Learning Invariances with Generalised Input-Convex Neural Networks

Vitali Nesterov 1  Fabricio Arend Torres 1  Monika Nagy-Huber 1  Maxim Samarin 1  Volker Roth 1

Abstract
Considering smooth mappings from input vectors to continuous targets, our goal is to characterise subspaces of the input domain, which are invariant under such mappings. Thus, we want to characterise manifolds implicitly defined by level sets. Specifically, this characterisation should be of a global parametric form, which is especially useful for different informed data exploration tasks, such as building grid-based approximations, sampling points along the level curves, or finding trajectories on the manifold. However, global parameterisations can only exist if the level sets are connected. For this purpose, we introduce a novel and flexible class of neural networks that generalise input-convex networks. These networks represent functions that are guaranteed to have connected level sets forming smooth manifolds on the input space. We further show that global parameterisations of these level sets can be always found efficiently. Lastly, we demonstrate that our novel technique for characterising invariances is a powerful generative data exploration tool in real-world applications, such as computational chemistry.

1. Introduction
A major goal of generative methods is the exploration of the data distribution by discovering novel and relevant examples beyond training data points. Typically, this involves some kind of inductive bias or side-information to allow an informed exploration of a domain of our interest. For instance, in chemical applications such additional information can be a chemical property, which facilitates guided discovery of novel molecular topologies or even restricts the exploration domain to some constant property value. These subspaces of the input domain are invariant with respect to smooth mappings $f : x \mapsto y$ from input vectors $x \in \mathbb{R}^n$ to continuous targets $y$ and can be implicitly defined by $\alpha$-level sets $L_\alpha = \{ x : f(x) = \alpha \}$. Our goal is to find a global parametric form for such manifolds.

A global parameterisation of level sets is a difficult topological problem. It exists only if the level sets of $f$ are connected. Figure 1 illustrates the parameterisation of a specific level set of a $(n-1)$-dimensional manifold in the input space. The property values are defined by a Rosenbrock function forming a curved valley. Even though the level sets are connected and closed it is not obvious how to find a global parameterisation directly in the input space. However, it can be guaranteed under certain constraints that such a global parametric form exists, and that it can be found efficiently in the bijectively transformed domain. Generated points of the level set can then be transformed back into the input space by the inverse mapping $x = h^{-1}(z)$.

In this work, we introduce a novel and flexible class of neural networks that generalise input-convex networks. By construction, these networks represent functions that are guaranteed to have connected level sets. For a global parameterisation of the level sets, we choose hyper-spherical coordinates, and we proof that this type of parameterisation always exists for level sets of functions represented by our novel class of neural networks. One of the main advantages of such a global parameterisation is that interpolation...
paths between points on a level set can be easily constructed. Our novel approach to characterise subspaces of the input domain forms a powerful technique for informed data exploration in diverse applications.

We summarise our contributions as follows:

1. We propose a novel class of neural networks that implements functions which always have connected level sets that can be parameterised globally.

2. The global parameterisation themselves can be efficiently constructed using hyper-spherical coordinates. They are highly useful tools for building grid-based approximations of level sets, for sampling points and finding interpolation paths on the level sets.

3. Based on both synthetic and real-world datasets, we show that our approach is a highly useful tool for exploratory data analysis under side-information.

2. Related Work

Existing methods for learning invariances usually belong to one of two categories: (i) generative latent variable models using a partitioned latent space, and (ii) numerical level set methods that directly operate in the input space.

2.1. Invariant Generative Models

The multi-level variational autoencoder (Bouchacourt et al., 2018) separates the latent space into a local and global feature space, where the former is only relevant for a subgroup and the latter shares common factors of variation.

A different approach is to use adversarial networks (Goodfellow et al., 2014), such as in Creswell et al. (2017); Lample et al. (2017), where the discriminator learns to predict attributes and the encoder learns to prevent it. As a result, there is no information about the property in the latent space and the model relies on additional provided input to construct data with a given property. In Klyks et al. (2018) a mutual information regulariser is learned to isolate sub-spaces for binary targets; Wieser et al. (2020) extends this approach to continuous targets. The cycle consistency (Zhu et al., 2017) technique, as an alternative, is also used for learning disentangled representations (Jha et al., 2018; Samarin et al., 2021).

While these models are to some extent capable of characterising invariances, they typically do not provide us a global parameterisation of the level sets, and not even the existence of a connected level set can be guaranteed in general.

2.2. Numerical Level Set Methods

 Classical numerical level set methods (Osher & Fedkiw, 2003; Sethian, 1999) attempt to directly find level sets in the input domain \(D \subset \mathbb{R}^d\) via boundary evolution. Let \(\Omega \subset \mathbb{R}^d\) be the enclosed part of an object defined by the boundary \(\partial \Omega\). The zero isovalue of a signed distance function \(\phi : \mathbb{R}^d \rightarrow \mathbb{R}\), defined as

\[
\phi(x) < 0, \quad \forall x \in \Omega \setminus \partial \Omega \\
\phi(x) = 0, \quad \forall x \in \Gamma := \partial \Omega \\
\phi(x) > 0, \quad \forall x \in D \setminus \mathbb{\Omega}
\]

with \(\mathbb{\Omega} := \Omega \cup \partial \Omega\), coincides with the boundary \(\Gamma\), with which we formally define a level set. By introducing the parameter \(t \in \mathbb{R}_+\) as pseudo-time, distinct level sets can be described via \(\phi(x, t) = 0\) of an evolving boundary \(\Gamma(t)\) moving in normal direction. The dynamics of the boundary are described by the Hamilton-Jacobi equation \(\frac{\partial \phi}{\partial t} + V \|\nabla \phi\| = 0\). The Hamilton-Jacobi equation is a special case of the convection equation \(\frac{\partial \phi}{\partial t} + v \cdot \nabla \phi = 0\), with velocity \(v = V \frac{\nabla \phi}{\|\nabla \phi\|}\) in normal direction of the level set and a scaling factor \(V\) of the velocity. Numerical solving methods Osher & Fedkiw (2003); Sethian (1999) evolve an initial boundary at \(t = 0\) over time for obtaining all level sets. Although these methods usually work well in low dimensions, they are difficult to apply in higher dimensions due to their reliance on discretisations of the input domain. Furthermore, a numerical boundary evolution provides only approximated solution curves, but not a parametric form of the level sets.

2.3. Preliminaries

The method proposed in this work for providing a global parameterisation of level sets relies on the composition of two types of functions, namely bijective mapping functions and (strictly) input-convex functions, in order to obtain a certain type of generalised input convexity that implies connected level sets.

2.3.1. Bijective Neural Networks

Normalising flows are generative models that rely on diffeomorphisms for warping base distributions into more complex ones. One of the most common applications is density estimation (Kobyzev et al., 2020; Papamakarios et al., 2019). As a result of this research direction, a wide range of flexible, efficient and trainable bijective network architectures has become readily available. More specifically, we rely on the autoregressive networks presented in (Germain et al., 2015) for our synthetic experiments.

2.3.2. Input Convex Neural Networks

Based on the Wasserstein Generative Adversarial Network (WGAN) (Arjovsky et al., 2017) which proposes usage of Lipschitz constraints, numerous methods such as WGAN-GP (Gulrajani et al., 2017) and WGAN-LP (Petzka et al.,
extend this idea to different kinds of K-Lipschitz constraints on the gradients. Spectral normalisation (Miyato et al., 2018) globally constrains the upper bound of the Lipschitz constant on the neural network weights. Sapkota & Bhattarai (2021) proposes an invex function approximator where the gradients are regularised w.r.t. the gradient direction given a reference invex function. These techniques provide only approximate solutions, while a more interesting approach has been introduced by Amos et al. (2017), where the network by construction guarantees input convexity providing a basis for our work. The Input Convex Neural Network (ICNN) (Amos et al., 2017) constructs convex functions applying non-negativity constraints on the neural network weights and using non-decreasing activation functions. We make use of ICNNs and extend it to a more generalised notion of input convexity in neural networks.

3. Methodology

An essential prerequisite for the existence of a global parameterisation of level sets is the connectedness of all level sets. In the following we show that such connectivity follows from a certain convexity-type constraint imposed on the functions that can be represented by a neural network. We then extend the discussion in the context of composite functions and the resulting properties of such, and show how to exploit these to characterise invariances in a global parametric form.

3.1. Generalisations of Convexity

Definition 1. Let $X \subseteq \mathbb{R}^n$ be an open set. The differentiable function $f : X \rightarrow \mathbb{R}$ is invex if there exists a vector function $\eta : X \times X \rightarrow \mathbb{R}^n$ such that $f(x) - f(y) \geq \eta(x,y)^T \nabla f(y), \forall x, y \in X$.

From the definition it follows that invexity is an extension of convexity: choosing $\eta(x,y) = x - y$ in Definition 1, invexity coincides with the definition of a differentiable convex function. An alternative definition of invexity is given by Ben-Israel & Mond (1986) in the following theorem:

Theorem 3.1. A differentiable function $f$ is invex if and only if every stationary point is a global minimum.

For dimension $n \geq 2$, invexity is a generalisation of pseudo-convexity, such that the set of pseudo-convex functions is strictly contained in the set of invex functions. For $n = 1$, pseudo-convexity is the same as invexity. Furthermore, a differentiable pseudo-convex function has the property that all stationary points are global minima, i.e. being invex, however, there also exists invex functions that are not pseudo-convex (Tanaka et al., 1989).

Quasi-convexity is another generalisation of convexity. The main characteristics of a quasi-convex function is that all of its sub-level sets are convex. The class of quasi-convex functions, however, has only a partial overlap with the class of invex functions. A classical example is $f(x) = x^3$, which is quasi-convex, but not invex because the saddle-point is at $x = 0$. It can be also shown that an invex function which fails to be pseudo-convex, cannot be quasi-convex either (Giorgi, 1990).

We now characterise the invexity of function compositions:

Theorem 3.2. Let $X \subseteq \mathbb{R}^n$ be an open set. Let $g : \mathbb{R} \rightarrow \mathbb{R}$ be a differentiable strictly monotone increasing function. Let $h$ be a differentiable injective function $h : X \rightarrow h(X) \subseteq \mathbb{R}^n$ with differentiable inverse $h^{-1}$. Let $f : h(X) \rightarrow \mathbb{R}$ be convex and differentiable. Then $F = g \circ f \circ h$ is invex on $X$.

Similar theorems already appeared in Keller et al. (2018) and Mishra & Giorgi (2008). In both cases, however, with some notation inaccuracies. Therefore, we add a formal proof here:

Proof. Convexity of $f = g^{-1} \circ F \circ h^{-1}$ implies $\forall x, y \in h(X)$:

$$
(g^{-1} \circ F \circ h^{-1})(x) - (g^{-1} \circ F \circ h^{-1})(y) \geq (x - y)^T \nabla (g^{-1} \circ F \circ h^{-1})(y) \quad (1)
$$

Using the chain rule, the right-hand side of this inequality can be written as:

$$
(x - y)^T J_{h^{-1}}(y) \nabla F(h^{-1}(y)) \cdot (g^{-1})'(F(h^{-1}(y))) \quad (2)
$$

where $J_{h^{-1}}$ denotes the Jacobian matrix of the inverse transformation $h^{-1}$, and $(g^{-1})'$ is the derivative of the scalar function $g^{-1}$.

Assume $x^*$ is a stationary point of $F$, i.e. $\nabla F(x^*) = 0$.

Since $h$ is injective, it induces a bijection onto its image $h(X)$ with inverse $h^{-1}$. Surjectivity of $h^{-1}$ then implies that there exists $y \in h(X)$ such that $h^{-1}(y) = x^*$. It follows that the r.h.s. is zero and

$$
(g^{-1} \circ F \circ h^{-1})(x^*) \geq (g^{-1} \circ F)(x^*) \forall x \in h(X). \quad (3)
$$

Since $g^{-1}$ is strictly monotone increasing, $(F \circ h^{-1})(x) \geq F(x^*) \forall x \in h(X)$.

Again, surjectivity of $h^{-1}$ implies that $\forall z \in X, \exists x \in h(X)$ such that $z = h^{-1}(x)$, and therefore, $F(z) \geq F(x^*) \forall z \in X$. Hence, every stationary point of $F$ yields a global minimum on $X$, such that $F$ is invex on $X$. \qed

It should be noted that in the setting of Theorem 3.2 and using the terminology in Horst (1984), $F = g \circ f \circ h$ can be called a “$(h, g^{-1})$-convex” function. Such functions belong
to the family of arcwise-convex functions, all of which are known to be invex given that all domain transformations are differentiable (Rueda, 1989).

The composition $g \circ f$ is by construction strictly quasi-convex, which follows from the strict monotonicity of $g$ and the convexity of $f$. It is also invex, which follows from the special case $h(x) = x$ in the above theorem. These properties are important to characterise invariances in a global parametric form as we intended to do.

### 3.2. Parameterisation of Level Sets

Given a differentiable function $f : \mathbb{R}^{n-1} \times \mathbb{R} \to \mathbb{R}$ and $(\theta, r) \mapsto f(\theta, r)$, let an $\alpha$-level set, induced by the equation $f(\theta, r) = \alpha$, be parameterisable if there exists a differentiable function $\gamma : \mathbb{R}^{n-1} \to \mathbb{R}$, such that $\gamma(\theta) = r$ and $f(\theta, \gamma(\theta)) = \alpha \forall \theta \in \mathbb{R}^{n-1}$. In other words, we require that the solutions of $f = \alpha$ can be described as a graph of the function $\gamma$. The parameterisation itself is provided by the mapping $\theta \mapsto (\theta, \gamma(\theta)) = \Gamma(\theta)$. Questions relating to local parameterisation in the neighbourhood of a given point can be answered by the implicit function theorem, showing that every smooth level set can be locally parameterised if the Jacobian is invertible.

Here, we are mainly interested in global parameterisations of level sets as it allows to interpolate between two data points on a level set and to find trajectories on the manifold induced in the input space. However, finding a global parameterisation is a difficult topological problem in general. Furthermore, a necessary condition for the existence of global parameterisations for the level set defined by $f(x, r) = \alpha$ is connectedness. In order to make $\gamma(\theta)$ well-defined everywhere, we also might want to look at compact level sets.

In general, connectedness and compactness might be difficult to characterise, however, we can benefit from invexity of our model $g \circ f \circ h$ and compute the parameterisation not directly in the input space $X$, but on its image $h(X)$ under the injective map $h$. We then can use $h^{-1}$ to map the level set back to the input space (see Figure 2). The advantage is that by construction, $g \circ f$ is strictly quasi-convex. The sub-level sets of strictly quasi-convex functions are strictly convex sets and therefore the level sets are always connected.

**Theorem 3.3.** Suppose $f$ is differentiable, strictly convex and $\text{dom}(f) = \mathbb{R}^{n}$. Let $g : \mathbb{R} \to \mathbb{R}$ be a differentiable strictly monotone increasing function. Given that $\inf((g \circ f)(x))$ is attainable implies that all sub-level sets $L_\alpha(g \circ f) = \{x : (g \circ f)(x) \leq \alpha\}$ are compact $\forall \alpha \in \mathbb{R}$.

**Proof.** For a differentiable, strictly convex $f$, the proof is given in Tejgorsk & Singer (2012). The statement follows from observing that given strict monotonicity of $g$, $\forall \alpha \in \mathbb{R}$, $\exists \alpha' \in \mathbb{R}$, such that $L_\alpha(g \circ f) = L_{\alpha'}(f)$.

Note that the compactness of the sub-level sets of $g \circ f$ implies compactness of their boundaries (i.e. the level sets).

### 3.3. Retrieval of Level Curves

Given that the infimum is attainable, and therefore becomes a minimum, the strategy is as follows: We first locate the minimum of the strictly quasi-convex function $g \circ f$ in the latent space via gradient descent. According to the Theorem 3.2, there will be exactly one stationary point $x^*$, which is the minimum. It defines the origin of our new coordinate system. To parameterise the $\alpha$-level set we use (hyper-)spherical coordinates in the form $\gamma(\theta; \Phi) = r$, where $r$ is the radius and $\theta \in [0, 2\pi)$ the azimuthal angle, assuming that all $n-2$ latitudes $\Phi = \{\phi_i\}, \phi \in [0, \pi]$ of the $n$-sphere are fixed. Thus, the parameterisation is provided by $((\theta, \Phi), \gamma(\theta, \Phi)) = R(\theta, \Phi)$. To get the first point on the level curve we initialise the azimuth $\theta$ and latitudes $\Phi$ and adjust the radius $r$ via line-search given a fixed $\alpha$-level. In order to interpolate on this level set between the initial point and any other point, we linearly interpolate the angles between these two coordinates and re-adjust the radius via line-search for each intermediate interpolation step. This is very efficient, because only a sequence of one-dimensional line-searches is required to retrieve points on a level curve.

Note that this (hyper-)spherical parameterisation always works for strictly quasi-convex functions. This is because their restrictions to lines are also strictly quasi-convex, and

---

**Figure 2.** A depiction of a level set parameterisation of the Rosenbrock function. (a) Parameterising a level curve of $g \circ f$ by first finding the minimum via gradient descent and then using polar coordinates centered at this minimum. The function values of $g \circ f$ are represented as coloured points on a grid in a two-dimensional latent space. The black curve indicates one specific level curve, parameterised in polar coordinates and found by linearly interpolating between $(\theta_i, \gamma(\theta_i))$ points on the level curve. (b) Transforming level set points back to the input space with $h^{-1}$. Note that a global parameterisation of the “banana”-shaped curve would not be possible with polar coordinates in the input space.
along every line starting in the origin, the function values are strictly monotone increasing with increasing radius, which implies that there is exactly one intersection of this line and the level set. As Figure 2a shows, the pointer slides along the curve intersecting the \(\alpha\)-level set only once for each angle \(\theta_i\), while in the input space, depicted in Figure 2b, the pointer crosses the corresponding level set multiple times.

3.3.1. Level Sets within a Subspace

In case the infimum is not attainable or we are only interested in a specific region of the input domain, the level sets within any defined subspace can be always parameterised in the same manner as described in Sec. 3.2. As shown previously, the minimum within any compact subspace \(\Omega \subseteq R^n\) can be always attained under the convexity argument. Finding the minimum on a subspace is a constrained optimisation problem of a quasi-convex and smooth function, which can be solved with projected gradient descent or other techniques. Given that the minimum in \(\Omega\) is attainable, the remaining steps are analogous to the strategy described before.

3.4. Invexity of Composite Neural Networks

Given an input-invex composition of functions \(g \circ f \circ h\), one can implement a bijective mapping \(h : X \rightarrow h(X)\) by any type of available bijective layers, such as described in Sec. 2.3.1. In our model we use an autoregressive flow (Germain et al., 2015). Function \(g\) can be any strictly monotone increasing function. We choose \(g\) to be weighted sums of simple strictly monotone increasing basis functions, such as identities or hyperbolic tangent functions, with positive weights. We consider a strictly input-convex neural network for \(f\). For this, we extend the theorem on input convexity of Amos et al. (2017) to strict input convexity:

**Theorem 3.4.** Define input \(z = h(x)\), parameters \(\tau = \{W_i^{(c)}, W_i^{(z)}, b_i\}, i \in \{1, \ldots, k - 1\}\), and output \(c_k = y\). Then the function \(f(z; \tau) = y\) recursively defined by

\[
c_{i+1} = \sigma \left( W_i^{(c)} c_i + W_i^{(z)} z + b_i \right)
\]

is strictly convex in input \(z\) if all weights \(W_i^{(c)}\) are non-negative (and no rows of \(W_i^{(c,z)}\) are all zeroes) and the activation function \(\sigma\) is a strictly convex and increasing function.

**Proof.** Analogous to the proof in Amos et al. (2017), the statement follows from observing that non-negative sums of strictly convex functions remain strictly convex and that compositions of a strictly convex and increasing function with a strictly convex function preserve strict convexity. See Appendix Sec. A.1 for more details on the proof.

Our experiments with classical bijective layers show that an end-to-end training in combination with an input-convex network often leads to practical problems related to local optima and overly complex transformations that have weak extrapolation capabilities. We therefore integrate our approach into a VAE framework, where the stochasticity in the network has a regularisation effect on the bijective mapping. Here, the invertible transformation \(h\) has both, the role of the encoder \(\mu_z(x) = h(x)\) and the decoder \(\mu_y(z) = h^{-1}(z)\). The input-convex network \(f\) forms an additional decoder \(\mu_y(x) = f(z)\) for the regression target \(y\). That gives us a big advantage in the ability to decode the sampled latent \(z\)-values with the inverse mapping.

4. Model

The variational autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014) is a deep latent variable model that combines amortised variational inference (encoder) with a generative process (decoder) to learn a probabilistic generative model. The decoder network parameterises the likelihood \(p(x|z)\) of the data \(x\) given a latent code \(z\). The encoder network maps the data \(x\) to the parameters of the approximate posterior \(q(z|x)\) of the latent variable \(z\). The variational distribution \(q\) is usually (but not necessarily) Gaussian, such that the encoder parameterises the first two moments of \(q(z|x)\). To enable an informed exploration of the latent space (Wieser et al., 2020; Keller et al., 2021; Samarin et al., 2021), the model can be extended with an additional decoder for a variable \(y\) as a side-information. The likelihood \(p(y|z)\) and the variational distribution \(q(z|x)\) aims to approximate \(p(z|x, y)\), while the encoder has only access to the input \(x\).

More specifically, our model is constructed in the following manner. We parameterise the encoder network with \(\phi\) and restrict it to an isotropic Gaussian \(q_{\phi}(z|x) = N(\mu_z(x), \sigma_z^2 I_{\text{dim}(z)})\), where \(\sigma_z\) is a trainable scale parameter and \(I\) is the identity matrix. The prior \(p(z) = N(0, I_{\text{dim}(z)})\) is Gaussian distributed. The joint distribution \(p_{\theta, \tau}(x, y|z)\) factorises into the likelihood of the input \(p_{\theta}(x|z)\) and the side-information \(p_{\tau}(y|z)\) given the latent representation \(z\). For a continuous variable \(x\), the likelihood is an isotropic Gaussian \(p_{\theta}(x|z) = N(\mu_z(x), \sigma^2 z I_{\text{dim}(z)})\) and is parameterised with \(\theta\). Since we consider a continuous target \(y\), the likelihood \(p_{\tau}(y|z)\) is a Gaussian and is parameterised with \(\tau\). In order to train our probabilistic model, we minimise the following objective function

\[
\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{p(x,y)} \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log p_{\theta, \tau}(x, y|z) \right] + \beta \cdot \mathbb{E}_{p(x)} \mathbb{D}_{\text{KL}} \left[ q_{\phi}(z|x) || p(z) \right],
\]

where \(\mathbb{D}_{\text{KL}}\) denotes the Kullback-Leibler divergence and \(\beta\) is a trade-off parameter to control the latent channel capacity (Higgins et al., 2016). The gradients of the expectation are
estimated using the reparameterisation trick (Kingma & Welling, 2013; Rezende et al., 2014). Note that for $\beta = 1$, Eq. (5) coincides with the negative variational lower bound of the joint marginal log likelihood $p(x, y)$.

4.1. Bijection through Cycle Consistency

For a real-world application it might be interesting to combine approximated bijective mappings with dimensionality reduction. Therefore, we alternatively use a cycle-consistency regularisation technique first introduced in the CycleGAN (Zhu et al., 2017), to approximate bijection in the encoder-decoder ensemble. In this model variation, both of them form two separate networks implementing standard forward layers, where the decoder approximates the inverse of the encoder. We compute cycle-consistency loss terms on the regression variable $y$ instead of on the latent variables $z$. This is adapted from Samarin et al. (2021), and is aimed to avoid adverse local minima during the training process.

Our loss terms in Eq. (6) are computed as follows. Given input $x_{\text{obs}}$ we get decoded $(x, y)$-pairs $x_{\text{obs}} \rightarrow z \rightarrow (x, y)$ and perform a cycle operation $x \rightarrow z' \rightarrow (x', y')$ to get the cycle prediction $y'$. We compute $||y - y'||_2$ on the training points to ensure consistency of input $x_{\text{obs}}$ and reconstruction $x$ w.r.t. property prediction $y$. In addition, we sample new latent codes $\tilde{z}$ uniformly at random and decode $\tilde{y}$ and $\tilde{y}'$ in the same manner. The loss term $||\tilde{y} - \tilde{y}'||_2$ enforces a smooth regularisation of the latent space beyond the training points.

$$L_{\text{cycle}} = ||y - y'||_2 + ||\tilde{y} - \tilde{y}'||_2$$ (6)

The complete objective function of the model is defined as follows

$$L = L_{\text{VAE}} + \gamma \cdot L_{\text{cycle}},$$ (7)

where $\gamma$ constant regulates the balance between the cycle and the reconstruction loss terms.

5. Experiments

Our experiments are organised in the following manner: First, we evaluate the capabilities of our model on synthetic data and show differences to the $\beta$-VAE (Higgins et al., 2016) with a cycle-consistency regularisation. Second, we evaluate our model on a modified version of Fashion-MNIST as well as on a chemical application.

5.1. Synthetic Dataset

The experiments on synthetic data provide an intuition for our model and show how we parameterise the level sets. Thereby, we focus on learning input invexity on a two-dimensional input. We consider these experiments as they allow to visualise the data and help to understand the idea on the first glance. The focus is on learning and visualising input invexity on a two-dimensional example. For our first experiment, we use a modified Rosenbrock function defined as

$$f(x_1, x_2) = \sqrt{\left(1 - 10x_1\right)^2 + 100 \left(10(x_2 - x_1^2)\right)^2},$$ (8)

which is a differentiable invex function with one stationary point, i.e. the global minimum, at $x_1 = x_2 = 0.1$. For the second set of experiments, we use a mixture of two Gaussians with means $\mu_1 = -0.2, \mu_2 = 0.2$ and (diagonal) covariances $\sigma_1^2 = \sigma_2^2 = 0.02 \cdot \text{Id}$. The property $y$ is multiplied by $-1$ to invert maxima to minima. For both data sets, we sample 1,600 data points $x \in \mathbb{R}^2$ on the $[-0.4, 0.4]^2$ grid and get the corresponding one-dimensional property values $y$ with either the Rosenbrock function or the Gaussian mixture.

For the experiments on synthetic data, we use two different models. As a baseline model, we use a $\beta$-VAE with approximated bijective mappings through cycle-consistency. The generalised input-convex model is based on the $\beta$-VAE, where the encoder is by construction bijective and the property decoder is a strict input-convex neural network (see Sec. 3.4). See Appendix Sec. A.3 for more details on the datasets and the model architecture.

In the first experiment, we demonstrate our results on the dataset sampled from the two-dimensional Rosenbrock function with one stationary point. To this end, we compare the baseline to the generalised input-convex model. Both of the models achieve similar results concerning the property prediction and input reconstruction. Since our model uses a bijective encoder, the encoder mapping can be directly inverted back to the input space. Despite comparable results, the baseline model provides no guarantees for the ex-
Figure 4. A depiction of the difference between the latent space arrangement of the β-VAE with cycle-consistency regularisation and our model. (a) Depicts parameterised level sets on the input convex latent space and (c) shows parameterised level sets on the unconstrained latent space. (b) and (d) depicts the corresponding reconstruction of the $(x, y)$-pairs.

Figure 5. Interpolation on the level set between two test-images from the Fashion-MNIST dataset. The left- and right-most images correspond to the reference images. In between, the reconstructions alongside the interpolation path on the level set are depicted.

Figure 3. Reconstruction of $X$ and $Y$ with a bijective mapping. Note that the objects on the level set smoothly transform from a pullover to trousers, but preserve the fixed rotation angle. In Figure 5, we demonstrate that the objects on the level set smoothly transform from a pullover to trousers, but preserve the fixed rotation angle.

5.2. Rotated Fashion-MNIST

As a more challenging dataset, we use a modified version of the Fashion-MNIST (Xiao et al., 2017). In order to create a continuous property value $y$ for each item, we rotate the images uniformly at random, $U([-45°, 45°])$, around their centre. The images are zero-padded to a size of $36 \times 36$ pixels to avoid cropping artefacts as a side effect of the rotation. We also excluded bags, ankle boots and sandals from the dataset. Due to a 1296-dimensional input space, we make use of the pseudo-bijective VAE (see Sec. 4.1) to learn a more compact 40-dimensional latent representation. Further details on the data and the model architecture are provided in Appendix Sec. A.4.

In the experiment, we interpolate between two test images of the same rotation angle but different class labels, i.e. between a pullover and trousers. Our level set choice is a $36°$ counter clockwise rotation. In Figure 5, we demonstrate that the objects on the level set smoothly transform from a pullover to trousers, but preserve the fixed rotation angle.

5.3. Small Organic Molecules

As a real-world application, we use the QM9 dataset (Ramakrishnan et al., 2014) which includes 133,885 small organic molecules. The molecules consist of up to nine heavy atoms (C, O, N and F), not including hydrogen atoms. Each molecule includes chemical properties which are calculated
with the Density Functional Theory methods. In our experiments, we select a subset with a fixed stoichiometry ($C_7O_2$) which consists of 16,302 molecules. We choose the band gap energy as the property of our interest.

We represent the molecules as a graph $G = (V, E)$, providing atom types and molecular bonds. The node set $V$ consists of up to 9 heavy atoms. The weighted edges set $E$ is represented as an upper triangular weighted adjacency matrix. Both graph components are one-hot encoded and include an additional blank symbol. We further employ a canonical ordering (O’Boyle, 2012) of the atoms for a standardised enumeration among all the molecules in the dataset (Nesterov et al., 2020). In total, the input space involves 189 dimensions. To learn a low-dimensional representation we approximate the bijective mapping (see Sec. 4.1), such that our latent space consists of 22 dimensions. To improve the performance, we compute the cycle loss on the differentiable approximation of the one-hot categorical distribution. See Appendix Sec. A.5 for more details on the model architecture.

Our training set consists of 13,800 molecules. On the test set of 2,500 molecules, we are able to reconstruct 97% of the graphs. For prediction of the band gap energies a MAE of 4.5 kcal mol$^{-1}$ is achieved. We compute the invariance as the MAE between the predicted property $y$ and the cycle prediction $y'$, for which we achieve 0.4 kcal mol$^{-1}$ on the test set. In an additional experiment, we interpolate between two reference molecules as shown in Figure 6. To do so, we first identify the largest cluster on the band gap energy values in our dataset. We then compute the mean of these cluster, which is approximately 138 kcal mol$^{-1}$ and define this as our level set of interest. Subsequently, we choose two reference molecules with a band gap close to 138 kcal mol$^{-1}$ and compute the interpolation path on the manifold spanned by the angles. In Figure 6, unique molecular graphs are depicted that can be found on the interpolation path. All these molecules could be retrieved in the QM9 database, such that we also know their ground-truth energies. Note that the indicated true property values are close to the fixed level set and are within the range of the model error.

6. Conclusion

This paper introduces a novel and flexible class of neural networks that generalise input-convex networks. Building on the notion of invexity, we have shown in an extensive proof that such networks learn mappings that imply connected level sets by construction. We motivate this requirement, as it guarantees the existence of a global parametric form of level sets. Our experimental results on synthetic and benchmark data, as well as for a real-world application in computational chemistry, demonstrate that such connected level sets are indeed learned and can be efficiently parameterised. This allows a practical interpolation between data points on a level set. As an additional feature, our model is also able to detect the absence of connected level sets in the input data. We further demonstrate that our network can be easily integrated into a generative framework. This enables characterising invariances for informed data exploration, providing a highly powerful tool for exploratory data analysis. On a chemical example, we show complex topological transformations of molecular structures on the level set of a fixed chemical property. The limitation of our method to single stationary points can be addressed by learning multiple convex functions on sub-domains in future work. Further extensions may include new kinds of bijective mappings and likelihood-based training with Normalising Flows.
References

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *International Conference on Machine Learning*, pp. 146–155. PMLR, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Ben-Israel, A. and Mond, B. What is invexity? *The ANZIAM Journal*, 28(1):1–9, 1986.

Blumenson, L. A derivation of n-dimensional spherical coordinates. *The American Mathematical Monthly*, 67(1): 63–66, 1960.

Bouchacourt, D., Tomioka, R., and Nowozin, S. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Creswell, A., Mohamied, Y., Sengupta, B., and Bharath, A. A. Adversarial information factorization. *arXiv preprint arXiv:1711.05175*, 2017.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889. PMLR, 2015.

Giorgi, G. A note on the relationships between convexity and invexity. *The ANZIAM Journal*, 32(1):97–99, 1990.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Horst, R. On the convexification of nonlinear programming problems: An applications-oriented survey. *European Journal of Operational Research*, 15(3):382–392, 1984.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Jha, A. H., Anand, S., Singh, M., and Veeravarapu, V. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 805–820, 2018.

Keller, S. M., Murezzan, D., and Roth, V. Invexity preserving transformations for projection free optimization with sparsity inducing non-convex constraints. In *German Conference on Pattern Recognition*, pp. 682–697. Springer, 2018.

Keller, S. M., Samarin, M., Arend-Torres, F., Wieser, M., and Roth, V. Learning extremal representations with deep archetypal analysis. *International Journal of Computer Vision*, 129(4):805–820, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Klys, J., Snell, J., and Zemel, R. Learning latent subspaces in variational autoencoders. *arXiv preprint arXiv:1812.06190*, 2018.

Kobyzev, I., Prince, S., and Brubaker, M. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., and Ranzato, M. Fader networks: Manipulating images by sliding attributes. *arXiv preprint arXiv:1706.00409*, 2017.

Mishra, S. K. and Giorgi, G. *Invexity and optimization*, volume 88. Springer Science & Business Media, 2008.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Nesterov, V., Wieser, M., and Roth, V. 3dmolnet: a generative network for molecular structures. *arXiv preprint arXiv:2010.06477*, 2020.

Osher, S. and Fedkiw, R. *Level set methods and dynamic implicit surfaces*, volume 1. Springer New York, 2003.

O’Boyle, N. M. Towards a universal smiles representation—a standard method to generate canonical smiles based on the inchi. *Journal of cheminformatics*, 4(1):22, 2012.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.
Petzka, H., Fischer, A., and Lukovnicov, D. On the regularization of wasserstein gans. *arXiv preprint arXiv:1709.08894*, 2017.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, pp. 2. Citeseer, 2014.

Rueda, N. G. Generalized convexity in nonlinear programming. *Journal of Information and Optimization Sciences*, 10(2):395–400, 1989.

Samarin, M., Nesterov, V., Wieser, M., Wieczorek, A., Parbhoo, S., and Roth, V. Learning conditional invariance through cycle consistency. In *DAGM German Conference on Pattern Recognition*, pp. 376–391. Springer, 2021.

Sapkota, S. and Bhattarai, B. Input invex neural network. *arXiv preprint arXiv:2106.08748*, 2021.

Sethian, J. A. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

Tanaka, Y., Fukushima, M., and Ibaraki, T. On generalized pseudoconvex functions. *Journal of Mathematical analysis and applications*, 144(2):342–355, 1989.

Telgarsky, M. and Singer, Y. A primal-dual convergence analysis of boosting. *Journal of Machine Learning Research*, 13(3), 2012.

Wieser, M., Parbhoo, S., Wieczorek, A., and Roth, V. Inverse learning of symmetry transformations. *Advances in Neural Information Processing Systems*, 2020.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.
A. Appendix

A.1. Proof of Strict Input Convexity

Figure 7. Illustration of the (strictly) Input Convex Neural Network. The figure is adapted from Amos et al. (2017).

We use the following notation:

- Input $z \in \mathbb{R}^d$ with $d$ input dimensions, which in the main text is given by the bijective mapping $z = h(x)$
- Hidden activations $c_i \in \mathbb{R}^{l_i}$, with $l_i$ being the width of the $i$-th layer and $c_k$ being the output $y$, i.e. $c_k = y$
- $k$ layers, $i \in \{0, 1, ..., k-1\}$
- Weights $W_i^{(c)} \in \mathbb{R}^{l_i+1 \times l_i}$, only defined for $i > 0$ or $W_0^{(c)} = 0$
- Input injection weights $W_i^{(z)} \in \mathbb{R}^{l_i+1 \times d}$ and bias terms $b_i \in \mathbb{R}^{l_i}$
- Element-wise activation function $\sigma = (\sigma_1, ..., \sigma_{l_{i+1}})$, i.e. $\sigma_j : \mathbb{R} \rightarrow \mathbb{R}$
- Parameters $\tau = \{W_{0:k-1}^{(z)}, W_{1:k-1}^{(c)}, b_{0:k-1}\}$

Let us restate the theorem in this notation:

**Theorem 3.4 (revised).** The function $f(z; \tau) = y$ recursively defined by

$$c_{i+1} = \sigma \left( W_i^{(c)} c_i + W_i^{(z)} z + b_i \right)$$

is strictly convex in input $z$ if all weights $W_i^{(c)}$ are non-negative (and no rows of $W_i^{(c,z)}$ are all zeroes), and the activation function $\sigma$ is a strictly convex and increasing function.

**Remark.** We need to show four parts, where all statements are w.r.t. input $z$:

1. Non-negative sums of strictly convex functions remain strictly convex;
2. Sums of strictly convex functions and affine functions preserve strict convexity;
3. Composition of a strictly convex and increasing function with a strictly convex function remains strictly convex;
4. Composition of a strictly convex function with an affine function is strictly convex (first layer).

**Proof.** The proofs of these statements follow from extension of standard properties of convex functions to strict convexity (Boyd & Vandenberghe, 2004). As the activation function operates element-wise, we can restrict ourselves to the case where the resulting function maps to the reals, $f : \mathbb{R}^d \rightarrow \mathbb{R}$. Therefore, let $a, b \in \mathbb{R}_{>0}$, $\lambda \in [0,1]$ and $x, y \in \mathbb{R}^d$, $x \neq y$, in the following.

**Part 1:** Let $f = ag + bh$, with $g : \mathbb{R}^d \rightarrow \mathbb{R}$ and $h : \mathbb{R}^d \rightarrow \mathbb{R}$ being strictly convex functions, i.e.

$$g(\lambda x + (1 - \lambda)y) < \lambda g(x) + (1 - \lambda)g(y),$$

with the same holding true for $h$. For $f$ we obtain

$$f(\lambda x + (1 - \lambda)y) = ag(\lambda x + (1 - \lambda)y)$$
$$+ bh(\lambda x + (1 - \lambda)y)$$
$$< \lambda ag(x) + (1 - \lambda)ag(y)$$
$$+ \lambda bh(x) + (1 - \lambda)bh(y)$$
$$= \lambda (ag(x) + bh(x))$$
$$+ (1 - \lambda)(ag(y) + bh(y))$$

which proves that $f$ is strictly convex.

**Part 2:** Let $f = ag + h$, with $g : \mathbb{R}^d \rightarrow \mathbb{R}$ being strictly convex functions and $h : \mathbb{R}^d \rightarrow \mathbb{R}$ being an affine function, i.e.

$$h(x) = Cx + d$$

with $C \in \mathbb{R}^{l \times d}$ and $d \in \mathbb{R}$. For $f$ we obtain

$$f(\lambda x + (1 - \lambda)y) = ag(\lambda x + (1 - \lambda)y)$$
$$+ h(\lambda x + (1 - \lambda)y)$$
$$< \lambda ag(x) + (1 - \lambda)ag(y)$$
$$+ C(\lambda x + (1 - \lambda)y) + d$$
$$= \lambda ag(x) + (1 - \lambda)ag(y)$$
$$+ \lambda Cx + (1 - \lambda)Cy$$
$$+ \lambda d + (1 - \lambda)d$$
$$= \lambda (ag(x) + Cx + d)$$
$$+ (1 - \lambda)(ag(y) + Cy + d)$$
$$= \lambda f(x) + (1 - \lambda)f(y)$$

which proves that $f$ is strictly convex.
Part 3: Let \( f = h \circ g \), with \( g : \mathbb{R}^d \to \mathbb{R} \) and \( h : \mathbb{R} \to \mathbb{R} \) (the activation function) being strictly convex functions. As before, we have

\[
g(\lambda x + (1 - \lambda)y) < \lambda g(x) + (1 - \lambda)g(y). \tag{21}
\]

for both \( g \) and \( h \). We can show that

\[
f(\lambda x + (1 - \lambda)y) = h(g(\lambda x + (1 - \lambda)y)) \tag{22}
\]

\[
< h(\lambda g(x) + (1 - \lambda)g(y)) \tag{23}
\]

\[
< \lambda h(g(x)) + (1 - \lambda)h(g(y)) \tag{24}
\]

\[
= \lambda f(x) + (1 - \lambda)f(y) \tag{25}
\]

which again provides that \( f \) is strictly convex.

Part 4: For the first layer,

\[
z_1 = \sigma \left( W_0^{(x)} x + b_0 \right), \tag{26}
\]

we need to show that \( f = h \circ g \) is strictly convex with \( h : \mathbb{R} \to \mathbb{R} \) (the activation function) being strictly convex, and \( g : \mathbb{R}^d \to \mathbb{R} \) being the affine function \( g(x) = Cx + d \) with \( C \in \mathbb{R}^{1 \times d} \) and \( d \in \mathbb{R} \) as before. Furthermore, we require that \( g \) is not just constant, i.e. \( C \neq (0, \ldots, 0) \) (no rows are all zeroes). For an affine function we can show that

\[
g(\lambda x + (1 - \lambda)y) = C(\lambda x + (1 - \lambda)y) + d \tag{27}
\]

\[
= \lambda C x + (1 - \lambda)y + \lambda d + (1 - \lambda)d \tag{28}
\]

\[
= \lambda(Cx + d) + (1 - \lambda)(Cy + d) \tag{29}
\]

\[
= \lambda g(x) + (1 - \lambda)g(y), \tag{30}
\]

i.e. the affine function is both convex and concave. Therefore,

\[
f(\lambda x + (1 - \lambda)y) = h(g(\lambda x + (1 - \lambda)y)) \tag{31}
\]

\[
= h(\lambda g(x) + (1 - \lambda)g(y)) \tag{32}
\]

\[
< \lambda h(g(x)) + (1 - \lambda)h(g(y)) \tag{33}
\]

\[
= \lambda f(x) + (1 - \lambda)f(y), \tag{34}
\]

i.e. \( f \) is strictly convex.

\[\square\]

A.2. Parameterisation of Level Sets

Let \( X \in \mathbb{R}^n \) be cartesian coordinates in a \( n \)-dimensional Euclidean space that are mapped to a (hyper-)spherical coordinate system \( \{ r, \varphi_1, \varphi_2, \ldots, \varphi_{n-1} \} \) with

\[
x_i \in \mathbb{R} \quad \forall i \in \{1, 2, \ldots, n\} \tag{35}
\]

\[
r \in \mathbb{R}_+ \tag{36}
\]

\[
\varphi_i \in [0, \pi] \quad \forall i \in \{1, 2, \ldots, n-2\} \tag{37}
\]

\[
\varphi_{n-1} \in [0, 2\pi) \tag{38}
\]

Then the mapping to and from a spherical coordinate system for a \( n \)-dimensional space is defined as follows (Blumenson, 1960):

\[
x_1 = r \cos(\varphi_1) \tag{39}
\]

\[
x_2 = r \sin(\varphi_1) \cos(\varphi_2) \tag{40}
\]

\[
\vdots
\]

\[
x_{n-1} = r \sin(\varphi_1) \cdots \sin(\varphi_{n-2}) \cos(\varphi_{n-1}) \tag{42}
\]

\[
x_n = r \sin(\varphi_1) \cdots \sin(\varphi_{n-2}) \sin(\varphi_{n-1}) \tag{43}
\]

\[
\varphi_1 = \arccos \left( \frac{x_1}{\sqrt{x_2^2 + x_3^2 + \cdots + x_1^2}} \right) \tag{44}
\]

\[
\varphi_2 = \arccos \left( \frac{x_2}{\sqrt{x_3^2 + x_4^2 + \cdots + x_2^2}} \right) \tag{45}
\]

\[
\vdots
\]

\[
\varphi_{n-2} = \arccos \left( \frac{x_{n-2}}{\sqrt{x_{n-1}^2 + x_{n-2}^2}} \right) \tag{47}
\]

\[
\varphi_{n-1} = \frac{2 \arccot \left( \frac{x_{n-1} + \sqrt{x_{n-1}^2 + x_{n-2}^2}}{x_n} \right)} \tag{48}
\]

A.3. Synthetic Experiments

A.3.1. Architecture and Training

The baseline model is a \( \beta \)-VAE with approximated bijective mappings through cycle-consistency. For the encoder and decoder, we use two fully-connected hidden layers with 1024 neurons each. The property decoder consists of two fully-connected hidden layers with 512 neurons each. Each of the hidden layers use a ReLU activation. The number of latent dimensions is set to \( d_z = 2 \). The model further involves cycle-consistency regularisation as described in Sec. 4.1, to approximate bijective mappings.

The generalised input-convex model is based on the \( \beta \)-VAE. The encoder is by construction bijective and consists of four autoregressive flow layers with 128 neurons each and an ELU activation. Since the encoder is directly invertible, a separate decoder network is not included. The property decoder is a strictly input-convex neural network (see Sec. 3.4). It consists of four hidden layers with 512 units each and softplus as a strictly convex activation.

The generalised input-convex model with an approximated bijective mapping is based on the \( \beta \)-VAE. For the encoder and decoder, we use two fully-connected hidden layers with 1024 neurons each. The property decoder consists of two fully-connected hidden layers with 512 neurons each. Each of the hidden layers use a ReLU activation function. The property
decoder is a strictly input-convex neural network (see Sec. 3.4). It consists of four hidden layers with 512 units each and softplus as a strictly convex activation.

All of the models are trained with an Adam optimiser, a learning rate of $10^{-4}$, and a batch size of 250. The $\beta$ parameter is annealed by a factor of 0.99 after every 30 epochs and the $\gamma$ constant is set to 0.01.

A.4. Rotated Fashion-MNIST: Architecture and Training

For the rotated Fashion-MNIST experiment we use the following general architecture and training settings. For the full details of the implementation we refer to the code provided.

Encoder. The encoder consists of an initial $7 \times 7$ convolution layer, followed by six $3 \times 3$ convolution layers increasing number of filters. For the down-sampling, three of the $2 \times 2$ convolutions have a stride of 2. In-between the non-striding convolutions, Batch Normalisation (Ioffe & Szegedy, 2015) was used. The flattened output is fed into a fully connected network that parameterises mean and standard deviation of the latent distribution $q(z|x)$.

Image Decoder. The image decoder consists of 3 transposed convolutions with a kernel size of $3 \times 3$ and a stride of 2. Each transposed convolution is followed by $3 \times 3$ convolutions with Batch Normalisation in between. The image likelihood, i.e. the reconstruction loss, is Gaussian with mean given by the decoder network and the covariance as a $\sigma I$, with $\sigma$ being a learnable parameter, $I$ the identity matrix.

Property Decoder. The property decoder is a 5 layered fully input-convex neural network followed by a hyperbolic tangent activation. The output is then scaled to the range from $-\pi/4$ to $\pi/4$ (i.e. the range of possible rotations). As activation function we use softplus.

Training. We train for 30 epochs with Adamax (Kingma & Ba, 2014), using a mini-batch size of 32 and a latent space dimension of 35. Weights of the two reconstruction losses, as well as the cycle-consistency loss were determined experimentally. For the weight of the KL-Divergence, we started with an initial value of 1 and slowly decreased it using an exponential cooling function, multiplying the weight by 0.95 each 20 mini-batches.

A.5. Chemical Experiments

The QM9 dataset (Ramakrishnan et al., 2014) includes 133,885 organic molecules. The molecules consist of up to nine heavy atoms (C, O, N, and F), not including hydrogen. Each molecule includes corresponding geometric, energetic, electronic and thermodynamic properties chemical properties. These properties have been computed with the Density Functional Theory methods. In our experiments, we select a subset with a fixed stoichiometry ($C_7O_2$) which consists of 16,302 molecules. As the chemical property, we choose the band gap energy.

A.5.1. Architecture and Training

For the encoder and decoder we use two dense layers including 1024 units with an ELU activation. The property decoder is a fully input-convex neural network. It consists of four hidden layers with 512 units each and a softplus activation. Each of the networks has an additional output layer with a linear activation. The $\beta$ parameter is annealed by a factor of 0.99 after every 30 epochs and the $\gamma$ constant is set to 0.01.