Analyzing Inverse Problems with Invertible Neural Networks

Lynton Ardizzone[1], **Jakob Kruse**[1], **Sebastian Wirkert**[2], **Daniel Rahner**[3], **Eric W. Pellegrini**[3], **Ralf S. Klessen**[3], **Lena Maier-Hein**[2], **Carsten Rother**[1], **Ullrich Köthe**[1]
[1]Visual Learning Lab Heidelberg, [2]German Cancer Research Center (DKFZ),
[3]Zentrum für Astronomie der Universität Heidelberg (ZAH)
[1]lynton.ardizzone@iwr.uni-heidelberg.de,
[2]s.wirkert@dkfz-heidelberg.de, [3]daniel.rahner@uni-heidelberg.de

Abstract

In many tasks, in particular in natural science, the goal is to determine hidden system parameters from a set of measurements. Often, the forward process from parameter- to measurement-space is a well-defined function, whereas the inverse problem is ambiguous: one measurement may map to multiple different sets of parameters. In this setting, the posterior parameter *distribution*, conditioned on an input measurement, has to be determined. We argue that a particular class of neural networks is well suited for this task – so-called Invertible Neural Networks (INNs). Although INNs are not new, they have, so far, received little attention in literature. While classical neural networks attempt to solve the ambiguous inverse problem directly, INNs are able to learn it jointly with the well-defined forward process, using additional latent output variables to capture the information otherwise lost. Given a specific measurement and sampled latent variables, the inverse pass of the INN provides a full distribution over parameter space. We verify experimentally, on artificial data and real-world problems from astrophysics and medicine, that INNs are a powerful analysis tool to find multi-modalities in parameter space, to uncover parameter correlations, and to identify unrecoverable parameters.

1 Introduction

When analyzing complex physical systems, a common problem is that hidden system parameters, which scientists would like to know, cannot be measured directly. For many of these problems, scientists have developed sophisticated theories on how measurable quantities arise from the hidden parameters. We will call such mappings the *forward process*. However, the *inverse* process is required to infer the hidden states of a system from measurements. Unfortunately, the inverse is often both intractable and ill-posed, since crucial information is lost in the forward process.

Because pairs of hidden parameters $\mathbf{x}$ and corresponding measurements $\mathbf{y}$ can be produced in great quantity via numerical simulation, it seems attractive to use machine learning, and especially neural networks, to find an approximation of the inverse process. But training a neural network directly for this task is of limited use because of the inherent ambiguity of $\mathbf{y} \rightarrow \mathbf{x}$. A predictor trained to return a point estimate of the hidden parameters for a given measurement will at best identify the most likely solution; and at worst take the average of multiple solutions with incompatible characteristics. There are network architectures which circumvent this problem by modeling the uncertainty of a solution, or predicting a distribution over possible outputs. However, all of these approaches face a hen-and-egg problem: formulating a supervised, discriminative loss over $\mathbf{x}$ (cf. Fig. 1) requires knowledge about the distribution of $\mathbf{x}$, which is precisely the distribution we wish to learn.

We propose to instead learn the well-defined forward process with an *invertible* neural network (INN) which provides the inverse for free. INNs are characterized by three properties: (i) The mapping

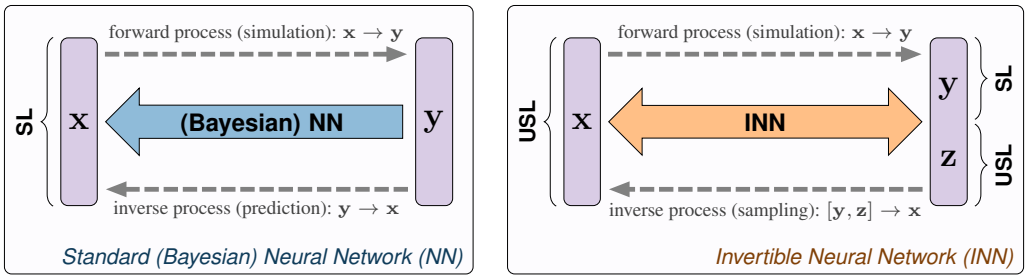


Figure 1: **Abstract comparison of standard discriminative approach *(left)* and our approach *(right)*.** The standard direct approach requires a discriminative, supervised loss (**SL**) term between predicted and true $\mathbf{x}$, causing problems when $\mathbf{y} \to \mathbf{x}$ is ambiguous. Our network uses a supervised loss only for the well-defined forward process $\mathbf{x} \to \mathbf{y}$. Generated $\mathbf{x}$ are required to follow the prior $p(\mathbf{x})$ by an unsupervised loss (**USL**), while the latent variables $\mathbf{z}$ are made to follow a Gaussian distribution, also by an unsupervised loss. See details in Section 3.3.

from inputs to outputs is bijective, i.e. its inverse exists, (ii) both forward and inverse mapping are efficiently computable, and (iii) the mappings have tractable Jacobians, so that probabilities can be transformed explicitly via the change-of-variables formula.

To counteract the inherent information loss of the forward process, we introduce additional *latent* output variables $\mathbf{z}$, which are trained to capture the information about $\mathbf{x}$ that is *not* contained in $\mathbf{y}$. Thus, our INN maps pairs $[\mathbf{y}, \mathbf{z}]$ of measurements and latent variables back to unique hidden parameter values $\mathbf{x}$. In addition, we train the network to shape the density $p(\mathbf{z})$ of the latent variables according to a Gaussian distribution. In other words, the posterior $p(\mathbf{x}|\mathbf{y})$ is reparametrized into a deterministic function $\mathbf{x} = g(\mathbf{y}, \mathbf{z})$ that transforms the known distribution $p(\mathbf{z})$ to $\mathbf{x}$-space, conditional on $\mathbf{y}$. In practice, we fix $\mathbf{y}$ at an actual measurement and sample $\mathbf{z}$ repeatedly from $p(\mathbf{z})$, so that the the corresponding inverses form a sample of the desired posterior $p(\mathbf{x}|\mathbf{y})$.

Our main contributions are as follows:

- We are the first to recognize and exploit the potential of Invertible Neural Networks (INNs) as conditional generative models.
- We present a new INN architecture with a bi-directional training scheme, easily constructible from standard modules, which enables joint modeling of the forward and inverse processes.
- We are the first to apply INNs to real-world applications, specifically from astrophysics and medicine. In contrast to alternative approaches, our INN is able to find multi-modalities in parameter space, uncover parameter correlations, and identify unrecoverable parameters.

2 Related work

Modeling the conditional posterior of an inverse process is a classical statistical task that can in principle be solved by Bayesian methods. Unfortunately, exact Bayesian treatment of real-world problems is usually intractable. The most common (but expensive) solution is to resort to sampling, typically by a variant of Markov Chain Monte Carlo [36, 11]. If a model $\mathbf{y} = s(\mathbf{x})$ for the forward process is available, approximate Bayesian computation is often preferred, which embeds the forward model in a rejection sampling scheme for the posterior $p(\mathbf{x}|\mathbf{y})$ [40, 26, 46].

Variational methods offer a more efficient alternative, approximating the posterior by an optimally chosen member of a tractable distribution family [3]. Neural networks can be trained to predict accurate sufficient statistics for parametric posteriors [30, 39], or can be designed to learn a mean-field distribution for the network's weights via dropout variational inference [10, 24]. Both ideas can be combined [20] to differentiate between data-related and model-related uncertainty. However, the restriction to limited distribution families fails if the true distribution is too complex (e.g. when it requires multiple modes to represent ambiguous or degenerate solutions) and essentially counters the ability of neural networks to act as *universal* approximators. Conditional GANs (cGAN) [28, 18] overcome this restriction in principle, but are known to have difficulties achieving satisfactory diversity in practice [50].

Normalizing flows, introduced to the field of neural networks by [35, 34], have the potential to solve these problems. They gradually transform normal densities into target densities and rely on bijectivity to guarantee the validity of these mappings. Thus, they possess property (i) of an INN, and are usually designed to fulfill requirement (iii) as well. Besides variants of the original architectures [43, 2, 44], the most common realizations use *auto-regressive flows*, where the density is decomposed according to the Bayesian chain rule [23, 17, 12, 31, 29, 25, 38, 45]. These networks successfully learned unconditional generative distributions for artificial data and standard image sets (e.g. MNIST, CelebA, LSUN bedrooms), and some encouraging results for conditional modeling exist as well [29, 38, 31, 45].

Although flow-based networks are invertible in principle, actual computation of their inverse is too costly to be practical, i.e. property (ii) is not fulfilled. This precludes the possibility of bi-directional or cyclic training, which has been shown to be very beneficial in generative adversarial nets and auto-encoders [49, 9, 8, 41]. In fact, optimization for cycle consistency forces these models to converge to invertible architectures, making fully invertible networks a natural choice. True INNs can be built using *coupling layers*, as introduced in the NICE [6] and RealNVP [7] architectures. Despite their simple design and training, these networks were rarely studied: [5, 15] showed that minimization of an adversarial loss is superior to maximum likelihood training in RealNVPs, [13] used a NICE-like design as a memory-efficient alternative to residual networks, [19] demonstrated that the lack of information reduction from input to representation does not cause overfitting, and [22] extended the RealNVP architecture by introducing invertible 1x1 convolutions. This line of research inspired us to extend RealNVPs into conditional generative models, applicable to real-world inverse problems from natural and life sciences.

3 Methods

3.1 Problem specification

We consider a common scenario in natural and life sciences: Researchers are interested in a set of variables $\mathbf{x} \in \mathbb{R}^D$ describing some phenomenon of interest, but only variables $\mathbf{y} \in \mathbb{R}^M$ can actually be observed, for which the theory of the respective field provides a model $\mathbf{y} = s(\mathbf{x})$ for the forward process. Since the transformation from $\mathbf{x}$ to $\mathbf{y}$ incurs an information loss, the intrinsic dimension m of $\mathbf{y}$ is in general smaller than D, even if the nominal dimensions satisfy $M > D$. Hence we want to express the inverse model as a conditional probability $p(\mathbf{x}|\mathbf{y})$, but its derivation from the forward model is intractable in the applications we are going to address.

We aim at approximating $p(\mathbf{x}|\mathbf{y})$ by a tractable model $q(\mathbf{x}|\mathbf{y})$, taking advantage of the possibility to create an arbitrary amount of training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ from the known forward model $s(\mathbf{x})$ and a suitable prior $p(\mathbf{x})$. While this would allow for training of a standard regression model, we want to approximate the full posterior probability. To this end, we introduce a latent random variable $\mathbf{z} \in \mathbb{R}^K$ drawn from a multi-variate standard normal distribution and reparametrize $q(\mathbf{x}|\mathbf{y})$ in terms of a deterministic function g of $\mathbf{y}$ and $\mathbf{z}$, represented by a neural network with parameters θ:

$$\mathbf{x} = g(\mathbf{y}, \mathbf{z}; \theta) \quad \text{with} \quad \mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I_K). \tag{1}$$

Note that we distinguish between *hidden parameters* $\mathbf{x}$ representing unobservable real-world properties and *latent variables* $\mathbf{z}$ carrying information intrinsic to our model.

In contrast to standard methodology, we propose to learn the model $g(\mathbf{y}, \mathbf{z}; \theta)$ of the inverse process *jointly* with a model $f(\mathbf{x}; \theta)$ approximating the known forward process $s(\mathbf{x})$:

$$[\mathbf{y}, \mathbf{z}] = f(\mathbf{x}; \theta) = [f_{\mathbf{y}}(\mathbf{x}; \theta), f_{\mathbf{z}}(\mathbf{x}; \theta)] = g^{-1}(\mathbf{x}; \theta) \quad \text{with} \quad f_{\mathbf{y}}(\mathbf{x}; \theta) \approx s(\mathbf{x}). \tag{2}$$

Functions f and g share the same parameters θ and are implemented by a *single invertible neural network*. Our experiments show that joint *bi-directional training* of f and g avoids many complications arising in e.g. cGANs or Bayesian neural networks, which only learn $g(\mathbf{y}, \mathbf{z})$ or $q(\mathbf{x}|\mathbf{y})$ alone.

The relation $f = g^{-1}$ is enforced by the invertible network architecture, provided that the nominal and intrinsic dimensions of both sides match. When $m \leq M$ denotes the intrinsic dimension of $\mathbf{y}$, the latent variable $\mathbf{z}$ must have dimension $K = D - m$, assuming that the intrinsic dimension of $\mathbf{x}$ equals its nominal dimension D. If the resulting nominal output dimension $M + K$ exceeds D, we augment the input with a vector $\mathbf{x}_0 \in \mathbb{R}^{M+K-D}$ of zeros and replace $\mathbf{x}$ with the concatenation

$[\mathbf{x}, \mathbf{x}_0]$ everywhere. Combining these definitions, our network expresses $q(\mathbf{x}|\mathbf{y})$ as

$$q\big(\mathbf{x} = g(\mathbf{y}, \mathbf{z}; \theta) \,|\, \mathbf{y}\big) = p(\mathbf{z}) \left|J_{\mathbf{x}}\right|^{-1} \tag{3}$$

$$\text{with Jacobian determinant } J_{\mathbf{x}} = \det\left(\left.\frac{\partial g(\mathbf{y}, \mathbf{z}; \theta)}{\partial [\mathbf{y}, \mathbf{z}]}\right|_{\mathbf{y}, f_{\mathbf{z}}(\mathbf{x})} \right). \tag{4}$$

Using coupling layers according to [7], computation of $J_{\mathbf{x}}$ is simple, as each transformation has a triangular Jacobian matrix.

3.2 Invertible architecture

To create a fully invertible neural network, we follow the architecture proposed by Dinh et al. [7]: The basic unit of this network is a reversible block consisting of two complementary affine coupling layers. Hereby, the block's input vector $\mathbf{u}$ is split into two halves, $\mathbf{u}_1$ and $\mathbf{u}_2$, which are transformed by an affine function with coefficients $\exp(s_i)$ and t_i ($i \in \{1, 2\}$), using element-wise multiplication ($\odot$) and addition:

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp(s_2(\mathbf{u}_2)) + t_2(\mathbf{u}_2), \qquad \mathbf{v}_2 = \mathbf{u}_2 \odot \exp(s_1(\mathbf{v}_1)) + t_1(\mathbf{v}_1). \tag{5}$$

Given the output $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2]$, these expressions are trivially invertible:

$$\mathbf{u}_2 = (\mathbf{v}_2 - t_1(\mathbf{v}_1)) \odot \exp(-s_1(\mathbf{v}_1)), \qquad \mathbf{u}_1 = (\mathbf{v}_1 - t_2(\mathbf{u}_2)) \odot \exp(-s_2(\mathbf{u}_2)). \tag{6}$$

Importantly, the mappings s_i and t_i can be arbitrarily complicated functions of $\mathbf{v}_1$ and $\mathbf{u}_2$ and need *not* themselves be invertible. In our implementation, they are realized by a succession of several fully connected layers with leaky ReLU activations.

A deep invertible network is composed of a sequence of these reversible blocks. To increase model capacity, we apply a few simple extensions to this basic architecture. Firstly, if the dimension D is relatively small, but a complex transformation has to be learned, we find it advantageous to pad both the in- and output of the network with an equal number of zeros. This does not change the intrinsic dimensions of in- and output, but enables the network's interior layers to embed the data into a larger representation space in a more flexible manner. Secondly, we insert permutation layers between reversible blocks, which shuffle the elements of the subsequent layer's input in a randomized, but fixed, way. This causes the splits $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2]$ to vary between layers and enhances interaction among the individual variables.

3.3 Bi-directional training

Invertible networks offer the opportunity to simultaneously optimize for losses on both the in- and output domains to make training more effective. Hereby, we perform forward and backward iterations in an alternating fashion, accumulating gradients from both directions before performing a parameter update. For the forward iteration, we penalize deviations between simulation outcomes $\mathbf{y}_i = s(\mathbf{x}_i)$ and network predictions $f_{\mathbf{y}}(\mathbf{x}_i)$ with a loss $\mathcal{L}_{\mathbf{y}}\big(\mathbf{y}_i, f_{\mathbf{y}}(\mathbf{x}_i)\big)$. Depending on the problem, $\mathcal{L}_{\mathbf{y}}$ can be any supervised loss, e.g. squared loss for regression or cross-entropy for classification.

The loss for latent variables penalizes the mismatch between the joint distribution of network outputs $q\big(\mathbf{y} = f_{\mathbf{y}}(\mathbf{x}), \mathbf{z} = f_{\mathbf{z}}(\mathbf{x})\big) = p(\mathbf{x})/|J_{\mathbf{yz}}|$ and the product of marginal distributions of simulation outcomes $p\big(\mathbf{y} = s(\mathbf{x})\big) = p(\mathbf{x})/|J_s|$ and latents $p(\mathbf{z})$ as $\mathcal{L}_{\mathbf{z}}\big(p(\mathbf{y})\, p(\mathbf{z}), q(\mathbf{y}, \mathbf{z})\big)$.

We block the gradients of $\mathcal{L}_{\mathbf{z}}$ with respect to $\mathbf{y}$ to make sure that the resulting updates only affect the predictions of $\mathbf{z}$ and do not worsen the predictions of $\mathbf{y}$. Thus, $\mathcal{L}_{\mathbf{z}}$ enforces two things: firstly, the generated $\mathbf{z}$ must follow the desired normal distribution $p(\mathbf{z})$; secondly, $\mathbf{y}$ and $\mathbf{z}$ must be independent, and not encode the same information twice. Since $\mathcal{L}_{\mathbf{z}}$ is implemented in terms of Maximum Mean Discrepancy (see Sec. 3.4), which only requires samples from the distributions to be compared, the Jacobian determinants $J_{\mathbf{yz}}$ and J_s do not have to be known explicitly.

Although $\mathcal{L}_{\mathbf{y}}$ and $\mathcal{L}_{\mathbf{z}}$ should be sufficient in principle, a small amount of residual dependency between $\mathbf{y}$ and $\mathbf{z}$ remains in more complicated problem instances. This causes $q(\mathbf{x}|\mathbf{y})$ to deviate from the true posterior $p(\mathbf{x}|\mathbf{y})$. To prevent this, we also define a loss $\mathcal{L}_{\mathbf{x}}$ on the input side, implemented by MMD as well. It matches the prior data distribution $p(\mathbf{x})$ against the distribution of backward predictions $q(\mathbf{x}) = p\big(\mathbf{y} = f_{\mathbf{y}}(\mathbf{x})\big)\, p\big(\mathbf{z} = f_{\mathbf{z}}(\mathbf{x})\big)/|J_{\mathbf{x}}|$ through $\mathcal{L}_{\mathbf{x}}\big(p(\mathbf{x}), q(\mathbf{x})\big)$.

3.4 Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) is a kernel-based method for comparison of two probability distributions that are only accessible through samples [14]. While a trainable discriminator loss is often preferred for this task in high-dimensional problems, especially in GAN-based image generation, MMD also works well, is easier to use and much cheaper, and leads to more stable training [42]. The method requires a kernel function as a design parameter, and we found that kernels with heavier tails than Gaussian are needed to get meaningful gradients for outliers. We achieved best results with the Inverse Multiquadratic $k(\mathbf{x}, \mathbf{x}') = \alpha/(\alpha + \|\mathbf{x} - \mathbf{x}'\|_2^2)$, reconfirming the suggestion from [42].

4 Experiments

We first demonstrate the capability of our invertible architecture on well-behaved artificial problems and compare its performance to alternative approaches. Then, in Sections 4.2 and 4.3, we give results for two real-world applications, from the fields of medicine and astrophysics. Additional experiments and details on network architectures are provided in the supplementary material.

4.1 Experiments on artificial data

In our first experiment, a Gaussian mixture distribution with 8 labeled modes is arranged on the 2D plane, i.e. $\mathbf{x} \in \mathbb{R}^2$. For a given sample $\mathbf{x}$ from this mixture model, the forward process gives out the label $\mathbf{y}$ of the mode it was drawn from, encoded as an 8-dimensional one-hot vector. We compare three variants of this setting, shown in Fig. 2, where each mode is assigned a unique or shared label.

We train an INN as described in section 3, with the latent ($\mathbf{z}$) dimension set to $K = 2$ and the in- and output zero-padded to 16 dimensions. We use the same hyper-parameters and network architecture for all three setups, consisting of 3 invertible blocks, each built with 3-layer sub-networks for s_i and t_i. Fig. 2 shows that our model is able to recover the full Gaussian mixture distribution for each tested label configuration. The figure also shows results when either the loss on the $\mathbf{x}$-domain or on the $[\mathbf{y}, \mathbf{z}]$-side is switched off. We find that forward training alone ($\mathcal{L}_\mathbf{y}, \mathcal{L}_\mathbf{z}$) captures the conditional relationships, but places too much mass in unpopulated regions of $\mathbf{x}$-space. Conversely, pure inverse training ($\mathcal{L}_\mathbf{x}$) learns the correct $\mathbf{x}$-distribution, but loses all conditioning information. In addition, we show results from a cGAN [28] and a model with dropout sampling and a learned aleatoric error term [20], both trained directly on the inverse task. In an effort to match the expressive power, we fixed the architecture of both models to have approximately the same number of parameters as our INN.

The cGAN is trained to produce the correct sample distribution given a mode label $\mathbf{y}$ and noise $\mathbf{z}$ of dimensionality $K = 2$, with additional pretraining of the generator to avoid mode collapse. While it is able to capture the overall multimodal nature of the solution spaces after fine-tuning, the individual modes collapse to nearly one-dimensional structures, yielding visibly degraded distributions. For results that match those of the INN, the cGAN requires considerably more latent variables or a more complex architecture. To ensure that the differences between cGAN and INN are not merely due to different loss functions, we also train a cGAN-like generator using MMD-loss instead of adverserial loss, which yields significantly better samples than the cGAN, but is still inferior to the INN – see supplementary material. The dropout network only takes labels $\mathbf{y}$ as input and uses dropout sampling followed by a Gaussian pertubation according to the predicted aleatoric uncertainty. Since it cannot make use of meaningful latent space structure, it fails to represent any multimodality not explicitly encoded in the labels $\mathbf{y}$.

To analyze how the latent space is structured for this task, we choose a fixed label $\mathbf{y}$ and sample $\mathbf{z}$ from a dense grid. For each $\mathbf{z}$, we compute $\mathbf{x}$ through our inverse network and colorize this point in latent ($\mathbf{z}$) space according to the distance from the closest mode in $\mathbf{x}$-space. We can see that our network learns to shape the latent space such that each mode receives the expected fraction of samples (Fig. 3). Applying the same visualization to the cGAN's noise variable $\mathbf{z}$ reveals a far less structured layout of the latent space.

We perform a second experiment investigating the behavior of the same methods in higher dimensions, where our INN outperforms the alternatives as well, shown in the supplementary material.

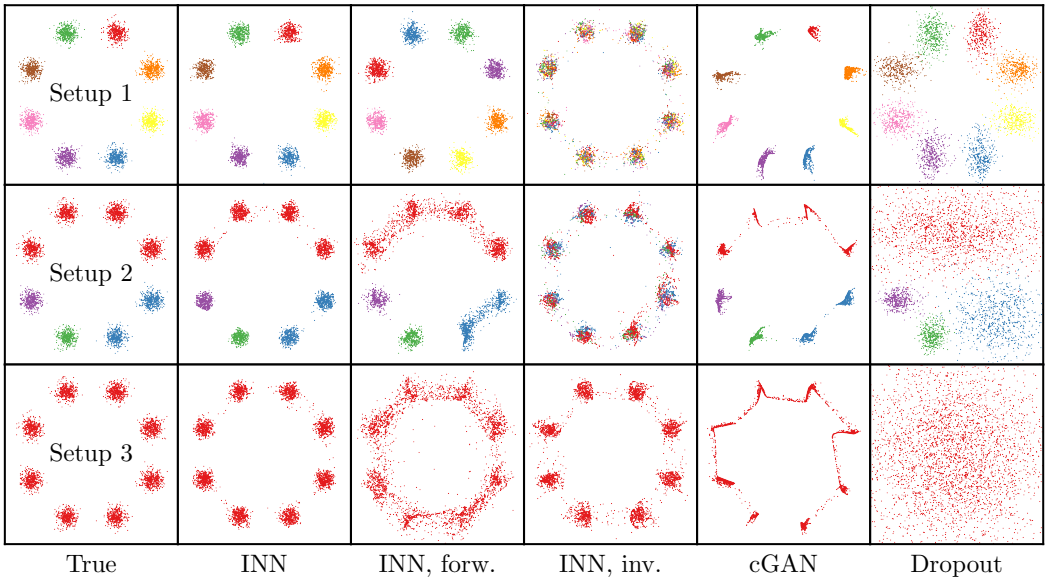


Figure 2: **Distribution of 2D points, conditioned on the label (i.e. color) of one or more modes.** Each row is a different setup, with all modes having separate labels *(Setup 1)*, some modes sharing the same label *(Setup 2)* and all modes having the same label *(Setup 3)*. The first column shows the ground truth, to its right are approximations by each model. Columns 3 and 4 show an ablation study, where the INN was trained using only the losses in the forward direction, and only the loss in the inverse direction.

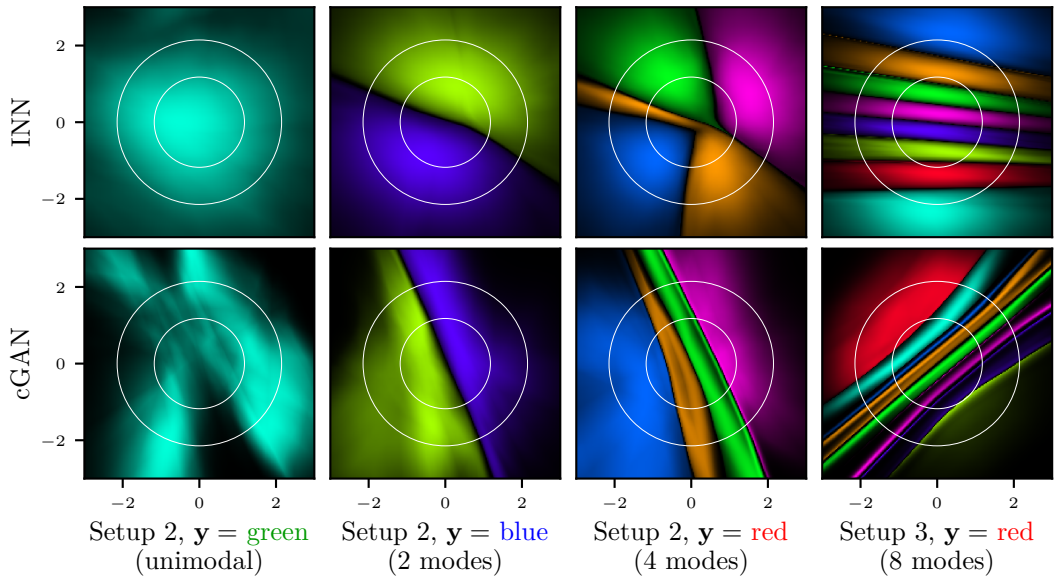


Figure 3: **Layout of latent space for one fixed label $\mathbf{y}$, colored by mode closest to $\mathbf{x} = g(\mathbf{y}, \mathbf{z})$.** For each latent position $\mathbf{z}$, the hue encodes which mode the corresponding $\mathbf{x}$ belongs to and the luminosity encodes how close $\mathbf{x}$ is to this mode. Note that colors used here do not relate to those in Fig. 2, and encode the position $\mathbf{x}$ instead of the label $\mathbf{y}$. The first three columns correspond to labels *green*, *blue* and *red* in Setup 2 of Fig. 2; the last column corresponds to label *red* in Setup 3. White circles mark areas that contain 50% and 90% of the probability mass of latent prior $p(\mathbf{z})$.

4.2 Functional parameter estimation from multispectral tissue images

In medical science, the functional state of biological tissue is of interest for many applications, such as tumor detection or verifying organ transplantation success. Tumors, for example, are expected to exhibit changes in oxygen saturation s_{O_2} [16]. Such changes influence the reflectance of the tissue, which can be measured by multispectral cameras [27]. We can simulate these measurements from a tissue model involving s_{O_2}, blood volume fraction v_{hb}, scattering magnitude a_{mie}, anisotropy g and tissue layer thickness d [47]. While these simulations can determine the reflectance spectrum ($\mathbf{y}$) for a given tissue, inverting the measurements to recover the underlying functional properties ($\mathbf{x}$) is an active field of research. State-of-the-art approaches [47, 48, 4] model the inverse process directly, but cannot quantify the estimate's uncertainty, which can be vital for an accurate diagnosis.

We train an INN for this problem, along with two ablations (only forward or only inverse training), as well as a regular neural net using the method of Kendall and Gal [20], with Monte-Carlo (MC) dropout and additional aleatoric error terms for each parameter. The latter also provides a point-estimate baseline, by disabling dropout and ignoring the aleatoric error. Lastly, we also compare to Approximate Bayesian Computation (ABC), using the method presented in [46], which is summarized in the supplementary material.

Results of our evaluation of these methods are presented in Table 1. We compute the deviation of point estimates $\hat{\mathbf{x}}$ from ground-truth values $\mathbf{x}^*$ in terms of the RMSE over test set observations $\mathbf{y}^*$

$$\text{RMSE}_{\text{Point}} = \sqrt{\mathbb{E}_{\mathbf{y}^*}\left[\|\hat{\mathbf{x}} - \mathbf{x}^*\|^2\right]} \tag{7}$$

and compare INN MAP solutions (maximum of the posterior) with a classical point estimate and INN ablations. For comparison of full posteriors between INN, MC dropout, and ABC, we compute the expectation of the squared error over the estimated posterior:

$$\text{RMSE}_{\text{Posterior}} = \sqrt{\mathbb{E}_{\mathbf{y}^*}\left[\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x}|\mathbf{y}^*)}\left[\|\mathbf{x} - \mathbf{x}^*\|^2\right]\right]} \tag{8}$$

RMSE scores are reported both for the main parameter of interest s_{O_2}, and the parameter subspace of s_{O_2}, v_{hb}, a_{mie}, which we found to be the only recoverable parameter dimensions. Secondly, we compute the calibration curves for the sampling-based methods, i.e. the fraction of ground truth inliers $q_{\text{inl.}}$ for a given confidence interval q. We define the calibration error to be the median of $|q_{\text{inl.}} - q|$ over all q. Lastly, we check the re-simulation error: we apply the simulation $s(\mathbf{x})$ to samples drawn from the posterior for fixed $\mathbf{y}$, and compare the simulation outcomes with the conditioning variable, averaged in the same way as Eqs. 7 and 8. All averages are taken over 5000 observations $\mathbf{y}$, each posterior uses 4096 samples, and the MAP is found using the mean-shift algorithm.

In terms of accuracy, we find that the INN's MAP-solution matches or outperforms the other methods. Disabling the $\mathcal{L}_{\mathbf{x}}$-loss only has a small negative impact on the accuracy. In contrast, if $\mathcal{L}_{\mathbf{y}}, \mathcal{L}_{\mathbf{z}}$ are switched off, the network fails completely, because the missing conditioning information is critical for the task, in line with the experiment in Fig. 2, column 4. For the calibration error, the INN also performs markedly better than the dropout network, due to the non-Gaussian priors visualized in Fig. 4. Despite the noisy posteriors, ABC performs best when the calibration curve is averaged over the entire test set. Regarding the re-simulation error, we find that the INN's MAP-solution reaches the limit of the simulation noise (last row), i.e. it presents a correct solution to the inverse problem while the classical network falls short. By construction, the re-simulation error of ABC is equal to the simulation noise.

Fig. 4 shows generated parameter distributions for one fixed measurement $\mathbf{y}$, comparing the INN, MC dropout, and ABC. All three methods use a sample count of $160\,000$ to produce smooth curves. Due to the sparse posteriors in the case of ABC, kernel density estimation was applied to its results, with a bandwidth of $\sigma = 0.1$. The results produced by the INN provide several new insights: First, we find that the posteriors for layer thickness d and anisotropy g match the shape of their priors, i.e. $\mathbf{y}$ holds no information about these parameters – they are unrecoverable. In contrast, MC dropout gives the illusion of recoverable parameters (albeit with high variance), because uniform posterior distributions are beyond its representational capabilities. Second, we find that the sampled distributions for the blood volume fraction v_{hb} and scattering amplitude a_{mie} are strongly correlated (rightmost plot). This phenomenon is not an analysis artifact, but has a sound physical explanation: As blood volume fraction increases, more light is absorbed inside the tissue. For the sensor to record the same intensities $\mathbf{y}$ as before, scattering must be increased accordingly. Since the MC dropout network employs a mean-field model, it cannot detect effects like this.

Table 1: **Results for the medical application** (see text for details). The simulation noise is determined by repeatedly simulating with the same parameters $\mathbf{x}$. The uncertainties of the median are determined via statistical bootstrapping.

Method	Error s_{O_2}	Error all	Calibration error	Re-simulation error
Point estimates (Eq. 7):				
Classical NN	0.056 ± 0.001	$\mathbf{0.56 \pm 0.01}$		0.95 ± 0.08
INN MAP solution	$\mathbf{0.037 \pm 0.001}$	0.58 ± 0.01		$\mathbf{0.34 \pm 0.03}$
INN MAP, only $\mathcal{L}_\mathbf{y}, \mathcal{L}_\mathbf{z}$	0.062 ± 0.001	0.74 ± 0.02		0.89 ± 0.06
INN MAP, only $\mathcal{L}_\mathbf{x}$	0.250 ± 0.037	1.69 ± 0.01		2.19 ± 0.13
Posteriors (Eq. 8):				
INN	0.094 ± 0.002	0.86 ± 0.01	$(1.5 \pm 0.3)\%$	0.48 ± 0.03
MC dropout [20]	0.172 ± 0.003	1.03 ± 0.01	$(9.3 \pm 1.2)\%$	0.65 ± 0.06
ABC	0.154 ± 0.002	0.90 ± 0.01	$\mathbf{(1.1 \pm 0.1)\%}$	0.34 ± 0.03
Simulation noise				0.34 ± 0.03

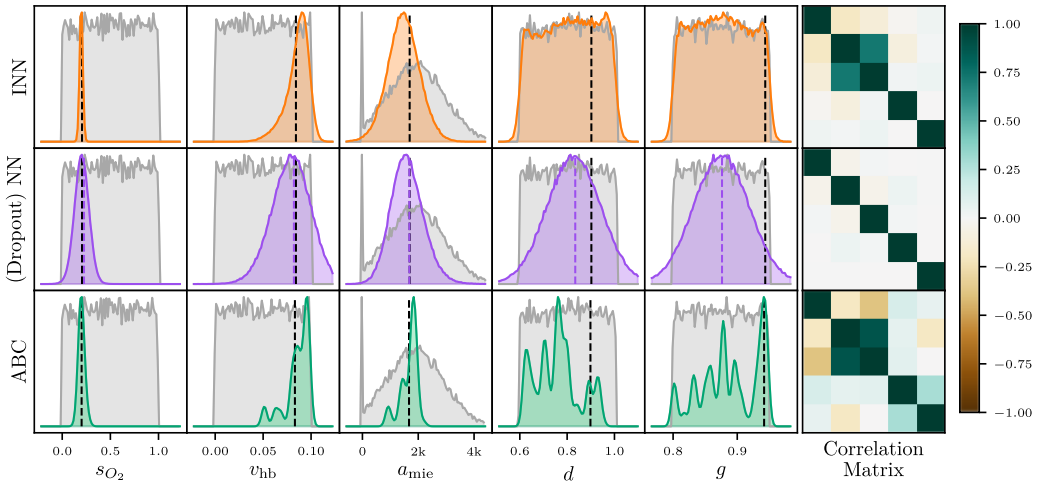
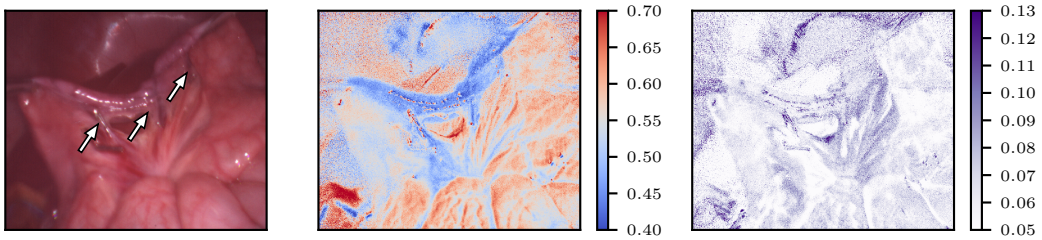


Figure 4: **Sampled distributions of 5 hidden parameters for fixed $\mathbf{y}$ in medical application** For a fixed observation $\mathbf{y}$, the top row shows INN-generated posterior distributions (*orange area*) over $\mathbf{x}$ which map to this $\mathbf{y}$. We compare this to ℓ^2-point estimates $\hat{\mathbf{x}}$ *(dashed purple line)* and their distribution via dropout and uncertainty sampling *(purple area)* in the middle row, as well as the estimate from ABC *(green area)* at the bottom. Ground truth values $\mathbf{x}^*$ *(dashed black line)* and prior of $\mathbf{x}$ over all data *(gray area)* are provided for reference.

The last row shows good agreement between INNs and ABC, but also highlights the main problem with the latter, namely its sampling efficiency. Although the same number of samples has been used for all three methods, the ABC posteriors are much more noisy, due to a low sample acceptance rate of $7.6 \cdot 10^{-5}$, i.e. less than 20 samples survive. Better results require a significantly larger training set, which is prohibitively expensive due to the slow simulation. While speed-ups could be achieved, e.g. by replacing the simulation with a fast (trained) forward model or by using a more efficient sampling technique, we did not pursue this further.

4.3 Impact of star clusters on the dynamical evolution of the galactic gas

Star clusters are born from a large reservoir of gas and dust that permeates the galaxy, the interstellar medium. The process is governed by the complex interplay of competing physical agents such as gravity, turbulence, magnetic fields, and radiation; with stellar feedback playing a decisive regulatory role [37]. To study this, astronomers measure emission lines from chemical elements such as hydrogen or oxygen [1, 21]. Their relative intensities depend on the ionization potential, the spectrum of the ionizing radiation, the gas density, and the absolute intensity of the radiation [32]. We investigate the dynamical feedback of young star clusters on their parental environment in a 1D model [33] which describes the complete temporal evolution of the system and allows us to generate synthetic emission



c) RGB image a) Median s_{O_2} b) Est. uncertainty

Figure 5: **INN applied to real footage to determine oxygenation s_{O_2} and uncertainty estimate.** The clips (*arrows*) on the connecting tissue cause lower oxygenation *(blue)* in the small intestine. Uncertainty is low in crucial areas and high only at some edges and specularities.

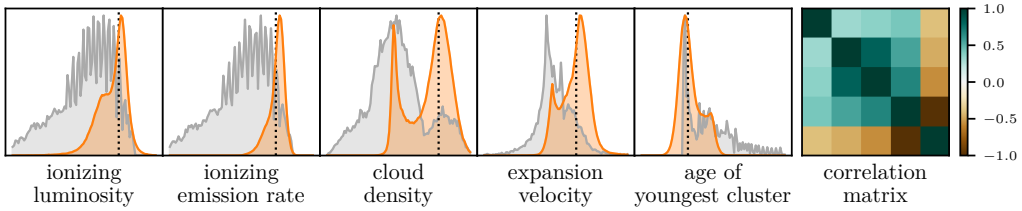


Figure 6: **Sampled distributions of 5 hidden parameters for fixed $\mathbf{y}$ in astrophysics application.** Colors as in Fig. 4, top row. The peculiar shape of the prior is due to the complex temporal evolution of the simulations. Our INN finds multiple modes as well as strong correlations in $p(\mathbf{x}|\mathbf{y})$.

line maps. Similar to the medical application above, the reverse of this mapping is highly degenerate and ill-posed. We train our forward model to predict the observable quantities $\mathbf{y}$ (e.g. emission line ratios) from composite simulation outputs $\mathbf{x}$ (e.g. ionizing luminosity and emission rate, cloud density, expansion velocity, and age of the youngest cluster in the system).

Results for one specific $\mathbf{y}$ are shown in Fig. 6. Note that our network recovers a decidedly multimodal distribution of $\mathbf{x}$ that visibly deviates from the prior $p(\mathbf{x})$. Note also the strong correlations in the system. For example, a given $\mathbf{y}$ may correspond to a young cluster with large expansion velocity, or to an older system that expands slowly. Finding these ambiguities in $p(\mathbf{x}|\mathbf{y})$ and identifying degeneracies in the underlying model are pivotal aspects of astrophysical research and the application of INNs has the potential to lead to a major breakthrough in this field.

5 Conclusion

We have presented a method for training Invertible Neural Networks to solve ambiguous inverse problems via conditional sampling. INNs were shown to excel in artificial experiments and provide meaningful insights for two real-world applications. While the correspondence between simulations and real measurements remains to be established, we share the excitement of the application experts to push INNs towards a generic tool, helping scientists from many different disciplines to better interpret their data and models, and to better plan their next experimental steps – be it modeling, measuring or simulation. A next step is to scale up the method to large data, such as images.

Acknowledgments

LA received funding by the Federal Ministry of Education and Research of Germany, project 'High Performance Deep Learning Framework' (No 01IH17002). JK, CR and UK received financial support from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 647769). SW and LMH received funding from the European Research Council (ERC) starting grant COMBIOSCOPY (637960). EWP, DR, and RSK acknowledge support by Collaborative Research Centre (SFB 881) 'The Milky Way System' (subprojects B1, B2 and B8), the Priority Program SPP 1573 'Physics of the Interstellar Medium' (grant numbers KL 1358/18.1, KL 1358/19.2 and GL 668/2-1) and the European Research Council in the ERC Advanced Grant STARLIGHT (project no. 339177)

References

[1] Jack A. Baldwin, Mark M. Phillips, and Roberto Terlevich. Classification parameters for the emission-line spectra of extragalactic objects. *PASP*, 93:5–19, February 1981.

[2] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv:1803.05649*, 2018.

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[4] Ela Claridge and Dzena Hidovic-Rowe. Model based inversion for deriving maps of histological parameters characteristic of cancer from ex-vivo multispectral images of the colon. *IEEE Trans Med Imaging*, November 2013.

[5] Ivo Danihelka, Balaji Lakshminarayanan, Benigno Uria, Daan Wierstra, and Peter Dayan. Comparison of maximum likelihood and GAN-based training of Real NVPs. *arXiv:1705.05263*, 2017.

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv:1410.8516*, 2014.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv:1605.08803*, 2016.

[8] Chris Donahue, Akshay Balsubramani, Julian McAuley, and Zachary C Lipton. Semantically decomposing the latent spaces of generative adversarial networks. *arXiv:1705.07904*, 2017.

[9] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv:1606.00704*, 2016.

[10] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv:1506.02158*, 2015.

[11] Dani Gamerman and Hedibert F Lopes. *Markov Chain Monte Carlo: Stochastic simulation for Bayesian inference*. Chapman and Hall/CRC, 2006.

[12] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

[13] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pages 2211–2221, 2017.

[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[15] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Bridging implicit and prescribed learning in generative models. *arXiv:1705.08868*, 2017.

[16] Douglas Hanahan and Robert A. Weinberg. Hallmarks of cancer: The next generation. *Cell*, 144(5):646–674, March 2011.

[17] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. *arXiv:1804.00779*, 2018.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017.

[19] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. *arXiv:1802.07088*, 2018.

[20] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

[21] Lisa J. Kewley, Michael A. Dopita, Claus Leitherer, Romeel Davé, Tiantian Yuan, Mark Allen, Brent Groves, and Ralph Sutherland. Theoretical evolution of optical strong lines across cosmic time. *The Astrophysical Journal*, 774(2):100, 2013.

[22] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv:1807.03039*, 2018.

[23] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[24] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[25] Alexander Kolesnikov and Christoph H. Lampert. PixelCNN models with auxiliary variables for natural image modeling. In *International Conference on Machine Learning*, pages 1905–1914, 2017.

[26] Jarno Lintusaari, Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic Biology*, 66(1):e66–e82, 2017.

[27] Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. *Journal of Biomedical Optics*, 19(1):10901, January 2014.

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

[29] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4797–4805, 2016.

[30] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, pages 1028–1036, 2016.

[31] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2335–2344, 2017.

[32] Eric W. Pellegrini, Jack A. Baldwin, and Gary J. Ferland. Structure and feedback in 30 Doradus. II. Structure and chemical abundances. *The Astrophysical Journal*, 738(1):34, 2011.

[33] Daniel Rahner, Eric W. Pellegrini, Simon C. O. Glover, and Ralf S. Klessen. Winds and radiation in unison: A new semi-analytic feedback model for cloud dissolution. *Monthly Notices of the Royal Astronomical Society*, 470:4453–4472, 10 2017.

[34] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[35] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv:1302.5125*, 2013.

[36] Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[37] Ralf S. Klessen and Simon C. O. Glover. Physical processes in the interstellar medium. *Saas-Fee Advanced Course*, 43:85, 2016.

[38] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. *arXiv:1701.05517*, 2017.

[39] N Siddharth, Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, and Philip HS Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

[40] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate bayesian computation. *PLoS computational biology*, 9(1):e1002803, 2013.

[41] Yunfei Teng, Anna Choromanska, and Mariusz Bojarski. Invertible autoencoder for domain adaptation. *arXiv:1802.06869*, 2018.

[42] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv:1711.01558*, 2017.

[43] Jakub M Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv:1611.09630*, 2016.

[44] Brian L Trippe and Richard E Turner. Conditional density estimation with bayesian normalising flows. *arXiv:1802.04908*, 2018.

[45] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

[46] Richard David Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 12(2):129–141, 2013.

[47] Sebastian J Wirkert, Hannes Kenngott, Benjamin Mayer, Patrick Mietkowski, Martin Wagner, Peter Sauer, Neil T Clancy, Daniel S Elson, and Lena Maier-Hein. Robust near real-time estimation of physiological parameters from megapixel multispectral images with inverse monte carlo and random forest regression. *International journal of computer assisted radiology and surgery*, 11(6):909–917, 2016.

[48] Sebastian J. Wirkert, Anant S. Vemuri, Hannes G. Kenngott, Sara Moccia, Michael Götz, Benjamin F. B. Mayer, Klaus H. Maier-Hein, Daniel S. Elson, and Lena Maier-Hein. Physiological Parameter Estimation from Multispectral Images Unleashed. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2017*, Lecture Notes in Computer Science, pages 134–141. Springer, Cham, September 2017.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *CVPR*, pages 2223–2232, 2017.

[50] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

Analyzing Inverse Problems with Invertible Neural Networks — Supplementary Material —

1 Generator with MMD loss

In the following, we present an alternative to the cGAN architecture, whereby the discriminator loss is replaced with an MMD loss. The MMD loss receives a concatenation of the generator output $\mathbf{x}$ and the label $\mathbf{y}$ it was supplied with, and compares these batch-wise with the concatenation of ground truth $(\mathbf{x}, \mathbf{y})$-pairs. This is effectively the same input that the learned discriminator also receives. Note, in contrast to this, the MMD loss of the INN only receives $\mathbf{x}$, and no information about $\mathbf{y}$. Fig. 7 shows the results, alongside the results of the INN and the cGAN from the main paper.

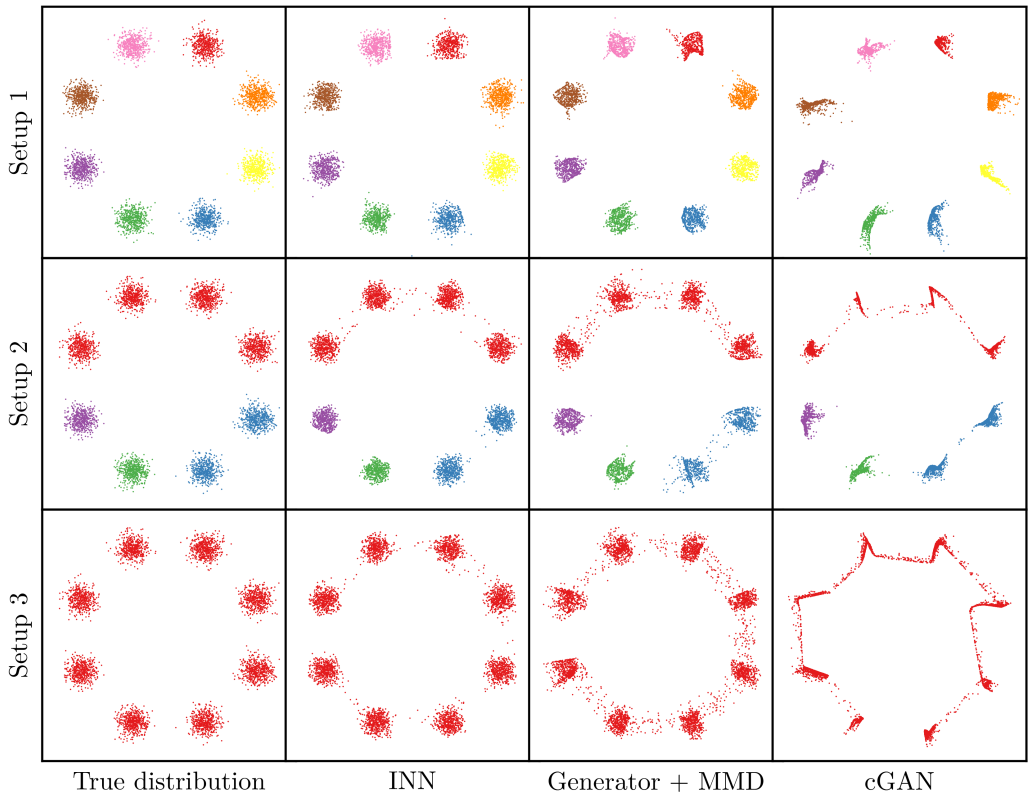


Figure 7: Comparison of INN, cGAN, and a cGAN-like generator trained with an MMD-loss instead of a discriminator.

2 Artificial data with complex forward process

In the main paper, we showed an artificial example in 2D. The complexity of that example lies in the shape of the prior, whereas the forward process is almost trivial. In the following, we present an additional example, where the forward process is more complex, and continuous as opposed to discrete. To construct this example, we begin with a standard normal distribution, which is then distorted and projected onto a line. This line is then again distorted to form a one-dimensional curve in 2D $\mathbf{y}$-space, shown in the top left panel of Fig. 8. The three points marked on the curve map to the sub-manifolds in $\mathbf{x}$-space indicated in the bottom right panel.

As shown in the top right, the INN solves the inverse problem correctly, using a latent dimensionality of $\dim(z) = 1$. For comparison, we also train an MMD-GAN with an equal number of parameters, which is also able to generate correct solutions. Both the dropout samping method and a cGAN did not converge to satisfactory solutions.

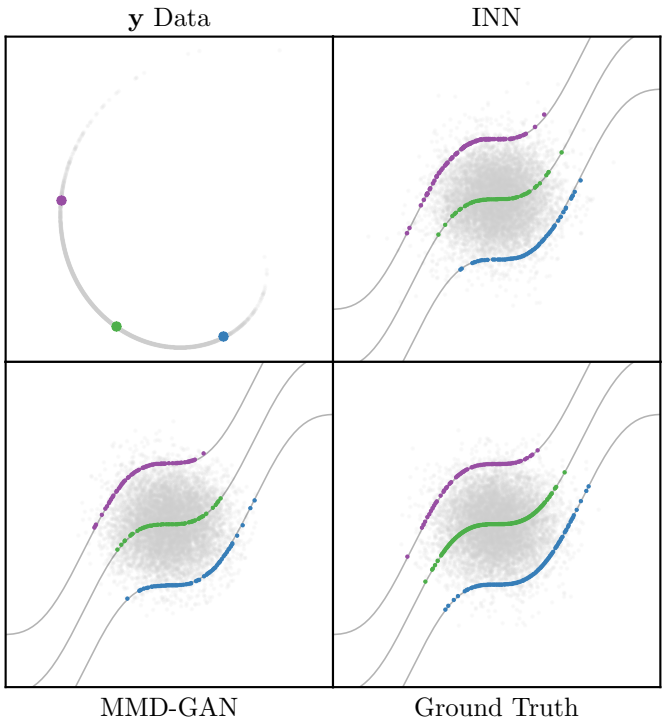


Figure 8: In clockwise order: $\mathbf{y}$-space of the artificial problem, $\mathbf{x}$-space with solution manifolds, MMD-GAN results, INN results

3 High-dimensional artificial data

In addition to the low-dimensional examples shown so far, we demonstrate an artificial example with a higher dimensionality. Hereby, we construct a Gaussian mixture model with 32 mixture components in 128-dimensional space to supply the hidden variables $\mathbf{x}$. The mixture components all have the same variance, and are scattered randomly around the origin, such that each mode has some overlap with the others. The forward process is given by 16 projections along random directions, producing the observations $\mathbf{y}$. This is in contrast to the example used in the main paper, where there was no overlap between mixture components, and the forward process was defined simply by mode labelling.

We now train four networks on this task: (i) INN, (ii) a Monte Carlo dropout network with learned aleatoric error term, (iii) a simple feed forward network for point estimates, and (iv) a generator with MMD loss on this problem – each with an approximately equal number of parameters in an attempt to ensure similar expressive power. We could not reach meaningful convergence on this task with

a cGAN, despite our best efforts. We compare these methods in various ways, summarized in the figures below.

Firstly, we analytically compute the linear subspace in the $\mathbf{x}$-domain that contains correct solutions to the inverse problem. For each possible solution sampled from the posterior of a given method, we can then calculate the closest point in this subspace. This gives us a measure of how far a method's solutions lie from the manifold of true solutions. This is tallied over many validation samples, and shown as a histogram in Fig. 9, top. We find that the INN produces by far the most accurate results. As the forward process is a matrix multiplication and therefore trivial to learn, the inverse is correct to a high accuracy. The point estimate method follows close behind, with the other methods performing significantly worse. Surprisingly, the generator trained with MMD performs the worst by far, although it visually worked very well in the low-dimensional artificial example.

Secondly, we pick a single validation data point, and sample from the estimated posterior of the sampling-based methods. We then perform principal component analysis (PCA) on these samples, the singular values of which are shown in Fig. 9, middle. We know that the posterior has $128 - 16 = 112$ dimensions, therefore it should only possess 112 non-zero singular values. We see that the INN learns to approximate this well, while the generator output collapses to a single point for a fixed $\mathbf{y}$, and the Monte Carlo dropout network samples solutions from the entire 128 dimensions, as it is not able to separate the correlated axes.

Lastly, we compare the quality of the priors produced when accumulating the network outputs over all validation samples, shown in Fig. 9, bottom. Naturally, the point estimates lie in extremely high likelihood regions, while the INN and Monte Carlo dropout network match the prior adequately. In fact, the dropout network matches the distribution better, as the prior will be composed of Gaussians by construction, whereas the shape has to be learned by the INN. The generator + MMD architecture is also centered around the correct region, but the distribution is much too broad, with both extremely high- and low-likelihood solutions occurring.

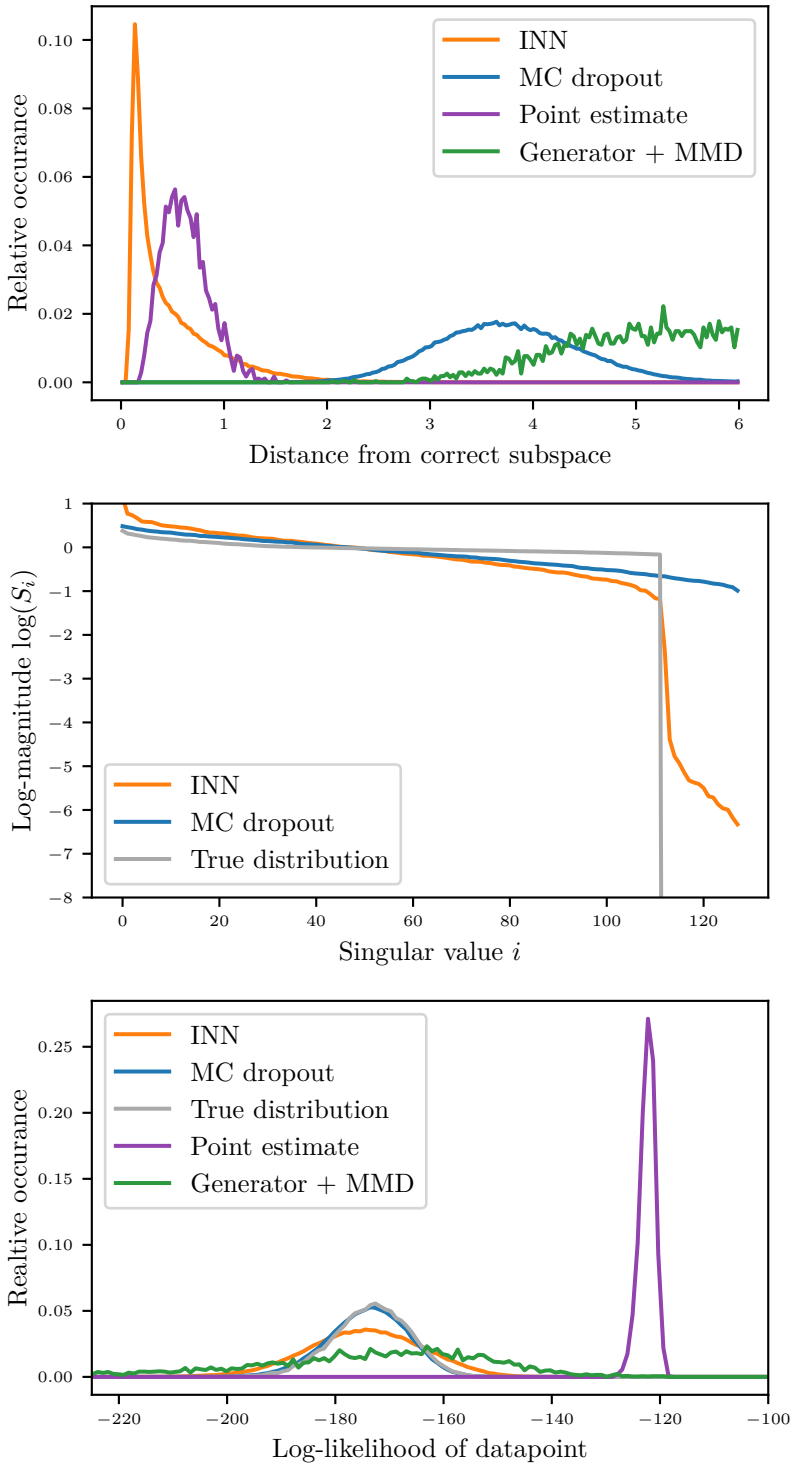


Figure 9: *From top to bottom:* (i) Distribution of ℓ^2-errors between the solutions to the inverse problem given by each method and the closest point in the subspace of true solutions. (ii) Magnitude of the singular values of samples conditioned on a single observation $\mathbf{y}$. (iii) Distribution of likelihoods of the outputs of each method compared to the true prior $p(\mathbf{x})$.

4 Calibration curve for tissue parameter estimation

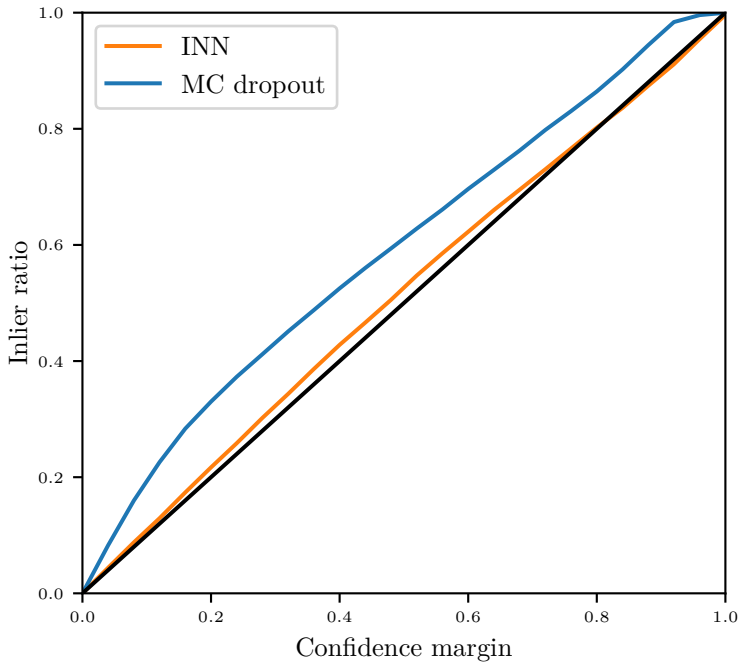


Figure 10: The full calibration curve, used to calculate the median calibration error in the main paper. We see that the curve of the INN is considerably closer to the true (*black*) curve.

5 Approximate Bayesian Computation algorithm

Algorithm 1: Approximate Bayesian Computation using a discrete training set

Result: Set of samples from the posterior $S_{\mathbf{y}} = \{\mathbf{x}_k \mid \mathbf{x}_k \sim p(\mathbf{x}|\mathbf{y})\}$
Training set S_{train}, number of samples N_{samples}, Gaussian noise in $\mathbf{y}$ with width $\sigma_{\mathbf{y}}$.
Initialize $S_{\mathbf{y}} \leftarrow \{\}$;
for $1 \leq i \leq N_{samples}$ **do**
  Draw with replacement $(\mathbf{x}_i, \mathbf{y}_i) \in S_{\text{train}}$;
  Compute $\rho_i = \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}\|^2}{2\sigma_{\mathbf{y}}^2}\right)$;
  Draw $r_i \sim \text{Unif}(0, 1)$;
  if $\rho_i \geq r_i$ **then**
    $S_{\mathbf{y}} \leftarrow S_{\mathbf{y}} \cup \{\mathbf{x}_i\}$;
  end
end

6 Details of datasets and network architectures

Table 2 summarizes the datasets used throughout the paper. The architecture details are given in the following.

6.1 Low-dimensional artificial data

INN: 3 invertible blocks, 3 fully connected layers per affine coefficient function with ReLU activation functions in the intermediate layers, zero padding to a nominal dimension of 16, Adam

Table 2: Dimensionalities and training set sizes for each experiment.

Section	training data	$\dim(\mathbf{x})$	$\dim(\mathbf{y})$	$\dim(\mathbf{z})$	see also
4.1	10^6	2	8	2	
4.2	15 000	13	8	13	[47]
4.3	8 772	19	69	17	[32]

optimizer, decaying learning rate from 10^{-3} to 10^{-5}, batch size 200. The inverse multiquadratic kernel was used for MMD, with $\alpha = 0.2$ in both $\mathbf{x}$- and $\mathbf{z}$-space.

Monte Carlo dropout: 6 fully connected layers with ReLU activations, Adam optimizer, learning rate decay from 10^{-3} to 10^{-5}, batch size 200, dropout probability $p = 0.2$.

cGAN: 6 fully connected layers for the generator and 8 for the discriminator, all with leaky ReLU activations. Adam was used for the generator, SGD for the discriminator, learning rates decaying from $2 \cdot 10^{-3}$ to $2 \cdot 10^{-6}$, batch size 256.

Generator with MMD: 8 fully connected layers with leaky ReLU activations, Adam optimizer, decaying learning rate from 10^{-3} to 10^{-6}, batch size 256. Inverse multiquadratic kernel, $\alpha = 0.5$.

6.2 High-dimensional artificial data

INN: 5 invertible blocks, with 2 fully connected layers and a leaky ReLU activation in the intermediate layer. Adam optimizer, learning rate decay from $2 \cdot 10^{-4}$ to $2 \cdot 10^{-6}$, batch size 200. No zero padding, inverse multiquadratic kernel with $\alpha = 100$.

Monte Carlo dropout/point estimate: 7 fully connected layers, ReLU activiations, Adam optimizer, learning rate decay from 10^{-3} to 10^{-6}, batch size 200.

Generator with MMD: 7 fully connected layers, leaky ReLU activiations, Adam optimizer, learning rate decay from 10^{-3} to 10^{-6}, batch size 256, inverse multiquadratic kernel, $\alpha = 100$.

6.3 Functional parameter estimation from multispectral tissue images

INN: 3 invertible blocks, 4 fully connected layers per affine coefficient function with leaky ReLUs in the intermediate layers, zero padding to double the original width. Adam optimizer, learning rate decay from $2 \cdot 10^{-3}$ to $2 \cdot 10^{-5}$, batch size 200. Inverse multiquadratic kernel with $\alpha = 1$, weighted MMD terms by observation distance with decaying $\gamma = 0.2$ to 0.

Monte Carlo dropout/point estimate: 8 fully connected layers, ReLU activations, Adam with decaying learning rate from 10^{-2} to 10^{-5}, batch size 100, dropout probability $p = 0.2$.

6.4 Impact of star clusters on the dynamical evolution of the galactic gas

INN: 5 invertible blocks, 4 fully connected layers per affine coefficient function with leaky ReLUs in the intermediate layers, no additional zero padding. Adam optimizer with decaying learning rate from $2 \cdot 10^{-3}$ to $1.5 \cdot 10^{-6}$, batch size 500. Kernel for latent space: $k(\mathbf{z}, \mathbf{z}') = \exp(-\alpha \|\mathbf{z} - \mathbf{z}'\|_2)$ with $\alpha = 0.02$. Kernel for $\mathbf{x}$-space: $k(\mathbf{x}, \mathbf{x}') = -\|\mathbf{x} - \mathbf{x}'\|_{1/2}^{1/4}$. Due to the complex nature of the prior distributions, this was the kernel found to capture the details correctly, whereas the peak of the inverse multiquadratic kernel was too broad for this purpose.