invariant to rotations—each $\hat{\mathbf{x}}$ is at the same arbitrary angle $\approx 135°$ regardless of the orientation of $\mathbf{x}$—and (ii) reconstructs rotated digits in the correct orientation.

2. The threshold is 60° because each '2' can be transformed into each other '2' by a rotation of 60° or less.

Table 3: True distribution for θ given $\mathbf{x}$ and $\hat{\mathbf{x}}$.

$\mathbf{x}$	$\hat{\mathbf{x}}$	$p(\theta \mid \mathbf{x}, \hat{\mathbf{x}})$
2	2	$\delta(\theta - 0°)$
2	2	$\delta(\theta - 30°)$
2	2	$\delta(\theta + 30°)$
8	8	$0.5 \cdot \delta(\theta - 0°) + 0.5 \cdot \delta(\theta - 180°)$
8	8	$0.5 \cdot \delta(\theta - 30°) + 0.5 \cdot \delta(\theta + 150°)$
8	8	$0.5 \cdot \delta(\theta + 30°) + 0.5 \cdot \delta(\theta - 150°)$



(a) θ (b) $\theta \mid \hat{\mathbf{x}}$

Figure 4: $p_{\boldsymbol{\psi}}(\theta)$ or $p_{\boldsymbol{\psi}}(\theta \mid \hat{\mathbf{x}})$— —and $p(\theta)$ or $p(\theta \mid \hat{\mathbf{x}})$— .

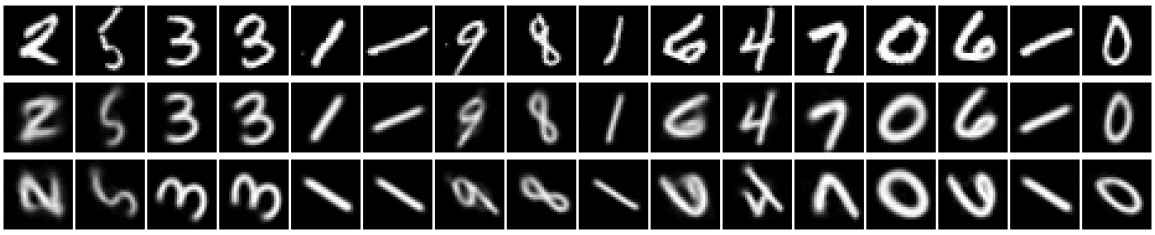


Figure 6: Learning invariance to rotations, $\theta \sim \mathcal{U}(-45°, 45°)$, of the MNIST dataset. **Top:** Samples $\mathbf{x}$ from the test set. **Mid:** Reconstructions of $\mathbf{x}$. **Bot:** prototypes $\hat{\mathbf{x}}$ for each $\mathbf{x}$.

Figure 5 shows that the model behaves sensibly in the presence of different transformation intensities—as the digits are rotated more, the model predicts a wider range of angles. Note that as the degree of rotation in the data increases, so does the range of angles predicted by our model. This is in contrast to methods such as Benton et al. (2020) which learn a 'maximum data augmentation' which decreases as the data is transformed to a greater degree.

4. Conclusion

We have presented a method for learning a generative model with representations that are invariant to symmetries. We discussed various pitfalls for invariant generative models, which motivated our design choices. We also provided some initial qualitative results, which suggest that the model can learn invariant representations for rotated MNIST digits and behaves reasonably under varying degrees of rotation.

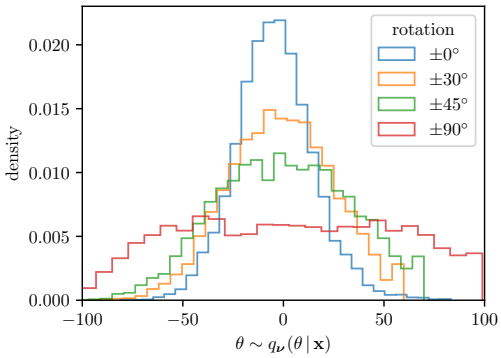


Figure 5: Samples from $q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \mid \mathbf{x})$ for digits rotated by $\theta \sim \mathcal{U}(\theta_{\min}, \theta_{\max})$.

However, this is all a first step towards learning invariant generative models. Important next steps include (i) quantitative results with comparisons to standard generative models, (ii) extension to a wider range of transformations (e.g., other affine transformations, colour transformations), (iii) ablation studies for all of our design choices, (iv) scaling up to more interesting problems than rotated digits, and (v) investigating our motivating example of data efficiency for invariant versus standard generative models.

References

Gregory W. Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks from training data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/cc8090c4d2791cdd9cd2cb3c24296190-Abstract.html.

Alexander Immer, Tycho F. A. van der Ouderaa, Vincent Fortuin, Gunnar Rätsch, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable laplace approximations. *CoRR*, abs/2202.10638, 2022. URL https://arxiv.org/abs/2202.10638.

Eric T. Nalisnick and Padhraic Smyth. Learning priors for invariance. In Amos J. Storkey and Fernando Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 366–375. PMLR, 2018. URL http://proceedings.mlr.press/v84/nalisnick18a.html.

Pola Elisabeth Schwöbel, Martin Jørgensen, Sebastian W. Ober, and Mark van der Wilk. Last layer marginal likelihood for invariance learning. *CoRR*, abs/2106.07512, 2021. URL https://arxiv.org/abs/2106.07512.

Tycho F. A. van der Ouderaa and Mark van der Wilk. Learning invariant weights in neural networks. *CoRR*, abs/2202.12439, 2022. URL https://arxiv.org/abs/2202.12439.

Mark van der Wilk, Matthias Bauer, S. T. John, and James Hensman. Learning invariances using the marginal likelihood. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9960–9970, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/d465f14a648b3d0a1faa6f447e526c60-Abstract.html.

Max Welling. Do we still need models or just more data and compute? *University of Amsterdam, April*, 20, 2019.

Appendix A. Additional Experimental Details

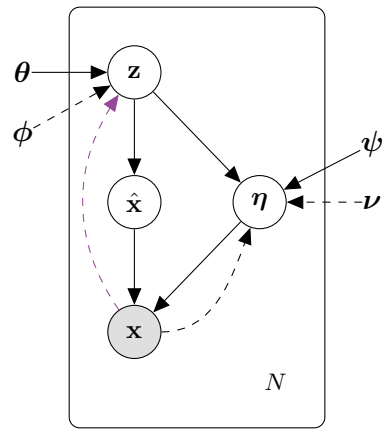


Figure 7: Generative (solid lines) and inference (dashed lines) models. Purple lines represent invariant NNs.

The full generative model for our experiments, modelling $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$ with a VAE, is:

$$\mathbf{z} \sim \mathcal{N}\left(\boldsymbol{\mu},\, \boldsymbol{\sigma} \cdot \mathbb{I}\right), \tag{7}$$

$$\hat{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \,|\, \mathbf{z}), \tag{8}$$

$$\boldsymbol{\eta} \sim p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \,|\, \mathbf{z}), \tag{9}$$

$$\mathbf{x} \sim p(\mathbf{x} \,|\, \hat{\mathbf{x}},\, \boldsymbol{\eta}). \tag{10}$$

The full graphical model is shown in Figure 7. We use the following ELBO for jointly learning the generative parameters $\{\boldsymbol{\theta},\, \boldsymbol{\psi}\}$ and the variational parameters $\{\boldsymbol{\phi},\, \boldsymbol{\nu}\}$:

$$\log p(\mathbf{x}) = \log \iiint p(\mathbf{x},\, \hat{\mathbf{x}},\, \mathbf{z},\, \boldsymbol{\eta})\, d\hat{\mathbf{x}}\, d\mathbf{z}\, d\boldsymbol{\eta} \tag{11}$$

$$= \log \iiint p(\mathbf{x} \,|\, \hat{\mathbf{x}},\, \boldsymbol{\eta})\, p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \,|\, \mathbf{z})\, p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \,|\, \mathbf{z})\, p(\mathbf{z})\, d\hat{\mathbf{x}}\, d\boldsymbol{\eta}\, d\mathbf{z} \tag{12}$$

$$= \log \iiint p(\mathbf{x} \,|\, \hat{\mathbf{x}},\, \boldsymbol{\eta})\, p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \,|\, \mathbf{z})\, p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \,|\, \mathbf{z})\, p(\mathbf{z})\, \frac{q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \,|\, \mathbf{x})}{q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \,|\, \mathbf{x})} \frac{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})} d\hat{\mathbf{x}}\, d\boldsymbol{\eta}\, d\mathbf{z} \tag{13}$$

$$= \log \mathop{\mathbb{E}}_{q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \,|\, \mathbf{x})\, q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})\, p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \,|\, \mathbf{z})} \left[\frac{\mathcal{N}(\mathbf{x} \,|\, \mathcal{T}_{\boldsymbol{\eta}}(\hat{\mathbf{x}}), \sigma\mathbb{I})\, p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \,|\, \mathbf{z})\, p(\mathbf{z})}{q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \,|\, \mathbf{x})\; q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})} \right] \tag{14}$$

$$\geq \mathop{\mathbb{E}}_{q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \,|\, \mathbf{x})\, q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})\, p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \,|\, \mathbf{z})} \left[\log \mathcal{N}(\mathbf{x} \,|\, \mathcal{T}_{\boldsymbol{\eta}}(\hat{\mathbf{x}}), \sigma\mathbb{I}) + \log p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \,|\, \mathbf{z}) + \log p(\mathbf{z}) - \log q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \,|\, \mathbf{x}) - \log q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})\right] \tag{15}$$

$$\equiv \mathcal{L}\left(\boldsymbol{\theta},\, \boldsymbol{\psi},\, \boldsymbol{\phi},\, \boldsymbol{\nu}\right). \tag{16}$$

We estimate the ELBO with a single Monte-Carlo sample. We follow Benton et al. (2020); van der Ouderaa and van der Wilk (2022); Immer et al. (2022) in constructing approximately invariant NNs via expectation. Specifically, we construct an invariant encoder $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$ from a non-invariant encoder $\hat{q}_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$:

$$q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}) \equiv \mathbb{E}_{\boldsymbol{\eta}}\left[\hat{q}_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathcal{T}_{\boldsymbol{\eta}}(\mathbf{x}))\right]. \tag{17}$$

Extended Abstract Track

We approximate this expectation using 10 Monte-Carlo samples.

We use `jax` with `flax`, `optax`, and `distrax` to implement our model.Our VAE encoder $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is a 3-layer CNN, with {64, 128, 256} channels. parameterising a heteroskedastic Gaussian distribution. The VAE decoder $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{z})$ is a 3-layer CNN, with transposed convolutions of {256, 128, 64} channels, parameterising a homoskedastic Gaussian distribution. The latent dimension is 20. $p_{\boldsymbol{\psi}}(\boldsymbol{\eta} \mid \mathbf{z})$ is a Neural Spline Flow with four bins, two flow layers, and base distribution $\mathcal{N}\big(\mu_{\mathrm{NN}_{\boldsymbol{\psi}}}(\mathbf{z}), \sigma_{\mathrm{NN}_{\boldsymbol{\psi}}}(\mathbf{z})\big)$ where the $\mathrm{NN}_{\boldsymbol{\psi}}$ is a two-layer MLP with hidden sizes {64, 32}. $q_{\boldsymbol{\nu}}(\boldsymbol{\eta} \mid \mathbf{x})$ follows the same structure. For both flow models, each flow layer is parameterised by a 2-layer MLP of with hidden sizes {64, 32}.

All neural networks use `relu` activation functions. The VAE encoder and decoder use LayerNorm. We use the `adamw` optimizer with a initial learning rate of 1×10^{-4} and a weight decay of 1×10^{-4}. We train for 15000 steps with a batch size of 256. We linearly increase the LR to 1×10^{-3} over the first 500 training steps and then use a cosine decay to reduce it back to 1×10^{-4} by the end of training. We multiply the VAE KL-divergence by a factor β, which has an initial value of 10 and is decayed to 1, using a cosine schedule, by the end of training.