INAUGURAL-DISSERTATION
zur
Erlangung der Doktorwürde
der
Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften
der
Ruprecht - Karls - Universität
Heidelberg

vorgelegt von

Ardizzone, Lynton, MSc.
aus Stuttgart

Tag der mündlichen Prüfung: _____

Conditional Invertible Generative Models for Supervised Problems

Betreuer: apl.-Prof. Ullrich Köthe

Abstract

Invertible neural networks (INNs), in the setting of normalizing flows, are a type of unconditional generative likelihood model. Despite various attractive properties compared to other common generative model types, they are rarely useful for supervised tasks or real applications due to their unguided outputs. In this work, we therefore present three new methods that extend the standard INN setting, falling under a broader category we term *generative invertible models*. These new methods allow leveraging the theoretical and practical benefits of INNs to solve supervised problems in new ways, including real-world applications from different branches of science. The key finding is that our approaches enhance many aspects of *trustworthiness* in comparison to conventional feed-forward networks, such as uncertainty estimation and quantification, explainability, and proper handling of outlier data.

Zusammenfassung

Invertierbare neuronale Netze (INNs), in ihrer Anwendung als sogenannte “Normalizing Flows", sind ein Typ unkonditionales generatives Likelihood-Modell. Trotz einiger nützlicher Eigenschaften, die INNs im Vergleich zu anderen generativen Modellen besitzen, können sie selten für überwachtes Lernen und echte Anwendungen eingesetzt werden, da ihre Outputs nicht gelenkt werden können. In dieser Arbeit präsentieren wir deshalb drei neue Methoden, die das Standardsetting von INNs erweitern. Wir bezeichnen diese breitere Kategorie als *generative invertierbare Modelle*. Diese neuen Modelle machen sich die theoretischen und praktischen Vorteile von INNs zunutze, um Aufgaben des überwachten Lernens zu lösen, unter anderem auch mehrere erfolgreiche Anwendungen in den Naturwissenschaften. Das Kernergebnis besteht in der Beobachtung, dass unsere Methoden in vielen Aspekten die *Vertrauenswürdigkeit* (“Trustworthiness”) der Vorhersagen steigern, unter anderem die Schätzung und Quantifizierung von Unsicherheiten; die Erklärbarkeit; sowie der korrekte Umgang mit “Outliern", also dem Netzwerk komplett unbekannte Beispiele.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Ulli Köthe and my co-supervisor Prof. Carsten Rother for their guidance and constructive feedback. Their confidence in my abilities and my work were key for completing this thesis.

I extend my appreciation to my second examiner, Prof. Klaus Maier-Hein, as well as the other members of the examination board, for their thorough review and clear judgement.

I am grateful to all my colleagues at the CVL for both their practical advice and intellectual exchange, particularly Jakob Kruse, Radek Mackowiak, Jens Müller, and Felix Draxler, as well as Tim Adler at the DKFZ.

Furthermore, I want to recognize all researchers involved in the many collaborations during my work, at the IMSY group at the DKFZ; the ITA, ITP, IUP, and Psychological Institute at Heidelberg University; as well as at the IWES at Hannover University; the IKIM at University Duisbourg-Essen; the DFKI Saarbrücken; the IAK at TU Dresden; and the Fraunhofer ITWM.

Contents

1 Introduction

1.1. Problems with deep learning in real-world use

In terms of pure task performance, deep learning has made tremendous leaps in the past decade. There are few fields where this is as clearly shown as in computer vision, where the feats achieved with modern deep learning methods may have seemed more fitting for science-fiction as recently as two decades ago. For simple image classification, deep neural networks are arguably on par with human performance (Hu et al., 2020). Not only that, but the pre-deep-learning state of the art in classification is now easily surpassed even by one-shot deep learning models, that only receive a single example per class (Hu et al., 2020). In image generation, completely synthetic faces have become almost indiscernible from real ones (Karras et al., 2019). Deep learning models are even able to extract audio from imperceptible fluctuations and vibrations in video footage (Davis et al., 2014) or accurately reconstruct the surrounding environment using only the specular reflections of a bag of snacks on video (Park et al., 2020).

Besides computer vision, the same development is found in other fields of applied computer science as well: In natural language processing, recently introduced deep generative models (so-called transformers) are able to solve various tasks at a previously state-of-the-art level with a single general model by simply posing the task as a text prompt, and can synthesize longer and longer texts with coherent content and meaning (Brown et al., 2020). In the field of reinforcement learning, fully self-taught deep-learning based systems are now able to beat the best humans at Go, thought to be one of the last classic games where humans can retain the upper hand, while discovering completely new but effective strategies unthought of in the millennia-long history of the game (Silver et al., 2017). In machine learning research in general, it has become quite rare to find publications that do not make use of deep learning in some respect.

In light of these astounding capabilities, deep learning is being adopted more and more into most areas of society (Zhang et al., 2021). In this thesis, we place special focus on use in natural sciences and engineering research (Stevens et al., 2020), and to an extent also in industry (Bughin et al., 2018), both areas where deep learning adoption is growing quickly and promises to be especially fruitful. When we speak of 'real-world use' of deep learning in the following, we refer primarily to these areas.

With methods and applications transitioning from academic deep learning research to real-world uses, certain shortcoming are beginning to become more painfully apparent to practitioners (Cheatham et al., 2019; Stevens et al., 2020). These shortcomings mostly do not affect the pure task performance, but are orthogonal to it, rather impacting the reliability, trustworthiness, and ultimately the usefulness of deep learning systems. Not only have many of these aspects been slow to catch up to the year-over-year improvements in task performance, but in fact, some areas have even regressed over the last decade.

Li et al. (2021) compile an impressively exhaustive summary and analysis of these problems, citing almost 500 sources detailing the requirements for safe, reliable, and trustworthy use of deep learning, and how current systems struggle. In the following, we describe four central points, all of which remain largely unsolved but are critical for the long-term success of deep learning in real-word use cases, and which are especially relevant in this thesis.

Uncertainty quantification Many types of deep learning models do not output a measure of uncertainty by default. Especially for regression-based problems, the network requires some modified output modality to accommodate for the uncertainty, as well as an adapted training scheme. It is far from clear how to best implement this, and the most widely spread choice lies in simple and limited scalar uncertainty measures (Kendall and Gal, 2017). For classification, the mere output modality for uncertainty is less of a problem, as the commonly used softmax outputs naturally provide an uncertainty estimate, but even this breaks down for more complex discrete problems such as semantic segmentation.

This is also not where the issue ends: simply because a model supports uncertainty estimates in terms of its output modality, this does not mean that the produced estimates faithfully reflect the actual uncertainty of the model. This is quantified and examined using the concept of uncertainty calibration. Mis-calibration can mean that the model systematically makes under- or over-confident uncertainty estimates, the latter of course being much more dangerous for practical applications. It is well documented, e. g. by Guo et al. (2017), that the more complex a deep learning model gets, the worse the quality of the uncertainty estimates, and the more often a model makes vastly overconfident predictions.

Lives will be in danger if a medical deep learning system predicts with 99.9% confidence that a patient is healthy, even though the system is correct 80% of the time in the best of cases. In fact, most would argue that correcting the faulty confidence estimate should take priority over improving the average task performance from 80% up to e. g. 83%. Concerningly, a large majority of practical deep learning research focuses almost solely on the latter.

Handling out-of-distribution data An unavoidable fact of real-world use is the occurrence of anomalous situations and inputs that were never encountered during training. By definition, it is not possible to directly prepare or test a model for such anomalous inputs (also called out-of-distribution or OoD inputs).

Of course, the most desirable outcome is for the model to explicitly detect OoD inputs, alerting the user, stopping the machine in question, etc. This is a much harder problem than just performing binary classification between in-distribution and OoD data, as by definition OoD data is never exhaustively covered by the training data.

Even if a model does not support the detection, the least that could be expected is for it to make uncertain predictions on such inputs. Disappointingly, the opposite is the case, standard deep learning models frequently make highly confident but nonsensical predictions on OoD data (Ovadia et al., 2019).

An especially dangerous class of OoD inputs are so-called adversarial attacks. Hereby, the image is altered in an imperceptible or insignificant way to the human eye. However, the alteration is crafted in such a way that a targeted deep learning system fails dramatically,

e. g. by predicting a completely wrong class with extremely high confidence. Neither is it necessary to access the underlying model parameters for this to work (Bhambri et al., 2019), nor does the whole image have to be altered, a well-placed 'adversarial sticker' on a road-sign can be enough (Brown et al., 2017).

As with the uncertainty estimates, dealing with the different kinds of OoD inputs seems much more important for real-world use than improving the task performance on a controlled test set by some small percentage.

Diverse solutions We commonly require diversity in cases with very high-dimensional outputs, such as depth estimation, segmentation, colorization, etc. In such cases, there are often multiple competing but completely different plausible solutions, e. g. colorizing a grayscale image of a car to have a blue roof and blue doors, or red roof and red doors, but excluding e. g. a blue roof and red doors.

There is a fluent transition between uncertainty estimation and diverse solutions, but each end of the spectrum often requires different treatment and methods in practice. In cases requiring diverse solutions, a simple uncertainty output such as pixel-wise marginal distributions is of no use. Instead, we would want a network to generate multiple different competing hypotheses, each ideally with an assigned probability or weight. This setting is where the use of already existing generative models such as conditional GANs would seem the most obvious, but in practice hardly any works exist where GANs produce useful diverse outputs on such supervised problems. VAEs at least provide some exceptions to this, as shown e. g. by Kohl et al. (2018), but their use is far from widespread.

Explainability Deep neural networks are notorious for being 'black boxes', meaning that their inner workings are hard to understand or visualize. Both the training dynamics and the representations and decision processes occurring in deep learning models are poorly understood. Gilpin et al. (2018) provide an organized and categorized list of existing ideas and approaches for explainability.

Such techniques are absolutely necessary, because deep learning models often view the inputs in counter-intuitive and unintended ways. For instance, Geirhos et al. (2018); Brendel and Bethge (2018); Hermann and Lampinen (2020) demonstrate that counter to common belief, standard classification networks on ImageNet decide almost completely based on the texture of an object, not its actual shape or arrangement. Such artefacts and unintended behaviours can be extremely hard to discover and identify if the model functions as a black box. This is illustrated by the poignant fact that it took six years following the original AlexNet publication (Krizhevsky et al., 2012) until the surprising observations by Geirhos et al. (2018) mentioned above were brought to light, at which point the AlexNet paper had already amassed a five-digit citation count.

As a real-world example, Zech et al. (2018) demonstrate how a model used to classify diseases in X-ray images makes decisions by recognizing the model of X-ray machine used to take a picture, inferring likely diseases from the department of the hospital that the machine is located in (e. g. predicting heart disease if the image is from a machine stationed in the cardiology department). Such biases can be avoided to a degree through caution during data acquisition, exhaustive and careful testing, and robust models, but explaining the models decisions serves as a constant and final check.

Even barring such mistakes, many would argue that it is a societal requirement for deep learning systems to explain any decisions impacting human lives. This demand has already prompted legislative action in the EU (Goodman and Flaxman, 2017), starting to put a 'right to explanation' into law.

Lastly, explainability can help scientists who use deep learning in their research to better understand their data, the relevant factors in physical processes, important regions in an organism for some function, and so on. We provide several real examples of this later in the thesis.

1.2. The niche of normalizing flows and INNs

Discussing these existing problems for real-world applications, we find that generative models hardly play a role, being seen as even more unpredictable and fickle as feed-forward regression or classification models. Especially in computer vision, their real-world applications are mostly limited to entertainment, novelty apps, and gimmicks such as FaceApp, Wombo.ai, Artbreeder, and so on, where catastrophic failure has no consequences or even serves to heighten the amusement of the user.[1] In this thesis however, we take an opposing approach and embrace generative models for supervised tasks where feed-forward discriminative (i. e. non-generative) models would otherwise be used.

To clear up the basic terminology beforehand, normalizing flows are a certain type of generative model, and invertible neural networks (INNs) are simply the type of network used in normalizing flows. INNs can also be trained in other ways that do not conform to the normalizing flow setup as we demonstrate later in this thesis. For simplicity, we will refer to the models used in this thesis as 'invertible generative models', as they use different training methods and setups, but all have the use of INNs in common.

Compared to other types of generative models, specifically GANs and VAEs, normalizing flows have some unique advantages which are discussed in greater detail in the next chapter. In the past, they have most commonly been used for unsupervised, unconditional generative modeling. Both the construction of normalizing flows and the training process are theoretically and conceptually simpler and more elegant, avoiding the common issues faced by other model types due to the approximations or compromises in training procedure (see section 2.3). The invertibility of the networks used in normalizing flows also allows for interactive image manipulation and explainability in simple and intuitive ways (Kingma and Dhariwal, 2018; Jacobsen et al., 2019). Their connections to information theory make normalizing flows a prime choice for information-based out-of-distribution detection (Choi et al., 2018), compression tasks (Hoogeboom et al., 2019), or information-theoretic training objectives (Müller et al., 2021).

Arguably, normalizing flows are also hampered to some degree by the requirement for invertibility that is responsible for their advantages in the first place: it tends to increase the computational cost and complicates modifications of the setup (such as using conditioning information). INN architectures are less mature and readily available than much more widely-used standard feed-forward networks. Certainly, the existing deficits of normalizing

[1] Note, since writing the thesis, diffusion-based image generation services such as Stability.ai, Midjourney, etc. have become viable and popular. While these tools produce more impressive results than any preceding methods, the point above still stands.

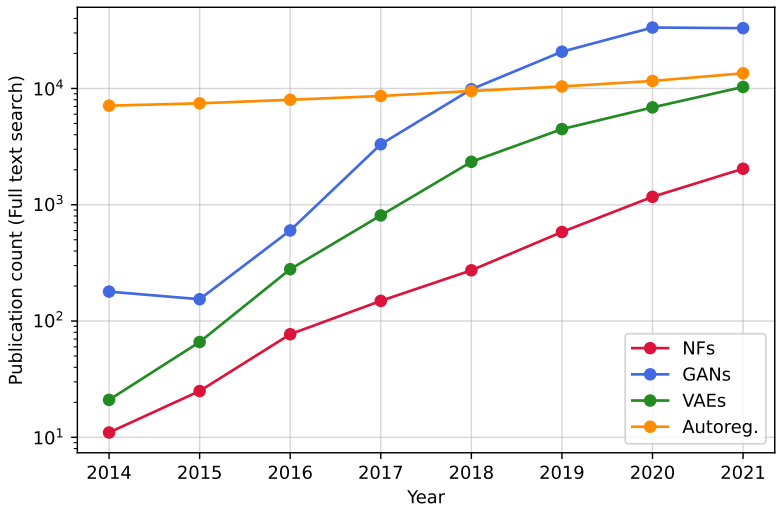


Figure 1.1.: Publications per year mentioning different kinds of generative models, as indexed by google scholar. Note the logarithmic scale. This should not be taken as a rigorous survey, but a rough indication of research trends. The search strings were `normalizing flow`, `generative adversarial [WORD]` (many different terms are used instead of 'network'), `variational (autoencoder|auto-encoder)`, and `autoregressive (model|flow)`.

flows are in no small part due to the fact that they entered the research community later than GANs or VAEs for instance, and have received less research attention in total. As a result, other types of generative models are still more prevalent in most areas.

We aim to capture these broad research trends by counting the number of publications discussing each type of generative model, shown in figure 1.1. The numbers should of course be taken with a grain of salt due to the rudimentary evaluation methods. Among other things, we expect all curves to be biased towards growth due to the field of deep learning as a whole growing.

According to publication volume recorded in figure 1.1, GANs are currently the most discussed models, but have seen little to no growth in recent years, perhaps indicating that the method has matured, or research attention is beginning to shift to other methods. Variational autoencoders (VAEs) seem to be going through a similar process with some delay: discussion volume is behind that of GANs, still growing, but no longer with the same exponential rate as pre-2018. The high prevalence of autoregressive models may surprise deep learning researchers, but autoregressive models (learned or handcrafted) are popular in many other branches of science, engineering and mathematics. Despite high absolute publication volume, the topic of autoregressive models has seen little growth over the past decade.

Most most important for this thesis, normalizing flows have been the least discussed type of model in the literature, with one or two orders of magnitude less attention than e. g. GANs at any point in time. At the same time however, they are experiencing the most consistent growth, with the number of publications discussing normalizing flows almost exactly doubling every year since 2014. Taking these number as reference, the current position of normalizing flows in research could be compared to that of GANs roughly in the

year 2017, a time where some of the most impressive and useful results from GANs had yet to be presented. Considering this, normalizing flows could be regarded as a promising niche in the field of generative modeling, with a large untapped potential for further development.

1.3. Goals and outline of the thesis

This thesis combines the two topics discussed so far: we claim that using invertible generative models for tasks usually reserved for supervised feed-forward networks is an overlooked way of addressing many of the problems of reliability, trustworthiness, and usefulness found in real-world applications.

Certainly, a single approach or thesis can not completely solve the challenge of trustworthiness, encompassing problems which are some of the most well-known and extensively researched issues in supervised deep learning. However, we do find that solutions to the issues arise in a much more natural and intuitive way with invertible generative models than is the case for existing feed-forward networks. This is supported by many successful uses of the presented methods in actual scientific research by others.

In chapter 2, we first summarize the theoretical and practical background of normalizing flows and INNs and current research directions. After that, we present three different methods, each used for different kinds of tasks and applications: chapter 3 presents a flexible way to produce detailed uncertainty estimates or diverse solutions on a wide range of problems. It is based on the following (largely overlapping) publications:

- Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019b
- Lynton Ardizzone, Jakob Kruse, Carsten Lüth, Niels Bracher, Carsten Rother, and Ullrich Köthe. Conditional invertible neural networks for diverse image-to-image translation. In *DAGM German Conference on Pattern Recognition*, pages 373–387. Springer, 2020a

In chapter 4, we then present are more specialized method, solving inverse problems in a interpretable and intuitive way, based on the following work:

- Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *Intl. Conf. on Learning Representations*, 2019a

Chapter 5 shows how to use invertible generative models for classification problems, improving uncertainty estimates and enabling explainability and out-of-distribution detection. It is based on the following works:

- Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. *Advances in Neural Information Processing Systems*, 33, 2020b
- Radek Mackowiak, Lynton Ardizzone, Ullrich Kothe, and Carsten Rother. Generative classifiers as a basis for trustworthy image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2971–2981, 2021

Finally, chapter 6 summarizes the contributions made in the course of the thesis, relates them once again to the open issues discussed in the introduction, and gives an outlook on the future of invertible generative models in real-world use cases.

2 Background

2.1. Normalizing flows

Normalizing flows are a type of generative model, meaning their task is approximating probability distributions. Historically, a setup similar to normalizing flows was first proposed by Deco and Brauer (1995); Hyvärinen and Pajunen (1999). The term 'normalizing flow' specifically was only popularized later by Tabak et al. (2010); Tabak and Turner (2013), although without using deep neural networks necessarily. Rippel and Adams (2013) were first to apply deep neural networks to this setup, followed later by more effective methods such as Dinh et al. (2014); Tomczak and Welling (2016); Dinh et al. (2016), etc., each discussed later.

To simplify the discussion in the following, we assume all variables to be real-valued vectors in euclidean space, $\mathbb{R}^D$. As is customary, we denote random variables as upper case letters ($X : \Omega \to \mathbb{R}^D$, etc.). and concrete instantiations of the random variables as lower case (real vectors $x \in \mathbb{R}^D$, etc.). The probability density function of a RV is written as $p(X)$, the evaluated density as $p(x)$ or $p(X{=}x)$ in ambiguous cases.

For some variable X of interest, the basic task of generative modelling is to produce some approximate density $q(X)$, with the goal of being as similar to $p(X)$ as possible, measured by some objective. In practice, $p(X)$ is rarely known explicitly or even knowable in principle, instead only a training set of M examples $\mathcal{X}_{\text{train}} = \{x^{(i)}\}_{i=1}^{M}$ is provided. The generative model can either be implemented such that $q(X)$ can be evaluated directly at any point of interest x (in which case we would speak of density estimation), or the model can be accessible through some way of producing new samples from $q(X)$. In principle, the latter is possible given the former through Monte-Carlo methods, but in practice the two are very different problems, especially in high-dimensional spaces. As we will see later, normalizing flows can do both at the same time.

2.1.1. Core idea of normalizing flows

The basic method employed in a normalizing flow is the same as with most other types of deep generative models, namely reparametrization: Instead of modeling a complex probability density $q(X)$ directly, it is implicitly expressed through a transformation between a so-called latent space Z, and data space X. Prescribing a distribution $p(Z)$ in latent space at the same time implies some $q(X)$ through the transformation. Other widely used generative models such as GANs have this basic structure in common, but differ in their theoretical and practical properties as well as the way they are trained.

The main feature setting normalizing flows apart from other methods is that the transformation between X and Z, which we will call f, is bijective. More accurately, both f

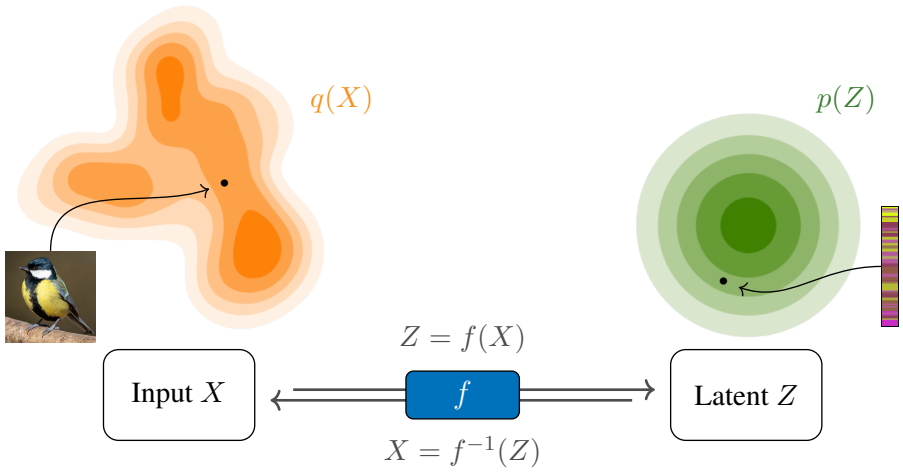


Figure 2.1.: Basic setup of normalizing flows: The complex distribution $q(X)$ on the left hand side is modeled by warping, stretching, and shaping the simple latent reference distribution $p(Z)$ on the right hand side. The transformation between the two is invertible, allowing for exact computation.

and its inverse must differentiable almost everywhere. For simplicity, this assumption is always included when speaking of 'invertible transformations' etc. below. Note that the invertibility also implies that X and Z have the same dimensionality.

How to parametrize such an invertible transformation in a learnable way will be discussed in section 2.2. For now we simply assume that f is chosen from some suitably large family of invertible mappings $\mathcal{F}$. The use of an invertible transformation allows for the expressed probability $q(X)$ to be easily and exactly computed at any point x using the change-of-variables formula. This is the central equation for normalizing flows, condensing their core idea:

$$q(x) = p\big(Z = f(x)\big) \left| \det \frac{\partial f}{\partial x} \right| . \tag{2.1}$$

Hereby, $\partial f / \partial x$ is the Jacobian matrix. We will simply write this as J for short in the following. Intuitively, the Jacobian term takes care of conservation of probability mass when the transformation stretches or squeezes space, i. e. changes the volume. The equation as a whole shows how choosing a different $f \in \mathcal{F}$ implies a different probability density function $q(X)$. When we write $q(X)$ in the following, this dependence on f is always implicitly given, as if imagining a subscript $q_f(X)$. Importantly, we can use this kind of model in two different ways: Firstly, we can use it for density estimation, inputting existing samples x at test time, and computing their probability under the model according to equation (2.1), i. e. we can perform density estimation. Secondly, we can sample new points from $q(X)$ at test time (in practice a very different task for high dimensionalities), by first sampling $z^* \sim p(Z)$, and then mapping the latent-space-samples back to X-space by inverting the transformation:

$$x^* := f^{-1}(z^*) \sim q(X) \tag{2.2}$$

The 'flow' in the term 'normalizing flow' refers to the transformation f. The chosen

latent distribution $p(Z)$ is usually a standard normal, so $q(X)$ is 'normalized' by the flow, hence '*normalizing* flow'.

Also note that normalizing flows can be formulated in a more general way using measures and transport theory (Bogachev et al., 2005). In that way, they can also be applied to non-euclidean spaces (Gemici et al., 2016), among other things. However, we will not expand on this in the context of this thesis.

2.1.2. Fundamental difference to other generative models

It is worth briefly discussing the main difference in model setup between normalizing flows, GANs, VAEs, and autoregressive models to better understand each and put them in context. We include a more in-depth discussion of the practical implications of these differences later in section 2.3.

For a GAN, the latent dimensionality is usually much smaller than the data dimensionality, and the transformation from latent space to data space is not invertible. This means that $q(X)$ does not exist as a real-valued probability density, and the change-of-variables formula cannot be applied. As a result, the model is usually *only* capable of sampling, not of density estimation. This also means a second network (the so-called discriminator) is necessary to estimate some measure of difference between $p(X)$ and $q(X)$, which is in itself often mathematically ill-defined.

In the case of a VAE, the model is variational, meaning the mapping between X and Z is probabilistic. To make the model tractable, a simple form is chosen:

$$p(Z|x) = \mathcal{N}\left(\mu(x), \sigma(x)\right). \tag{2.3}$$

The so-called encoder network (a standard feed-forward network) simply outputs the μ and σ. The mapping from Z back to X requires a second network, called the decoder. The variational setting does not allow for the exact model likelihood to be computed, only a bound on the likelihood can be derived. During training, this results in the so-called ELBO loss function, which can produce various artefacts (Bousquet et al., 2017).

Comparing these two most widely used types of generative models to normalizing flows, we can see that they firstly each require two networks, while normalizing flows only require one network with special restrictions for invertibility. Secondly, neither can compute the model's likelihood exactly, complicating both the model itself as well as the training procedure.

Finally, we also mention autoregressive models, which in some aspects (especially the exact likelihood) are most similar to normalizing flows. Autoregressive models take a somewhat different approach to generative modelling than the previous methods: instead of a single transformation f between data- and latent-space, they use the chain rule to decompose the probability of the vector X into a chain of probabilities for each dimension X_j, conditioned on all previous dimensions:

$$q(X) = q(X_1) \prod_{j=2}^{D} q\big(X_j \,|\, X_1, \ldots, X_{j-1}\big) \tag{2.4}$$

Each term is essentially a small one-dimensional conditional generative model, which can be simpler to implement than a single much larger transformation f. Each term is often

modeled in a similar way to normalizing flows, see section 2.3. When given an input x, all terms of the chain can be evaluated in parallel, making these models especially suitable for density estimation. Sampling on the other hand is computationally very expensive, having to iterate though each term in succession.

2.1.3. Training normalizing flows with maximum likelihood

A normalizing flow in the form described above does not imply or require a certain training scheme. It can in fact be trained in various ways, e. g. with a GAN-loss (Grover et al., 2017) or as an auto-encoder (Schirrmeister et al., 2018). We explore two further non-standard training schemes in chapters 4 and 5. However, there is one training method that is by far the most common and popular for normalizing flows, which is afforded by their unique capability to compute the exact model likelihood: so-called maximum likelihood training.

Maximum likelihood training means the objective during training is to maximize the probability assigned to the training samples by the model. For reasons both practical and fundamental, log-probabilities are used. As the logarithm is strictly monotonic in $(0, \infty)$, this does not change the optimum. The maximum likelihood loss $\mathcal{L}_{\mathrm{ML}}$ is then defined as

$$\mathcal{L}_{\mathrm{ML}} = \mathbb{E}_{X \sim p(X)} \big[-\log q(X) \big] = -\int p(x) \log q(x) \mathrm{d}x \tag{2.5}$$

and its empirical counterpart over the training set $\mathcal{X}_{\mathrm{train}}$ as

$$\widehat{\mathcal{L}}_{\mathrm{ML}} = -\frac{1}{M} \sum_{x^{(i)} \in \mathcal{X}_{\mathrm{train}}} \log q(x^{(i)}) \tag{2.6}$$

To compute the loss in practice, we can substitute the models density in equation (2.6) using equation (2.1), and simplify further by assuming the standard choice for a Gaussian latent space, $p(Z) = \mathcal{N}(0, 1)$:

$$\widehat{\mathcal{L}}_{\mathrm{ML}} = \frac{1}{M} \sum_{x^{(i)} \in \mathcal{X}_{\mathrm{train}}} -\Big(\log p\big(Z = f(x^{(i)})\big) + \log \det \left| J(x^{(i)}) \right| \Big) \tag{2.7}$$

$$= \frac{1}{M} \sum \frac{\| f(x^{(i)}) \|_2^2}{2} - \log \det \left| J(x^{(i)}) \right| + const. \tag{2.8}$$

When minimizing the loss, there is one important fact to consider: in principle, the true and the empirical objective have dramatically different minima. The empirical loss is minimized if only the finite number of training samples are assigned mass in the form of high peaks, and all other points receive likelihood as close to 0 as possible. If this is the minimum that results from the training process, we speak of *overfitting*. The desired behaviour is different: the model resulting from the training procedure should be at least similar to the minimum of the exact (but unknowable) objective. We would commonly say we want the model to *generalize* well.

Achieving good generalization is by no means automatic or natural: the construction of the family of functions $\mathcal{F}$ being optimized over should exclude or attenuate unwanted solutions, known as inductive bias. Various choices about the training procedure (initialization, optimization algorithm, data augmentation, etc.), broadly falling under the term regularization, further aid in finding a desired well-behaved local minimum over an unwanted one.

To formalize this more, the training process consists of finding the transformation $\hat{f}$ that leads to the smallest loss:

$$\hat{f} = \underset{f \in \mathcal{F}}{\texttt{optimize}}\left(\widehat{\mathcal{L}}_{\mathrm{ML}}\right) \underset{\sim}{\in} \underset{f \in \mathcal{F}}{\arg\min}\, \mathcal{L}_{\mathrm{ML}} \tag{2.9}$$

With the symbol $\underset{\sim}{\in}$ meaning we want $\hat{f}$ to be approximately equal to a member of the argmin-set ('approximate' in a vaguely defined way). Neither does nor should the optimization procedure (`optimize`) find a true minimum of the empirical loss $\widehat{\mathcal{L}}_{\mathrm{ML}}$, nor does the minimum of $\widehat{\mathcal{L}}_{\mathrm{ML}}$ coincide with that of $\mathcal{L}_{\mathrm{ML}}$. While some theoretical results exist, satisfying the approximation to an adequate degree is largely a matter of engineering in practice. The better the approximation holds, the better we expect the model to generalize.

2.1.4. Max likelihood training as minimizing Kullback-Leibler-divergence

The maximum likelihood objective has been used extensively for over a century in a wide range of different fields, and has been exhaustively studied. There are several ways of interpreting its use and effectiveness. For the first interpretation of maximum likelihood training, we write the Kullback-Leibler (KL) divergence between the true distribution and the one expressed by the model:

$$\begin{aligned} D_{\mathrm{KL}}\big(p(X)\big\|q(X)\big) &= \mathbb{E}_{X\sim p(X)}\left[\log\left(\frac{p(X)}{q(X)}\right)\right] && (2.10)\\ &= \mathbb{E}_{X\sim p(X)}\left[\log p(X)\right] - \mathbb{E}_{X\sim p(X)}\left[\log q(X)\right] && (2.11) \end{aligned}$$

The first term in this sum is completely independent of the model, and therefore needs not be included in the training process. The second term we can identify as the maximum likelihood loss function.

One important conclusion from this is that the distance metric that maximum likelihood training is implicitly optimizing for is exactly the KL divergence. This informs the characteristics, particularities, and failure modes of the resulting model, compared to generative models that use other distance metrics such as Jensen-Shannon-divergence, earth mover's distance, different Wasserstein metrics, maximum mean discrepancy, etc. (see for instance Huang et al. (2017) for a more in-depth discussion). The practical effects of this are discussed later in section 2.3.

The second conclusion to draw from this interpretation is that we can use the value of the maximum likelihood loss to easily and objectively compare between different normalizing flow models, architectures, and training improvements. The only missing ingredient to compute the KL divergence numerically is the unknown offset represented by the first term in equation (2.11). Because it is identical for all models, it still allows for a direct relative comparison between models without it being known. That is to say a model with a smaller KL divergence will always have a smaller maximum likelihood loss than a model with a larger KL divergence. This way to measure and compare the approximation quality is quantitative, reliable, and principled. Other generative deep-learning models have no such quality measures, instead relying on heuristics such as the Frechet inception distance (FID, Heusel et al., 2017). It should be noted however that comparisons using the maximum likelihood loss in this way are not completely without caveats (Theis et al., 2015).

2.1.5. Max likelihood in the Bayesian context

To not complicate the discussion unnecessarily when casting maximum likelihood training in a Bayesian perspective, we will prematurely use a result from the next section, which is that the functions $f \in \mathcal{F}$ are parametrized by real-valued parameter vectors θ (or Θ, seen as a random variable) from a compact parameter-space $\mathcal{P}_\Theta \subset \mathbb{R}^K$

We can then describe our knowledge about the likelihood of the model parameters that we can glean from the training set using Bayesian statistics. Specifically, we express the probability of model parameters using Bayes theorem:

$$p(\theta \mid \mathcal{X}_{\text{train}}) \propto p(\mathcal{X}_{\text{train}} \mid \theta) p(\theta) \tag{2.12}$$

The evidence in the denominator, $p(\mathcal{X}_{\text{train}})$, can be safely ignored since it stays constant when varying θ. With *i.i.d.* data, the likelihood on the right hand side can be decomposed. We make a slight adjustment to the notation, writing $p(x|\theta)$ as $q_\theta(x)$ to conform with the previous discussion (although the dependence on θ or f was always implicit above):

$$p(\mathcal{X}_{\text{train}} \mid \theta) = \prod_{x^{(i)} \in \mathcal{X}_{\text{train}}} q_\theta(x^{(i)}) \tag{2.13}$$

Taking the logarithm as usual results in

$$\log p(\theta \mid \mathcal{X}_{\text{train}}) = \sum_{x^{(i)} \in \mathcal{X}_{\text{train}}} \log q_\theta(x^{(i)}) + \log p(\theta) + const. \tag{2.14}$$

The first resulting term is again equal to the empirical maximum likelihood loss, albeit negative missing the normalization constant $1/M$. Assuming for a moment a uniform prior, i. e. ignoring the second term as constant, we can see that the maximum a posteriori estimate for the model parameters of the left hand side, i. e. $\hat{\theta} = \arg\max_\theta p(\theta|\mathcal{X}_{\text{train}})$, corresponds exactly to the minimum of the maximum likelihood loss.

Furthermore, the Bayesian interpretation offers a more principled motivation for regularization schemes: in the Bayesian picture, weight regularization corresponds to the prior term $\log p(\theta)$ in equation (2.14). For example, the most widely used $L2$ weight regularization consists of adding $\mathcal{L}_{\text{reg.}} = \tau \|\theta\|_2^2$ to the overall objective during training, τ being the strength of the regularization. It is simple to verify that this corresponds exactly to a Gaussian prior over weights,

$$p(\Theta) = \mathcal{N}\left(0, \sqrt{1/2\tau}\right). \tag{2.15}$$

Lastly, the Bayesian interpretation of maximum likelihood training allows connecting normalizing flows with the existing results from Bayesian statistics, treating normalizing flows as any other Bayesian model. For instance, it allows using tools such as *Bayesian information criterion* (BIC) or the *widely applicable information criterion* (WAIC) to make well-founded and justified statements about overfitting, generalization, and out-of-distribution detection (Choi et al., 2018). The connection is also important in establishing normalizing flows as a tool for Bayesian practitioners (Radev et al., 2020).

2.2. Invertible network architectures

The previous section focused on the more theoretical principles of normalizing flows. At the center of this was the function f, member of a learnable, invertible class of functions $\mathcal{F}$. However, we previously omitted any details how to implement this class of functions in practice, which will be the focus of this section. From the construction and usage of normalizing flows in the previous section, we can first derive the following properties that the members of $\mathcal{F}$ must fulfill:

- The functions must be bijective and both the forward and inverse must be differentiable almost everywhere ('almost' w. r. t. Lebesgue-measure in our case).
- From a practical computational standpoint, the inverse must be easy to compute.
- The log-Jacobian-determinant must also be readily computable. Note that the full Jacobian matrix is not required. In fact, many of the methods discussed below can *only* compute the determinant, not the full matrix.
- The parametrization must be such that $\mathcal{F}$ is a sufficiently general class of functions able to represent complex and very high-dimensional invertible transformations.
- The parametrization must also be suitable for mini-batch gradient-descent in the same way that standard feed-forward neural networks are, to make the optimization tractable.

A deep learning model constructed and parametrized in a way that it fulfills these properties is typically labeled an invertible neural network, or INN. While normalizing flows are not the only use for INNs (see e. g. Gomez et al., 2017), they are by far the most widespread.

2.2.1. Coupling-based INNs

So-called coupling blocks are currently the most popular way to construct INNs. An INN typically consists of a number of such blocks chained together, in a similar way to layers in a standard feed-forward network. The number of coupling blocks required for a sufficiently expressive INN varies between 4 for simple and low-dimensional data to over 100 for very large INNs (Kingma and Dhariwal, 2018).

For the sake of discussion, we call the input to a coupling block x and the output u. These do not coincide with the in- and outputs to the INN for a network with more than one block. The essential idea of the coupling block is that the input x and output u are each split into two halves, x_1, x_2 and u_1, u_2. Each half is transformed through the *coupling operation*, which we will call T_1, T_2, conditioned by the other half. This coupling operation is a simple function (often element-wise) with a known inverse and Jacobian determinant, but the conditioning can be arbitrarily complex and non-linear (this is where the INN gets its expressive power). As a simple example for the time being, we can imagine element-wise addition with learned, conditional coefficients $t(u_2)$ (more examples further below):

$$u_1 = x_1 + t(x_2) =: T_1(x_1; x_2) \tag{2.16}$$

$$x_1 = u_1 - t(x_2) =: T_1^{-1}(u_1; x_2) \tag{2.17}$$

Note that the second conditioning argument, x_2, stays the same. For the complete coupling block, this is performed on both halves. The entire 'trick' to make the block invertible is

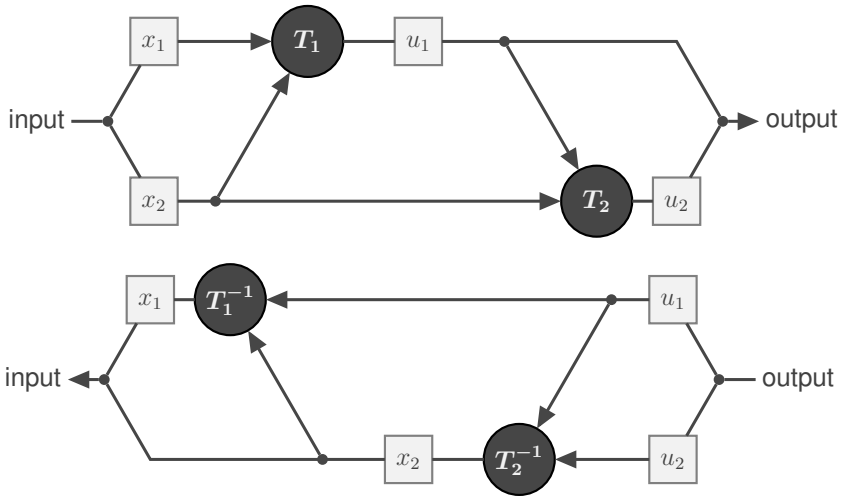


Figure 2.2.: Illustration of a general coupling block, using a coupling operation T. Top: block run forward (left to right). Bottom: same block inverted (right to left).

that the output u_1 of the first step is used as the condition for the second step in place of x_1 (note the second line):

$$u_1 = T_1(x_1; x_2) \tag{2.18}$$
$$u_2 = T_2(x_2; u_1) \tag{2.19}$$

Such a coupling block and is illustrated in the top half of figure 2.2.

Inverting the block is just as simple and cheap as computing the forward pass: starting from only the outputs of the forward pass, u_1, u_2, the second step in equation (2.19) can be inverted first, providing x_2. This in turn allows for the first step to be inverted as well:

$$x_2 = T_2^{-1}(u_2, u_1) \tag{2.20}$$
$$x_1 = T_1^{-1}(u_1, x_2) \tag{2.21}$$

The inversion is shown in the bottom half of figure 2.2. As we can see, the coupling block as a whole makes a complex transformation tractably invertible, as the T functions never have to be inverted w. r. t. the second conditioning argument.

To compute the whole block's Jacobian determinant, it is simplest to write each step in equation (2.19) as a block matrix, factoring the Jacobian into the two steps:

$$\frac{\partial u}{\partial x} = \frac{\partial [u_1, u_2]}{\partial [x_1, x_2]} = \begin{pmatrix} \frac{\partial T_1}{\partial x_1} & \frac{\partial T_1}{\partial x_2} \\ 0 & \mathbb{1} \end{pmatrix} \begin{pmatrix} \mathbb{1} & 0 \\ \frac{\partial T_2}{\partial u_1} & \frac{\partial T_2}{\partial x_2} \end{pmatrix} \tag{2.22}$$

The zero- and unit-matrices in the bottom/top half come from the variable that is not affected by each step. Marked in red are the derivatives w. r. t. the conditioning argument, which can not be computed in a tractable way. However, as we only require the determinant of the Jacobian, we can express the overall determinant as a product of block matrix determinants, where each intractable term is canceled with each 0, leaving the following:

$$\det\left(\frac{\partial u}{\partial x}\right) = \det\left(\frac{\partial T_1}{\partial x_1}\right) \det\left(\frac{\partial T_2}{\partial x_2}\right) \tag{2.23}$$

Recall that the coupling operations $T_{1,2}$ were specifically chosen so that the Jacobian determinant w. r. t. the main argument is known. In the literature, it is often stated that they must be triangular, but this is not strictly the case as long as the determinant can be cheaply computed. To summarize the result: while the full Jacobian of the coupling block can not be readily computed, we can compute its determinant more easily by decomposing the Jacobian, and computing the determinants of each component separately.

It bears mentioning at this point that some authors in the literature define a coupling block as only one of the two steps of the coupling block described here, and viewing swapping the two halves between steps as a separate network component.

There is a variety of operations that can be used for coupling mechanism in practice. T_1 and T_2 are usually identically constructed, but each has separate learnable parameters $\theta_{1,2}$. For simplicity, we omit the indices and write T and θ (meaning T_1 and θ_1). Most commonly, some kind of coefficients are predicted from u_2 by shallow feed-forward subnetworks, and these coefficients then determine the actual transformation. The most simple example already used above is additive coupling, introduced in the NICE architecture (Dinh et al., 2014):

$$T(x_1; x_2) = x_1 + t_\theta(x_2). \tag{2.24}$$

Here, t_{θ_1} is the subnetwork with network weights θ_1. This is the same mechanism as a standard ResNet (He et al., 2016), except that the variables are split into the two halves. Notably, the Jacobian determinant of this is always exactly 1 (c. f. equation (2.23)), demonstrating a restricted expressive power: additive coupling is by construction volume preserving, meaning that it can only shear, shift, and rotate distributions, and not freely stretch or squeeze them. For instance, this implies that the modeled distribution will have the same number of modes as the latent distribution (one in the Gaussian case), and can not be freely multimodal.

As an improvement to the NICE architecture, Dinh et al. (2016) introduce the real-NVP (i. e. real non-volume-preserving) INN. Here, the coupling blocks use affine coupling instead of additive, with the subnetwork outputting both additive and multiplicative coefficients t and s:

$$T(x_1; x_2) = x_1 \odot \exp\big(s_\theta(x_2)\big) + t_\theta(x_2), \tag{2.25}$$

with $\odot$ representing element-wise multiplication. In practice, a joint vector of s and t coefficients is output by a single subnetwork, to save computation power and make use of parameter sharing. The exponential in equation (2.25) ensures that input x_1 is always multiplied with a positive, non-zero value, otherwise invertibility would not be guaranteed. The inversion is given just as simply as the forward computation through

$$T^{-1}(u_1; x_2) = \big(u_1 - t_\theta(x_2)\big) \odot \exp\big(-s_\theta(x_2)\big) \tag{2.26}$$

The log-Jacobian-determinant is equally convenient to compute, consisting simply of the sum of the s_θ-coefficients. Using such affine coupling as opposed to purely additive greatly increases the expressive power of the model and allows for a qualitatively larger family of functions to be expressed. Nevertheless, attention must be paid concerning numerical issues if the transformations become too extreme (Behrmann et al., 2020). Due to its special importance to the field as a whole and this thesis in particular, the affine coupling block is illustrated in figure 2.3.

Beyond this, there is also a number of other schemes for the coupling mechanism. To briefly list the most notable:

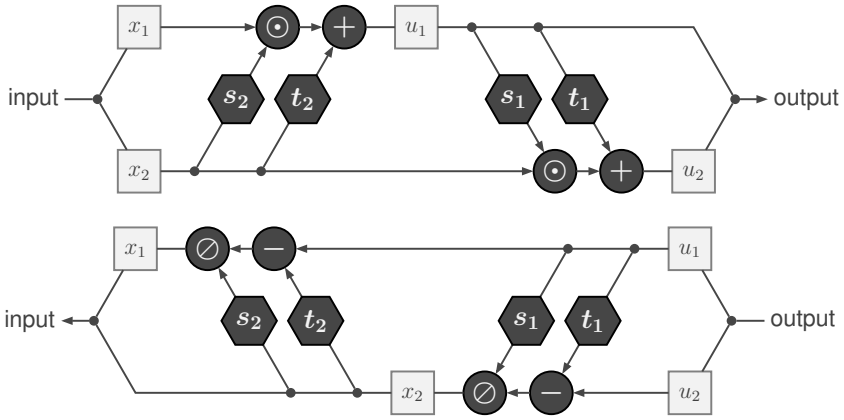


Figure 2.3.: Affine coupling block as introduced by Dinh et al. (2016). Top: block run forward (left to right). Bottom: same block inverted (right to left). In practice, $s_{1,2}$ and $t_{1,2}$ would be merged into a single subnetwork each predicting both the s and t coefficients at the same time.

- Non-linear squared flow (Ziegler and Rush, 2019): A more expressive mechanism compared to affine coupling. Coefficients a, b, c, d, g are all predicted by a subnetwork (omitting the full notation for clarity, $a_\theta(x_2) =: a$ etc., and taking all operations element-wise):

$$T(x_1; x_2) = a + bx_1 + \frac{c}{1 + (dx_1 + g)^2} \tag{2.27}$$

  The coefficients have to be restricted in certain ways through regularization, as the equation is not invertible for all combinations of values.

- Mixture-CDF (Ho et al., 2019): The subnetwork outputs the parameters of a mixture of logistic distributions. The coupling function is then the cumulative density function (CDF) of this mixture, exploiting the fact that CDFs are monotonic and analytically known in the case of logistic distributions.

- SOS-flows (Jaini et al., 2019): Applies to both autoregressive models and coupling blocks. The authors use some long-standing algebra theorems to parametrize polynomials of arbitrary degree in a way that they are guaranteed to be monotonically increasing. It is however computationally more expensive to invert.

- GIN (Sorrenson et al., 2020): A modification to standard affine coupling to make it volume-preserving. It is significantly more expressive than NICE (additive coupling), because only the overall determinant is constrained to 1, not every diagonal entry of the Jacobian. It is used for applications where the theory or method requires a volume-preserving transformation.

While each of these methods has specific advantages or special use cases, the past years of research progress have shown that standard affine coupling as introduced by Dinh et al. (2016) is still the most practically useful for a wide range of applications, especially in high dimensionalities. It is more expressive than NICE to a meaningful degree, while the more complex coupling schemes offer diminishing returns despite higher computational cost. Further reaffirming this choice, Teshima et al. (2020) prove theoretically that a series of affine coupling blocks is enough to be a universal approximator for diffeomorphisms.

2.2.2. Non-coupling INN architectures

Besides coupling-based INNs, various other ideas exist how to parametrize a sufficiently flexible, learnable invertible function (Teng et al., 2018; Hoogeboom et al., 2020; Song et al., 2019b). In the following, we discuss two such methods in more detail that have seen some adoption in the wider research community.

i-ResNet The concept of the i-ResNet stems from two observations: Firstly, mathematically speaking, standard ResNet blocks (He et al., 2016) will represent an invertible function as long as the residual subnetworks r_θ have a Lipschitz constant smaller than 1:

$$u = x + r_\theta(x) \tag{2.28}$$

Note that the input is not split into halves in contrast to a coupling block, making the function more expressive, but not in general invertible for Lipschitz constants > 1. Secondly, during training of a normalizing flow, the INN does not have to be inverted. This suggests that forward and inverse computation do not necessarily have to be symmetrical in terms of computation cost, at least within reasonable limits. After all, the network will typically have to generate much fewer samples during test time running inverted than processing samples at training time running forward (often $> 10^7$).

Based on this, Behrmann et al. (2019) introduce the *i-ResNet*: An INN consisting of completely standard ResNet block, but with a penalty on their Lipschitz constant. The Jacobian determinant can not be readily computed in the same way as with standard coupling blocks, so this is performed numerically using Hutchinson's trace estimator, the details of which are omitted here. Despite using this computationally more expensive and less accurate method to determine the Jacobian determinant, the authors claim this is offset by the more expressive architecture and show that the estimator's precision does not meaningfully impact training. At test time, each block of the network is inverted with a numerical fixed-point algorithm.

Neural ODEs and FFJORD It is possible to interpret the i-ResNet (or ResNets in general) as a discrete step Euler integration, with each residual block representing one integration step. Numbering the features between layers with t makes this clearer: the output x_{t+1} of a residual block is then

$$x_{t+1} = x_t + r(x_t; \theta_t), \tag{2.29}$$

with θ_t referring to the parameters of the residual subnetwork r of that block.

Neural ODEs (Chen et al., 2018) present a natural extension of this interpretation, looking at residual Network architectures through the lens of ODEs beyond fixed step Euler integration. For this, all residual subnetworks $r(\cdot, \theta_t)$ are replaced by a single neural network with parameters θ and an additional 'time' input t: $r(x, t; \theta)$. This is effectively a *Hypernetwork* (Ha et al., 2016), combining all the separate residual subnetworks, and selecting each using t. The main difference to before is that t can be made continuous instead of discrete. Implied by the Euler scheme interpretation of the ResNet, performing a forward pass of the model is equivalent to solving the following ODE:

$$\frac{\partial x}{\partial t} = r(x, t; \theta). \tag{2.30}$$

The main benefit of neural ODEs is in the fact that the forward pass can be computed using a variety of state-of-the art ODE solvers that dynamically adjust their step sizes to an ideal interval, instead of the fixed-step Euler scheme. Importantly for the case of normalizing flows, firstly the process is guaranteed to be invertible under weak restrictions placed by the Picard-Lindelöf-theorem, secondly the time-inverted intergration is just as cheap to compute as the forward direction. Numerical inversion algorithms or Lipschitz-constraints as with the i-ResNet are not necessary. In the ODE interpretation, the non-invertibility in a standard ResNet is simply an artefact of the coarse and inaccurate fixed-step integration. Grathwohl et al. (2018) further improve on the method introduced by Chen et al. (2018), greatly simplifying and accelearting the computation of the Jacobian determinant by adapting Hutchinson's trace estimator to the setting.

2.2.3. Spatial structure in INNs

In practice, especially in scientific applications and computer vision, we commonly want to process grid-structured data, such as images, videos, volumes, etc. Just as with standard feed-forward networks, we need to take this structure into account in the form of shift-invariant or -equivariant building blocks for our INN. We also want to change the resolution and number of features throughout the network to go from low-level to abstract information.

The standard for shift-equivariant operations used in feed-forward networks is the convolution, and elementary component in virtually all networks working on grid-structured data such as images. Reproducing this behaviour in coupling-based INNs is easily possible by making the subnetworks, i. e. the primary learnable component, convolutional. To make sure that dimensions are coupled with dimensions that are close in the image or grid, it is not plausible to split the input in the center or randomly (although this can be experimented with as a regularization technique, see Kirichenko et al. (2020)). Instead, the split is performed along the channel-direction, for instance u_1 would contain the red channel of the whole image, and u_2 the green and blue channel of the entire image. Taking proper padding into account, the coupling coefficients produced by the subnetworks will match the image height and width. Note that recently, Hoogeboom et al. (2020) as well as Song et al. (2019b) also proposed other possibilities for equivariant learned operations, modeling invertible convolution kernels directly. However, these methods have not seen widespread practical use yet.

Shift-invariant operations in standard feed-forward networks are most kinds of pooling operations. Local max- or mean-pooling is invariant to small shifts in the input, and global mean-pooling, as used e. g. in a standard ResNet (He et al., 2016), is fully shift-invariant. However, by definition, a shift-invariant operation can never be bijective (two shifted inputs are not allowed to produce the same outputs). Instead, re-ordering or re-shaping of feature space in a repeating pattern is used to reduce the resolution and increase the feature channels (e. g. Dinh et al. (2014) or Jacobsen et al. (2018) for two possibilities), or the entire feature space is flattened into a single vector. There have been some attempts to introduce at least some aspect of shift-invariance to these operations. For instance, Jacobsen et al. (2019) introduce a discrete cosine transform (DCT) as an alternative to global mean pooling, so that the translation-dependence is limited to only some of the output dimensions, while some are shift-invariant. A further example is found in section 3.5 of this thesis, presenting an invertible alternative to local mean-pooling operations.

One more common practice introduced early in the development of normalizing flows

by Dinh et al. (2016) aims to better deal with the large number of dimensions present in image data by using skip connections: after a segment of the INN, commonly before the resolution is reshaped, half of the variables are split off, and directly become part of the latent space. The desired behaviour is that these feature dimensions will contain low-level detail information, that does not have to be transformed any further, while only the remaining dimensions are processed into more abstract information. This results in a segmented latent space where half of the variables where processed by the first segment of the INN, a quarter were processed by the first two segments, and so on. Overall, this measure can help reduce the computational effort of the INN.

2.3. Normalizing flows in the context of generative modeling

With normalizing flows being one of the less popular types of generative models (c.f. figure 1.1), the aim of this section is to compare their properties and results from recent normalizing flow research to the more common types of generative models to better understand the advantages and possible use cases of normalizing flows.

Normalizing flows and GANs Under the assumption of a perfect discriminator (and other strong assumptions), the standard GAN training procedure corresponds to the generator minimizing the Jensen-Shannon-divergence to the true distribution (Nowozin et al., 2016). In reality however, these assumptions are far from satisfied during training, and work is ongoing to theoretically characterize which objective GANs are actually optimizing for (Huang et al., 2017).

The empirical observations however are unambiguous: GANs suffer severely from a phenomenon called mode-collapse (Thanh-Tung and Tran, 2020), and normalizing flows do not. Mode collapse describes the situation where a generative model assigns close to zero mass to parts of the distribution, either missing certain modes completely or collapsing to a lower-dimensional manifold. This may not be apparent or even detectable for natural image generation, but it certainly rules out the use of GANs for many tasks where covering the full distribution is important, including uncertainty estimation, diverse solutions, out-of-distribution detection, concept discovery, and more.

The KL divergence that normalizing flows implicitly optimize for carries a huge penalty for not assigning mass where it should be ($q \to 0 \implies p \log(p/q) \to \infty$), but has no direct penalty for falsely assigning mass where there should be none ($p \to 0 \implies p \log(p/q) \to 0$), only indirectly by the density being too low in other places. This makes normalizing flows (as well as VAEs to a lesser extent) practically immune to mode collapse (Bousquet et al., 2017), and modeling errors will lead the outputs to be safely under-confident rather than over-confident and limited (see also section 3.8 and chapter 4). A real-world practical example demonstrating this difference in behaviours of the two types of models is given e. g. by Wang et al. (2017) in robotics.

While the ensured diversity and mode-coverage of normalizing flows is no doubt an advantage for many applications, this is not the case for all: for realistic image generation, having full mode-coverage seems less important than (or even detrimental to) generating visually pleasing images. This includes tasks such as in-painting, object replacement, etc. While GANs produce more impressive, realistic and complex images year over year (Karras

et al., 2017, 2019; Brock et al., 2019), flow-based models have struggled to produce results with comparable visual quality. As the most prominent example, Kingma and Dhariwal (2018) invested a large amount of research and computation power into producing a normalizing flow with the best possible image fidelity, with such a large model that it requires training on 64 GPUs in parallel. However, the visual quality of the results is at best middling compared to competing GANs, which produced higher-resolution and more realistic images using a fraction of the model parameters.

A systematic comparison of GANs and normalizing flows for image generation in particular was performed by Danihelka et al. (2017) as well as Grover et al. (2017). The authors essentially confirm that the differences between the types of models are not due to the invertibility restrictions, but rather the fundamental training setup. The GAN discriminator effectively focuses on aspects of the distribution that humans also prioritize, while the KL divergence implied by maximum likelihood training weights visually less important aspects more highly. Kirichenko et al. (2020) further support these observations.

It is possible that these findings discouraged further research in the direction of realistic image generation with normalizing flows, although some recently emerging publications have begun to revisit the topic (Yu et al., 2020). Most recent research on normalizing flows however seems to focus mostly on lower-dimensional problems and data, and more methodological contributions, using datasets such as MNIST and CIFAR-10 as well as scientific data.

Normalizing flows and VAEs In terms of practical use, VAEs are perhaps best comparable to normalizing flows. They too represent a bi-directional mapping between latent- and data-space, except that they are variational, i. e. the two mappings are probabilistic. To make VAEs tractable, certain simplifying assumptions are made in their construction, which well-known and extensively examined problems (Alemi et al., 2018; Bousquet et al., 2017), most notably the well-known blurry reconstructions and samples.

Due to the similar concept of VAEs and normalizing flows, various authors have combined the methods in some way: Firstly, an some of the earliest applications of normalizing and autoregressive flows were to improve the latent distributions of VAEs, attempting to correct for their shortcomings. (Rezende and Mohamed, 2015; Kingma et al., 2016; Tomczak and Welling, 2016, 2017; Berg et al., 2018). Secondly, adversarial autoencoders (as opposed to VAEs) are not variational, which allows using an INN as encoder and decoder at the same time, demonstrated by Schirrmeister et al. (2018) (although the resulting model could just as fittingly be described as an adversarial normalizing flow). Finally, Nielsen et al. (2020) combine invertible and variational operations in a single methodological framework, seamlessly bridging normalizing flows and VAEs.

Normalizing flows and autoregressive models Each conditional distribution in autoregressive models is often implemented using the same coupling mechanisms as normalizing flows (see section 3.2), mapping between each variable and a 1D 'latent space' with the coupling function. In this setting, we often speak of *autoregressive flows*. Important advancements were contributed by Germain et al. (2015); Kingma et al. (2016); Uria et al. (2016); Papamakarios et al. (2017); Huang et al. (2018a).

Comparing them from this viewpoint, the fundamental difference between autoregressive and normalizing flows is simply that the former have a triangular Jacobian for the entire

transformation they represent, while normalizing flows have a more or less free-form Jacobian matrix (when using coupling-based INNs, the transformation often consists of many steps alternating upper- and lower-diagonal Jacobian matrices).

This indeed makes a fundamental theoretical difference: fully triangular transformations can approximate any distribution (Bogachev et al., 2005), but unlike coupling networks clearly can not approximate any diffeomorphism (Teshima et al., 2020). This rules them out for many uses beyond the standard setup, where special structure or properties of the latent space or transformation have to be enforced, such as Sorrenson et al. (2020); Köhler et al. (2020), etc. and chapters 4 and 5 of this thesis. A recent publication by Kruse et al. (2021) introduces a coupling block with a sufficiently expressive fully triangular Jacobian, blurring the lines between the two types of models. A second shortcoming of autoregressive flows as opposed to normalizing flows is that they are very expensive to produce samples from, and instead best used for density estimation. Inverse autoregressive flows use autoregressive architectures with the opposite choice 'forward' direction, leading to very fast inverse, but slow forward computation. This may be attractive if the target distribution can *only* be evaluated, but not sampled from, or combined with autoregressive flows as a forward-inverse pair (most famously Oord et al., 2018).

In practice, deep autoregressive models have proven especially successful in domains where a sequential structure of the data naturally fits the sequential decomposition by the model, such as speech synthesis (Oord et al., 2016a). For images, more advanced schemes are necessary: naively decomposing the 2D image into a 1D sequence of variables results in an unnatural and inconvenient ordering of the pixels. Various schemes exist to correct for this problem, but complicate the model setup (Papamakarios et al., 2017; Oord et al., 2016c; Kolesnikov and Lampert, 2017).

In conclusion, comparing normalizing flows to the other types of generative models in this way serves to explain their standing as a fast growing niche in the field of generative modelling. With the current state of development, they may not be the first choice for general purpose data generation due to architectural restrictions, higher computational cost, and deficits in realistic image generation. However, as the large number of successful methods and applications shows, including those in this thesis, they have the potential to be extremely powerful and flexible tools with capabilities that no other type of generative model can supply. It also begun to become apparent though a number of recent high-profile publications that normalizing flows can be of just as much use for large and competitive computer vision applications when they are not used completely stand-alone, but combined in some advantageous way with more standard feed-forward components (Rombach et al., 2020; Pumarola et al., 2020; Chen et al., 2021).

3 Conditional Invertible Neural Networks

We consider a supervised problem with the goal to predict some variable X from an observed input Y. Most standard deep-learning solutions (especially for regression) take the approach of making some 'best guess' $\hat{x}$ for the answer, using a feed-forward network g: $\hat{x} = g(y)$. During training, the output $\hat{x}$ and the ground-truth solution x are compared using some function $D(\hat{x}, x)$, for instance the $L2$-distance. The choice of D will determine the behaviour and failure modes of the prediction model.

Works like those of Kendall and Gal (2017) show how this can be understood from a Bayesian perspective: in this view, even a feed-forward model implicitly estimates a Bayesian posterior for the problem, $q(X|Y)$. Hereby, the choice of D implicitly defines some simple parametric posterior. For instance, the standard $L2$ loss

$$D(\hat{x}; x) = \|\hat{x} - x\|_2^2 = \|g(y) - x\|_2^2 \tag{3.1}$$

implies a uniform-width Gaussian posterior centered around $\hat{x}$, as $D(\hat{x}; x)$ is the logarithm of this posterior evaluated at x:

$$\log q(x|y) = \log \mathcal{N}\big(x; g(y), \sigma\big) \propto \|g(y) - x\|_2^2 \tag{3.2}$$

This interpretation highlights a basic problem with such standard feed-forward supervised approaches: Without knowledge of the shape of the posterior, we can not know which supervised loss function to choose. If the loss function that does not agree with the posterior, the resultung model will make inaccurate or misleading predictions, and possible uncertainty estimates may be nonsensical. It therefore seems obvious how and why generative models should be used instead: a conditional generative model can generate a posterior $q(X|Y)$ completely freely. It will not suffer from the unnecessarily restrictive parametrization of the posterior implied by feed forward networks, and can addresses uncertainty quantification and diverse solutions in a uniform way.

How much more powerful and useful a completely free multimodal and multidimensional posterior really is, will become most clear through the examples in the course of the thesis. But we already note at this stage for how multiple separate and distinct solutions, linear or non-linear correlations and dependencies in solution space, quantile-based uncertainty intervals, among many other things, can only be expressed using such a generative posterior rather than a restrictive parametric one, especially if it is only implicit.

Our quantitative and qualitative comparisons to other methods later clearly demonstrate that *invertible* generative models are best suited for this role, but we can also argue for them beforehand: their unique capability for both density estimation and cheap sampling open up different ways of accessing the uncertainty distributions in a single model, for instance computing marginals and finding the maximum a posteriori estimate exactly using

density estimation, and producing diverse solution hypotheses or computing a Monte-Carlo expectation of some quantity of interest through sampling. Also, as discussed previously in section 2.3, we don't expect invertible generative models to miss modes or regions of the uncertainty distribution like GANs do, and our experiments confirm the theoretical knowledge that VAEs smear, blur, and distort detailed uncertainty distributions in both high- and low-dimensional spaces.

We see the plain reason why previous works have not realized this method in the fact that no flexible and expressive enough conditional invertible generative models exist to be used in this way. It is not immediately obvious how best to introduce conditioning information to otherwise unconditional standard normalizing flows. The remainder of this chapter is therefore based on the overlapping publications Ardizzone et al. (2019b) and Ardizzone et al. (2020a): we will first introduce the core ideas how to make normalizing flows and INNs into conditional models, and demonstrate these ideas with toy experiments. We then enhance the method to deal with more varied and complex types of conditions, demonstrating its capabilities on two computer vision problems with diverse solutions. Finally, we give an overview of several publications from different scientific fields making use of the presented methods, showcasing key examples how scientist profit from using invertible generative models in practice.

3.1. Core idea

For conditional GANs (cGANs) and conditional VAEs (cVAEs), introducing a condition is simple: the condition is given as an input to generator and discriminator or encoder and decoder. These networks are then able to process the condition in a useful way, e. g. extracting useful high-level information from complicated conditions such as images, time series, etc. Simply using the condition as an input like this is not possible with normalizing flows and INNs. To see this, we refer back to the change-of-variables formula for normalizing flows, equation (2.1):

$$q(x) = p(Z = f(x)) \left| \det \frac{\partial f}{\partial x} \right|$$

We find that if we simply use the condition y along with x as an additional input to the INN (as we do for cGANs, cVAEs), the normalizing flow will model the joint probability $q(x, y)$:

$$q(x, y) = p(Z = f(x, y)) \left| \det \frac{\partial f}{\partial [x, y]} \right| . \tag{3.3}$$

This is of little use for conditional modeling, neither for conditional density estimation (although it might be possible to divide by a known $p(y)$ in certain cases), and especially not for conditional sampling. This being said, we refer to Kruse et al. (2021), who developed the *HINT* architecture, using some of the advancements made in this chapter. HINT receives the joint input $[x, y]$, and allows density estimation and sampling of $p(x, y)$, $p(x|y)$ and $p(y)$, all with a single model, by inducing a certain dependency pattern in the coupling blocks allowing to decompose the models density.

Comparing equation (2.1) and equation (3.3), we can recognize the following possibility for a conditional normalizing flow model: We provide y in the form of an auxiliary input $f(x; y)$, indicated by the semicolon, meaning that the invertibility is only w. r. t. x:

$$z = f(x; y) \quad \Longleftrightarrow \quad x = f^{-1}(z; y) \tag{3.4}$$

Putting this into the change-of-variables formula gives us

$$q(x \mid y) = p(Z = f(x; y)) \left| \det \frac{\partial f}{\partial x} \right| . \tag{3.5}$$

In contrast to the joint input $f(x, y)$, z has the same dimensionality as x, and likewise we compute the determinant only of $\partial f / \partial x$. This becomes more intuitive to grasp if we temporarily write y as a subscript, as in $f_y(x) \coloneqq f(x; y)$:

$$q(x \mid y) = p(Z = f_y(x)) \left| \det \frac{\partial f}{\partial x} \right| . \tag{3.6}$$

Written like this, the equation looks like a completely standard normalizing flow, but with a 'selection variable' y, that selects a different invertible transformation based on the auxiliary input. We call an INN that has such an auxiliary conditioning input a *conditional invertible neural network*, or cINN for short.

To train such a conditional model, no further modifications are necessary: maximum likelihood training works just as well for conditional distributions:

$$\mathcal{L}_{\mathrm{cML}} = \mathbb{E}_{p(X,Y)} \Big[-\log q(X \mid Y) \Big] = \int -p(x, y) \log q(x \mid y) \mathrm{d}x \mathrm{d}y \tag{3.7}$$

and its empirical counterpart

$$\widehat{\mathcal{L}}_{\mathrm{cML}} = -\frac{1}{M} \sum_{[x^{(i)}, y^{(i)}] \in \mathcal{X}_{\mathrm{train}}} \log q\big(x^{(i)} \mid y^{(i)}\big). \tag{3.8}$$

Correspondence with the conditional KL divergence and conditional Bayesian MLE holds up in the same way as in section 2.1.5 and section 2.1.4.

Finally, we also want to note that there is a second, complementary way of making a normalizing flow conditional, but using a standard *unconditional* INN. That is by making the latent distribution conditional on y instead:

$$p(x \mid y) = p(Z = f(x) \mid y)|J| \tag{3.9}$$

This is the construction that will be used in chapters 4 and 5. There are some trade-offs and considerations between the two possibilities, which will be discussed in the later chapters.

3.2. Conditional affine coupling blocks

To produce a cINN in practice, we modify the affine coupling block as introduced by Dinh et al. (2016), as this is the most popular INN architecture. However, our method also carries over directly to other types of coupling blocks (e. g. Dinh et al., 2014; Kingma and Dhariwal, 2018, etc.) or i-ResNet blocks (Behrmann et al., 2019). As a reminder of section 3.2, each standard coupling block splits its input x into two parts $[x_1, x_2]$ and applies affine transformations between them that each have strictly upper or lower triangular Jacobian matrices:

$$\begin{aligned} u_1 &= x_1 \odot \exp\big(s_1(x_2)\big) + t_1(x_2) \\ u_2 &= x_2 \odot \exp\big(s_2(u_1)\big) + t_2(u_1) \, . \end{aligned}$$

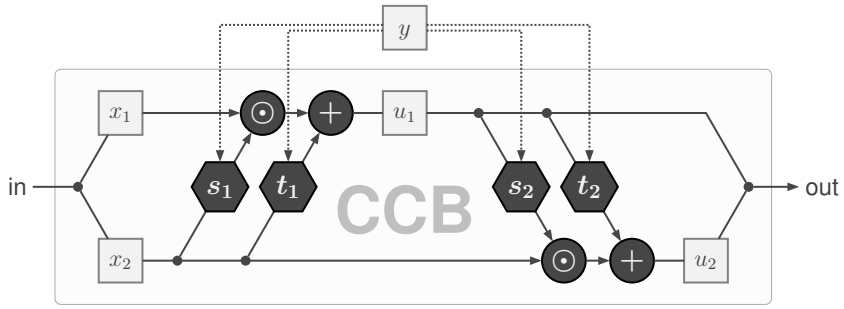


Figure 3.1.: Schematic illustration of one conditional affine coupling block (CCB), as described in the text.

The outputs $[u_1, u_2]$ are concatenated again and passed to the next coupling block. The internal functions s_j and t_j are be represented by arbitrary neural networks, and are only ever evaluated in the forward direction, even when the coupling block is inverted:

$$\begin{aligned} x_2 &= \big(u_2 - t_2(u_1)\big) \oslash \exp\big(s_2(u_1)\big) \\ x_1 &= \big(u_1 - t_1(x_2)\big) \oslash \exp\big(s_1(x_2)\big) \ . \end{aligned}$$

To make this coupling block conditional, we note that the subnetworks s_j and t_j are never inverted. This means we can concatenate conditioning data y to their inputs without losing the invertibility, replacing $s_1(x_2)$ with $s_1(x_2, y)$ etc. as follows:

$$\begin{aligned} u_1 &= x_1 \odot \exp\big(s_1(x_2, y)\big) + t_1(x_2, y) \\ u_2 &= x_2 \odot \exp\big(s_2(u_1, y)\big) + t_2(u_1, y) \ . \end{aligned} \tag{3.10}$$

This conditional coupling block (CCB) design is illustrated in figure 3.1. Notably, the inversion still functions the same way, when y is also known for the inverse. And just as with standard coupling blocks, as derived in section 2.2 or in more detail by Dinh et al. (2016), the log-determinant of the $\partial u / \partial x$ Jacobian matrix is still simply the sum over all dimensions of $s_1(x_2, y)$ and $s_2(u_1, y)$.

In practice, there are some considerations for making a joint vector of $x_{1,2}$ and y. Unstructured vectors can be trivially concatenated, and images or arrays of matching sizes can be concatenated along the channel-axis. In real applications however, y and x might have very different sizes, structures, and data types which are not easily compatible – class labels, images, vectors, time series, graphs, unordered sets, point clouds, to name just a few. This problem will be solved in by the changes in section 3.4.

3.3. Experiment – conditional MNIST generation

As a first experiment for conditional coupling blocks, and conditional maximum-likelihood training, we train a cINN on the MNIST dataset. MNIST is a collection of 28×28 pixel greyscale images of handwritten digits (this will be x in our case), together with a discrete label from 0-9 (this is the condition y in our case). For the sake of the experiment, we flatten the images into unstructured vectors of length $28^2 = 784$. We then construct a cINN of 24 coupling blocks using fully connected subnetworks s and t. We provide the labels y in form of a one-hot vector, i. e. a binary $\{0, 1\}^K$ vector where only the entry corresponding to the

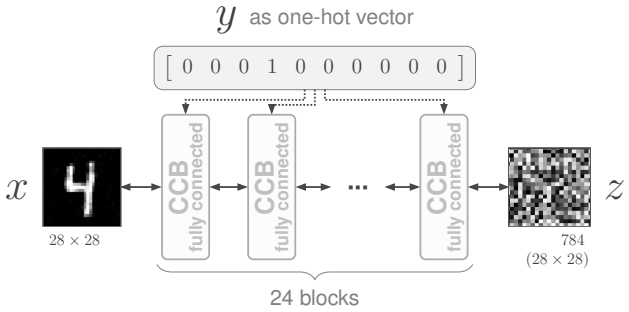


Figure 3.2.: Schematic overview of the cINN used for conditional MNIST generation.

label is 1, the rest 0. This one-hot conditioning vector is simply concatenated onto the usual input of the subnetworks, giving them an input size of $\dim(x)/2+\dim(y) = \frac{784}{2}+10 = 402$ (c. f. figures 3.1 and 3.2). For data augmentation we only add a small amount of Gaussian noise to the images ($\sigma = 0.02$), as described later in section 3.5.

We then train the model with the conditional maximum-likelihood loss as explained above. After training, we generate samples from the model by sampling a random latent vector z^* from a standard normal distribution $\mathcal{N}(0, 1)$, and mapping it back to image space using the inverse network, along with different labels y^* as conditions. Such samples are shown in figure 3.3. Interestingly, the generated digits retain a characteristic style when changing the condition but leaving the latent vector z^* fixed. In other words, the style is disentangled from the class. This is in no way guaranteed, theoretically it is just as possible for the model to decompose into 10 essentially separate and different models, one for each class. In that case, the latent space of each class would be structured differently, and inter-class transfer of latent vectors would be meaningless. This property, in conjunction with our network's invertibility, can directly be used for style transfer, as demonstrated in figure 3.5: an existing image from the test set, along with its corresponding label, is transformed into a latent vector. This latent vector is then mapped back through the inverted network, but using a different condition. This allows generating the other digits in the same unique style as the existing input image. The methodology was later used by Rombach et al. (2020) for style transfer on larger images.

In figure 3.4 we further illustrate the structure of latent space. Not only are style and class disentangled from each other, but the style dimensions themselves are also disentangled. We can identify some axes in latent space with interpretable meanings, that linearly interpolate between aspects of the style, while most axes simply encode the image noise. After all, the intrinsic dimension of MNIST is known to be much lower than 784. Note that while the latent space is learned without supervision, we found the shown in the figure in a semi-automatic fashion: We perform principle component analysis (PCA) on the latent vectors of the test set, without the noise augmentation. This allows us to identify the subspace of meaningful style dimensions. Some meaningful axes in this subspace are shown in the figure. As with the class-style disentanglement, there is no theoretical reason to expect such linear style-style disentanglement to occur; infinitely many solutions where the different style elements are mixed non-linearly are just as valid for the cINN to learn.

It is currently not clear how these two disentanglement properties arise in the cINN, whether they are due to inductive bias or some other property. Recent work of Sorrenson et al. (2020) introduces a special type of conditional normalizing flow called *GIN*. Using this special model, the authors are able go give theoretic guarantees under which disentangle-

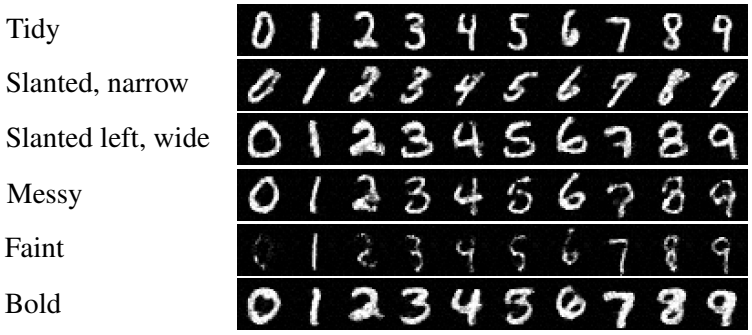


Figure 3.3.: MNIST samples from our cINN conditioned on digit labels. All ten digits within one row $(0, \dots, 9)$ were generated using the same latent code z, but changing condition y. We see that each z encodes a single style consistently across digits, while varying z between rows leads to strong differences in writing style.

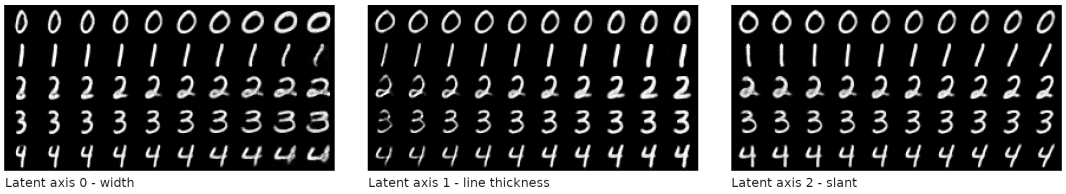


Figure 3.4.: Axes in our MNIST model's latent space, which linearly encode the style attributes width, thickness and slant.

ment in latent space must occur. However, standard cINNs explicitly violate the conditions required for the guaranteed disentanglement of GIN, and further research is needed in this area.

3.4. Introducing a conditioning network

While the basic cINN from the previous sections works well if the condition is dense, informative, and simple to incorporate to the subnetworks (such as class labels), problems arise for more general settings. As already mentioned, conditions such as graphs, time-series, etc. can not be easily fed into the subnetworks. More generally, we also often expect that some higher-level features need to be extracted from y for the conditioning to be effective, e. g. global semantic information from an image, overarching behaviour in a time-series, and so on. In such cases, feeding the condition y directly into the CCBs would not be sufficient: even if all the dimensionalities or image sizes were by chance compatible and aligned, it would place an unreasonable burden on the s and t networks, as they would need to re-learn any higher-level features in each coupling block. After all, the conditioning information is not passed depth-wise through the cINN in the same way that x is. Note that cGANs or cVAEs do not suffer from this issue, the condition can be processed freely by their feed-forward networks.

Instead, we propose extracting useful high-level features from the condition through some learned function $\varphi(y)$, which maps from the domain of y to a real-valued feature space. In practice, this can take different forms: in section 3.6, φ is a feed-forward ResNet that extracts pyramid of both visual and semantic features at different resolutions; in sec-

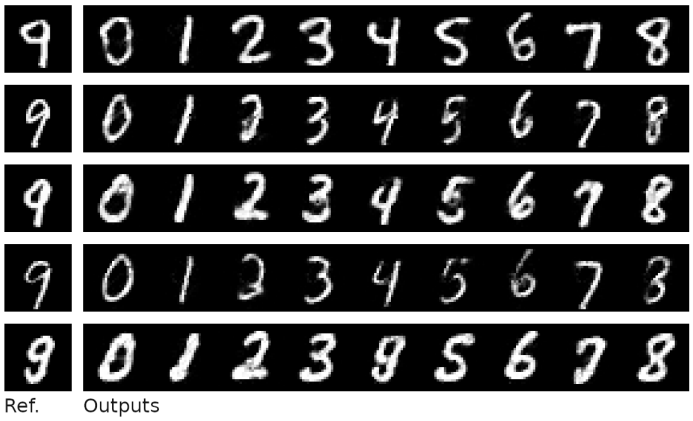


Figure 3.5.: To perform style transfer, we determine the latent code $z = f(x; y, \theta)$ of a test image *(left)*, then use the inverse network $g = f^{-1}$ with different conditions $\hat{y}$ to generate the other digits in the same style, $\hat{x} = g(z; \hat{y}, \theta)$.

tion 3.7 it is a network producing a single feature map with semantic information in each pixel; in section 3.8.6 it is a permutation-invariant network producing a vector of learned summary statistics; it can also be a graph neural network, transformer network, neural process, or any other kind of feature extractor.

While it is certainly possible to use a pre-trained feature extractor, or any self-supervised or unsupervised representation learning approach to train φ, we are more interested in training φ and the conditional coupling blocks jointly from scratch. For this, we simply back-propagate the conditional maximum-likelihood loss $\mathcal{L}_{\mathrm{cML}}$ through the feature extraction. Intuitively speaking, the more useful the learned features are for the cINN's task, the lower the $\mathcal{L}_{\mathrm{cML}}$ loss will become. Therefore, the conditioning network is encouraged to extract useful features.

We can formalize this using the information-theoretical concept of mutual information (MI). MI quantifies the amount of information that two variables share, in other words, how informative one variable is about the other. For any two random variables a and b, It can be written as the KL-divergence between joint and factored distributions: $I(a, b) = D_{\mathrm{KL}}(p(a, b) \| p(a)p(b))$. With this, we can derive the following proposition, details and proof are found in the appendix:

Proposition 1. *Let $\hat{\theta}$ be the INN parameters and $\hat{\varphi}$ the conditioning network that jointly minimize $\mathcal{L}_{\mathrm{cML}}$. Assume that the INN $f(\cdot; \cdot, \theta)$ is from a family of universal density approximators $\mathcal{F}$, defined in Assumption 1 (appendix), and φ is optimized over a family $\mathcal{G}_0$ defined in Assumption 2 (appendix). Then it holds that*

$$I\big(x, \hat{\varphi}(y)\big) = \max_{\varphi \in \mathcal{G}_0} I\big(x, \varphi(y)\big) \tag{3.11}$$

In other words, the learned features will be the ones that are maximally informative about the generated variable x. Importantly, the assumptions about the conditioning networks family $\mathcal{G}_0$ are quite weak and do not say anything about its representational power: the features will be as informative as possible within the limitations of the conditioning network's architecture and number of extracted features.

We can go a step further under the assumption that the power of the conditioning network and number of extracted features are large enough to reach the global minimum of the loss (sufficient condition given by Assumption 3, appendix). In this case, we can also

show that the cINN as a whole will learn the true conditional distribution by minimizing the loss (proof in appendix):

Proposition 2. *Assume φ has been optimized over a family $\mathcal{G}_1$ of universal approximators and $\dim(c) \geq \dim(y)$ (Assumption 3, appendix), and the INN is optimized over a family of universal density approximators $\mathcal{F}$ (Assumption 1, appendix). Then the following holds for $(x, y) \in \mathcal{X}$, where $\mathcal{X}$ is the joint domain of the true training distribution $p(x, y)$:*

$$q(x|\hat{\varphi}(y), \hat{\theta}) = p(x|y) \tag{3.12}$$

In our experiments and the work of other authors building on ours, we find that the conditioning network is a critical part for using cINNs in practice. Even in cases where the data-types and -structures between x and y are directly compatible, using a conditioning network improves the performance. Our theoretical results confirm that the conditioning network will learn something useful in all situations. We therefore include the conditioning network in the definition of a cINN from here on, i. e. a cINN means the series of conditional coupling blocks combined with a conditioning network.

3.5. Improvements to convolutional cINNs

In this section, we highlight some of the changes and engineering techniques needed to achieve meaningful results on computer vision problems using cINN, or INNs in general. Due to INNs being much less prevalent than feed-forward architectures, and normalizing flows being used less than other types of generative models, there is not yet a set of established techniques, tweaks, and standard architectures to ensure good performance on computer vision tasks. Instead, many publications on normalizing flows use toy data such as 2D probability densities or MNIST.

Image dequantization. All digital data must necessarily be quantized, and for most digital image formats, this quantization is especially drastic: each color channel is encoded by an 8-bit unsigned integer, assigning one of 255 brightness levels. This quantization is known to cause problems in training normalizing flow-based models, due to the mismatch of observably discrete input data but continuous latent space (Theis et al., 2015). To avoid this, it is common to add a small amount of noise to the input images, most often uniform noise with an amplitude corresponding to the quantization level. While this procedure is not new to this thesis, we perform ablation studies below in section 3.7 and further experiments with noise types and scales along with a more detailed theoretical examination in section 5.3. Overall, we find that it depends on the image size, complexity, and image statistics whether the de-quantization is necessary for stable training and good results.

Channel mixing. Dinh et al. (2016) who first introduced deep affine coupling-based INNs performed the same split in each coupling block. In the years since, various ideas have been presented about how to mix the variables in a learnable way between each coupling block. For instance, Tomczak and Welling (2016) suggest using so-called householder reflections to parametrize orthogonal rotation matrices, however with unreliable benefits for many applications. Kingma and Dhariwal (2018) instead manage to parametrize arbitrary invertible matrices, however leading to instabilities and singularities in practice, requiring training restarts and careful tuning.

Performing several direct comparison experiments, we find that for our applications, the best performance and stability can be reached by simply using random fixed orthogonal matrices to rotate the variables. For images, only the feature channels are rotated with the same matrix for each pixel. While it might seem preferable to learn the mixing of variables as with previous works, we find little to no quantitative improvement and a detrimental effect on the training stability. More research is needed to fully understand these effects, and whether learnable rotations lead to more powerful flows or if random fixed ones are equally useful. At least for the case of 'iterative Gaussianization' (a precursor to normalizing flows), work by Laparra et al. (2011) confirms that the use of random rotations does not make a qualitative difference in the expressiveness of the flow. Similarly, Draxler et al. (2020) find no difference between random and specially chosen rotation matrices with end-to-end training, and also reveals that a standard coupling block can already express a rotation up to $\pi/2$ on its own.

s-coefficients final activation. One problem with the affine coupling architecture is the exponential slope of the coupling block output w. r. t. the multiplicative shubnetwork s. This can very easily lead to training instability, even if great care is taken with the intialization and other hyperparameters.

The technique we use to prevent this is an extension to the one used in RealNVP (Dinh et al., 2016): Instead of directly using s as the multiplicative affine coefficient, we instead use

$$\tilde{s} = \alpha \tanh\left(\frac{s(u)}{\alpha}\right). \tag{3.13}$$

The hyper-parameter α is introduced by us and not present in previous works by Dinh et al. (2016) and others. It can be used to adjust the degree of regularization, interpolating between stability (small α) and expressivity (large α). For $\alpha \to 0$, the coupling block reverts to a NICE additive block, and for $\alpha \to \infty$ it is equal to having no regularization on the s-coefficients at all. Dividing by α in the argument of the tanh-function ensures the gradient of $\tilde{s}$ w. r. t. s is always 1 at the zero crossing on initialization. The regularization only becomes active for large outputs of s, while for $\tilde{s} \ll \alpha$, we have $\tilde{s} \approx s$.

Gradient clipping First used extensively for recurrent neural networks, gradient clipping describes the practice of limiting the gradients of networks parameters, or the norm thereof, to a certain value, truncating the gradients if they surpass this value (Pascanu et al., 2013). We find that gradient clipping is vital for the stability of training large cINNs at effective learning rates, as the model either quickly diverges without, or the learning rate has to be set prohibitively low.

Alexanderson and Henter (2020) further examine why this practice is necessary. They find that rare long-tail inputs lying far outside the Gaussian in latent space produce sporadic spikes in the gradients' magnitude. The authors also suggest replacing the latent distribution with a Student t-distribution, making gradient clipping unnecessary, and slightly speeding up convergence as a result. For GANs, gradient clipping does not seem to have the same stabilizing effect as for normalizing flows, as documented by Brock et al. (2019, p. 34)

Haar wavelet invertible downsampling The checkerboard-reordering of dimensions is intended to have the same use as pooling layers in feed-forward networks: reducing the

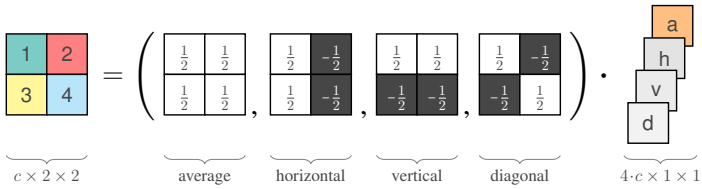


Figure 3.6.: Haar wavelet downsampling reduces spatial dimensions & separates lower frequencies (a) from high (h,v,d).

spatial resolution and instead adding additional feature channels. In the case of INNs, the increase in channels is directly connected to the decrease in resolution, as the total number of dimensions has to stay fixed.

However, while the invertible reordering of dimensions may satisfy these nominal requirements, it does not satisfy the actual purpose of pooling layers in feed-forward networks, that is to summarize or agglomerate features in a locally shift-invariant way: If a 2×2 patch of pixels with C channels is simply stacked into a single pixel with $4C$ channels as is typical, these additional feature channels will each have very similar contents, and all be affected by aliasing artifacts. Specifically, if the input is shifted by a single pixel, the contents of each feature channel are completely swapped around.

Instead, we therefore propose to process features using Haar-wavelets instead. Haar wavelets are a discrete wavelet transformation, decomposing each 2×2 pixel patch into four filtered channels. The use of Haar wavelets specifically is motivated by the fact that the first resulting output of the wavelet transform is equivalent to mean-pooling, and therefore has the same behaviour as pooling operations in feed forward networks. The other three channels contain spatial gradient information of the patch, horizontal, vertical and diagonal. As such, we expect these to contain lower-level detail information, with the commonalities by the first channel subtracted out. The Haar wavelet transform is illustrated in figure 3.6

The Haar-wavelet decomposition into mean and spatial gradients also works in harmony with the skips connection sometimes used to save memory and parameters in INNs, splitting off a portion of the features to directly become part of the latent code. In this case, it the resulting detail-channels can be split off, only further processing the mean-pooled portion of channels.

3.6. Experiment – Day to night

In the following section, we demonstrate cINNs on an image-to-image problem with diverse solutions. Image-to-image translation for natural images was first demonstrated with GAN-based models Isola et al. (2017), later refined by works such as Wang et al. (2018). It was also extended to the unpaired setting by Zhu et al. (2017a). However, these models are generally not able to produce diverse outputs. Several works attempt to prevent this mode collapse in image-to-image GANs through specialized architectures and regularization Zhu et al. (2017b); Park et al. (2019); Lee et al. (2018) with varying success. A hybrid approach between GAN and autoencoder is used in Ulyanov et al. (2018) for diversity. While these approaches do lead to visual diversity, there is currently no way to verify if they cover the entire distribution, compare models quantitatively, and the other issues discussed in section 2.3.

We apply a cINN with a conditioning network to this task of image-to-image translation to address these shortcomings of GAN-based approaches. The target image domain is the input/generated output x of the invertible part of the cINN, and the source image y is input into the conditioning network. The conditioning network transforms the original condition y into a pyramid of feature tensors $c^{(k)}$ that the INN uses at the different resolution levels k. For this, we simply use a standard ResNet-18 feed-forward network (He et al., 2016), denoted as φ. The activations of the ResNet just before each pooling layer constitute the levers of the feature pyramid. The invertible part consists of 8 coupling blocks in total, and five Haar-wavelet downsampling operations spaced in between. The subnetworks consist of three convolutions, with ReLU activations and batch normalization after the first two convolutions. We refer to 3.7 for a more detailed illustration of how the conditioning network and invertible part of the cINN are combined.

To demonstrate the models capabilities, we train on the popular day-to-night dataset (Laffont et al., 2014). It contains webcam images from approximately 100 different locations, taken at 10-20 times during the day and night each. This results in about 200 combinations of day-night pairs per location. The test set consists of 5 new locations unseen during training. As data augmentation, we randomly resize and crop the images to 128×128 pixels, in addition to the de-quantization noise. In our setting, we use the day-images as the condition y, and the night-images as the generated x. We train for $175\,000$ iterations using the Adam optimizer with a batch-size of 48, and leave the learning rate fixed at 0.001 throughout. These training parameters are comparable to those of standard feed-forward models.

Despite the relatively small training set, we see little signs of overfitting, and the model generalizes well to the test set of new locations. Previously, Sun et al. (2019) also found low overfitting and good generalization on small training sets using INNs. Several samples by the model are shown in Fig. 3.8. The cINN correctly recognizes populated regions and generates lights there, as well as freely synthesizing diverse cloud patterns and weather conditions. At the same time, the edges and structures (e.g. mountains) are consistently and precisely aligned with the conditioning image. The features learned by the conditioning network are visualized in Fig. 3.9. Hereby, linearly independent features were extracted via PCA. The figure shows one example of a feature from each of the first three levels of the pyramid, clearly demonstrating how each level contains features relevant for the corresponding scale of generation, e. g. edges and texture at the high-resolution stages, and semantic information at the low resolution stages.

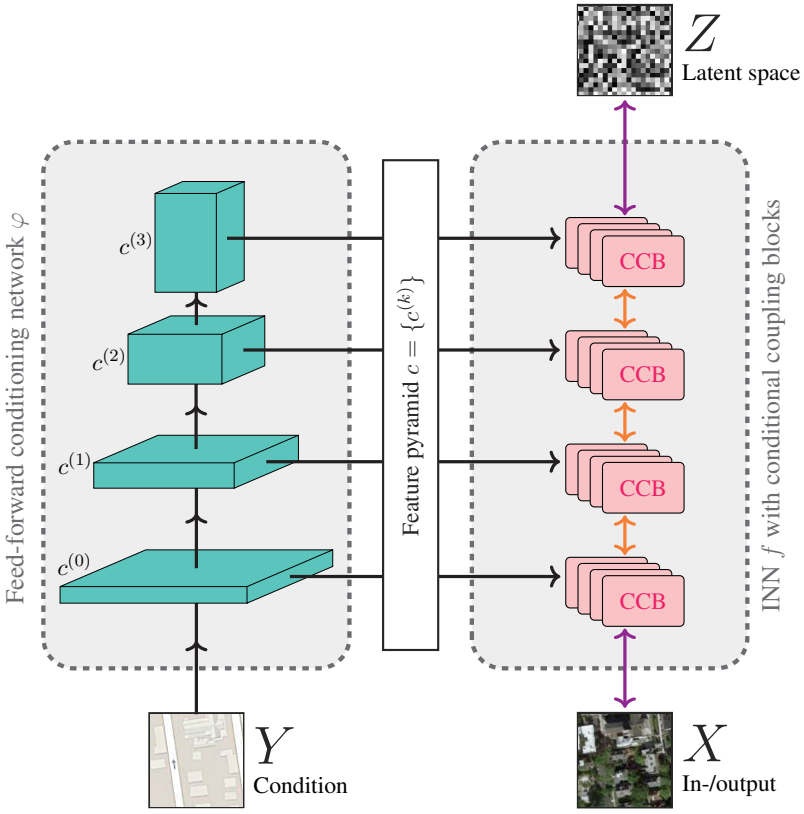


Figure 3.7.: Illustration of the cINN. It consists of a feed-forward conditioning network (*left half*), and an invertible part (*right half*).
Black arrows: connections always in the same direction.
Green boxes: extracted feature maps $c^{(k)}$.
Purple arrows: invertible connections, depending on training/testing.
Orange arrows: invertible wavelet downsampling.
Pink blocks: conditional coupling blocks (CCBs).

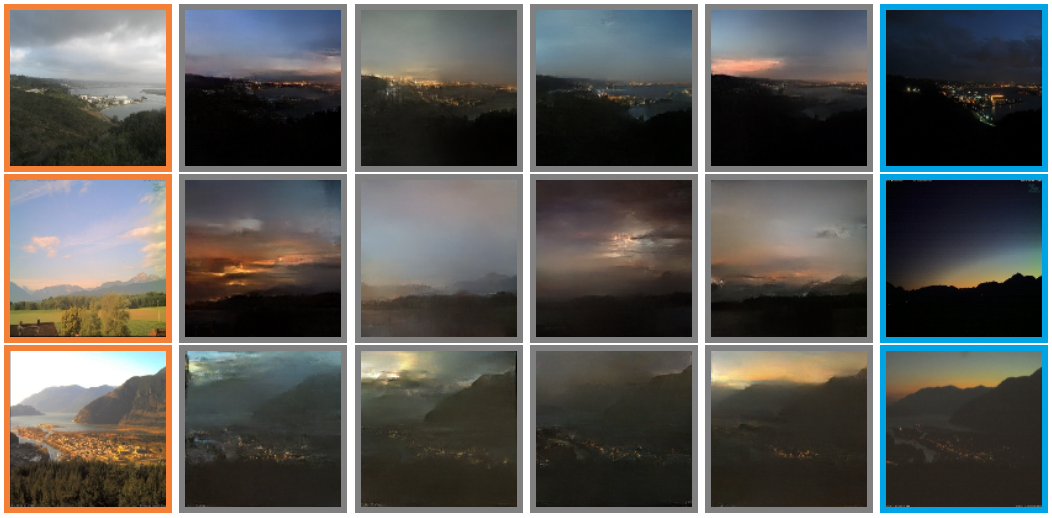


Figure 3.8.: Examples of **conditions** y (left), three **generated samples** (middle), and the **original image** x (right).

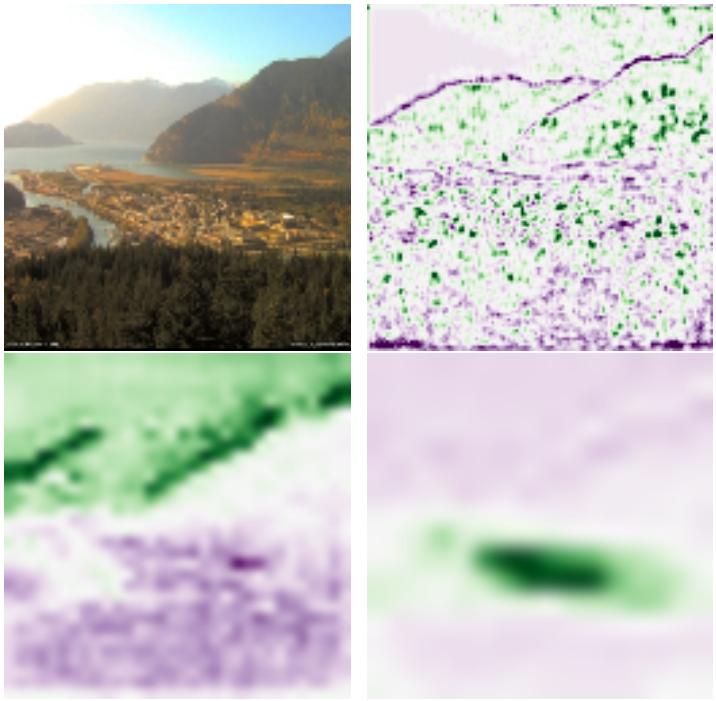


Figure 3.9.: Conditioning image (top left), and extracted features from different levels of the pyramid. From left to right, top to bottom: 1st level, precise edges and texture; 2nd level, foreground/background; 3rd level, populated area.

3.7. Experiment – Diverse image colorization

For a more challenging task, we turn to colorization of natural images. State-of-the-art regression models for colorization produce visually near-perfect images (Iizuka et al., 2016), but do not account for the ambiguity inherent in this inverse problem. To address this, models would ideally define a conditional distribution of plausible color images for a given grayscale input, instead of just returning a single "best" solution. Popular existing approaches for diverse colorization predict per-pixel color histograms from a U-Net (Zhang et al., 2016) or from hypercolumns of an adapted VGG network (Larsson et al., 2016). However, sampling from these local histograms independently can not lead to a spatially consistent colorization, requiring additional heuristic post-processing steps to avoid artefacts.

In terms of generative models, both VAEs (Deshpande et al., 2017) and cGANs (Isola et al., 2017; Cao et al., 2017) have been proposed for the task. However, their solutions in no way reach the quality of the regression-based models, and cGANs in particular often lack diversity. Approaches using auto-regressive models (Guadarrama et al., 2017) or CRFs (Royer et al., 2017) also exist, but these methods are computationally very expensive and often unable to scale to realistic image sizes. Ulyanov et al. (2018) use a custom autoencoder-like adversarial architecture to ensure diversity, producing the some of the best diverse colorization results so far to our knowledge. However, their experiments are limited to a data set with only cars, and just three latent dimensions, leading to global, but no local diversity. For images with multiple or more complex objects, three dimensions are not expected to suffice in covering the solution space.

We therefore argue for the use of cINNs for this task: the conditioning network can perform the task of recognizing the semantic content, and can be based on existing colorization solutions, while the invertible part of the cINN uses this information to produce images with covering the full range of locally and globally diverse solutions. The invertibility also allows for intuitive and interactive editing and manipulation, as we demonstrate in the following.

Fro the problem setup, we represent the images in Lab color space. The commonly used RGB color space has been found to be unsuitable for the task by previous works: not only are distances in RGB-space very inconsistent with perceived color difference, but the RGB representation is also over-complete for the task (the grayscale image is contained in RGB), introducing an unnecessary additional difficulty. Instead, the Lab color space represents images through a grayscale ('luminance') channel L, the condition, and two channels a and b containing the color information, the generated output. We train on the ImageNet dataset (Russakovsky et al., 2015) for the greatest possible variance in objects and image types. As the color channels do not require as much resolution as the luminance channel, we condition on 256×256 pixel grayscale images, but generate 64×64 pixel color information. This is in accordance with the majority of existing colorization methods.

For the conditioning network φ, we start with the same VGG-like architecture as Zhang et al. (2016), and pre-train it as a feed-forward colorization model using their code. We then cut off the network before the second-to-last convolution, resulting in 256 feature maps of size 64×64 from the grayscale image L. To form the feature pyramid, we then extend this by a series of strided convolutions, ReLU activations, and batch normalization layers specific to each resolution level, along with a final convolution specific to each coupling block, i. e. each coupling block k receives its own specialized conditioning (small hexagons in figure 3.11). The ablation study in figure 3.17 confirms that the conditioning network is absolutely necessary to capture semantic information: directly conditioning the invertible part on the grayscale image, the cINN simply produces random colors ignoring image content.

The invertible part of the model consists of 22 convolutional CCBs, with three downsampling steps in between. After that, the features are flattened, followed by 8 fully connected CCBs. To conserve memory and computation, we adopt the latent skip connections first presented by Dinh et al. (2014): after each wavelet downsampling step, we split off some of the detail-channels. These are not processed any further, but fed into a skip connection and concatenated directly onto the latent output vector. The reasoning behind this is the following: The high resolution stages have a smaller receptive field and less expressive power, so the channels split off early correspond to local structures and noise. More global information is passed on to the lower resolution sections of the INN and processed further. Overall, the generative performance of the network is not meaningfully impacted, while reducing the computational cost and memory requirements. The effect of sampling the different parts of z individually while keeping the others fixed is shown in figure 3.18. As we can see, the local partitions of latent space encode image noise or the color of details, the intermediate partition contains the color of the objects themselves, and the most global and abstract partition encodes global attributes such as overall saturation or tints. The architecture is illustrated in figure 3.11.

For training, we again use the Adam optimizer and a batch-size of 48. We train for roughly $250\,000$ iterations, stopping when the validation loss increases, indicating overfitting. The initial learning rate is 10^{-3}, decreasing by a factor of 10 at $100\,000$ and $200\,000$

Figure 3.10.: Diverse colorized images which our network generated for the same grayscale image. One of them contains the original ground truth colors, but which? Solution at the bottom of next page.

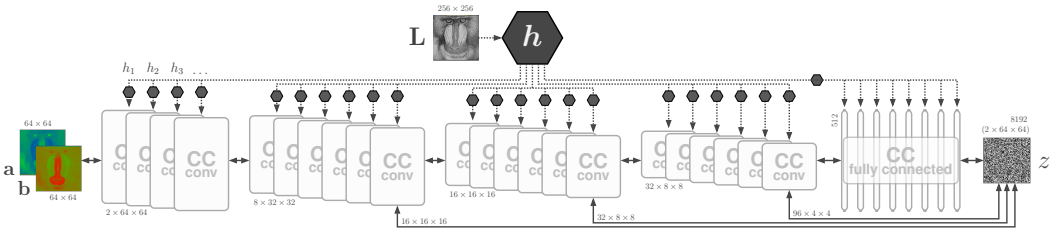


Figure 3.11.: cINN model for diverse colorization. The conditioning network h consists of a truncated VGG Simonyan and Zisserman (2014) pretrained to colorize ImageNet, with separate convolutional heads $h_1, h_2, h_3, \ldots$ tailoring the extracted features to each individual conditional coupling block (CC). After each group of coupling blocks, we apply Haar wavelet downsampling (figure 3.6) to reduce the spatial dimensions and, where indicated by arrows, split off parts of the latent code z early.

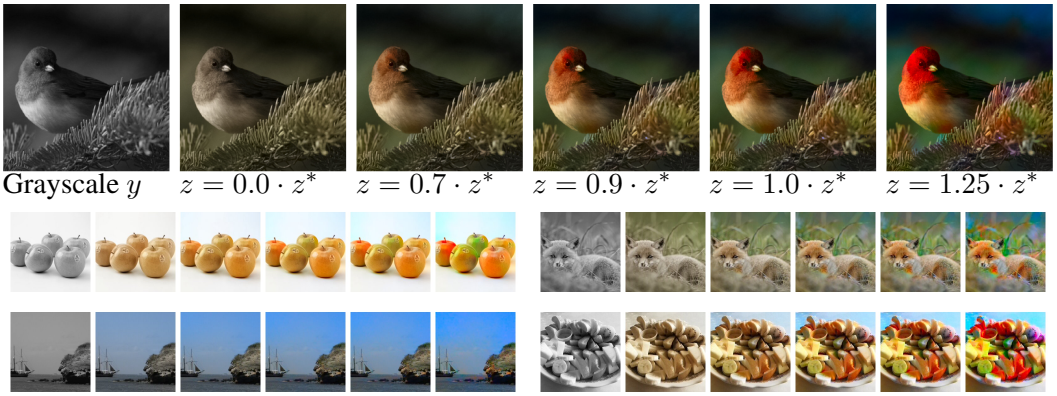


Figure 3.12.: Effects of linearly scaling the latent code z while keeping the condition fixed. Vector z^* is "typical" in the sense that $\|z^*\|^2 = \mathbb{E}\big[\|z\|^2\big]$, and results in natural colors. As we move closer to the center of the latent space ($\|z\| < \|z^*\|$), regions with ambiguous colors become desaturated, while less ambiguous regions (e.g. sky, vegetation) revert to their prototypical colors. In the opposite direction ($\|z\| > \|z^*\|$), colors are enhanced to the point of oversaturation.

iterations. At inference time, we use joint bilateral upsampling Kopf et al. (2007) to match the resolution of the generated color channels a, b to that of the luminance channel L. This produces visually slightly more pleasing edges than bicubic upsampling, but has little to no impact on the results. It was not used in the quantitative results table, to ensure an unbiased comparison.

Latent space interpolations and color transfer are shown in figures 3.12 and 3.13, with more experiments in the appendix. Various diverse samples and failure cases along with a qualitative comparison to other generative models are shown in figures 3.14 to 3.16.

The work Ardizzone et al. (2020a) contains a detailed quantitative comparison along with these qualitative experiments. These confirm the observations from the figures. In summary, we find that the cINN produces the best sample diversity by a large margin while also improving over the other generative methods in terms of image quality. The only superior method in terms of quality is the feed-forward regression model, which has no diversity by construction and only produces muted colors that are 'safe' in terms of $L2$-loss.

The middle image in the bottom row of figure 3.10 shows the original color image.

Figure 3.13.: For color transfer, we first compute the latent vectors z for different color images (L, a, b) *(top row)*. We then send the same z vectors through the inverse network with a new grayscale condition L^* *(far left)* to produce a transfer of colorization a^*, b^* *(bottom row)*. Differences between reference and output color (e.g. bright pink rose) can arise from mismatches between the reference colors a, b and the intensity prescribed by the new condition L^*.

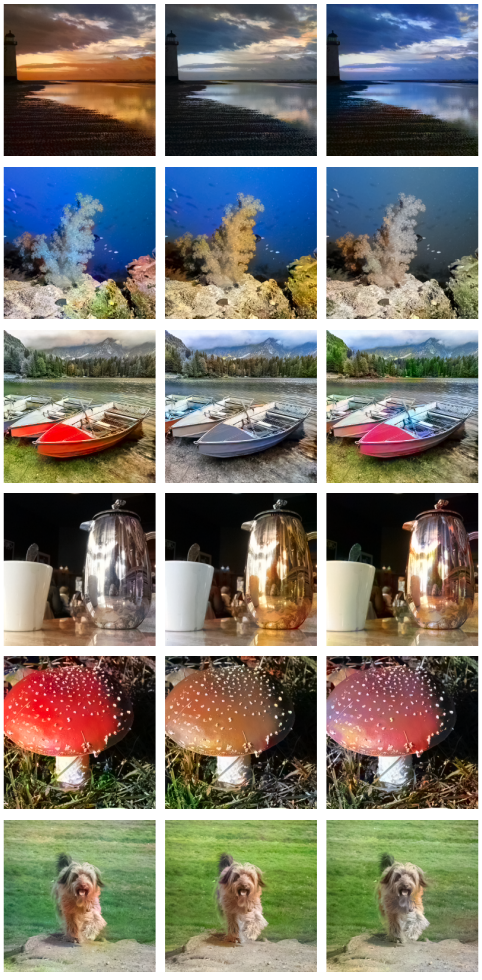


Figure 3.14.: Diverse colorizations produced by our cINN.

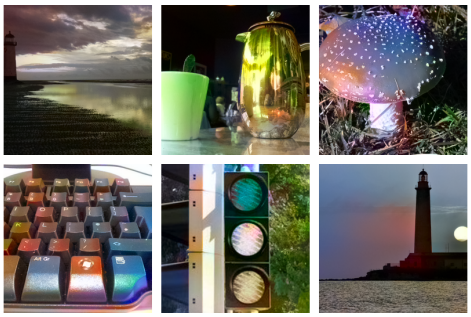


Figure 3.15.: Failure cases of our method. *Top:* Sampling outliers. *Bottom:* cINN did not recognize an object's semantic class or connectivity.

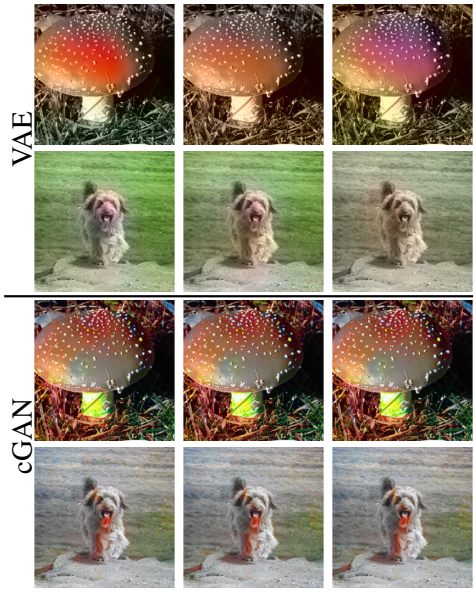


Figure 3.16.: Other methods have lower diversity or quality, and suffer from inconsistencies in objects, or color blurriness and bleeding (cf. figure 3.14, bottom).

Figure 3.17.: In an ablation study, we train a cINN using the grayscale image directly as conditional input, without a conditioning network φ. The resulting colorized images largely ignore semantic content which leads to exaggerated diversity. More ablations are found in the appendix.

Figure 3.18.: Sampling the resolutions levels. First row: $z^{(0)}$, introduces global hue and saturation shifts. Second row: $z^{(1)}$, determines the colors of the object. Third, fourth row: $z^{(2)}$, $z^{(3)}$, imperceptible changes on pixel-level.

3.8. cINNs Applied in Science

Beyond the computer-vision focused tasks from the previous sections, cINNs have successfully been used in a number of scientific and engineering applications in the time since their publication (Ardizzone et al., 2019b). Their properties make them uniquely suited for scientific tasks, including the flexibility in input modality afforded by the conditioning network, the stable, repeatable, and principled training procedure, and the convenience of accessing the full posterior through both sampling and density estimation.

The following section therefore briefly summarizes a number of works from different scientific fields. Note that the contributions to these works in the context of the thesis were minor, consisting of e. g. data pre-processing routines, assistance or advice in model setup, hyper-parameter tuning and visualization. They are showcased here mainly to demonstrate the effectiveness and advantages of using cINNs in the context of scientific research, each subsection highlighting a different strength or advantage of the cINN in practice.

3.8.1. Astrophysics

Astrophysics is a field for which cINNs can be especially useful. Electromagnetic waves (light, radio waves, etc.) are the only signal that can be observed for most objects outside the solar system. All other properties and structure of objects have to be reconstructed from these signals alone, leading to many ambiguities and sources of uncertainty, and a large demand for advanced data analysis methods. As a result, several publications in astrophysics have already made successful use of cINNs.

Kang et al. (2022) use cINNs and a synthetic dataset to predict the properties of star-forming gas clouds (such as age, mass, etc.) from the intensity of observable spectral lines. Notably, some of the parameters predicted by the cINN are discrete ordinal quantities (such as the number of star clusters that form within a cloud). This is achieved by de-quantizing the discrete values with Gaussian noise, and re-quantizing the samples at test time. Garcia Satorras et al. (2021), among others, provide some additional methodological insight and justification for this technique. Figure 3.19 shows some examples of highly ambiguous cases exhibiting two distinct solutions. Being able to access the full posterior allows for pinpointing the source of the degeneracy to a single uncertain parameter, informing which types of measurements would be able to resolve the ambiguity.

Ksoll et al. (2020) address a similar problem, but predicting the properties of individual stars. The authors again find certain types of stars with ambiguous multi-modal posteriors. They demonstrate how existing knowledge from other sources allows ruling out certain peaks in the posterior, thus collapsing the degeneracy down to one well-defined solution. This is only afforded by the full flexible posterior provided by the cINN, not e. g. by parameter-wise scalar uncertainty estimates.

Haldemann et al. (2022) predict the internal structure and composition of exoplanets from external measurements, a highly ambiguous inverse problem. As existing Markov Chain Monte-Carlo (MCMC) methods are already able to solve the problem, the focus is placed on the computational speed-up possible through cINNs. The authors verify that the cINN results are almost equally accurate as the MCMC posteriors, while being multiple orders of magnitude faster. This computational efficiency is especially helpful in astronomy, as huge catalogues of objects has to be examined, and high computational costs can tangibly hinder research progress.

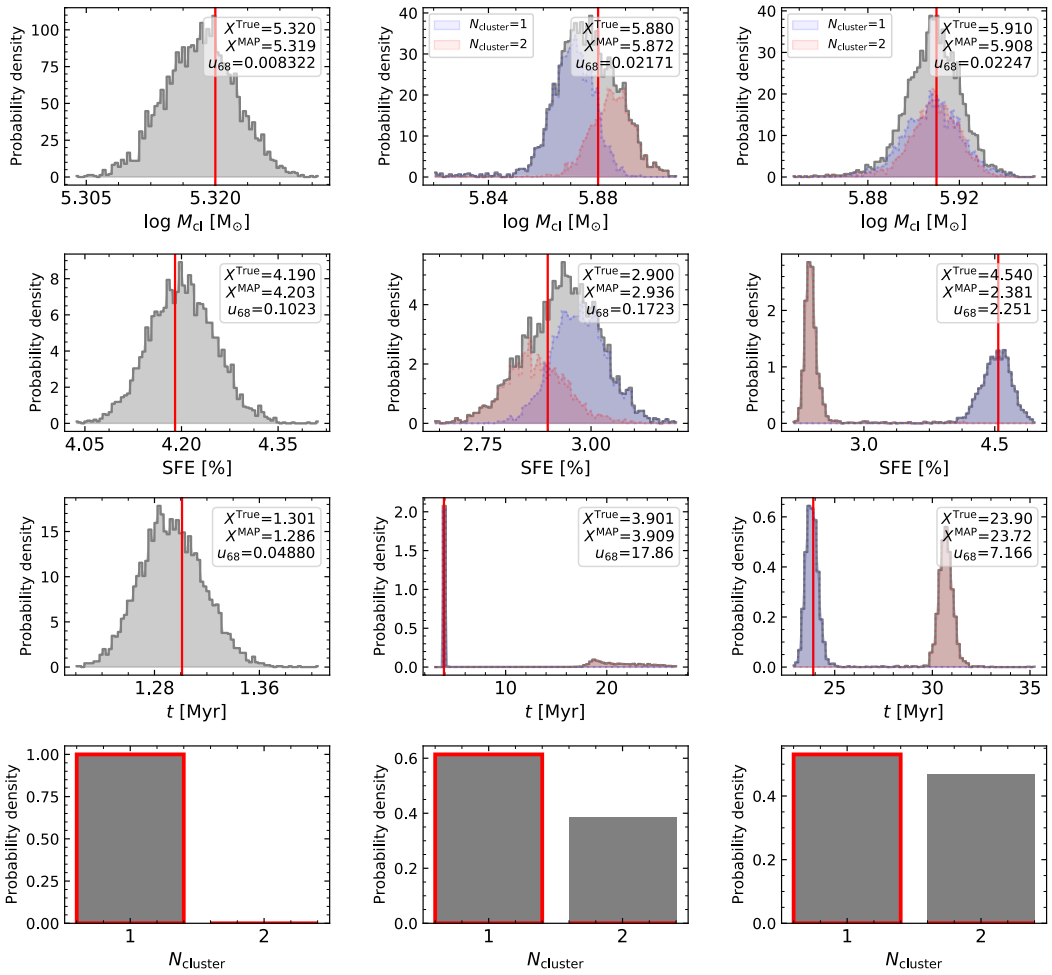


Figure 3.19.: Figure adapted from Kang et al. (2022), characterizing star-forming gas clouds. Showing the posterior marginalized over a selection of four parameters of interest (rows) for three different sets of observations (columns). M_{cl}: mass of cloud, SFE: star formation efficiency, t: age, N_{cluster}: number of formed star clusters (discrete). The produced posteriors exhibit varied multimodal shapes, including configurations where one peak is extremely sharp and the other spread out wide (center column, third row). For the second and third columns, the posterior is further conditioned by only selecting samples with $N_{\mathrm{cluster}} = 1$ (blue) or $N_{\mathrm{cluster}} = 2$ (red), revealing that the multi-modal degeneracy in the other parameters is caused by the uncertainty in N_{cluster}.

3.8.2. Ambiguous computed tomography registration

Another field requiring reliable and accurate uncertainty predictions is medical imaging, or medicine in general. The following work by Trofimova et al. (2020) addresses a common task related to image registration in surgery: before a surgery, a detailed 3D computed tomography (CT) scan of a patient is performed. During the surgery itself, instruments, implants, etc. can be guided by taking 2D X-ray images on the fly. The task of registration is to find the precise location and orientation of the 2D X-ray image relative to the complete 3D scan made beforehand, allowing the surgeons to better judge the location of instruments in the body.

As Trofimova et al. (2020) note, existing solutions attempt to find some single best fit, not taking into account the ambiguities caused by symmetries of the human body. The authors therefore propose using a cINN to perform the task in an uncertainty- and ambiguity-aware way. In this case, the conditioning network performs the most complex parts of the task, extracting joint features from the 3D and 2D scans using a refined and purpose-made feed-forward network architecture. The invertible part of the cINN only needs to generate simple pose vectors with five degrees of freedom conditioned on these features. Figure 3.20 shows some examples of the results, showcasing how the cINN is able to differentiate between ambiguous and unambiguous cases. This example demonstrates the flexibility and ease-of-use of the cINN setup, able to quickly adapt existing task-specific feed-forward architectures in the form of the conditioning network.

3.8.3. Photo-acoustic imaging

Nölke et al. (2021) apply cINNs to multispectral photo-acoustic imaging (PAI), a new emerging technique in medical imaging. PAI in general uses a setup similar to ultrasound imaging. However, instead of sending out ultrasound waves, the recording head sends out laser pulses. These pulses slightly heat the tissue in the body according to its absorption and scattering properties, producing minute shock waves inside the body, known as the *photoacoustic effect*. These shock waves are then recorded as sound by the recording head. Complex numerical algorithms are used to reconstruct a 3D image of the absorbance inside the body, revealing structures invisible to all other imaging techniques. Zackrisson et al. (2014) fittingly describe the method as 'light in and sound out' in their seminal work on the method (as opposed to 'sound in and sound out' for classical ultrasound imaging).

Multispectral PAI specifically uses laser pulses with different wavelengths, allowing for reconstruction of tissue parameters inside the body from the spectral response of the observed tissue, in this case specifically blood oxygenation.

Nölke et al. (2021) apply a cINN to simulated multispectral PAI measurements, with the task of recovering the blood oxygenation of a blood vessel surrounded by tissue. The method is applied to different simulated scenarios, illustrated in figure 3.21: With multiple blood vessels in the same volume and only a single detector, the problem is highly ambiguous, and the cINN is unable to recover the properties of the target blood vessel. The authors then show how using three detectors instead of one (or moving one detector to three different positions and combining the measurements) can resolve this ambiguity once again.

This case serves to demonstrate how the cINN can serve as a method for experimental design, allowing researchers to examine which experimental setups will produce degener-

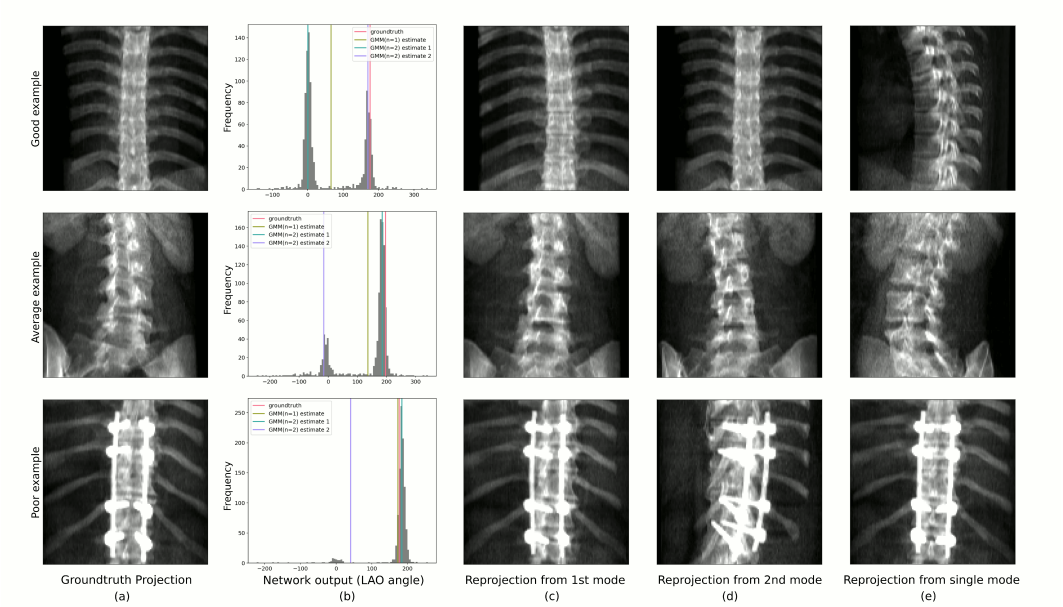


Figure 3.20.: Figure adapted from Trofimova et al. (2020) concerning 3D registration of 2D X-ray images. The rows show examples with varying degrees of ambiguity. The left most image is the input. The second column shows the posterior over one of the pose angles. Indicated in red is the ground truth, in yellow the mean of the posterior (the solution that would result from a model trained with $L2$-regression), in green and blue the two solutions indicated by the two peaks of the posterior (fitted Gaussian mixture model). The remaining three columns show the simulated X-ray image resulting from each of the predicted solutions. Indicated by the first column of re-projected images, the higher posterior peak always represents a plausible and sensible solution. The second column uses the lower posterior peak, only producing a plausible solution for the first symmetric example (the peak is correspondingly less pronounced in the posterior for the lower two examples). The third column shows that the unimodal $L2$-solution is only valid for the last asymmetrical case.

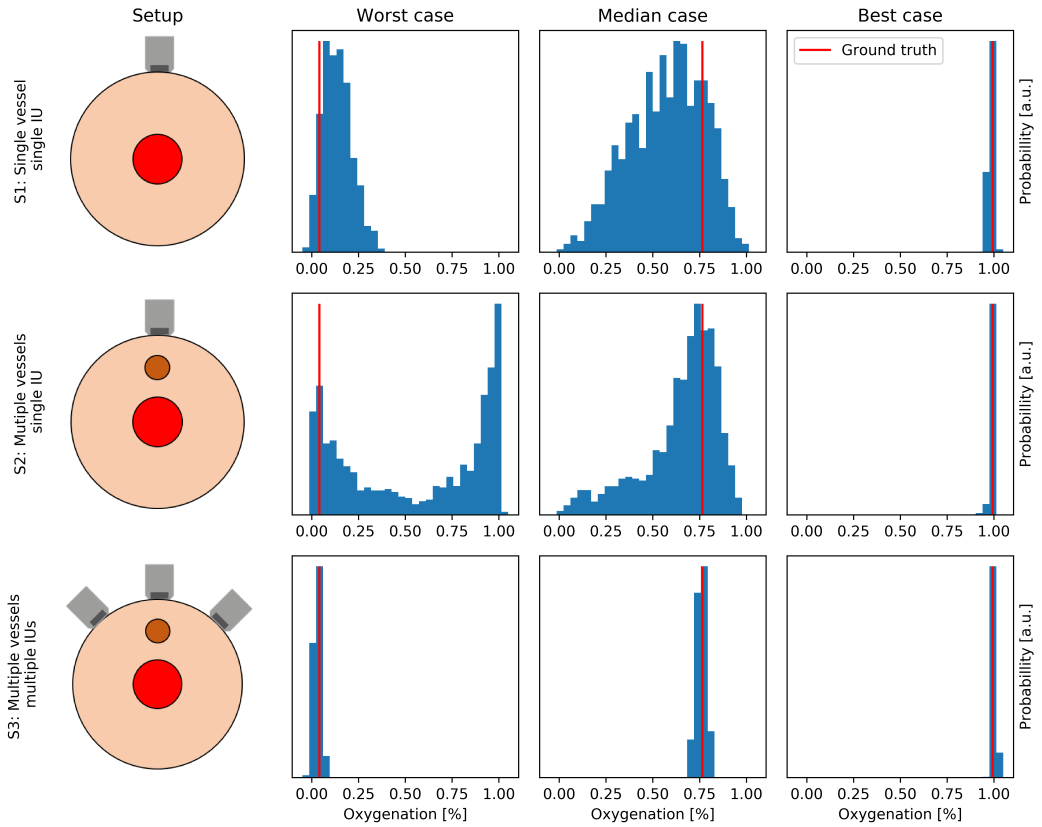


Figure 3.21.: Figure adapted from Nölke et al. (2021). For three different experimental photoacoustics setups (illustration first column), three representative posteriors over the blood oxygenation are shown. Introducing additional detectors (bottom row) allows resolving the inherent ambiguity in the problem.

ate results *in silico*, informing decisions on how to implement the real measurement setup beforehand.

3.8.4. Characterization of wind turbine blades

The recent publication by Noever-Castelos et al. (2021) concerns the characterization of wind turbine blades. The manufacturing process of such blades introduces various imprecisions and variations, making each produced blade unique, so that the safety and stability must be assured for each blade individually. Naturally, testing it under extreme load including material failure can only be performed *in silico* (the blade can not be deployed if it is destroyed in the testing procedure). While simulation methods are sufficiently advanced and accurate to perform these tests, they require a detailed virtual model of each individual blade, including the material properties and physical parameters of each component, called a 'digital twin' (see figure 3.22 for a cross-section example). This includes many quantities that can not be measured directly, such as the dimensions of internal components, material densities and elasticity.

To produce the digital twin, an advanced pipeline is used in practice, consisting of non-destructive physical measurements followed by two processing steps: the three dimensional shape and profile of the turbine blade are first measured by a laser scanner. The blade

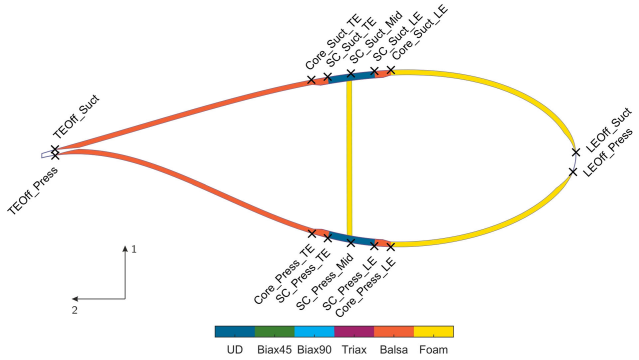


Figure 3.22.: Figure from Noever-Castelos et al. (2021): Profile of a wind turbine blade, sowing different components consisting of different materials (colors). The construction and material choice changes across the length of the blade.

is then attached to an apparatus that induces oscillations with various frequencies and in various directions and orientations. The amplitude and phase of the induced oscillations at each position along the blade is then recorded across a large range of frequencies. The first processing step consists of a sophisticated algorithm incorporating known physics principles, used to compute several properties at each point along the blade from the vibration behaviour, including shear- and elasticity-modulus, moments of inertia, mass distribution over the length, etc. The second processing step, and the one addressed with a cINN by Noever-Castelos et al. (2021), has the goal of reconstructing the 'digital twin' of the blade from this intermediate representation. As opposed to the first step, this second step can be more ambiguous and ill-posed: the properties and shapes of the various internal and external building materials must be resolved even though the intermediate representation only describes the blade as a whole.

The application of a cINN to this problem is shown to come with several advantages in practice: a large dataset can be easily produced by simulating the physical measurements for randomly generated turbine blades in order to train the cINN. Reliable uncertainty estimates and confidence intervals, as those produced by the cINN, are of course especially important for applications such as these, where the worst-case scenario must be considered, and not just the most likely 'best-guess' solution.

An especially salient example of the cINN's capabilities is shown in figure 3.23: while most of the parameters can be recovered quite accurately, indicated by the matching small uncertainty estimates and tight spread around the true value, two parameters in particular, concerning the densities of two different building materials, are highly degenerate, with the uncertainty interval almost matching the prior distribution. Figure 3.24, shows the generated posterior over these two material densities in 2D, which reveals that a tight linear correlation exists between the two. What appeared like two degenerate parameters is in fact only one dimension of degeneracy along an off-axis direction, while the orthogonal direction can be predicted very precisely.

Comparing to the construction schematic reveals that the parameters in question are the densities of two thin sheets of material laminated directly together. Because they overlap almost completely, only their overall mass can be determined (orthogonal to the spread in figure 3.24), but not how the mass is divided up between them, causing the observed degeneracy. While it seems evident in retrospect, neither the existence nor the reason for this 1-dimensional degeneracy affecting two parameters was known beforehand.

The work thereby demonstrates not only the importance of cINNs for determining reliable uncertainties for safety-relevant applications, but also how it can identify and help explain hidden degeneracies in the application.

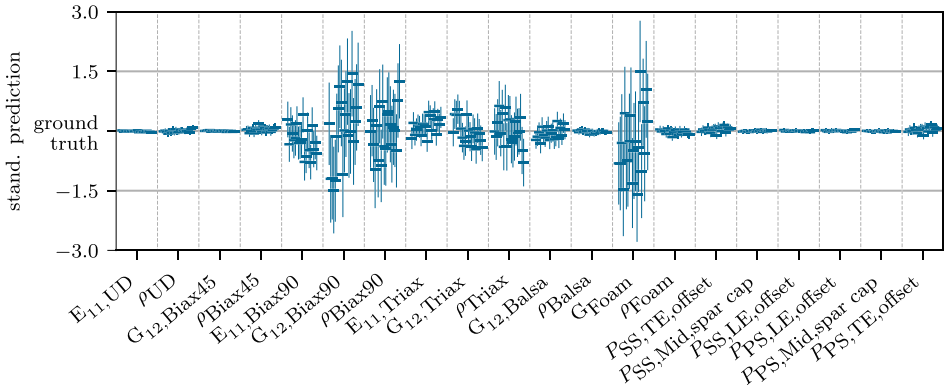


Figure 3.23.: Figure from Noever-Castelos et al. (2021). Each section on the X-axis indicates a different estimated parameter. The points and error bars correspond to posteriors of 20 test-set examples, indicating mean and variance. Each posterior is shifted so that the ground-truth value is at 0, and the vertical axis units are scaled to the width of the prior (a posterior width of 2 indicates that the parameter is unrecoverable). The spread around the ground-truth value corresponds correctly to the estimated uncertainties.

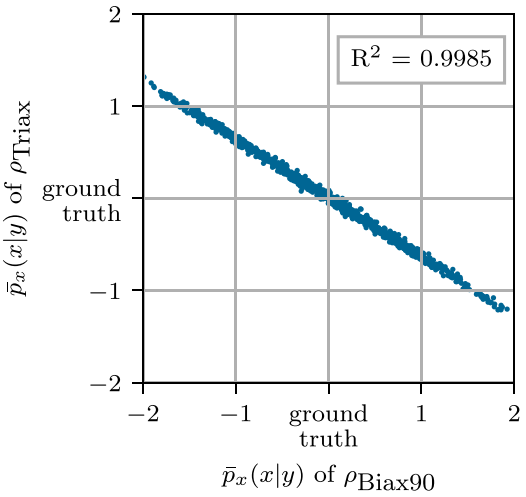


Figure 3.24.: Figure from Noever-Castelos et al. (2021). Posterior in 2D over two of the most uncertain parameters in figure 3.23. The tightly correlated posterior indicates that the two unrecoverable parameters are caused by a single 1D degeneracy (along the spread), while a certain linear combination of the parameters can be precisely predicted (orthogonal to the spread).

3.8.5. High-energy particle physics

To discover new and previously unknown effects and mechanisms, particle physics experimentalists are interested in collision events with increasingly high energies, usually termed 'hard scattering' events in context. However, the particles resulting from such events almost instantly decay into many lower-energy 'soft' particles which are observed by highly complex detectors. The detector itself introduces further measurement noise and uncertainty. The common approach for examining new hypotheses or models of unknown hard scattering events is to perform Monte-Carlo simulations of the measurements that would result from the hypothesis, and statistically comparing the simulation outcomes to the actually observed measurements.

As Bellagente et al. (2020) argue, in many cases it may be preferable to be able to compare directly in the space of 'hard scattering' where the events of interest occur, not in the space of resulting measurements, leading to simpler, computationally cheaper, and more direct and intuitive evaluation of new physics models. Instead of comparing simulated measurements to real ones, the goal is to invert the measurement process (so-called *unfolding*) and then comparing directly to the hypothesis in question. In machine learning terms, a domain transfer occurs in both directions, where only the former can be explicitly modeled (hard scattering to measurement), while the 'unfolding' (measurement to hard scattering) has remained challenging.

The highly probabilistic nature of the observation and measurement process naturally requires a probabilistic way of expressing the inverse, hence Bellagente et al. (2020) choose cINNs for the task. They demonstrate promising results in a number of different settings: the cINN produces highly accurate and computationally cheap unfolding results and compares favourably to existing GAN-based approaches for the task.

The application demonstrates the use of cINNs as a tool for probabilistic domain transfer, in a similar way to transforming between the day and night domains in section 3.6. Reliable and accurate distributions are arguably even more critical here than for computer vision problems, seeing how extensively statistical methods and hypothesis tests are used in particle physics.

3.8.6. Bayes-Flow

In a number of settings, using psychology as one of the main examples, the observed conditioning input is an unordered set with varying size containing multiple observations of the same system. For instance, we could imagine survey results from different groups of people, from which the parameters of an underlying psychological model should be determined for each group. Writing the result from each participant k as $y^{(k)}$, we want to condition the generated distributions on the set of all observations, $\{y^{(k)}\}_{k=1}^{K}$. The standard cINN restricts the conditioning input to a fixed size, i. e. a fixed number of observations K, which is simply not realistic when surveying different groups etc.

The goal of the work by Radev et al. (2020) is therefore to model a conditional density $q(x \mid \{y^{(k)}\}_{k=1}^{K})$ where K can change during training and test-time. The model is termed 'Bayes-Flow' as it is targeted to Bayesian practitioners in psychology and related fields.

To enable this, Radev et al. (2020) replace the usual conditioning network with a permutation invariant *summary-network*, i. e. a feed-forward network that extracts features

from a variable-sized set of inputs in a permutation invariant way, $\varphi(\{y^{(k)}\})$. In practice, this is implemented with a small feed-forward network ψ that is first applied to all members of the set independently, the resulting feature vectors are then summed, and finally this sum is further processed by a feed-forward network ρ:

$$\varphi(\{y^{(k)}\}) = \rho\left(\sum_k \psi\big(y^{(k)}\big)\right) \tag{3.14}$$

This technique was first introduced by Zaheer et al. (2017) as 'deep sets'. Sannai et al. (2019, Theorem 2.1) show that such a mechanism is a universal approximator for all permutation-invariant functions, and it has previously most prominently been used in neural processes, a variational generative model conditioned on sets (Garnelo et al., 2018b,a; Kim et al., 2018).

Radev et al. (2020) verify and demonstrate the capabilities of the Bayes-Flow on several relevant models from psychology and statistics, such as the Ricker model and the Lévy-Flight model (see publication). It was later also successfully applied to epidemiology (Radev et al., 2021), determining the underlying epidemiological parameters of the COVID-19 pandemic in Germany. Schmitt et al. (2021) further extend the method to be able to detect model mis-specification, i. e. whether the observations at test time actually stem from the same process that was used to produce the training data, by performing out-of-distribution detection on the output of the summary network.

Bayes-Flow demonstrates how the cINN architectures can be adapted easily and effectively and extended to new types of conditional modelling by replacing the conditioning network with new types of feature extractors.

3.9. Conclusion

This chapter demonstrated how modified INNs (cINNs) trained as normalizing flows, can serve as a powerful and flexible tool for conditional generative modelling. Their use in a series of successful academic publications in other fields shows that their advantages go beyond academic interest in machine learning.

The main strength of cINNs is in uncertainty quantification and diverse solutions. Beyond this, the non-parametric, multi-dimensional posteriors provide some degree of explainability, as demonstrated by the example in section 3.8.4, although it is not possible to directly visualize the hidden features and inner processes of the cINNs.

However, we highlight that the basic principle in using the cINNs to solve problems is shared with all other conditional generative approaches. The 'only' contribution are the improved capabilities of cINNs to model conditional distributions. In the following two chapters, we will instead use invertible generative models to address problems in two fundamentally different ways to existing approaches, yielding further advantages.

4 INNs for Constrained Inverse Problems

In the previous chapter, specifically section 3.8, we can recognize a trend: Many of the applications are so-called inverse problems, that follow a similar scheme. There is some underlying system whose parameters or state x we wish to recover, but which cannot be measured directly. Instead, we only have access to some observations y arising from that system. For many of these cases, including those from section 3.8, scientists have developed sophisticated theories on how the measurable quantities y arise from the hidden system parameters x. We will call such mappings the *forward process*. For this forward process, numeric simulations are often readily available, accurate, and computationally fast. However, the *inverse* process is required to infer the hidden states of a system from measurements. While the forward process is well-defined and can be modeled accurately, the inverse is often both intractable and ill-posed, since crucial information is lost in the forward process. We consider such problems specifically in the following section, loosely following the publication Ardizzone et al. (2019a).

Modeling the posterior of an inverse process taking the ambiguities into account is a classical statistical task that can in principle be solved by Bayesian methods. However, 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 (Robert and Casella, 2004; Gamerman and Lopes, 2006). If a model $y = s(x)$ for the forward process is available, approximate Bayesian computation (ABC) is often preferred, which embeds the forward model in a rejection sampling scheme for the posterior $p(x|y)$ (Sunnåker et al., 2013; Lintusaari et al., 2017; Wilkinson, 2013). In the previous chapter, we demonstrated that generative models, especially invertible generative models, can address the task in a much more efficient way. However, we note that they do not make use of the special structure of inverse problems that exhibit a well-posed and simple forward process.

In the chapter, we consider whether the asymmetry in difficulty beween forward and inverse process can be exploited in some way. If it is easier for scientists to model the forward rather than the inverse problem, it stands to argue that the same is true vor deep learning models. This is precisely the approach we will follow in this chapter: We can train an invertible model on the well-understood forward process $x \rightarrow y$ and get the inverse $y \rightarrow x$ for free by running it backwards at test time. To counteract the inherent information loss of the forward process, we use additional latent noise variables z, which capture the information about x that is *not* contained in y. Thus, the INN learns to associate hidden parameter values x with unique pairs $[y, z]$ of measurements and latent variables. Forward training optimizes the mapping $[y, z] = f(x)$ and implicitly determines its inverse $x = f^{-1}(y, z) = g(y, z)$. Additionally, we make sure that the density $p(z)$ of the latent variables is shaped as a Gaussian distribution, as with standard normalizing flow models. Figure 4.1 shows a standard deep learning prediction model on the left (able to be replaced

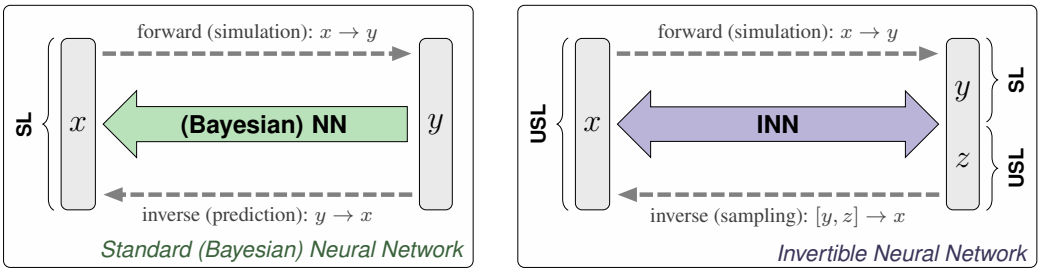


Figure 4.1.: Abstract comparison of standard approach *(left)* and ours *(right)*. The standard direct approach requires a discriminative, supervised loss (**SL**) term between predicted and true x, causing problems when $y \to x$ is ambiguous. Our network uses a supervised loss only for the well-defined forward process $x \to y$. Generated x are required to follow the prior $p(x)$ by an unsupervised loss (**USL**), while the latent variables z are made to follow a Gaussian distribution, also by an unsupervised loss. See details in Section 4.1.2.

by a Bayesian neural network to obtain uncertainty estimates), and the proposed INN-based method on the right.

This modification of the normalizing flow model is specific to the types of inverse problems described above and makes unique use of the INN's bidirectional nature. Despite their automatic invertibility, INNs have rarely made good use of bidirectional training, which has been shown to be very beneficial in generative adversarial nets and auto-encoders (Zhu et al., 2017a; Dumoulin et al., 2016; Donahue et al., 2017; Teng et al., 2018). In fact, optimization for cycle consistency forces such models to converge to invertible architectures, making fully invertible networks a natural choice.

4.1. Method

4.1.1. Problem specification

To re-state our problem setting in a complete way, we imagine researchers are interested in a set of variables $x \in \mathbb{R}^D$ describing some system of interest, but only variables $y \in \mathbb{R}^M$ can actually be observed, for which the theory of the respective research field provides a model $y = s(x)$ for the forward process, e. g. in form of a simulation program. As the notation suggests, in this chapter, we exclusively examine deterministic forward models, or at least deterministic for all practical purposes. In this case, since the transformation from x to y incurs an information loss, the intrinsic dimension m of y must be smaller than D, even if the nominal dimensions M satisfy $M > D$. As before, we want to express the inverse model as a conditional probability $p(x \mid y)$. The mathematical derivation of this probability from the forward model is intractable in the applications we are going to address. Note that for the case described, $p(x \mid y)$ is a singular distribution without a Radon-Nikodym probability density. In the same way as the δ-distribution, it will have some infinitely sharp peaks. This is not a problem in itself, but it has some implications for the training procedure below, e. g. precluding maximum likelihood training.

As usual, aim at approximating $p(x \mid y)$ by a tractable model $q(x \mid y)$, taking advantage of the possibility to create an arbitrary amount of training data $\{(x_i, y_i)\}_{i=1}^N$ from the known forward model $s(x)$ and a suitable prior $p(x)$. 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 $z \in \mathbb{R}^K$ drawn from a multivariate standard

normal distribution and reparametrize $q(x \mid y)$ in terms of a deterministic function g of y and z, represented by a neural network with parameters θ:

$$x = g(y, z; \theta) \quad \text{with} \quad z \sim p(z) = \mathcal{N}(z; 0, I_K). \tag{4.1}$$

Note that we distinguish between *hidden parameters* x representing unobservable real-world properties and *latent variables* z carrying information intrinsic to our model. Choosing a Gaussian prior for z poses no additional limitation, as proven by the theory of nonlinear independent component analysis (Hyvärinen and Pajunen, 1999).

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

$$[y, z] = f(x; \theta) = [f_y(x; \theta), f_z(x; \theta)] = g^{-1}(x; \theta) \quad \text{with} \quad f_y(x; \theta) \approx s(x). \tag{4.2}$$

Functions f and g share the same parameters θ and are implemented by an INN. Our experiments show that joint *bi-directional training* of f and g avoids many complications arising in e. g. cVAEs, Bayesian neural networks, and to some extent cINNs, which all have to learn the forward process implicitly. Instead, using the bidirectional approach we can apply a supervised loss directly on the forward process, ensuring its consistency.

The relation $f = g^{-1}$ is naturally enforced by the invertible network architecture, provided that the nominal and intrinsic dimensions of both sides match. Note that no modification to standard (unconditional) INN architectures is necessary, unlike with the cINN. When $m \leq M$ denotes the intrinsic dimension of y, the latent variable z must have dimension $K = D - m$, assuming that the intrinsic dimension of x equals its nominal dimension D. If the resulting nominal output dimension $M + K$ exceeds D, we augment the input with a vector $x_0 \in \mathbb{R}^{M+K-D}$ of zeros and replace x with the concatenation $[x, x_0]$ everywhere.

4.1.2. Bi-directional training

Invertible networks offer the opportunity to simultaneously optimize for losses on both the in- and output domains (Grover et al., 2017), which allows for more effective training. 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 $y_i = s(x_i)$ and network predictions $f_y(x_i)$ with a loss $\mathcal{L}_y\big(y_i, f_y(x_i)\big)$. Depending on the problem, $\mathcal{L}_y$ 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 $q\big(y = f_y(x), z = f_z(x)\big)$ and the product of marginal distributions of simulation outcomes $p\big(y = s(x)\big)$ and latents $p(z)$ as $\mathcal{L}_z\big(q(y, z), p(y)\, p(z)\big)$.

We block the gradients of $\mathcal{L}_z$ with respect to y to ensure the resulting updates only affect the predictions of z and do not worsen the predictions of y. Thus, $\mathcal{L}_z$ enforces two things: firstly, the generated z must follow the desired normal distribution $p(z)$; secondly, y and z must be independent upon convergence (i.e. $p(z \mid y) = p(z)$), and not encode the same information twice. In appendix Sec. B.1, we prove the following proposition:

Proposition 1 *If an INN $f(x) = [y, z]$ is trained as described, and both the supervised loss $\mathcal{L}_y = \mathbb{E}[(y - f_y(x))^2]$ and the unsupervised loss $\mathcal{L}_z = D\big(q(y, z), p(y)\, p(z)\big)$ reach zero, sampling according to equation (4.1) with $g = f^{-1}$ returns the true posterior $p(x \mid y^*)$ for any measurement y^*.*

Although $\mathcal{L}_y$ and $\mathcal{L}_z$ are sufficient asymptotically, a small amount of residual dependency between y and z remains after a finite amount of training. This causes $q(x \mid y)$ to deviate from the true posterior $p(x \mid y)$. To speed up convergence, we also define a loss $\mathcal{L}_x$ on the input side, implemented again by MMD. It matches the marginal distribution of backward predictions $q(x)$ against the prior data distribution $p(x)$ through $\mathcal{L}_x\big(p(x), q(x)\big)$. In we find in the appendix, Sec. B.1, $\mathcal{L}_x$ will be zero when the forward losses $\mathcal{L}_y$ and $\mathcal{L}_z$ have converged to zero. Thus, incorporating $\mathcal{L}_x$ does not alter the optimum, but improves convergence in practice. It does not compare conditional distributions, so it does not train the network directly for the problem. The $\mathcal{L}_x$ loss can be replaced with a loss comparing conditional distributions to get a behaviour more similar to the cINN.

Finally, if we use padding on either network side, loss terms are needed to ensure no information is encoded in the additional dimensions. We *a)* use a squared loss to keep those values close to zero and *b)* in an additional inverse training pass, overwrite the padding dimensions with noise of the same amplitude and minimize a reconstruction loss, which forces these dimensions to be ignored.

4.1.3. Maximum mean discrepancy

Maximum Mean Discrepancy (MMD) is a kernel-based method for comparison of two probability distributions that are only accessible through samples (Gretton et al., 2012). 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 (Tolstikhin et al., 2017). 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 $k(x, x') = 1/(1 + \|(x - x')/h\|_2^2)$, reconfirming the suggestion by Tolstikhin et al. (2017). Since the magnitude of the MMD depends on the kernel choice, the relative weights of the losses $\mathcal{L}_x$, $\mathcal{L}_y$, $\mathcal{L}_z$ are adjusted as hyperparameters, such that their effect is about equal.

4.2. Experiments

We first demonstrate the capabilities of the bidirectionally trained INNs on two well-behaved synthetic problems and then show results for two real-world applications from the fields of medicine and astrophysics. Additional details on the datasets and network architectures are provided in the appendix. In the following, we compare the bidirectional INN approach with the (non-learned) approach of approximate Bayesian computation (see below), as well as common feed-forward alternatives. In principle, a cINN could also be used to adress the tasks. A direct comparison between bidirectionally trained INNs and cINNs on the same tasks along with a more detailed quantitative comparison with many additional feed-forward methods are presented by Kruse et al. (2019), and not part of this thesis.

4.2.1. Artificial data – Gaussian mixture

To test basic viability of INNs for inverse problems, we train them on a standard 8-component Gaussian mixture model $p(x)$. The forward process is very simple: The first four mixture

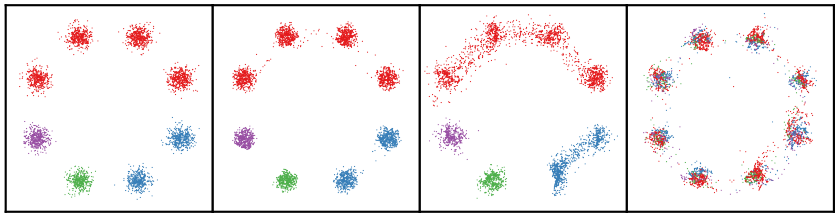


Figure 4.2.: Viability of INN for a basic inverse problem. The task is to produce the correct (multi-modal) distribution of 2D points x, given only the color label y^*. When trained with all loss terms from Sec. 4.1.2, the INN output matches ground truth almost exactly *(2nd image)*. The ablations *(3rd and 4th image)* show that we need $\mathcal{L}_y$ and $\mathcal{L}_z$ to learn the conditioning correctly, whereas $\mathcal{L}_x$ helps us remain faithful to the prior, ignoring the condition.

components (clockwise) are assigned label $y =$ **red**, the next two get label $y =$ **blue**, and the final two are labeled $y =$ **green** and $y =$ **purple** (Fig. 4.2). The true inverse posteriors $p(x \mid y^*)$ consist of the mixture components corresponding to the given one-hot-encoded label y^*. We train the INN to directly regress one-hot vectors y using a squared loss $\mathcal{L}_y$, so that we can provide plain one-hot vectors y^* to the inverse network when sampling $p(x \mid y^*)$. We observe the following: (i) The INN learns very accurate approximations of the posteriors and does not suffer from mode collapse. (ii) The coupling block architecture does not reduce the network's representational power – results are similar to standard networks of comparable size as shown in figure 4.4. (iii) Bidirectional training works best, whereas forward training alone (using only $\mathcal{L}_y$ and $\mathcal{L}_z$) captures the conditional relationships properly, but places too much mass in unpopulated regions of x-space. Conversely, pure inverse training (just $\mathcal{L}_x$) learns the correct x-distribution, but loses all conditioning information.

Using feed-foward networks of comparable size to the INN (number of parameters), we train a series of other conditional generative methods to qualitatively compare their behaviour in figure 4.4.

cVAE There is some similarity between the training setup of our method (Fig. 4.3, *right*) and that of cVAE (Fig. 4.3, *left*), as the forward and inverse pass of an INN can also be seen as an encoder-decoder pair. The main differences are that the cVAE learns the relationship $x \to y$ only indirectly, since there is no explicit loss for it, and that the INN requires no reconstruction loss, since it is bijective by construction. The problems inherent to VAEs discussed in section 2.3 leads to the generated distributions being too spread out.

cVAE-IAF We adapt the cVAE to use Inverse Autoregressive Flow (Kingma et al., 2016) between the encoder and decoder, making it more similar to a normalizing flow model. On the Gaussian mixture toy problem, the trained cVAE-IAF generates correct posteriors on par with our INN, see Fig. 4.4, but the model is larger and more complex.

cGAN Training a conditional GAN of network size comparable to the INN (counting only the generator) and only two noise dimensions turned out to be challenging. Even with additional pre-training to avoid mode collapse, the individual modes belonging to one label are reduced to nearly one-dimensional structures.

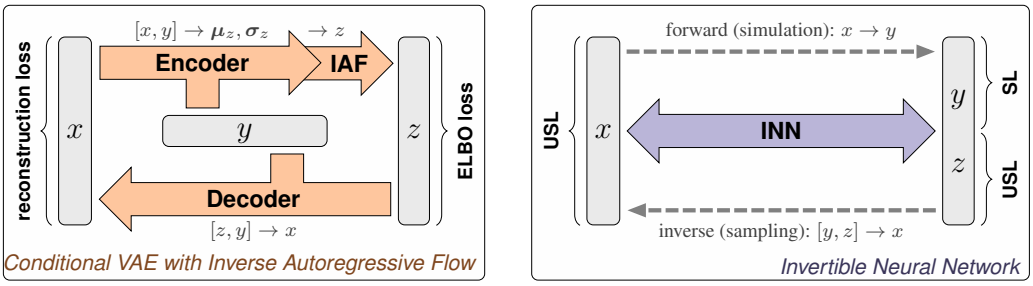


Figure 4.3.: Abstraction of the cVAE-IAF training scheme compared to our INN from Fig. 4.1. For the standard cVAE, the IAF component is omitted.

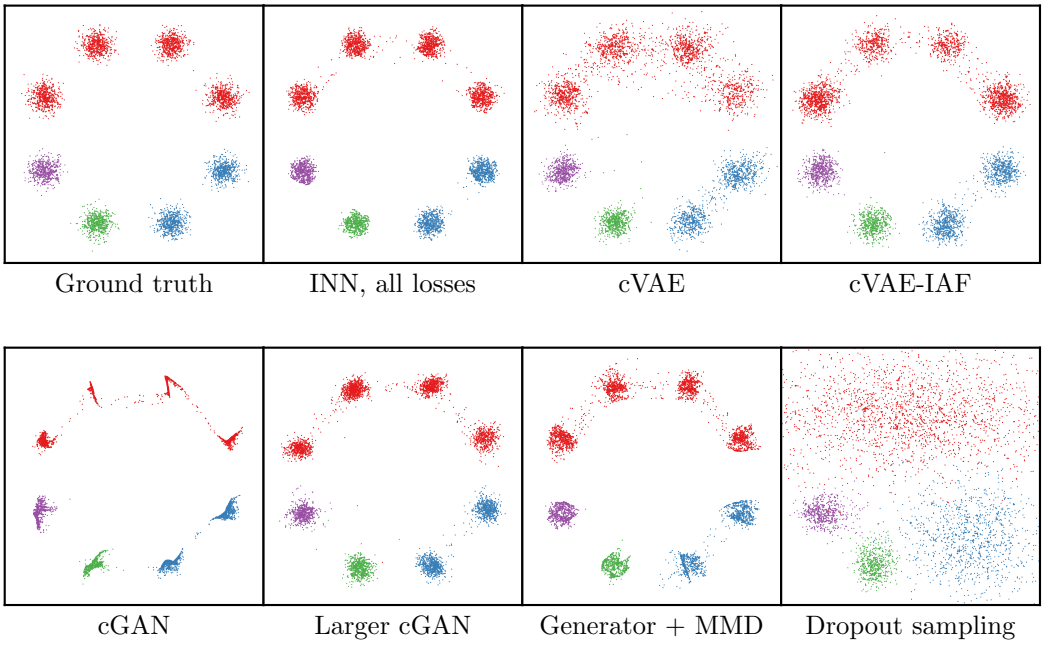


Figure 4.4.: Results of several existing methods for the Gaussian mixture toy example.

Larger cGAN In order to match the results of the INN, we trained a more complex cGAN with 2M parameters instead of the previous 10K, and a latent dimension of 128, instead of 2. To prevent mode collapse, we introduced an additional regularization: an extra loss term forces the variance of generator outputs to match the variance of the training data prior. With these changes, the cGAN can be seen to recover the posteriors reasonably well.

Generator + MMD Another option is to keep the cGAN generator the same size as our INN, but replace the discriminator with an MMD loss (cf. Sec. 4.1.3). This loss receives a concatenation of the generator output x and the label y it was supplied with, and compares these batch-wise with the concatenation of ground truth (x, y)-pairs. Note that in contrast to this, the corresponding MMD loss of the INN only receives x, and no information about y. For this small toy problem, we find that the hand-crafted MMD loss dramatically improves results compared to the standard cGAN.

Dropout sampling The method of dropout sampling with learned error terms (Kendall and Gal, 2017) is by construction not able to produce multi-modal outputs, and therefore fails on this task.

Latent space analysis

To analyze how the latent space of our INN is structured for this task, we choose a fixed label y^* and sample z from a dense grid. For each z, we compute x through our inverse network and colorize this point in latent (z) space according to the distance from the closest mode in x-space. We can see that our network learns to shape the latent space such that each mode receives the expected fraction of samples, shown in figure 4.5.

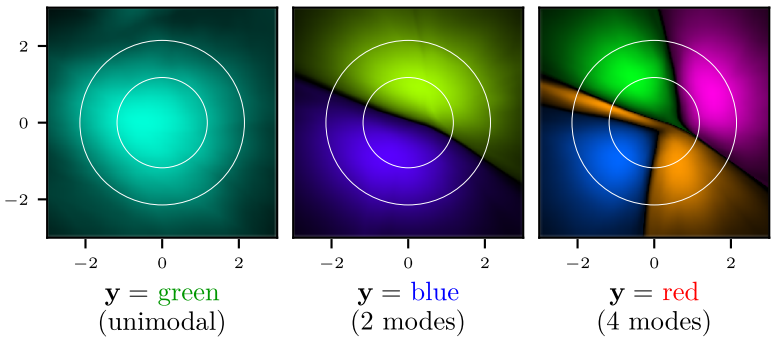


Figure 4.5.: Layout of INN latent space for one fixed label y^*, colored by mode closest to $x = g(y^*, z)$. For each latent position z, the hue encodes which mode the corresponding x belongs to and the luminosity encodes how close x is to this mode. Note that colors used here do not relate to those in Fig. 4.2, and encode the position x instead of the label y. The three columns correspond to labels *green*, *blue* and *red* Fig. 4.2. White circles mark areas that contain 50% and 90% of the probability mass of latent prior $p(z)$.

4.2.2. Artificial data – inverse kinematics

For a task with a more complex and continuous forward process, we simulate a simple inverse kinematics problem in 2D space: An articulated arm moves vertically along a rail and rotates at three joints. These four degrees of freedom constitute the parameters x. Their priors favor a pose with $180°$ angles and centered origin. The forward process is to calculate the coordinates of the end point y, given a configuration x. The inverse problem asks for the posterior distribution over all possible inputs x that place the arm's end point at a given y position. More specifically, the dataset is constructed using Gaussian priors $x_i \sim \mathcal{N}(0, \sigma_i)$, with $\sigma_1 = 0.25$ and $\sigma_2 = \sigma_3 = \sigma_4 = 0.5 \triangleq 28.65°$. For the arm lengths, we choose $l_1 = l_2 = 0.5$ and $l_3 = 1.0$. The forward process can be analytically expressed as

$$y_1 = x_1 + l_1 \sin(x_2) + l_2 \sin(x_3 - x_2) + l_3 \sin(x_4 - x_2 - x_3) \tag{4.3}$$

$$y_2 = l_1 \cos(x_2) + l_2 \cos(x_3 - x_2) + l_3 \cos(x_4 - x_2 - x_3), \tag{4.4}$$

and is illustrated in figure 4.6, *right*.

An example for a fixed y^* is shown in Fig. 4.6, where we compare our INN to a cVAE (see figure 4.3 for conceptual comparison of architectures). Adding Inverse Autoregressive

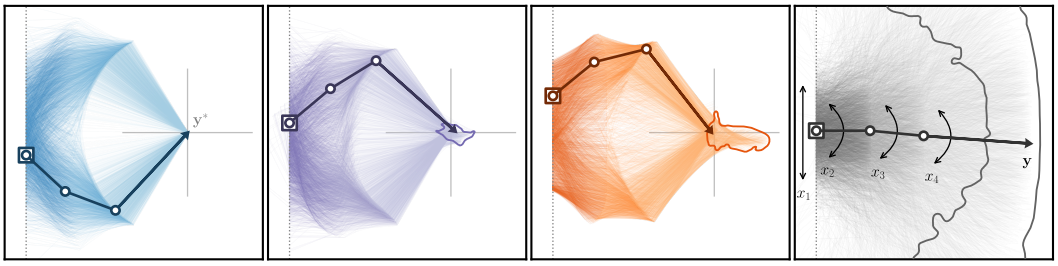


Figure 4.6.: Distribution over articulated poses x, conditioned on the end point y^*. The desired end point y^* is marked by a gray cross. A dotted line on the left represents the rail the arm is based on, and the faint colored lines indicate sampled arm configurations x taken from the true (ABC) or learned (INN, cVAE) posterior $p(x \mid y^*)$. The prior *(right)* is shown for reference. The actual end point of each sample may deviate slightly from the target y^*; contour lines enclose the regions containing 97% of these end points. We emphasize the articulated arm with the highest estimated likelihood for illustrative purposes. This example is a particularly difficult case where the differences between methods are most visible. More representative examples and quantitative evaluation is given below in figure 4.7 and table 4.1.

Flow (IAF, Kingma et al., 2016) does not improve cVAE performance in this case (see table 4.1). The chosen y^* is a difficult example, because it is an unlikely end point position according to the prior $p(x)$; it is outside the prior's 97% quantile area (Fig. 4.6, *right*), so that $p(x \mid y^*)$ considerably deviates from $p(x)$ and has a strongly bi-modal posterior $p(x \mid y^*)$. Only few training samples have been observed in this region, requiring good generalization. Easier, more representative examples are shown in figure 4.7, where the difference is not as easily visible by eye, but clearly measurable quantitatively.

In the inverse kinematics case, due to the computationally cheap forward process, we can use approximate Bayesian computation (ABC) to sample from the (approximate) ground truth posterior. While there is a whole field of research concerned with ABC approaches and their efficiency-accuracy trade-offs, our use of the method here is limited to the essential principle of rejection sampling. When we require N samples of x from the posterior $p(x \mid y^*)$ conditioned on some y^*, there are two basic ways to obtain them:

Threshold: We set an acceptance threshold ϵ, repeatedly draw x-samples from the prior, compute the corresponding y-values (via simulation) and keep those where $\text{dist}(y, y^*) < \epsilon$, until we have accepted N samples. The smaller we want ϵ, the more simulations have to be run, which is why we use this approach only for the experiment in Sec. 4.2.2, where we can afford to run the forward process millions or even billions of times.

Quantile: Alternatively, we choose what quantile q of samples shall be accepted, and then run exactly N/q simulations. All sampled pairs (x, y) are sorted by $\text{dist}(y, y^*)$ and the N closest to y^* form the posterior. This allows for a more predictable runtime when the simulations are costly, as in the medical application in Sec. 4.2.3 where $q = 0.005$.

Compared to ground truth, we find that both INN and cVAE recover the two symmetric modes well. However, the true end points of x-samples produced by the cVAE tend to miss the target y^* by a wider margin. This is because the forward process $x \to y$ is only learned implicitly during cVAE training. A trained cGAN fails completely: it misses the bi-modal

posterior entirely, sampling instead from a distribution much closer to the prior. It is not included in the further experiments.

Table 4.1.: Quantitative evaluation of the inverse kinematics experiment. The performance measurements are explained in the text below and in section 4.2.3.

Method	Mean re-sim. err.	Median re-sim. err.	Calibration err.
cVAE	0.0368	0.0307	7.78%
cVAE-IAF	0.0368	0.0307	7.81%
INN	0.0139	0.0113	0.96%

We use two quantitative measurements to affirm the success of the method, the results of which are given in table 4.1. Firstly, we compute the re-simulation error: We apply the simulation $s(\hat{x})$ to samples from the posterior, and compare the simulation outcome to the conditioning y^*. If the posterior is faithful, each sample, when re-simulated, should map exactly back to y^*, and we measure this in euclidian distance. Second, we compute the calibration error. In short, the calibration error measures to what degree the confidence intervals implied by the posterior agree with the actual performance (i. e. is the ground truth value inside the 68%-interval in 68% of cases, etc.). In detail, we compute the fraction of ground truth inliers $\alpha_{\text{inl.}}$ for corresponding α-confidence-region ($\alpha \in [0, 1]$) of the marginal posteriors in each dimension of x. The reported error is then the median of $|\alpha_{\text{inl.}} - \alpha|$ over all α. The calibration error and alternative measurements is discussed in greater detail in chapter 5.

The resulting observation in table 4.1 is that the INN is a large improvement over the cVAE-methods in both re-simulation error as well as calibration, due to the INN being directly trained in a supervised way to guarantee consistency of the posterior.

4.2.3. Estimating parameters of biological tissue

After demonstrating the viability on synthetic data, we apply our method to two real world problems from medicine and astronomy. In medical science, the functional state of biological tissue is of interest for many applications. Tumors, for example, are expected to show changes in oxygen saturation s_{O_2} (Hanahan and Weinberg, 2011). Such changes cannot be measured directly, but influence the reflectance of the tissue, which can be measured by multispectral cameras (Lu and Fei, 2014). Since ground truth data can not be obtained from living tissue, we create training data by simulating observed spectra y from a tissue model x involving s_{O_2}, blood volume fraction v_{hb}, scattering magnitude a_{mie}, anisotropy g and tissue layer thickness d (Wirkert et al., 2016). This model constitutes the forward process, and traditional methods to learn point estimates of the inverse (Wirkert et al., 2016, 2017; Claridge and Hidovic-Rowe, 2013) are already sufficiently reliable to be used in clinical trials. However, these methods can not adequately express uncertainty and ambiguity, which may be vital for an accurate diagnosis.

Competitors. We train an INN for this problem, along with two ablations (as in Fig. 4.2), as well as a cVAE with and without IAF (Kingma et al., 2016) and a network using the method of Kendall and Gal (2017), with dropout sampling and additional aleatoric error terms for each parameter. The latter also provides a point-estimate baseline (classical

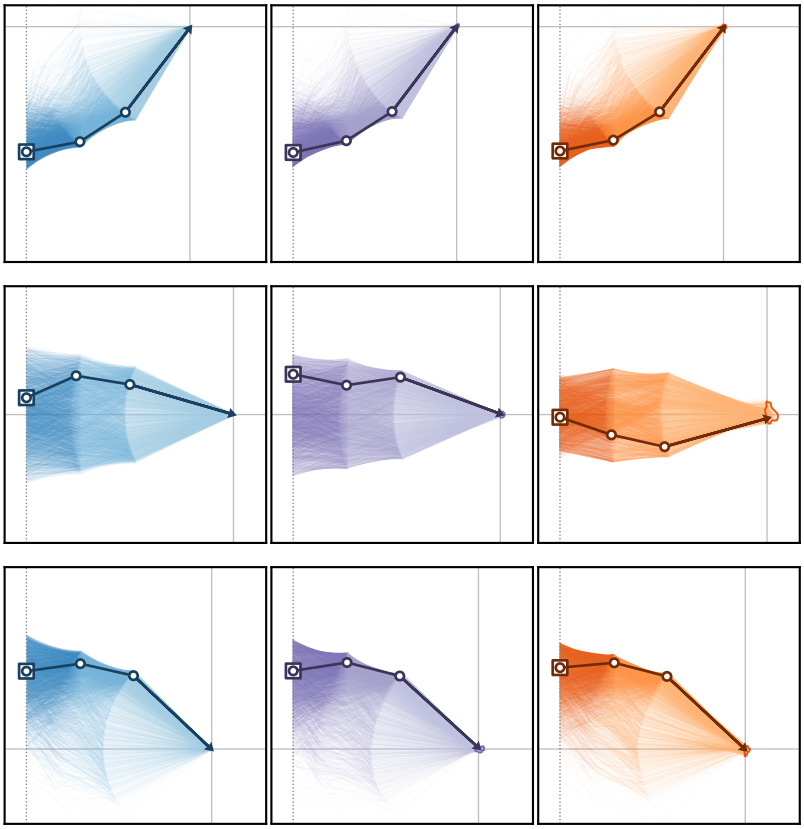


Figure 4.7.: Posteriors generated for typical, less challenging observations y^* than in Fig. 4.6.

NN) when used without dropout and error terms, which matches the current state-of-the-art results by Wirkert et al. (2017). Finally, we compare to ABC, approximating $p(x \mid y^*)$ with the 256 samples closest to y^*. Note that with enough samples, ABC would produce the true posterior. We performed 50 000 simulations to generate samples for ABC at test time, taking one week on a GPU, but still observe strong statistical noise and inconsistencies in the posteriors. This is because ecause ABC has no learned component and can not generalize from a smaller number of seen examples. The learning-based methods are trained within minutes, on a training set of 15 000 samples generated offline.

Error measures. We are interested in both the accuracy (point estimates), and the shape of the posterior distributions. For point estimates $\hat{x}$, i.e. MAP estimates, we compute the deviation from ground-truth values x^* in terms of the RMSE over test set observations y^*, RMSE $= \sqrt{\mathbb{E}_{y^*}[\|\hat{x} - x^*\|^2]}$. The 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 parameters. Again, we compute the re-simulation error as well as the calibration error. All values are computed over 5000 test-set observations y^*, or 1000 observations in the case of re-simulation error. Each posterior uses 4096 samples, or 256 for ABC; all MAP estimates are found using the mean-shift algorithm.

Quantitative results. Evaluation results for all methods are presented in Table 4.2. The INN matches or outperforms other methods in terms of point estimate error. Its accuracy deteriorates slightly when trained without $\mathcal{L}_x$, and entirely when trained without the

Table 4.2.: Quantitative results in medical application. We measure the accuracy of point/MAP estimates as detailed in Sec. 4.2.3. Best results within measurement error are **bold**, and we determine uncertainties ($\pm$) by statistical bootstrapping. The parameter s_{O_2} is the most relevant in this application, whereas *error all* means all recoverable parameters (s_{O_2}, v_{hb} and a_{mie}). Re-simulation error measures how well the MAP estimate $\hat{x}$ is conditioned on the observation y^*. Calibration error is the most important, as it summarizes correctness of the posterior shape in one number; see figure 4.10 for more calibration results.

Method	MAP error s_{O_2}	MAP error all	MAP re-simulation error	**Calibration error**
NN (+Dropout)	0.057 ± 0.003	$\mathbf{0.56 \pm 0.01}$	0.397 ± 0.008	1.91%
INN	$\mathbf{0.041 \pm 0.002}$	$\mathbf{0.57 \pm 0.02}$	$\mathbf{0.327 \pm 0.007}$	**0.34%**
INN, only $\mathcal{L}_y, \mathcal{L}_z$	0.066 ± 0.003	0.71 ± 0.02	0.506 ± 0.010	1.62%
INN, only $\mathcal{L}_x$	0.861 ± 0.033	1.70 ± 0.02	2.281 ± 0.045	3.20%
cVAE	0.050 ± 0.002	0.74 ± 0.02	$\mathbf{0.314 \pm 0.007}$	2.19%
cVAE-IAF	0.050 ± 0.002	0.74 ± 0.03	$\mathbf{0.313 \pm 0.008}$	1.40%
ABC	*0.036 ± 0.001*	*0.54 ± 0.02*	*0.284 ± 0.005*	*0.90%*
Simulation noise			0.129 ± 0.001	

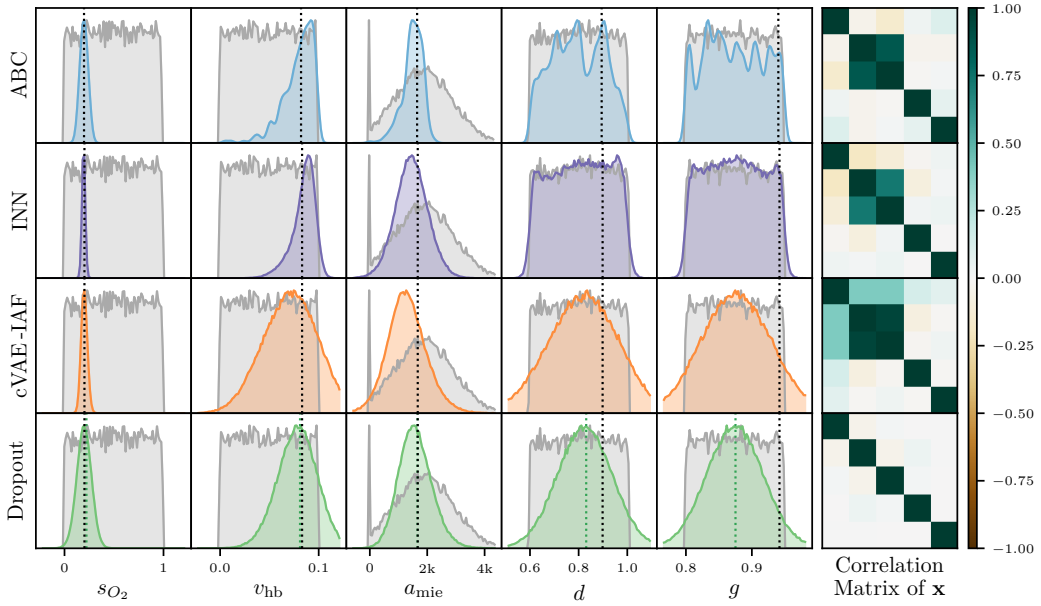


Figure 4.8.: Sampled posterior of 5 parameters for fixed y^* in medical application. For a fixed observation y^*, we compare the estimated posteriors $p(x \mid y^*)$ of different methods. The bottom row also includes the point estimate *(dashed green line)*. Ground truth values x^* *(dashed black line)* and prior $p(x)$ over all data *(gray area)* are provided for reference.

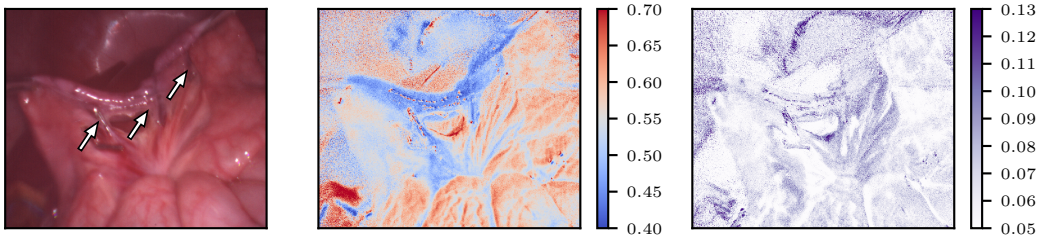
conditioning losses $\mathcal{L}_y$ and $\mathcal{L}_z$, just as in Fig. 4.2. For our purpose, the calibration error is the most important metric, as it summarizes the correctness of the whole posterior distribution in one number (also see figure 4.10). Here, the INN has a big lead over cVAE(-IAF) and Dropout, and even over ABC due to the low ABC sample count. Figure 4.10 plots the calibration error, $\alpha_{\text{inliers}} - \alpha$, against the level of confidence α in more detail. Negative values mean that a model is overconfident, while positive values say the opposite.

Qualitative results. Fig. 4.8 shows generated parameter distributions for one fixed measurement y^*, comparing the INN to cVAE-IAF, Dropout sampling and ABC. The three

former methods use a sample count of 160 000 to produce smooth curves. Due to the sparse posteriors of 256 samples 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 relevant insights: First, we find that the posteriors for layer thickness d and anisotropy g match the shape of their priors, i.e. y^* holds no information about these parameters – they are unrecoverable. This finding is supported by the ABC results, whereas the other two methods misleadingly suggest a roughly Gaussian posterior. 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 y^* as before, scattering must be increased accordingly. Since the MC dropout network employs a mean-field model, it cannot detect effects like this.

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.

Figure 4.9 shows the results when the INN trained in Sec. 4.2.3 is applied pixel-wise to multispectral endoscopic footage. In addition to estimating the oxygenation s_{O_2}, we measure the uncertainty in the form of the 68% confidence interval.



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

Figure 4.9.: INN applied to real footage to predict oxygenation s_{O_2} and uncertainty. 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.

4.2.4. Estimating parameters of star cluster formation

For a real-world astronomy application, we use a similar problem setting as already mentioned in section 3.8.1: Star clusters are born from a large reservoir of gas and dust that permeates the Galaxy, the interstellar medium (ISM). The densest parts of the ISM are called molecular clouds, and star formation occurs in regions that become unstable under their own weight. 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 (S. Klessen and C. O. Glover, 2016). To characterize the

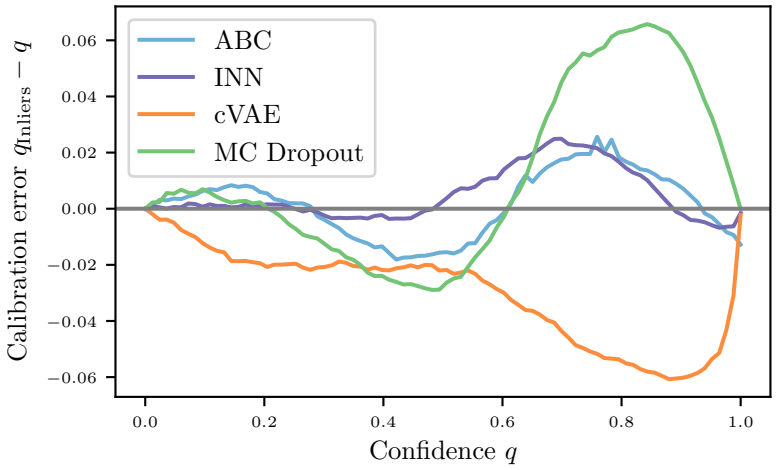


Figure 4.10.: Calibration curves for all four methods compared in Sec. 4.2.3.

impact of the energy and momentum input from young star clusters on the dynamical evolution of the ISM, astronomers frequently study emission lines from chemical elements such as hydrogen or oxygen. These lines are produced when gas is ionized by stellar radiation, and their relative intensities depend on the ionization potential of the chemical species, the spectrum of the ionizing radiation, the gas density as well as the 3D geometry of the cloud, and the absolute intensity of the radiation (Pellegrini et al., 2011). Key diagnostic tools are the so-called BPT diagrams (after Baldwin et al., 1981) emission of ionized hydrogen, H+ , to normalize the recombination lines of O++ , O+ and S+ (see also Kewley et al., 2013). We investigate the dynamical feedback of young star clusters on their parental cloud using the WARPFIELD 1D model developed by Rahner et al. (2017). It follows the entire temporal evolution of the system until the cloud is destroyed, which could take several stellar populations to happen. At each timestep we employ radiative transfer calculations (Reissl et al., 2016) to generate synthetic emission line maps which we use to train the neural network. Similar to the medical application from Section 4.2.3, the mapping from simulated observations to underlying physical parameters (such as cloud and cluster mass, and total age of the system) is highly degenerate and ill-posed. As an intermediary step, we therefore train our forward model to predict the observable quantities y (emission line ratios) from composite simulation outputs x (such as ionizing luminosity and emission rate, cloud density, expansion velocity, and age of the youngest cluster in the system, which in the case of multiple stellar populations could be considerably smaller than the total age). Using the inverse of our trained model for a given set of observations y^*, we can obtain a distribution over the unobservable properties x of the system.

Results for one specific y^* are shown in Fig. 4.11. Note that the INN recovers a decidedly multimodal distribution of x that visibly deviates from the prior $p(x)$, as well as finding strong correlations in solution space. For example, the measurements y^* investigated may correspond to a young cluster with large expansion velocity, or to an older system that expands more slowly. Finding these ambiguities in $p(x \mid y^*)$ and identifying degeneracies in the underlying model are pivotal aspects of astrophysical research, and a method to effectively approximate full posterior distributions has the potential to lead to a major breakthrough in this field.

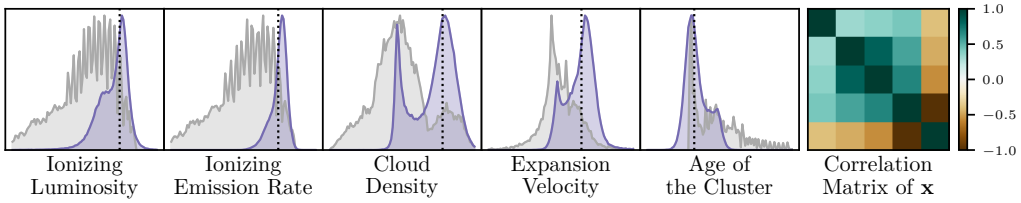


Figure 4.11.: Astrophysics application. Properties x of star clusters in interstellar gas clouds are inferred from multispectral measurements y. We train an INN on simulated data, and show the sampled posterior of 5 parameters for one y^* (colors as in Fig. 4.8, second row). The peculiar shape of the prior is due to the dynamic nature of these simulations. We include this application as a real-world example for the INN's ability to recover multiple posterior modes, and strong correlations in $p(x \mid y^*)$.

4.3. Conclusion

In summary, we see the following fundamental advantages of the INN-based method compared to alternative approaches: Firstly, one can learn the forward process and obtain the more complicated inverse process '*for free*', as opposed to other conditional generative models (especially cGANs, cVAEs), which focus only on the inverse and learn the forward process only implicitly. This improves the consistency of the posterior samples with the observation in inverse problems. Secondly, similar to the cINN, the learned posteriors are not restricted to a particular parametric form, in contrast to classical feed-foward variational methods. Kruse et al. (2019) perform a more in-depth comparison of different methods for inverse problems, confirming the results from this thesis.

One limitation of the presented bidirectional INN training is the use of the MMD loss, the effectiveness of which breaks down in very high-dimensional spaces such as for images. Without further improvements, the application of the bidirectional INNs is therefore constrained to lower-dimensional tasks, as the ones demonstrated above. Maximum-likelihood-based training as that offered by the cINN is less sensitive in this respect.

5 Information Bottleneck INN

In this chapter, we once again deviate from standard maximum likelihood training, and instead use the Information Bottleneck (IB) principle, a tool from information theory, to train our normalizing flows. Due to training with IB, we will call these models IB-INNs. Specifically, our models will be class-conditional, i. e. the condition y will be a discrete class-label. We find that the IB training technique has special implications using the model as a so-called generative classifier, a certain way to use generative models for classification discussed in more detail below. In the first section, we will discuss the background and existing work surrounding both the information bottleneck as well as generative classifiers. After that, we can restate in more detail the goal and significance of the work in this chapter, connecting the two concepts. Then, we introduce our methodology and theoretical results, before demonstrating the advantages and capabilities or the IB-INNs in various experiments.

The chapter is based in large part on the two publications Ardizzone et al. (2020b) and Mackowiak et al. (2021).

5.1. Background and Introduction

5.1.1. The Information Bottleneck

With the Information Bottleneck being an information theoretic concept, we first briefly summarize some basic ideas from information theory. We also recall the notation introduced in section 2.1, writing random variables as upper case letters and simple vectors as lower case, as this distinction plays a more important role in this chapter.

Imagining some random variable U, the Shannon information entropy is the first concept discussed. Intuitively, is quantifies the amount of information that a random variable carries. It can be computed as

$$h(U) = -\int \log p(U) \mathrm{d}p(U) \tag{5.1}$$

Here, $\mathrm{p}(U)$ means using the probability as an integration measure. In simple terms, to $p(U)\mathrm{d}U$ if U is continuous, otherwise equal to a sum over the discrete values that U can take. More details and explanation are provided in the common literature on information theory, e. g. Cover and Thomas (2012). Cross-entropy uses a different probability density $q(U)$ in addition to $p(U)$, computed as

$$h_q(U) = -\int \log q(U) \mathrm{d}p(U) \tag{5.2}$$

The connections to normalizing flows is easily apparent, for instance the negative log-likelihood loss is exactly the same as the cross-entropy between the model's density and the true one. For instance, as the cross-entropy is an upper bound on the entropy, we can use minization of the cross-entropy (or negative log-likelihood) as a bounded minimization of the information entropy of some variable.

The second concept introduced here is the Mutual Information (MI) between two different random variables U and V. Intuitively speaking, the MI quantifies the amount of information that two variables share. The intuitive understanding carries over to the definition, the MI is computed by adding up the information entropy of each variable, $h(U) + h(V)$, and subtracting the information of the joint distribution $h(U, V)$. Any duplicated or shared information is added twice (in $h(U)$ and $h(V)$), but only subtracted once. We therefore write the MI as

$$I(U, V) = h(U) + h(V) - h(U, V). \tag{5.3}$$

Inserting the definitions of the information entropy, we get the following explicit formula:

$$I(U, V) = \int \log \left(\frac{p(U, V)}{p(U)p(V)} \right) \mathrm{d}p(U, V) \tag{5.4}$$

From this, we find that we can also formulate the definition using the KL-divergence, specifically between the joint and factored distributions:

$$I(U, V) = D_{\mathrm{KL}}\Big(p(U, V) \Big\| p(U)p(V)\Big) \tag{5.5}$$

The Information Bottleneck (IB) objective is formulated using the mutual information, and allows for an information-theoretic view of various signal-processing systems, among them deep neural networks. We consider the setting where we have some observed input variable X, and want to predict some output Y from it, the standard setup in supervised learning. For simplicity, we limit the discussion to the common case of discrete Y (i.e. class labels), discussion possible extensions to continuous Y later. The IB then postulates existence of a latent space Z, where all information flow between X and Y is channeled through (hence the method's name). In order to optimize predictive performance, IB attempts to maximize the mutual information $I(Y, Z)$ between Y and Z. Simultaneously, it strives to minimize the mutual information $I(X, Z)$ between X and Z, forcing the model to ignore irrelevant aspects of X which do not contribute to classification performance and only increase the potential for overfitting. The objective can thus be expressed as

$$\mathcal{L}_{\mathrm{IB}} = I(X, Z) - \beta\, I(Y, Z)\ . \tag{5.6}$$

The trade-off parameter β is crucial to balance the two aspects.

The IB in this formulation was originally introduced by Tishby et al. (2000) as a general tool for information-theoretic optimization of compression methods. The idea was expanded on by Chechik et al. (2005); Gilad-Bachrach et al. (2003); Shamir et al. (2010) and Friedman et al. (2001). A relationship between IB and deep learning was first proposed by Tishby and Zaslavsky (2015), and later experimentally examined by Shwartz-Ziv and Tishby (2017), who use IB for the understanding of neural network behavior and training dynamics. Several works have since attempted to leverage the IB principle to train or regularize feed-forward classification models $p(Y|X)$. For instance, a close relation of IB to dropout, disentanglement, and variational auto-encoding was discovered by Achille and

Soatto (2018), which led them to introduce Information Dropout as a way to take advantage of IB in discriminative models. Secondly, the Variational Information Bottleneck (VIB, Alemi et al., 2017; Kolchinsky et al., 2017) provides a feasible approximation in form of an upper bound for the IB. These models are shown to have improved robustness in various aspects compared to standard-trained discriminative classifiers, such as less overfitting and resilience to adversarial attacks.

In this chapter, we will consider the relationship between X and Y from the exact opposite perspective as previous works – instead of using the IB for a standard prediction model $p(Y|X)$, we train an invertible neural network (INN) as a conditional generative likelihood model $p(X|Y)$, i. e. as a conditional normalizing flow. In this case, X is the variable of which the likelihood is predicted, and Y is the class condition. As with the models presented in previous chapters, it is a generative model because one can sample from the learned $p(X|Y)$ at test time to generate new examples from any class. However in this chapter, we focus on optimal likelihood estimation for existing inputs, not the generating aspect.

5.1.2. Generative classifiers

The idea of Generative Classifiers (GCs) is not usually associated with the information bottleneck, only in the course of this chapter will it be revealed that the IB and GCs fit together. GCs are in purely a way how to use conditional generative models for solving classification tasks, regardless of model type or training procedure.

In the vast majority of cases in deep learning, classification is performed through so-called discriminative classifiers (DCs), e. g. softmax classification. This means they directly predict the class probabilities $p(Y|X)$. In contrast, GCs use conditional generative models $p(X|Y)$. The fundamental difference in output modality is illustrated in figure 5.1. The posterior class probabilities are indirectly inferred at test time by Bayes' rule in the following way:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{\mathbb{E}_{p(Y)}\left[p(X|Y)\right]}. \tag{5.7}$$

Because DCs optimize prediction performance directly, they achieve better results in terms of their predictive performance. In fact, because of the increased complexity of the task, the application of GCs has so far been limited to very simple datasets such as MNIST and SVHN. For any practical image classification tasks, DCs are used exclusively, due to their superior discriminative performance. Despite this, GCs are said to have various advantages over DCs in principle.

Despite hardly being used, GCs are considered to have various advantages over DCs, which align with the term *trustworthiness* (Huang et al., 2018b). In the following, we pick three principle aspects which where already briefly discussed in chapter 1, relating them to the differences between DCs and GCs.

Uncertainty quantification: The issues in uncertainty calibration of DCs have been widely documented (Guo et al., 2017): the DCs model for $p(Y|X)$ tend to be most accurate near decision boundaries where it matters, but deteriorates away from them, where deviations incur no noticeable loss. Similarly, Out-of-distribution data is not handled properly at test

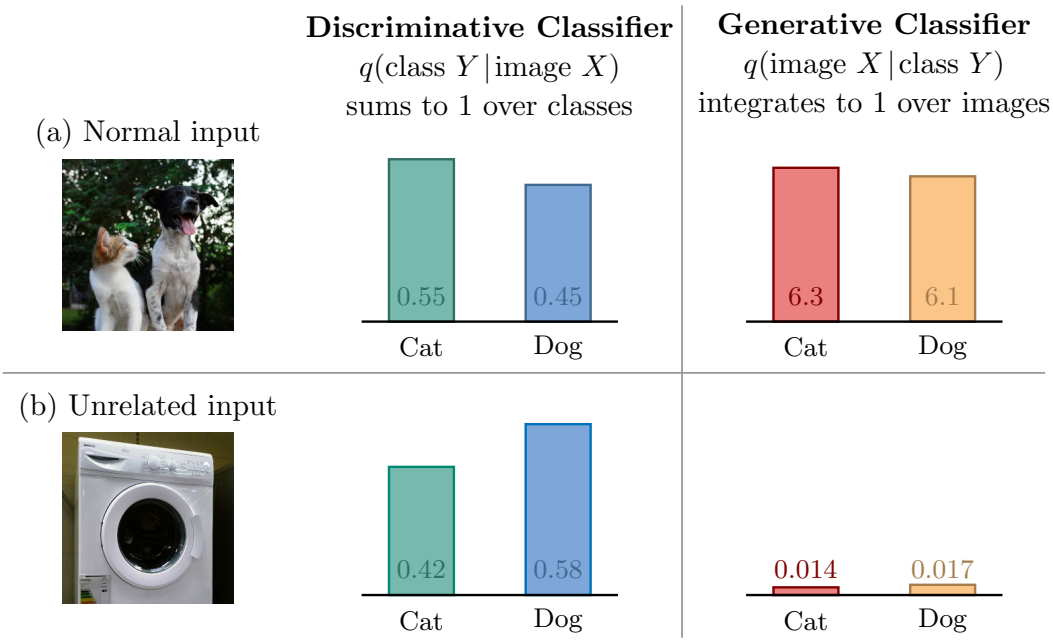


Figure 5.1.: Example of one advantage of generative classifiers: The class posterior of a DC always sums up to 1, while the likelihoods of the GC do not have this restriction, constituting inherently more informative outputs. For instance, the GC can show if a prediction is uncertain because the input agrees with both classes, or with neither. For the DC, there is no difference between the two cases.

time, DCs often make highly confident predictions on OoD data (Ovadia et al., 2019). In contrast, GCs model full likelihoods $p(X|Y)$ and thus implicitly full posteriors $p(Y|X)$, which should in principle lead to improvements in uncertainty estimates on both fronts.

Explainability: DCs based on deep neural networks are notorious for being 'black boxes', prompting many developments in the field of explainable AI. In the taxonomy laid out in Gilpin et al. (2018), most commonly used algorithms fall into categories I or II: post-hoc methods that visualize how a network processes information (I), or that show its internal representations (II). The explanations can vary depending on the chosen method, and there is no guarantee that the results faithfully reflect what the DC is doing internally.

In contrast, GCs bring to mind Feynman's mantra "What I cannot create, I do not understand". As GCs are able to model the input data itself, not just the class posteriors, they have fundamentally more informative outputs. For instance, GCs allow us to tell if a decision between two classes is uncertain because the input agrees well with *both* classes, or with *neither* (see Fig. 5.1). In addition, most GCs have interpretable latent spaces with meaningful features, allowing for the actual decision process to be directly visualized without post-hoc techniques. Therefore, it could be argued that GCs belong to category III of the explainability taxonomy Gilpin et al. (2018), i.e. methods that intrinsically work in an explainable way, without relying on additional algorithms.

Robustness: A second large concern about the practical use of deep learning based classification systems is their robustness, which can have different meanings, depending on the context and area of research. In particular, GCs have been assumed to be superior to DCs in

terms of generalization under dataset shifts Ulusoy and Bishop (2006); Raymond and Riccardi (2007). In contrast, DCs often make highly confident predictions, even when the input is completely unrelated to the training data Hendrycks and Dietterich (2019); Szegedy et al. (2014); DeVries and Taylor (2018). In addition, a big advantage of GCs is their capability to explicitly identify out-of-distribution (OoD) inputs in a natural way, thus indicating when a decision should not be trusted. Returning to the example from Fig. 5.1, we can imagine it is easy to detect the OoD input by using a threshold on the predicted likelihood. GCs have also been applied in the context of adversarial attacks, where robustness is often understood as the difficulty of finding a successful attack. For a more 'robust' model in this sense, the adversarial perturbations will be larger, and in the best case the attack will fail entirely. Furthermore, GCs were found to be more robust towards adversarial attacks Li et al. (2019) and allow for their explicit detection Ghosh et al. (2019).

The current state of research into GCs can best be understood by discussing the historical development: Long before the advent of deep learning, an in-depth analysis of the trade-offs between discriminative and generative models was first performed by Ng and Jordan (2001) and was later extended by Ulusoy and Bishop (2006); Raymond and Riccardi (2007). Including theoretical and experimental examinations, works are largely in agreement that GCs are more robust and more explainable. Works by Bouchard and Triggs (2004); Bishop and Lasserre (2007); Xue and Titterington (2010); Bishop (2007) investigated the possibility of balancing the strengths of both methods via a hyper-parameter. The existence of a trade-off is not self evident, as in principle, a more accurate generative model should generally also lead to better classification. In this work, we find that the IB can represent this trade-off, when applied to generative likelihood models used as GCs. Overall, we see from literature in the pre-deep learning era, that the performance gap and trade-offs were balanced enough for there to be no clear winner between GCs and DCs, and usage depended on the problem setting and solution requirements.

With the proliferation of deep learning, the task performance of DCs made huge leaps forward, clearly outclassing GCs. As a result, GCs have been used very rarely in recent years. Some of the few exceptions being application to natural language processing (Yogatama et al., 2017), or for robustness against missing data (Hwang et al., 2019). In a recent series of research, (Li et al., 2019; Schott et al., 2019a,b) all address adversarial robustness using GCs. Their models are shown to be more robust against adversarial attacks and able to detect them. However, they are limited to simple datasets such as MNIST and SVHN, and do not scale to problems with more than approx. 10 classes, or to natural images, a problem further examined by Fetaya et al. (2020). For the work of Schott et al. (2019b) specifically, a completely separate model is trained for each class, which is not feasible for tasks with more classes.

Trying to circumvent the issues with the predictive performance of GCs, Jacobsen et al. (2019) introduce a modified architecture that allows for mixed generative/discriminative training, by using some latent dimensions for discriminative classification and others for generative modeling. However, the model's likelihood is no longer tractable and the theoretical underpinning and implications are unclear. The same goes for Lee et al. (2019), who modify the problem by training a GC on features previously extracted from a standard feed-forward network.

GCs should furthermore be clearly distinguished from so-called hybrid models (Raina et al., 2004): these commonly only model the marginal $p(X)$ and jointly perform *discriminate* classification using a shared feature-space. The actual task is still trained discrimi-

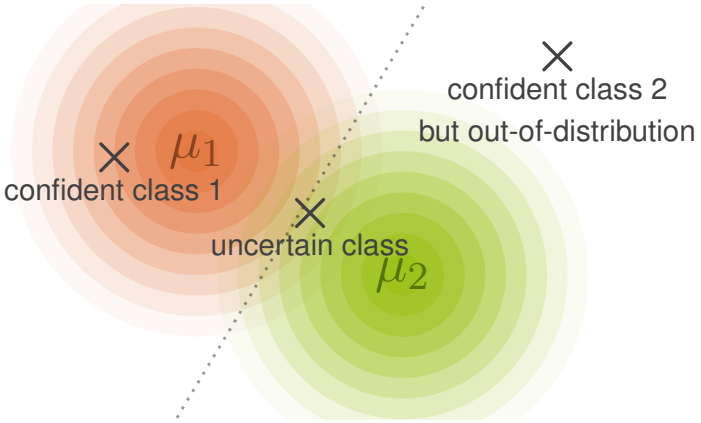


Figure 5.2.: Illustration of the latent output space of a generative classifier. The two class likelihoods for $Y = \{1, 2\}$ are parameterized by their means $\mu_{\{1,2\}}$ in Z. The dotted line represents the decision boundary. A confident, an uncertain, and an out-of-distribution sample are illustrated.

natively, and their main application is semi-supervised learning. Important examples are Kingma et al. (2014); Chongxuan et al. (2017); Nalisnick et al. (2019c); Grathwohl et al. (2019). Hybrid models have some fundamental differences to GCs, e.g. that the conditional likelihoods are not directly modeled and the latent space has no explicit class structure usually present in GCs.

Concerning OoD detection with generative models in general, the authors of Nalisnick et al. (2019b) and later Kirichenko et al. (2020) observed that likelihood models trained on natural images fail to detect certain OoD inputs, and may perform significantly worse than random. This problem is addressed e.g. by Nalisnick et al. (2019a); Choi et al. (2018); Serrà et al. (2019); Song et al. (2019a); Zhang et al. (2020), where different OoD scores are introduced that correct for these shortcomings. These works only consider unconditional likelihood models for OoD detection, while a separate classifier is still needed to perform the actual task. GCs combine both these steps into a single model, simplifying the process and potentially improving OoD detection at the same time. It also means it is harder for the class content to factor in to both OoD detection and explainability techniques when using unconditional models, as we demonstrate later in the chapter.

In summary, GCs showed some promising qualities in the past, that have not been able to be leveraged since the widespread adoption of deep learning. To this day, it has remained unclear if GCs even manifest their advantages in more complex tasks, and whether they can become competitive to DCs in task performance. For example, Fetaya et al. (2020) find while GCs can successfully detect adversarially attacked MNIST images, this already fails for the CIFAR-10 dataset. Fetaya et al. (2019) explicitly cast doubt on whether GCs can ever be used for high-dimensional input data at all.

5.1.3. The IB-INN

Applying the IB to generative invertible models is not completely straight forward. The perfectly deterministic and invertible forward pass is at odds with the assumptions of lossy and probabilistic processes of the IB, a problem that Amjad and Geiger (2019) expand

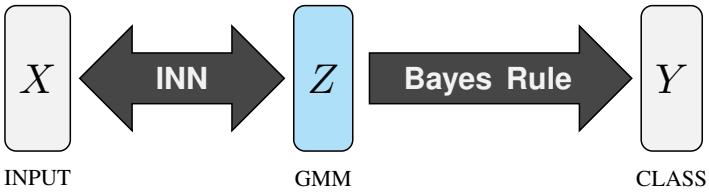


Figure 5.3.: The Information Bottleneck Invertible Neural Network (IB-INN) as a generative classifier.

on further. As is revealed in this chapter, the de-quantization noise augmentation used by default for normalizing flows is the key to resolving this issue, effectively introducing a small amount of controlled information loss.

For the conditional model itself, we use a latent conditional model as in the previous chapter instead of a cINN, which we find is a technical requirement for the method. Specifically, instead of the usual latent space $p(Z)$, we prescribe a learnable conditional latent space $p(Z|Y)$. A conditional latent distribution then implies the conditional output distribution. Due to the discrete nature of Y, we simply choose a Gaussian mixture model as the latent distribution, with one mixture component per class. figure 5.2 illustrates the decision process in such a conditional latent space Z. The uncertain in-distribution and uncertain out-of-distribution cases can each be identified with the top and bottom row in figure 5.1.

In figure 5.3, we show on a high level how the setup relates to the usual understanding of the IB: The INN transforms the inputs to a latent space Z. Due to the conditional structure of latent space, the class-posterior can be recovered using Bayes' theorem. Again, the main difference is that the IB-INN produces a class-conditional likelihood of the inputs, and not the class-posterior directly.

The main finding of this chapter is that the IB training procedure pairs exceptionally well with the concept of generative classifiers (GCs): the IB-INNs are uniquely suited for the use as GCs. As we examine in more detail later on, this is due to the $I(Z, Y)$-term, which explicitly encourages a recognizable class-structure in latent space, as opposed to standard maximum likelihood training for conditional generative models. Thereby, the trade-off parameter β in the IB-INN loss smoothly interpolates between the advantages of GCs (accurate posterior calibration and outlier detection), and those of DCs (superior task performance). Empirically, at the right setting for β, the model only suffers a minor degradation in classification accuracy compared to DCs while exhibiting more accurate uncertainty quantification than pure DCs or GCs.

We will derive our IB-based loss function in detail in section 5.2, and discuss the implications of using it to train normalizing flows. We then demonstrate the efficacy of the IB-INN as a GC on the CIFAR10 and CIFAR100 datasets in section 5.3, leading to good classification performance at the same time as the GC advantages. In section 5.4, we then scale the method to the ImageNet dataset, a level of complexity where GCs have not been applied until now. On both datasets, we examine which of the properties ascribed to GCs in the past actually manifest when using them in realistic problem settings beyond e. g. MNIST digits, with surprising results.

5.2. Method

Assumption 1 in the appendix provides some weak assumptions about the domains of the random variables and their distributions. Full proofs for all results are also provided in the appendix.

Our models have two kinds of learnable parameters. Firstly, an invertible neural network (INN) with parameters θ maps inputs X to latent variables Z bijectively: $Z = g_\theta(X) \Leftrightarrow X = g_\theta^{-1}(Z)$. Assumption 2 in the Appendix provides some explicit assumptions about the network, its gradients, and the parameter space, which are largely fulfilled by standard invertible network architectures, including the affine coupling architecture we use in the experiments. Secondly, a Gaussian mixture model with class-dependent means μ_y, where y are the class labels, and unit covariance matrices is used as a reference distribution for the latent variables Z:

$$q(Z \mid Y) = \mathcal{N}(\mu_y, \mathbb{I}) \quad \text{and} \quad q(Z) = \sum\nolimits_y p(y)\, \mathcal{N}(\mu_y, \mathbb{I}). \tag{5.8}$$

For simplicity, we assume that the label distribution is known, i.e. $q(Y) = p(Y)$.

Our derivation rests on a quantity we call *mutual cross-information* CI (in analogy to the cross-entropy):

$$CI(U, V) = \mathbb{E}_{u,v \sim p(U,V)} \left[\log \frac{q(u, v)}{q(u)q(v)} \right]. \tag{5.9}$$

Note that the expectation is taken over the true distribution p, whereas the logarithm involves model distributions q. In contrast, plain mutual information uses the same distribution in both places. Our definition is equivalent to the recently proposed predictive $\mathcal{V}$-information (Xu et al., 2020), whose authors provide additional intuition and guarantees. The following proposition (proof in Appendix) clarifies the relationship between mutual information I and CI:

Proposition 2 *Assume that $q(.)$ can be chosen from a sufficiently rich model family (e.g. a universal density estimator, see Assumption 2). Then for every $\eta > 0$ there is a model such that $\left|I(U, V) - CI(U, V)\right| < \eta$ and $I(U, V) = CI(U, V)$ if $p(U, V) = q(U, V)$.*

As the true mutual information can only be estimated and not explicitly computed for empirical data, we replace both mutual information terms $I(X, Z)$ and $I(Y, Z)$ in the IB objective (equation (5.6)) with the mutual cross-information CI, and derive optimization procedures for each term in the following subsections.

5.2.1. INN-Based Formulation of the I(X,Z)-Term in the IB Objective

Estimation of the mutual cross-information $CI(X, Z)$ between inputs and latents is problematic for deterministic mappings from X to Z (Amjad and Geiger, 2018), and specifically for INNs, which are bijective by construction. In this case, the joint distributions $q(X, Z)$ and $p(X, Z)$ are not valid Radon-Nikodym densities and both CI and I are undefined. Intuitively, I and CI become infinite, because p and q have an infinitely high delta-peak at $Z = g_\theta(X)$, and are otherwise 0. For the IB to be applicable, some information has to be discarded in the mapping to Z, making p and q valid (Radon-Nikodym-)densities. In

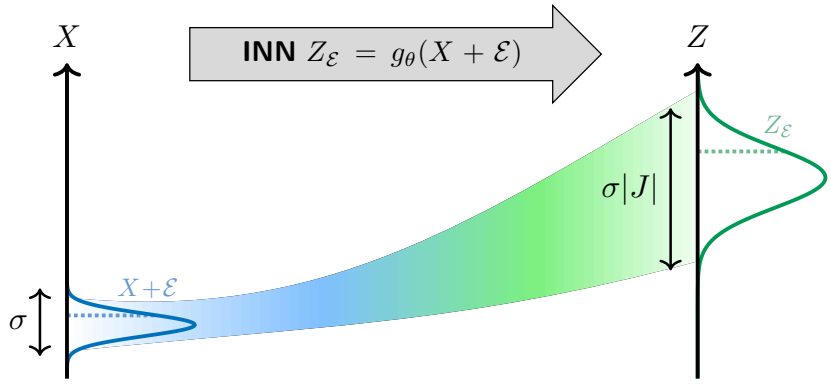


Figure 5.4.: The more the noise is amplified in relation to the noise-free input, the lower the mutual cross-information between noisy latent vector $Z_\mathbb{E}$ and noise-free input X.

contrast, normalizing flows rely on all information to be retained for optimal generative capabilities and density estimation.

Our solution to this seeming contradiction comes from the practical use of normalizing flows. Here, a small amount of noise is commonly added to dequantize X (i.e. to turn discrete pixel values into real numbers), to avoid numerical issues during training. If the noise is smaller or equal to the quantization step size ΔX, the capability for density estimation is not affected. We adopt this approach to artificially introduce a minimal amount of information loss: Instead of feeding X to the network, we input a noisy version $X' = X + \mathbb{E}$, where $\mathbb{E} \sim \mathcal{N}(0, \sigma^2\mathbb{I}) = p(\mathbb{E})$ is Gaussian with mean zero and covariance $\sigma^2\mathbb{I}$. For a quantization step size ΔX, the additional error on the estimated densities caused by the augmentation has a known bound decaying with $\exp(-\Delta X^2/2\sigma^2)$ (see Appendix). We are interested in the limit $\sigma \to 0$, so in practice, we choose a very small fixed σ, that is smaller than ΔX. This makes the error practically indistinguishable from zero. The INN then learns the bijective mapping $Z_\mathbb{E} = g_\theta(X + \mathbb{E})$, which guarantees $CI(X, Z_\mathbb{E})$ to be well defined. Minimizing this CI according to the IB principle means that $g_\theta(X + \mathbb{E})$ is encouraged to amplify the noise $\mathbb{E}$, so that X can be recovered less accurately, see figure 5.4 for illustration. If the global minimum of the loss is achieved w.r.t. θ, I and CI coincide, as $CI(X, Z_\mathbb{E})$ is an upper bound (also cf. Prop. 2):

Proposition 3 *For the specific case that* $Z_\mathbb{E} = g_\theta(X + \mathbb{E})$*, it holds that* $I(X, Z_\mathbb{E}) \leq CI(X, Z_\mathbb{E})$.

Our approach should be clearly distinguished from applications of the IB to DCs, such as Alemi et al. (2017), which pursue a different goal. There, the model learns to ignore the vast majority of input information and keeps only enough to predict the class posterior $p(Y \mid X)$. In contrast, we induce only a small, explicitly adjustable loss of information to make the IB well-defined. As a result, the amount of retained information in our generative IB-INNs is orders of magnitude larger than in DC approaches, which is necessary to represent accurate class-conditional likelihoods $p(X \mid Y)$.

We now derive the loss function that allows optimizing θ and μ_y to minimize the noise-augmented $CI(X, Z_\mathbb{E})$ in the limit of small noise $\sigma \to 0$. Full details are found in appendix.

We decompose the mutual cross-information into two terms

$$CI(X, Z_{\mathbb{E}}) = \mathbb{E}_{p(X),p(\mathbb{E})}\big[-\log q\big(Z_{\mathbb{E}}=g_\theta(x+\varepsilon)\big)\big] + \underbrace{\mathbb{E}_{p(X),p(\mathbb{E})}\big[\log q\big(Z_{\mathbb{E}}=g_\theta(x+\varepsilon)\,\big|\,x\big)\big]}_{:=A}.$$

The first expectation can be approximated by the empirical mean over a finite dataset, because the Gaussian mixture distribution $q(Z_{\mathbb{E}})$ is known analytically. To approximate the second term, we first note that the condition $X = x$ can be replaced with $Z = g_\theta(x)$, because g_θ is bijective and both conditions convey the same information

$$A = \mathbb{E}_{p(X),p(\mathbb{E})}\big[\log q\big(Z_{\mathbb{E}} = g_\theta(x+\varepsilon)\,\big|\,Z = g_\theta(x)\big)\big].$$

We now linearize g_θ by its first order Taylor expansion,

$$g_\theta(x+\varepsilon) = g_\theta(x) + J_x\varepsilon + O(\varepsilon^2),$$

where $J_x = \frac{\partial g_\theta(X)}{\partial X}\big|_x$ denotes the Jacobian at $X = x$. Going forward, we write $O(\sigma^2)$ instead of $O(\varepsilon^2)$ for clarity, noting that both are equivalent because we can write $\varepsilon = \sigma n$ with $n \sim \mathcal{N}(0, \mathbb{I})$, and $\|\varepsilon\| = \sigma\|n\|$. Inserting the expansion into A, the $O(\sigma^2)$ can be moved outside of the expression: It can be moved outside the $\log$, because that has a Lipschitz constant of $1/\inf q(g_\theta(X + \mathbb{E}))$, which we show is uniformly bounded in the full proof. The $O(\sigma^2)$ can then be exchanged with the expectation because the expectation's argument is also uniformly bounded, finally leading to

$$A = \mathbb{E}_{p(X),p(\mathbb{E})}\big[\log q\big(g_\theta(x) + J_x\varepsilon\,\big|\,g_\theta(x)\big)\big] + O(\sigma^2).$$

Since ε is Gaussian with mean zero and covariance $\sigma^2\mathbb{I}$, the conditional distribution is Gaussian with mean $g_\theta(x)$ and covariance $\sigma^2 J_x J_x^T$. The expectation with respect to $p(\mathbb{E})$ is thus the negative entropy of a multivariate Gaussian and can be computed analytically as well

$$\begin{aligned} A &= \mathbb{E}_{p(X)}\left[-\frac{1}{2}\log\big(\det(2\pi e\sigma^2 J_x J_x^T)\big)\right] + O(\sigma^2) \\ &= \mathbb{E}_{p(X)}\big[-\log|\det(J_x)|\big] - d\log(\sigma) - \frac{d}{2}\log(2\pi e) + O(\sigma^2) \end{aligned}$$

with d the dimension of X. To avoid running the model twice (for x and $x + \varepsilon$), we approximate the expectation of the Jacobian determinant by 0^{th}-order Taylor expansion as

$$\mathbb{E}_{p(X)}\big[\log|\det(J_x)|\big] = \mathbb{E}_{p(X),p(\mathbb{E})}\big[\log|\det(J_\varepsilon)|\big] + O(\sigma),$$

where J_ε is the Jacobian evaluated at $x+\varepsilon$ instead of x. The residual can be moved outside of the $\log$ and the expectation because J_ε is uniformly bounded in our networks.

Putting everything together, we drop terms from $CI(X, Z_{\mathbb{E}})$ that are independent of the model or vanish with rate at least $O(\sigma)$ as $\sigma \to 0$. The resulting loss $\mathcal{L}_X$ becomes

$$\mathcal{L}_X = \mathbb{E}_{p(X),\,p(\mathbb{E})}\big[-\log q\big(g_\theta(x+\varepsilon)\big) - \log\big|\det(J_\varepsilon)\big|\big]. \tag{5.10}$$

Since the change of variables formula defines the network's generative distribution as

$$q_X(x) = q\big(Z = g_\theta(x)\big)\,\big|\det(J_x)\big|, \tag{5.11}$$

$\mathcal{L}_X$ is the negative log-likelihood of the perturbed data under q_X,

$$\mathcal{L}_X = \mathbb{E}_{p(X),p(\mathbb{E})}\big[-\log q_X(x+\varepsilon)\big]. \tag{5.12}$$

The crucial difference between $CI(X, Z_{\mathbb{E}})$ and $\mathcal{L}_X$ is the elimination of the term $-d\log(\sigma)$. It is huge for small σ and would dominate the model-dependent terms, making minimization of $CI(X, Z_{\mathbb{E}})$ very hard. Intuitively, the fact that $CI(X, Z_{\mathbb{E}})$ diverges for $\sigma \to 0$ highlights why $CI(X, Z)$ is undefined for bijectively related X and Z. In practice, we estimate $\mathcal{L}_X$ by its empirical mean on a training set $\{x_i, \varepsilon_i\}_{i=1}^N$ of size N, denoted as $\mathcal{L}_X^{(N)}$.

$$\mathcal{L}_X^{(N)} = \frac{1}{N}\sum_{i=1}^{N}\big[-\log q\big(g_\theta(x_i+\varepsilon_i)\big) - \log\big|\det(J_i)\big|\big] \tag{5.13}$$

where J_i is the Jacobian of g_θ evaluated at $x_i + \varepsilon_i$.

It remains to be shown that replacing $I(X, Z_{\mathbb{E}})$ with $\mathcal{L}_X^{(N)}$ in the IB loss Eq. 5.6 does not fundamentally change the solution of the learning problem in the limit of large N, small σ and sufficient model power. Sufficient model power here means that the family of generative distributions realizable by g_θ should be a universal density estimator (see Appendix, Assumption 2). This is the case if g_θ can represent increasing triangular maps (Bogachev et al., 2005), which has been proven for certain network architectures explicitly (e.g. Jaini et al., 2019; Huang et al., 2018a), including the affine coupling networks we use for the experiments (Teshima et al., 2020). Propositions 2 & 3 then tell us that we may optimize $CI(X, Z_{\mathbb{E}})$ as an estimator of $I(X, Z_{\mathbb{E}})$. The above derivation of the loss can be strengthened into

Proposition 4 *Under Assumptions 1 and 2, for any $\epsilon, \eta > 0$ and $0 < \delta < 1$ there are $\sigma_0 > 0$ and $N_0 \in \mathbb{N}$, such that $\forall N \geq N_0$ and $\forall 0 < \sigma < \sigma_0$, the following holds uniformly for all model parameters θ:*

$$\Pr\left(\left| CI(X, Z_{\mathbb{E}}) + d\log\sqrt{2\pi e\sigma^2} - \mathcal{L}_X^{(N)}\right| > \epsilon\right) < \delta$$

$$\textit{and}\quad \Pr\left(\left\| \frac{\partial}{\partial\theta} CI(X, Z_{\mathbb{E}}) - \frac{\partial}{\partial\theta}\mathcal{L}_X^{(N)}\right\| > \eta\right) < \delta$$

The first statement proves consistence of $\mathcal{L}_X^{(N)}$, and the second justifies gradient-descent optimization on the basis of $\mathcal{L}_X^{(N)}$. Proofs can be found in the appendix.

5.2.2. GMM-Based Formulation of the I(Z,Y)-Term in the IB Objective

Similarly to the first term in the IB-loss in Eq. 5.6, we also replace the mutual information $I(Y, Z)$ with $CI(Y, Z_{\mathbb{E}})$. Inserting the likelihood $q(z \mid y) = \mathcal{N}(z; \mu_y, \mathbb{I})$ of our latent Gaussian mixture model into the definition and recalling that $q(Y) = p(Y)$, this can be decomposed into

$$CI(Y, Z_{\mathbb{E}}) = \mathbb{E}_{p(Y)}\big[-\log p(y)\big] + \mathbb{E}_{p(X,Y),p(\mathbb{E})}\left[\log\frac{q\big(g_\theta(x+\varepsilon)\mid y\big)\,p(y)}{\sum_{y'} q\big(g_\theta(x+\varepsilon)\mid y'\big)\,p(y')}\right]. \tag{5.14}$$

In this case, $CI(Y, Z_{\mathbb{E}})$ is a lower bound on the true mutual information $I(Y, Z_{\mathbb{E}})$, allowing for its maximization in our objective. In fact, it corresponds to a bound originally proposed by Barber and Agakov (2003) (see their Eq. 3): The first term is simply the entropy $h(Y)$, because $p(Y)$ is known. The second term can be rewritten as the negative cross-entropy $-h_q(Y \mid Z_{\mathbb{E}})$. For $I(Y, Z_{\mathbb{E}})$, we would have the negative entropy $-h(Y \mid Z_{\mathbb{E}})$ in its place, then Gibbs' inequality leads directly to $CI(Y, Z_{\mathbb{E}}) \leq I(Y, Z_{\mathbb{E}})$.

The first expectation can be dropped during training, as it is model-independent. Note how the the second term can also be written as the expectation of the GMM's log-posterior $\log q(y \,|\, z)$. Since all mixture components have unit covariance, the elements of Z are conditionally independent and the likelihood factorizes as $q(z \,|\, y) = \prod_j q(z_j \,|\, y)$. Thus, $q(y \,|\, z)$ can be interpreted as a naive Bayes classifier. In contrast to naive Bayes classifiers in data space, which typically perform badly because raw features are not conditionally independent, our training enforces this property in latent space and ensures accurate classification. Defining the loss $\mathcal{L}_Y^{(N)}$ as the empirical mean of the log-posterior in a training set $\{x_i, y_i, \varepsilon_i\}_{i=1}^N$ of size N, we get

$$\mathcal{L}_Y^{(N)} = \frac{1}{N} \sum_{i=1}^{N} \log \frac{\mathcal{N}\big(g_\theta(x_i + \varepsilon_i); \mu_{y_i}, \mathbb{I}\big)\, p(y_i)}{\sum_{y'} \mathcal{N}\big(g_\theta(x_i + \varepsilon_i); \mu_{y'}, \mathbb{I}\big)\, p(y')}. \tag{5.15}$$

5.2.3. The IB-INN-Loss and its Advantages

Replacing the mutual information terms in Eq. 5.6 with their empirical estimates $\mathcal{L}_X^{(N)}$ and $\mathcal{L}_Y^{(N)}$, our model parameters θ and $\{\mu_1, ..., \mu_K\}$ are trained by gradient descent of the *IB-INN loss*

$$\mathcal{L}_{\text{IB-INN}}^{(N)} = \mathcal{L}_X^{(N)} - \beta\, \mathcal{L}_Y^{(N)} \tag{5.16}$$

In the following, we will interpret and discuss the nature of the loss function in Eq. 5.16 and form an intuitive understanding of why it is more suitable than the class-conditional negative-log-likelihood ('class-NLL') traditionally used for normalizing-flow type generative classifiers: $\mathcal{L}_{\text{class-NLL}} = -\mathbb{E} \log \big(q_\theta(x|y)\big)$. The findings are represented graphically in Fig. 5.5.

$\mathcal{L}_X$-term: As shown by Eq. 5.12, the term is the (unconditional) negative-log-likelihood loss used for normalizing flows, with the difference that $q(Z)$ is a GMM rather than a unimodal Gaussian. We conclude that this loss term encourages the INN to become an accurate likelihood model under the marginalized latent distribution and to ignore any class information.

$\mathcal{L}_Y$-term: Examining Eq. 5.15, we see that for any pair $(g(x + \varepsilon), y)$, the cluster centers $(\mu_{Y \neq y})$ of the other classes are repulsed (by minimizing the denominator), while $g_\theta(x + \varepsilon)$ and the correct cluster center μ_y are drawn together. Note that the class-NLL loss only captures the second aspect and lacks repulsion, resulting in a much weaker training signal. We can also view this in a different way: by substituting $q(x|y) \left|\det(J_x)\right|^{-1}$ for $q(z|y)$, the second summand of Eq. 5.14 simplifies to $\log q(y|x)$, since the Jacobian cancels out. This means that our $\mathcal{L}_Y$ loss directly maximizes the correct class probability, while ignoring the

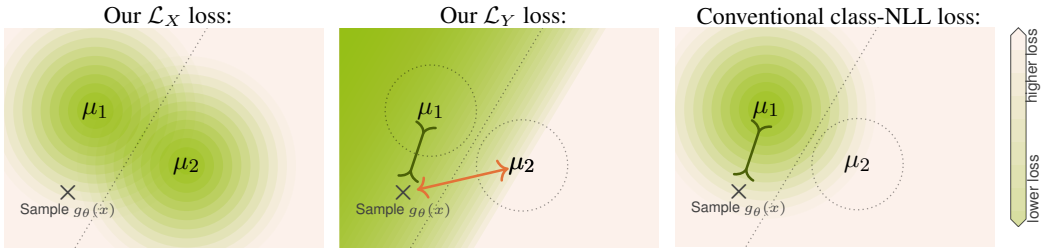


Figure 5.5.: Illustration of the loss landscape for our IB formulation *(left, middle)* and standard class-conditional negative-log-likelihood *(right)*. The loss is shown for an input x belonging to class $Y = 1$, green areas correspond to low loss. The orange arrows and black inverted arrows indicate repulsive and attractive interactions with the cluster centers. Crucially, standard NNL exerts no repulsive force.

data likelihood. Again, this improves the training signal: as Fetaya et al. (2019) showed, the data likelihood will otherwise dominate the class-NLL loss, so that lack of classification accuracy is insufficiently penalized.

Classical class-NLL loss: The class-NLL loss or an approximation thereof is used to train standard GCs. The IB-INN loss reduces to this case for $\beta = 1$, because the first summand in $\mathcal{L}_X$ (cf. Eq. 5.10) cancels with the denominator in Eq. 5.15. Then, the INN no longer receives a penalty when latent mixture components overlap, and the GMM looses its class discriminatory power, as Fig. 5.5 illustrates: Points are only drawn towards the correct class, but there is no loss component repulsing them from the incorrect classes. As a result, all cluster centers tend to collapse together, leading the INN to effectively just model the marginal data likelihood. This confirmed and explained in more detail by Fetaya et al. (2019), who show that indeed the class-NLL loss causes a vanishingly small training signal for the class separation. Similarly, Wu et al. (2019) found that $\beta = 1$ is the minimum possible value to perform classification with discriminative IB methods.

5.3. Experiments with CIFAR datasets

In the following, we examine the properties of the IB-INN used as a GC, especially the quality of uncertainty estimates and OoD detection. We construct our IB-INN by combining the design efforts of various works on INNs and normalizing flows. In brief, we use a Real-NVP architecture consisting of affine coupling blocks (Dinh et al., 2016), with added improvements from recent works (Kingma and Dhariwal, 2018; Jacobsen et al., 2019, 2018; Ardizzone et al., 2019b). A detailed description of the architecture is given in the work of Ardizzone et al. (2020b), and not included in this thesis. We learn the set of means μ_Y as free parameters jointly with the remaining model parameters in an end-to-end fashion using the loss in Eq. 5.16. The practical and numerically stable implementation of the losses is also discussed in Ardizzone et al. (2020b), and also not part of this thesis.

We apply two additional techniques while learning the model, label smoothing and loss rebalancing:

Label smoothing We observe that the individual class means μ_Y drift apart during training, because training with hard labels enforces the Gaussian mixture components to become perfectly separated. This can cause problems during training, as there is a high loss barrier between the clusters due to $\mathcal{L}_X$, preventing points from moving smoothly from one class to the other during training. To avoid this effect, we simply apply a small amount of label smoothing (Szegedy et al., 2016), where the one-hot training vectors are softened with $\alpha = 0.05$ in our case. We apply the same to all baseline comparison models.

Loss rebalancing To avoid the laborious process of adjusting hyperparameters for vastly different loss magnitudes, we employ a loss rebalancing scheme. This allows us to use the same hyperparameters when changing β between 5 orders of magnitude. Firstly, we divide the loss $\mathcal{L}_X$ by the number of dimensions of X, which approximately matches its magnitude to the $\mathcal{L}_Y$ loss. It also ensures that $\mathcal{L}_X$ remains in a similar range when changing e.g. the input image size, as it scales linearly with the number of input dimensions. We define a corresponding $\gamma := \beta / \dim(X)$ to stay consistent with the IB definition. Secondly, we scale the entire loss by a factor $2/(1+\gamma)$. This ensures that it keeps the same magnitude when changing γ.

$$\mathcal{L}_{\text{IB}}^{(N)} = \frac{2}{1+\gamma} \left(\frac{\mathcal{L}_X^{(N)}}{\dim(X)} - \gamma \, \mathcal{L}_Y^{(N)} \right) \tag{5.17}$$

Finally, the noise amplitude σ should be chosen to satisfy two criteria: it should be small enough so that the Taylor expansions in the loss for $\sigma \to 0$ are sufficiently accurate, and it should also not hinder the model's performance. Our ablation given later in figure 5.11 indicates that both criteria are satisfied when $\sigma \lessapprox 0.25 \Delta X$, with the quantization step size ΔX, so we fix $\sigma = 10^{-3}$ for the remaining experiments.

5.3.1. Baseline comparison methods

In addition to the IB-INN, we train several alternative methods. For each, we use exactly the same INN model, or an equivalent feed-forward ResNet model. Every method has the exact same hyperparameters and training procedure, the only difference being the loss function and invertibility.

Class-NLL: As a standard generative classifier, we train an INN with a GMM in latent space naively as a conditional generative model, using the class-conditional maximum likelihood loss:

$$\mathcal{L}_{\text{class-NLL}} = -\mathbb{E} \log \big(q_\theta(x|y) \big). \tag{5.18}$$

Secondly, we also train a regularized version to increase the classification accuracy. The regularization consists of leaving the class centroids μ_Y fixed on a hyper-sphere, forcing some degree of class-separation. As the radius becomes comparable to the typical intra-class distances, the training signal for the classification is amplified. We therefore choose $\sqrt{\dim(Z)}$ as radius.

Feed-forward: As a DC baseline, we train a standard ResNet (He et al., 2016) with softmax cross entropy loss. We replace each affine coupling block by a ResNet block, leaving all other hyperparameters the same.

i-RevNet (Jacobsen et al., 2018): To rule out any differences stemming from the constraint of invertibility, we additionally train the INN as a standard softmax classifier, by projecting the outputs to class logits with an additional fully connected layer. While the architecture is

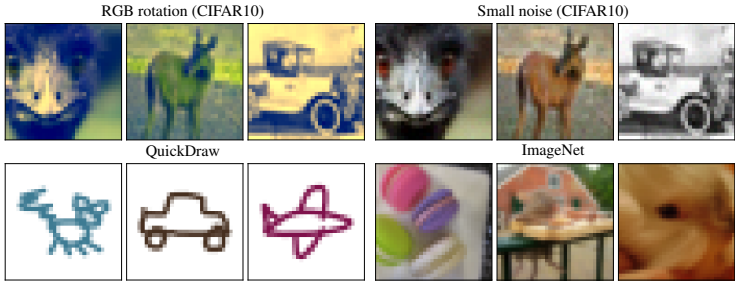


Figure 5.6.: Examples from each OoD dataset used in the evaluation. The inlier data are original CIFAR10 images.

invertible, it is not a generative model and trained just like a standard feed-forward classifier.
Variational Information Bottleneck (VIB): To examine which observed behaviours are due to the IB in general, and what is specific to GCs, we also train the VIB (Alemi et al., 2017), a feed-forward DC, using a ResNet. We convert the authors definition of β to our γ for consistency.

5.3.2. Experimental setup

In the following, we describe the scores used below in Table 5.1.
Bits/dim: The bits/dim metric is common for objectively comparing the performance of density estimation models such as normalizing flows, and is closely related to the KL divergence between real and estimated distributions. Details can be found e.g. in Theis et al. (2015). It is esentially just a rescaled version of the negative log-likelihood motivated by lossless compression quality.
Calibration error: The calibration curve measures whether the confidence of a model agrees with its actual performance. While the previous chapters considered regression problems, the calibration error is computed differently for classification. To formalize it, we use the Iverson bracket:

$$[C] := \begin{cases} 1 & \text{if } C \text{ is true;} \\ 0 & \text{otherwise,} \end{cases} \tag{5.19}$$

All prediction outputs are the binned according to their predicted probability P, the *confidence*. All outputs are included, not just the class with the highest predicted probability. For each bin, it is recorded for which fraction of these samples the prediction was actually correct, Q.

For this, we define bin edges b_i, with $i \in \{1, \dots, K+1\}$, so that $b_1 = 0$, $b_{K+1} = 1$, and $b_{i+1} > b_i$. The bins themselves are then half-open intervals between the bin edges: $B_i = [b_i, b_{i+1})$ with $i \in \{1, \dots, K\}$. In practice, we choose the b_i be spaced more tightly near high and low confidences, as this is where the bulk of the predictions are made:

```
concatenate(range(0.00, 0.05, stepsize=0.01),
            range(0.05, 0.95, stepsize=0.1),
            range(0.95, 1.00, stepsize=0.01))
```

The results are largely stable when changing these values within reasonable limits. We can now define $n^{(i)}$, the count of predictions within a confidence bin; as well as $n_c^{(i)}$, the count

of *correct* predictions in that bin:

$$n^{(i)} \coloneqq \sum_{x_j} \sum_{y'} \left[p(y'|x_j) \in B_i \right] \tag{5.20}$$

$$n_c^{(i)} \coloneqq \sum_{(x_j, y_j)} \sum_{y'} \left[p(y'|x_j) \in B_i \right] \cdot \left[\arg\max_{y'} (p(y'|x_j) = y_j \right] \tag{5.21}$$

where x_j and the (x_j, y_j)-pairs are from the test set.

We define the confidence P as the center of each bin, and the achieved accuracy in this bin as Q:

$$P_i = \frac{b_i + b_{i+1}}{2} \tag{5.22}$$

$$Q_i = \frac{n_c^{(i)}}{n^{(i)}} \tag{5.23}$$

For a perfectly calibrated model and negligible binning- and counting-noise, we expecet $P = Q$, e.g. predictions with 70% confidence are correct 70% of the time.

We use several metrics to measure deviations from this behaviour, largely in line with Guo et al. (2017). Specifically, we consider the expected calibration error (ECE, error weighted by bin count),

$$\mathrm{ECE} = \sum_i \frac{n^{(i)}}{n_{\mathrm{tot}}} |P_i - Q_i| \tag{5.24}$$

using the shorthand $n_{\mathrm{tot}} \coloneqq \sum_i n^{(i)}$. Secondly, the maximum calibration error (MCE, max error over all bins),

$$\mathrm{MCE} = \max_i |P_i - Q_i|. \tag{5.25}$$

Lastly, we also compute the integrated calibration error (ICE, area between perfect and actual behaviour),

$$\mathrm{ICE} = \sum_i (b_{i+1} - b_i) |P_i - Q_i| \tag{5.26}$$

We also include the geometric mean of all three (GME) as a single quantitative value:

$$GME = \sqrt[3]{\mathrm{ECE} \cdot \mathrm{MCE} \cdot \mathrm{ICE}}. \tag{5.27}$$

The geometric mean is used because it best accounts for the different magnitudes of the metrics.

Increased out-of-distribution (OoD) prediction entropy: For data that is OoD, we expect from a model that it returns uncertain class predictions, as it has not been trained on such data. In the ideal case, each class is assigned the same probability of $1/(\text{nr. classes})$. Ovadia et al. (2019) quantify this through the discrete entropy of the class prediction outputs $H(Y|X_{\mathrm{Ood}})$. To counteract the effect of less accurate models having higher prediction entropy overall, we report the difference between OoD and in-distribution test set

$$H(Y|X_{\mathrm{Ood}}) - H(Y|X_{\text{In distrib.}}) \tag{5.28}$$

OoD detection score: We use OoD detection capabilities intrinsically built in to GCs. For this, we apply the recently proposed typicality test (Nalisnick et al., 2019a). This is a hypothesis test that sets an upper and lower threshold on the estimated likelihood, beyond which batches of inputs are classified as OoD. Although the test can be applied to batches of images at once (either all OoD or all in-distribution), we apply the test to single input images (i.e. batch size 1). For quantification, we vary the detection threshold to produce a receiver operator characteristic (ROC), and compute the area under this curve (ROC-AUC) in percent. For short, we call this the *OoD detection score*. It will be 100 for perfectly separated in- and outliers, and 50 if each point is assigned a random likelihood by the model.

OoD datasets: The inlier dataset consist of CIFAR10/100 images, i.e. 32×32 colour images showing 10/100 object classes. Additionally, we created four different OoD datasets, that cover different aspects, see Fig. 5.6. Firstly, we create a random 3D rotation matrix with a rotation angle of $\alpha = 0.3\pi$, and apply it to the RGB color vectors of each pixel of CIFAR10 images. Secondly, we add random uniform noise with a small amplitude to CIFAR10 images, to see if minor alterations of the image statistics can also be detected as OoD. Thirdly, we use the QuickDraw dataset of hand drawn objects (Ha and Eck, 2017), and filter only the categories corresponding to CIFAR10 classes and color each grayscale line drawing randomly. Therefore the semantic content is the same, but the image modality is different. Lastly, we downscale the ImageNet validation set to 32×32 pixels. This constitutes a dataset of natural images with very similar image statistics to CIFAR10, but this time with completely different semantic content. However, we find that none of the models can reliably detect the ImageNet data as OoD, so we use this primarily to measure the prediction entropy behaviour on OoD data.

5.3.3. Results

Quantitative comparison. A comparison of all models is performed in table 5.1 for CIFAR10, and in table 5.2 for CIFAR100. We find that at the extreme $\gamma \to \infty$, the model behaves almost identically to a standard feed forward classifier using the same architecture (i-RevNet), and for $\gamma = 0$, it closely mirrors a conventionally trained GC, as the bits/dim are the same. We find the most favourable setting to be at $\gamma = 1$: Here, the classification error and the bits/dim each only suffer a 10% penalty compared to the extremes. The uncertainty quantification for IB-INN at this setting (calibration and OoD prediction entropy) is far better than for pure DCs. Against expectations, standard GCs also have worse calibration error. Our hypothesis is that their predictions are too noisy and inaccurate for a positive effect to be visible. For OoD detection, the IB-INN and standard GCs are all comparable, as we would expect from the similar bits/dim. Figure 5.7, and figure 5.12 in more detail, show the trade-off between the two extremes: at low γ, the OoD detection and uncertainty quantification are improved, at the cost of classification accuracy. The VIB behaves in agreement with the other DCs: it has consistently lower classification error but higher calibration error than the IB-INN. This confirms that the IB-INN's behaviour is due to the application of IB to GCs exclusively, not the IB-principle alone. This does not mean that the IB-INN should always be preferred over VIB, or vice versa. The main advantages of the VIB are the increased robustness to overfitting and adversarial attacks, aspects that we do not examine in this thesis.

Table 5.1.: Results on the CIFAR10 dataset. All models have the same number of parameters and were trained with the same hyperparameters. All values except entropy and overconfidence are given in percent. The arrows indicate whether a higher or lower value is better.

Model		Classif. err. (↓)	Bits/dim (↓)	Calibration error (↓)				Incr. OoD prediction entropy (↑)					OoD detection score (↑)				
				Geo. mean	ECE	MCE	ICE	Average	RGB-rot	Draw	Noise	ImgNet	Average	RGB-rot	Draw	Noise	ImgNet
IB-INN (ours)	$\gamma = 1$	10.27	5.25	**1.26**	**0.54**	**3.25**	**1.13**	**0.38**	**0.43**	**0.40**	**0.10**	**0.61**	68.76	**78.80**	67.30	77.19	54.59
	only $\mathcal{L}_X$ ($\gamma = 0$)	–	**4.80**	–	–	–	–	–	–	–	–	–	74.51	70.68	85.74	91.14	55.82
	only $\mathcal{L}_Y$ ($\gamma \to \infty$)	8.72	17.27	3.98	0.81	13.94	5.57	0.28	0.23	**0.40**	0.00	0.49	61.25	57.04	**90.29**	50.24	54.40
Stand. GC	Class-NLL	61.75	4.81	12.61	4.17	30.58	15.70	0.03	0.02	-0.06	0.02	0.12	**73.92**	70.65	83.31	**90.97**	55.76
	Class-NLL + regul.	40.04	4.83	24.75	7.13	70.63	30.11	0.01	0.00	-0.01	0.01	0.02	74.02	69.33	85.13	91.04	55.88
Pure DC	VIB ($\gamma = 1$)	6.83	–	6.66	0.81	26.56	13.75	0.17	0.14	0.23	0.00	0.32	–	–	–	–	–
	ResNet	**6.51**	–	6.23	0.76	29.29	10.92	0.18	0.16	0.20	0.00	0.34	–	–	–	–	–
	i-RevNet	9.22	–	4.19	0.79	16.68	5.54	0.24	0.09	0.38	0.00	0.51	–	–	–	–	–

Table 5.2.: Results on the CIFAR100 dataset. All models have the same number of parameters and were trained with the same hyperparameters. All values except entropy and overconfidence are given in percent. The arrows indicate whether a higher or lower value is better.

Model		Classif. err. (↓)	Bits/dim (↓)	Calibration error (↓)				Incr. OoD prediction entropy (↑)					OoD detection score (↑)				
				Geo. mean	ECE	MCE	ICE	Average	RGB-rot	Draw	Noise	ImgNet	Average	RGB-rot	Draw	Noise	ImgNet
IB-INN (ours)	only $\mathcal{L}_X$ ($\gamma = 0$)	–	**4.82**	–	–	–	–	–	–	–	–	–	70.03	63.35	87.45	85.12	50.99
	$\gamma = 0.1$	42.57	4.94	**2.60**	**0.58**	**7.04**	**4.28**	0.50	0.66	0.28	**0.35**	0.69	68.31	**66.53**	78.91	81.70	50.75
	only $\mathcal{L}_Y$ ($\gamma \to \infty$)	33.78	18.44	4.49	0.62	16.76	8.72	0.58	0.52	1.04	0.00	**0.77**	58.29	47.95	**99.37**	49.23	49.23
Stand. GC	Class-NLL	97.92	**4.82**	16.20	1.02	95.63	43.53	-0.04	-0.14	0.55	-0.53	-0.03	**70.26**	64.68	86.54	**85.19**	**51.09**
	Class-NLL + regul.	69.28	5.07	13.94	0.75	89.74	40.15	0.00	-0.00	-0.01	0.01	0.01	68.83	64.96	82.19	83.32	50.44
Stand. DC	ResNet	**29.27**	–	5.13	0.65	20.57	10.16	**0.60**	**0.68**	0.97	-0.00	0.74	–	–	–	–	–
	i-RevNet	37.54	–	5.18	0.63	19.85	11.09	0.51	0.32	**1.00**	-0.00	0.75	–	–	–	–	–

For CIFAR100 (table 5.2), the general behaviour is similar to that on CIFAR10: The IB-INN model which balances both loss terms performs significantly better in terms of uncertainty calibration than both standard GCs and DCs. It also performs OoD detection almost as well as pure GCs, with a much better classification error. There are two differences compared to the CIFAR10 case: Firstly, in terms of increase in predictive entropy on OoD data, there are much smaller differences between models (excluding the standard GCs). The standard ResNet has the best overall performance by a small margin. Note that the increase in prediction entropy is also influenced by the calibration and overall classification error of the model to some degree, so we are careful in drawing any conclusions from minor differences. Secondly, we find that the most advantageous trade-off regime is now at a lower value of γ. The only values trained for CIFAR100 were $\gamma \in \{0.1, 1, 10\}$, and we find that the models with γ set to 1 and 10 behave almost the same as the limit case $\gamma \to \infty$. The explanation for this is simple: due to the increased difficulty of the task, the $\mathcal{L}_Y$ loss is higher than for CIFAR10. Therefore, it has a larger influence at the same setting for γ compared to the CIFAR10 models.

Latent Space Exploration To better understand what the IB-INN learns, we analyze the latent space in different ways. Firstly, Fig. 5.8 shows the layout of the latent space GMM through a linear projection, chosen so that the projected class centers have the largest possible spread. We find that the clusters of ambiguous classes, e.g. truck and car, are connected in latent space to account for the inherent uncertainty of this decision. Secondly, Fig. 5.10 shows interpolations in latent space between two test set images, using models trained with different values of γ. We observe that for low γ, the IB-INN has a well structured latent space, leading to good generative capabilities and plausible interpolations. For larger γ, class separation increases and interpolation quality continually degrades. Finally, generated images can give insight into the classification process, visualizing how the model understands each class. If a certain feature is not generated, this means it does not contribute

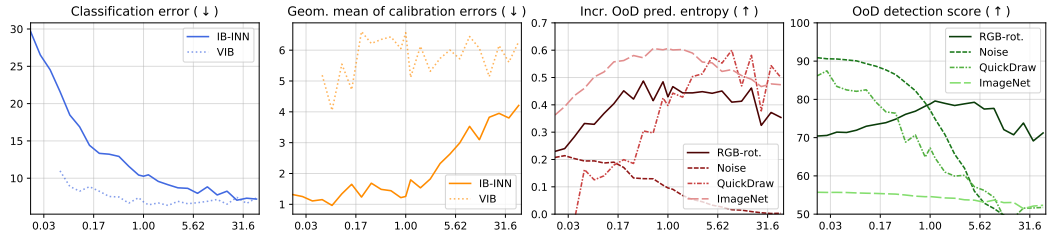


Figure 5.7.: Effect of changing the parameter γ between 0.02 and 50 (logarithmic x-axis) on the different performance measures (y-axis). The left two plots show the IB-INN and VIB, the right two plots only show the IB-INN. The VIB does not converge for $\gamma < 0.05$. The arrows indicate if a larger or smaller score is better. While classification accuracy improves with γ, the uncertainty measures generally grow worse. The trend of OoD detection and OoD entropy is less clear, and depends on the OoD dataset. The special case $\beta = 1$ (class-NLL) translates to $\gamma \approx 3 \cdot 10^{-4}$ (cf. table 5.1).

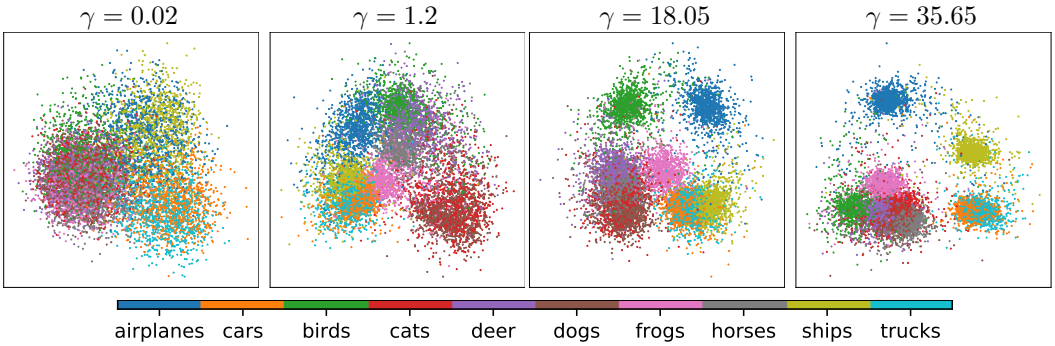


Figure 5.8.: GMM Latent space behaviour by increasing γ. The class separation increases with larger γ. Note that ambiguous classes (e.g. truck and car) remain connected to account for uncertainty.

positively to the likelihood, and in turn will be ignored for classification. Examples for this are shown in Fig. 5.9.

In Figure 5.13, we show the trajectory of a sample in latent space, when gradually increasing the angle α of the RGB-rotation augmentation used as an OoD dataset. It travels from in-distribution to out-of-distribution. Such images were never seen during training.

Choice of σ: Fig. 5.11 shows the behaviour for 25 different models trained with σ between 10^{-4} and 10^{0} (x-axis), and fixed $\gamma = 0.2$. We find that the loss values (left) and performance characteristics (middle) do not depend on σ below a threshold of about 10^{-3}, a value 4 times smaller than the quantization step size $\Delta X = 1/256$. Contrary to expectations from existing work on normalizing flows, the models performance does not decrease even when σ is 50 times smaller than ΔX. Detrimental effects might occur more easily if the quantization steps are larger, e.g. $\Delta X = 1/32$ as used by Kingma and Dhariwal (2018), or if the model were more powerful or less regularized (e.g. from the tanh-clamping we employ). The rightmost plot compares our approximation of $CI(X, Z_\varepsilon)$ with the asymptotic $I(X, Z_\varepsilon) + const.$ for $\sigma \to 0$, where the constant is unknown. The slope of the approximation agrees well for small σ, but breaks down for larger values.

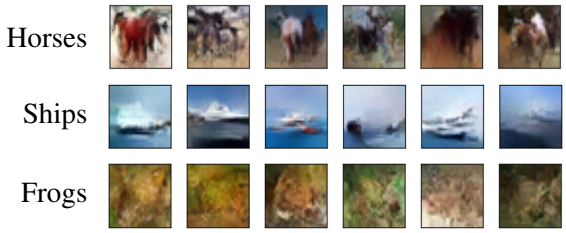


Figure 5.9.: Images are generated for three different classes, by sampling from the respective mixture component in latent space, and inverting the network (more examples in Appendix). This gives insight what happens during classification, see text: only textures are generated for the frog class, indicating that this is the only aspect used for classification.

Figure 5.10.: The columns show a latent space interpolation between two images (leftmost and rightmost). Each row shows a model with a different γ.

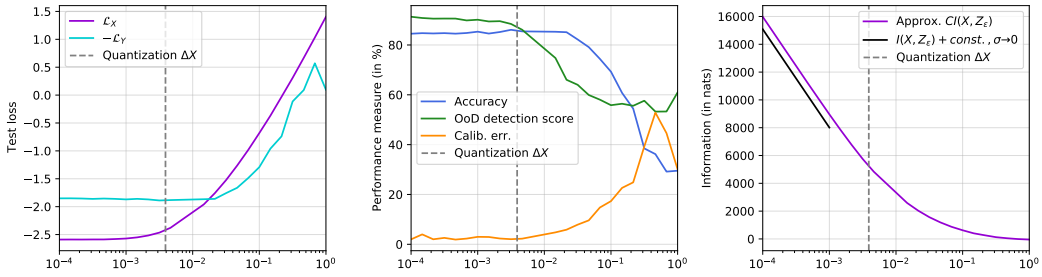


Figure 5.11.: From left to right: Changes in test-loss, performance metrics, and a comparison between approximation and known slope of the true mutual information for varying values of σ (x-axis)

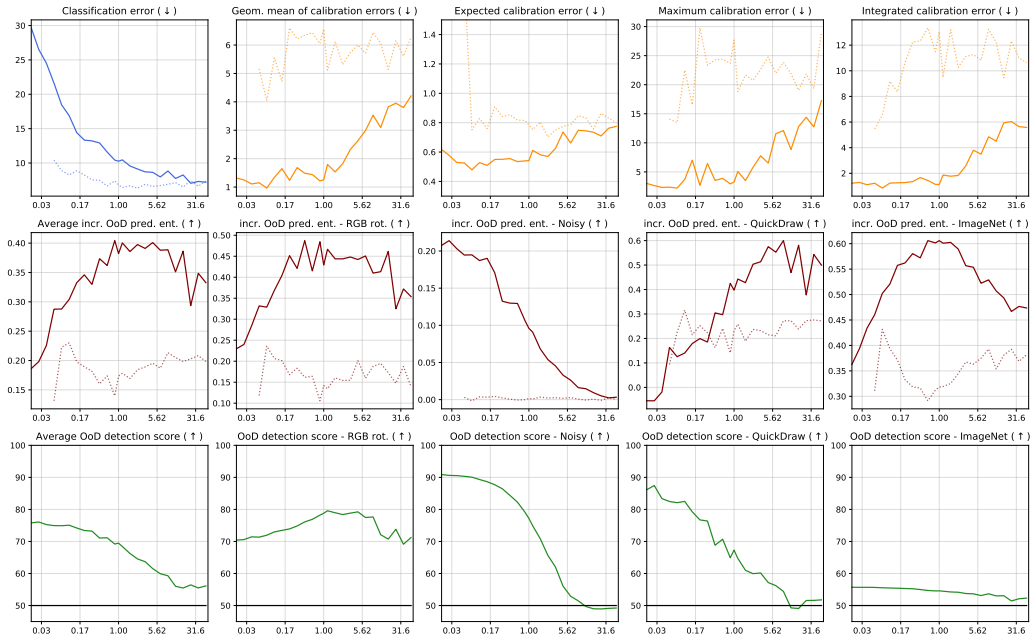


Figure 5.12.: Effect of changing the parameter $\tilde{\beta}$ (x-axis) on the different performance measures (y-axis). The arrows indicate if a larger or smaller score is better. The black horizontal line in the last row indicates random performance. The VIB results are added as dotted lines. The VIB does not converge reliably for values of $\gamma < 0.2$, producing some otiliers e.g. for expected calibration error. This is not to claim that the IB-INN is better than the VIB or vice versa. The comparison serves to show how the IB affects GCs and DCs differently.

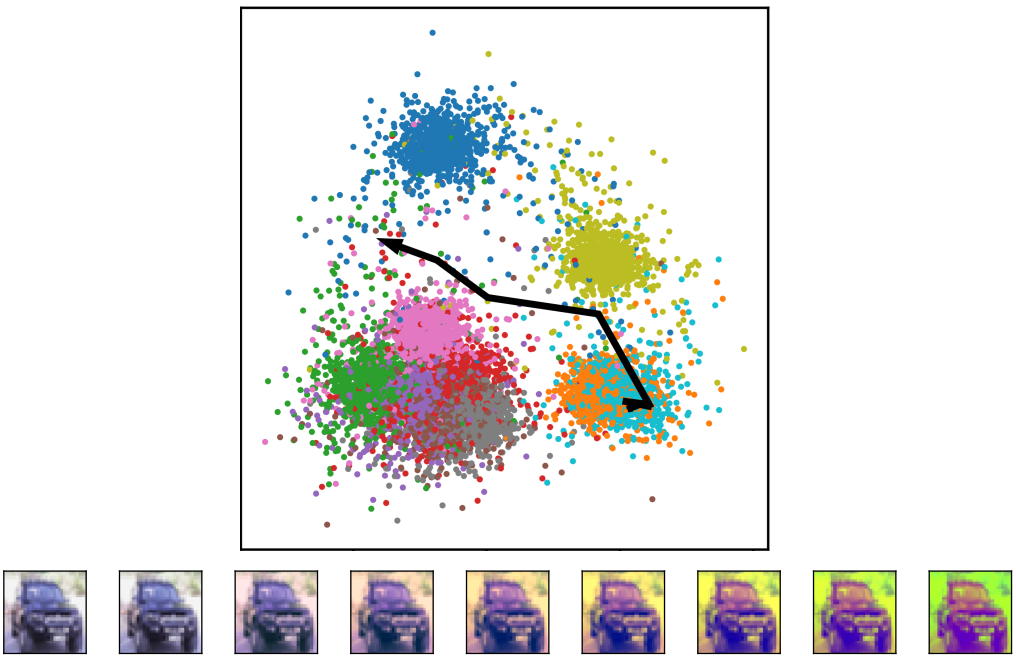


Figure 5.13.: The scatter plot shows the location of test set data in latent space. A single sample is augmented by rotating the RGB color vector as described in the text. The small images show the successive steps of augmentation, while the black arrow shows the position of each of these steps in latent space. We observe how the points in latent space travel further from the cluster center with increasing augmentation, causing them to be detected as OoD.

5.4. Experiments with ImageNet dataset

In the context of image classification or computer vision in general, the CIFAR images used in the previous section are very low-resolution and and the appearance and content of each class is comparitively uniform. In this way, the CIFAR data is not particulary representative of realistic real-world applications. Instead, we use much larger and more varied images in this section, namely from the ImageNet dataset (Russakovsky et al., 2015). Here, the input images are 224×224 pixels large, a dimensionality increase by a factor of 50, depicting 1000 different classes. To add additional complexity, some classes are so sililar that only human experts can distinguish them (such as three different breeds of Husky dogs), and others so dissimilar that they have almost nothing in common and never occur in the same context (such as cockroaches and oil tankers). As the data consist of natural images gathered semi-automatically from the internet, the images also span a range of image quality, appearence and image modality, as well as many images containing multiple objects despite having only one 'correct' label assigned.

The ImageNet dataset therefore serves as a more meaningful test of the capabilities of IB-INNs used as generative classifiers, representing a task complexity at which GCs have previously never been used to our knowledge.

5.4.1. Network Architecture

A more detailed description of the network architecture, reasoning for each choice and design principles is found in Mackowiak et al. (2021), we summarize the main points in the following. We construct the invertible network (INN) from affine coupling blocks as in previous chapters (Dinh et al., 2016), with various modifications from other recent works (Ardizzone et al., 2019a,b; Jacobsen et al., 2019; Kingma and Dhariwal, 2018). As invertible alternatives to 2×2 max-pooling and global mean-pooling, we use a Haar wavelet transform (section 3.5) and a DCT transform (Jacobsen et al., 2019) respectively.

Because of the similarities between affine coupling blocks and residual blocks as used in a ResNet, we match the design of the INN to that of a standard ResNet-50 wherever possible. The overall layout is summarized in table 5.3, c.f. (He et al., 2016, Table 1). Some differences arise due to the constraint of invertibility: the number of feature channels and the available receptive field vary between the two networks. The effective rather than maximum receptive field varies surprisingly, and is discussed further below. The invertibility is also associated with an extra cost of parameters and computation, summarized in table 5.4: Both in terms of network parameters, as well as FLOPs for one forward pass, the cost of the INN is about twice as high as a standard ResNet-50. We are optimistic that this overhead can be reduced in the future with more efficient INN architectures.

5.4.2. General Performance

We train several generative classifiers, with the following values for the hyperparameter $\gamma \in \{1, 2, 4, 8, 16, 32, \infty\}$. Again, γ controls how much the model focuses on the generative likelihood estimation aspect (low γ), vs. prioritizing good classification performance (high γ). In addition, we include a model trained with $\gamma = 0$, i.e. no classification at all, analogous to a standard normalizing flow, as well as a standard feed-forward ResNet-50 He et al. (2016), i.e. a pure DC.

Layer	Blocks	Im. size	Channels		R.F.	
			INN	ResNet	INN	ResNet
Input		224	3	3		
Entry flow	1	112	12	64	8	6
Pool (Haar/max)		56	48	64	10	10
Conv_2_x	3	56	48	256	34	34
Conv_3_x	4	28	192	512	106	90
Conv_4_x	6	14	768	1024	314	266
Conv_5_x	3	7	3072	2048	538	426
Pool (DCT/avg.)		1	150 528	2048	∞	∞

Table 5.3.: For each of the resolution levels in the INN and ResNet-50, the number of coupling/residual blocks and spatial size is given, along with the number of feature channels and the maximum possible receptive field (R.F.).

	ResNet	INN
Network parameters (M)	23.5	55.4
All parameters (M)	25.6	77.5
FLOPs (G)	4.07	9.08

Table 5.4.: Number of parameters and computational cost for each model. *'Network parameters'* only counts the coupling/residual blocks. *'All parameters'* additionally includes the fully connected output layer of the ResNet, and the parametrization of μ_y for the INN. The (M) and (G) indicates Mega and Giga respectively. For FLOPs, the fused multiply-add instruction (FMA) is counted as a single FLOP, as it is commonly a single instruction in modern computing architectures.

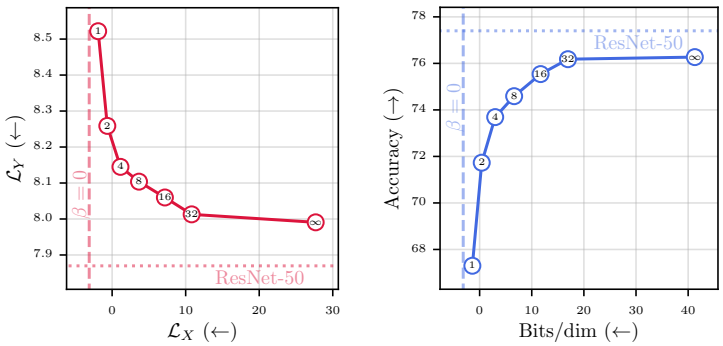


Figure 5.14.: Trade-off between the two losses $\mathcal{L}_X$ and $\mathcal{L}_Y$ (left), and between generative modeling accuracy in bits/dim, and top-1 accuracy (right). Each point represents one model, trained with a different γ. A standard ResNet has no $\mathcal{L}_X$ loss and is shown as a horizontal line. The model with $\gamma = 0$ (standard normalizing flow) is missing the $\mathcal{L}_Y$ loss and is shown as a vertical line. The small numbers inside the markers give the value of γ of that particular model.

The primary performance metrics used in Table 5.6 and Fig. 5.14 are firstly, the top-1 accuracy on the test set (in our case, the ILSVCR 2012 validation set (Russakovsky et al., 2015)). To compare with other methods, we adopt the same procedure for testing the performance, i. e. 10-crop testing. In detail, the image is resized so the shortest edge is 256 pixels, and then cropped to a square. Next, five 224×224-pixel patches are cropped from the corners and center, as well as their horizontally flipped versions, giving 10 crops in total. Typically, each of these 10 crops is passed through the network, and the logits are averaged before the final softmax operation. For a generative classifier, as there are no logits per se, the logit averaging is analogous to taking the geometric mean over the 10 input crops for both the denominator and enumerator of Bayes rule.

Secondly, for the generative likelihood estimation performance, we use the bits per dimension ('*bits/dim*') metric, as this is the prevalent evaluation metric for likelihood-based generative models such as normalizing flows. It quantitatively measures the accuracy of the density estimation (i.e. generative performance), explained e.g. in Theis et al. (2015), where a lower bits/dim corresponds to a more accurate generative model. Originating in models that estimate the density on discrete inputs (e.g. 255 brightness levels per RGB channel per pixel Oord et al. (2016b)), bits/dim measures how many bits would be needed on average to store the value of each input dimension, if an ideal encoding were formulated using the models estimated likelihoods. If the quality of the likelihoods is worse, the encoding will be less efficient w.r.t. to the real data, and more bits will be needed to store the inputs. This can be generalized for continuous density models such as standard normalizing flows, explained e.g. in Theis et al. (2015). Importantly, it can be shown that bits/dim is exactly proportional to the KL-divergence between the true and estimated densities, albeit with a constant offset:

$$\text{bits/dim} \propto D_{\text{KL}}(p(X) \,\|\, q_\theta(X)) + const. \tag{5.29}$$

However, the constant offset is fundamentally unknown and data dependent, so while bits/dim is a strong quantitative metric to compare different models against each other, it cannot quantify how good the model is on an absolute level.

In Table 5.6, we report the test losses and the two discussed performance metrics for the different models. Further shown in Fig. 5.14, changing γ moves smoothly between the limit cases of a feed-forward network, and a pure density estimation model along a pareto curve. The classification accuracy increases continuously with γ, but a minor gap remains to the feed-forward ResNet-50, in line with works such as Jacobsen et al. (2018). Simultaneously and as expected, the bits/dim get worse as we move away from a purely generative model ($\gamma = 0$).

Receptive field. While the maximum possible receptive field of the INN and a standard trained ResNet are roughly comparable (see Mackowiak et al. (2021) for precise computation), we see large differences in the effective receptive field.

To measure an effective receptive field, we determine the sensitivity of latent dimensions with respect to changes in each input pixel. Note that there is not universal definition of the effective recepetive field, so the following procedure is just one of several possibilities. We first pick a feature space column u, in the fatures before the DCT pooling layer. With the feature space having $H \times W \times 3072$ feature space, u is a $1 \times 1 \times 3072$ column. We choose a column from the center to avoid interactions with the edges. The individual features in the column are denoted u_l ($l = 1 \ldots 3072$). We now measure the gradient w.r.t.

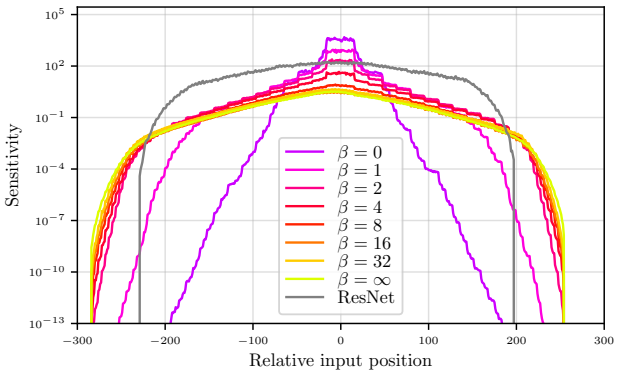


Figure 5.15.: Effective receptive field for each value of γ, just before the final pooling operation. Note the logarithmic sensitivity axis.

	IB-INN, $\beta =$							RN
	1.0	2.0	4.0	8.0	16.0	32.0	∞	
ECE (%)	0.16	0.16	0.16	0.17	0.16	0.17	0.17	0.17
MCE (%)	5.54	3.13	5.47	4.57	5.50	5.28	5.10	7.72
OCE	3.87	4.13	4.31	4.73	4.15	4.94	5.12	6.75

Table 5.5.: Calibration Errors for different values for γ and for the ResNet. Expected Calibration Error (ECE), Max Calibration Error (MCE), Overconfidence Calibration Error (OCE)

each channel k of each image pixel ij, x_{ijk}, for real input images x. We define the 'latent sensitivity' of the model at each position as the L_1 norm of the gradient of the features w.r.t. that input position, averaged over images from the test set:

$$\text{Sensitivity}(i,j) = \mathbb{E}_{x \in \text{test}} \left[\sum_{k=1}^{3} \sum_{l=1}^{3072} \left| \frac{\partial u_l}{x_{ijk}} \right| \right] \tag{5.30}$$

Note that the choice of the L_1 norm does not change the nature of the results, other choices result in similar profiles. The cross-sectional shape of this latent sensitivity over image coordinates represents the effective receptive field, and is shown in Fig. 5.15.

We observe that for low γ, the effective RF is very narrow. In fact it is almost as narrow as it could possibly be: for $\gamma \leq 4$, the FWHM of the sensitivity is only 64 pixels. This is the same we would get from only the downsampling steps, without any spatial convolutions (with 6 downsamplings, $2^6 = 64$). This could indicate that for the likelihood estimation, local details and structures are more important than any long-range features. This has recently also been confirmed in Kirichenko et al. (2020). It also gives a possible explanation for the trade-off in classification accuracy, as a higher receptive field would be beneficial to understanding the semantic image content. For higher values of γ, the shape of the effectiv receptive field more closely matches that of a standard trained ResNet (1.25 times wider, in line with the 1.25 times larger maximum possible RF).

Calibration Lastly, we examine the uncertainty calibration. We find that for such a large number of classes as ImageNet, the usual error measures are of little use: Especially the ECE is almost completely dominated by the many classes with predicted probabilities close

β	$\mathcal{L}_X^{(\text{test})}$ ($\downarrow$)	$\mathcal{L}_Y^{(\text{test})}$ ($\downarrow$)	Bits/dim ($\downarrow$)	Acc. (%) ($\uparrow$)	OCE ($\downarrow$)
1	-1.90	8.52	4.34	67.30	3.87
2	-0.65	8.26	6.14	71.73	4.13
4	1.14	8.14	8.72	73.69	4.31
8	3.66	8.10	12.35	74.59	4.73
16	7.17	8.06	17.43	75.54	4.15
32	10.81	8.01	22.68	76.18	4.94
∞	27.68	7.99	47.01	76.27	5.12
0	-3.11	–	2.59	–	–
ResNet	–	7.87	–	77.40	6.75

Table 5.6.: Test losses and metrics for models trained with different γ. Bits/dimension quantifies the performance of density estimation models (see text, smaller is better, i.e. more accurate generative model). As with the original ResNet, the classification accuracy uses 10-crop testing. OCE is the overconfidence error, i.e. how often confident predictions are wrong (see text, smaller is better).

to zero. Instead, we additionally introduce the overconfidence error 'OCE', wich only examines predictions with especially high confidence, which are of special importance for real world saftey and reliability. The OCE measures the normalized classification error of predictions with a high confidence over some threshhold $C \geq C_{\text{crit}} = 99.7\%$. Then, if the error rate in these cases is 1.2% for instance, although it should be at most 0.3% according to the chosen threshhold, this gives an OCE of $1.2/0.3 \approx 4$. Our findings are in line with previous works, in that the uncertainty calibration improves with lower γ and better generative capabilities Ardizzone et al. (2020b). A more detailed examination is found in the publication (Mackowiak et al., 2021), and not included in the thesis.

5.4.3. ImageNet out-of-distribution detection

While OoD detection was already performed in the previous section for CIFAR data in line with existing literature, we repeat the examination for ImageNet data in the following. We find that some of the phenomena commonly observed and discussed on CIFAR images (Nalisnick et al., 2019b; Kirichenko et al., 2020, etc.) are further amplified by the 50-fold increase in data dimensionality provided by ImageNet, so that the work in this section can help further understand and characterize these phenomena. We will first consider how to describe most existing OoD detection methods in a common hypothesis-testing framework, before applying this framework to detect different types of OoD data in ImageNet.

OoD detection as hypothesis test

For likelihood-based generative models, detecting OoD inputs is straight forward, by directly utilizing the estimated probability density q_θ: in principle, if an input is outside the support of the training data, and the model has learned the true distribution, the OoD sample should be assigned $\log q_\theta(x) = -\infty$. In practice, it is only required that OoD samples have lower likelihood scores than the training data. From here, any input with an inferred likelihood below a threshold can be treated as OoD. However, Nalisnick et al. (2019b) identify various special cases where OoD inputs have an unnaturally high log-likelihood score. This

prompted the development of a *typicality-test* in Nalisnick et al. (2019a), that uses both an upper and a lower threshold. Other works take similar approaches, modifying the score or changing the way the detection threshold is applied, such as Choi et al. (2018); Serrà et al. (2019); Song et al. (2019a); Zhang et al. (2020).

In the following, we note how any threshold-based OoD detection can be treated as a hypothesis-test, unifying existing approaches: Mathematically, a binary, threshold-based detection of any kind is equivalent to a hypothesis, with a p-value equal to the false-positive rate. However, the hypothesis-testing interpretation corresponds especially tightly to the OoD setting, as by nature of the problem only the scores of in-distribution data are explicitly known beforehand, corresponding to only the null-hypothesis being explicitly modeled in hypothesis testing.

In other words, the null-hypothesis is that an input is in-distribution, and if the score for this input lies above or below the set threshold, this hypothesis is rejected, with a p-value corresponding to the false-positive rate. The false-positive rate can be determined solely from the in-distribution data regardless of the type of OoD data (just as the p-value is not reliant on an explicit choice of alternative hypothesis).

Interpreting OoD detection in this context, including existing works, has some important implications. The formulation allows for the use of any kind of scalar value extracted from each input. Many existing works try to provide some theoretical underpinning of using a given score (Choi et al., 2018; Nalisnick et al., 2019a), each demonstrating superior detection capabilities in one aspect or another, claiming this is what is 'truly' out-of-distribution. However, in the hypothesis-testing framework, there is no fundamental difference between these scores and any other heuristically picked score, such as Schirrmeister et al. (2020); Serrà et al. (2019). As Kirichenko et al. (2020) point out, what is OoD should and does depend on human interpretation and the requirements of each application.

With our newfound interpretation of OoD detection as a hypothesis test, we also present a new detection method. In the same vein as the typicality test, we choose both an upper and a lower threshold. In the case of the typicality test, the thresholds are centered symmetrically around the mean log-likelihood of the training data (Fig. 5.16, middle). For our ImageNet models, we observe that the distribution of log-likelihood values in the training set is highly asymmetrical. Therefore, we determine the upper and lower thresholds according to equal quantiles on either side (Fig. 5.16, bottom).

To evaluate the OoD detection capabilities independent of the threshold, we vary the p-value of the test and produce a receiver operating characteristic (ROC) curve. The area under this curve (ROC-AUC), in percent, serves as a scalar measurement of the OoD detection capabilities, with ROC-AUC of 100% meaning that the OoD samples and in-distribution samples are perfectly separated, and a value at 50% or below indicates a random performance or worse.

Semantic- and style-likelihood

As the number of classes is typically smaller than the number of total dimensions (in the case of ImageNet, 1000 classes and $\approx 150\,000$ dimensions), the class centroids μ_y span a lower dimensional subspace in latent space, which we will index with c for 'class'. The corresponding null-space, indexed by s for 'style', does not affect the classification, mathematically speaking. In the following, we assume that all μ_y are linearly independent. This is

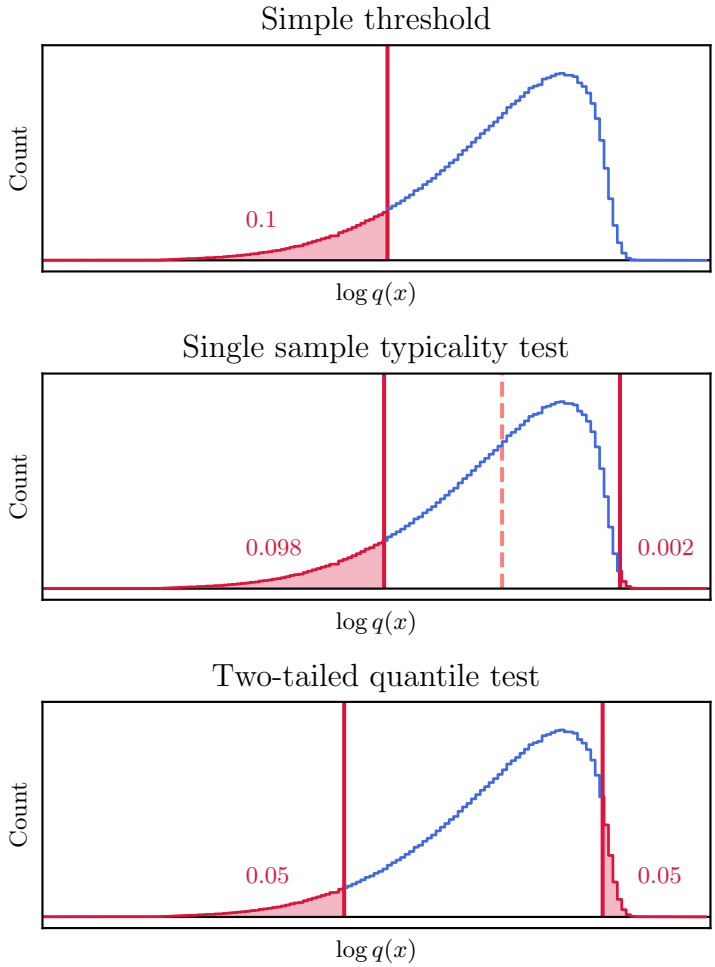


Figure 5.16.: Illustration of three different OoD tests based on the estimated likelihood. The curve shows the distribution of likelihood scores in the training set. The blue part counts as in-distribution, and the red part as OoD. The threshold is chosen such that the red area (false positive rate, p-value), is 0.1 in all three cases, for illustration. In practice, this would be chosen much lower, e.g. 0.01. The small red numbers indicate the fraction of training samples above and below each threshold. Top: simple single-tailed test, used e. g. by Nalisnick et al. (2019b). Middle: typicality test (Nalisnick et al., 2019a), the dashed line indicates the 'typical' score. Bottom: two-tailed quantile test, introduced and used in this chapter.

almost certainly the case in practice, but we still check this explicitly for each model. With the two linear subspaces, we can decompose z into two parts $z = z_c + z_s$, where z_c and z_s each lie in their respective subspaces. Importantly, because the null-space is always orthogonal to the row-space, it holds that $\|z\|^2 = \|z_c\|^2 + \|z_s\|^2$. Using this, we can decompose the conditional likelihood in latent space, $p(z \mid y)$. We write the normalization constant as A for convenience:

$$p(z \mid y) = A \exp\left(-\frac{1}{2}\|z - \mu_y\|^2\right) \tag{5.31}$$

$$= A \exp\left(-\frac{1}{2}\left(\|z_c - (\mu_y)_c\|^2 + \|z_s - (\mu_y)_s\|^2\right)\right) \tag{5.32}$$

$$= A \exp\left(-\frac{1}{2}\|z_c - \mu_y\|^2\right) \exp\left(-\frac{1}{2}\|z_s - 0\|^2\right) \tag{5.33}$$

$$= p(z_c \mid y)p(z_s) \tag{5.34}$$

Intuitively, we have factorized the likelihood in latent space into a class-term $p(z_c|y)$, and a class-independent style-term $p(z_s)$. This factorization is not only significant for the task of OoD detection, but also for many explainability techniques in the subsequent section.

Detecting ImageNet OoD

By definition, it is not possible to collect a representative and exhaustive set of possible out-of-distribution images, making quantitative method comparisons difficult. We therefore omit a quantitative evaluation in this section of the thesis, and instead refer to the work of Mackowiak et al. (2021): there, a pragmatic approach is taken, comparing the OoD detection quantitatively on some hand-selected, practically relevant and agreed-upon types of outliers, namely different kinds of image corruptions. In this section, we instead aim to show qualitatively which kinds of outliers are detected when using different scores.

To achieve this, we perform the following experiment: we simply perform OoD detection on the ImageNet test set, even though it is by definition in-distribution. We then visualize the false positives detected most clearly as out-of-distribution, i. e. the test images on the outer edges of the score-distribution, cf. figure 5.16. The characteristics observed in these false positives provide information about the type of outliers each score is sensitive to.

Qualitative results of this experiment are given in figures 5.17 to 5.19, showing the 10 most outlying false positives among the 50 000 ImageNet test set images.

When using the estimated log-likelihood as a score in figure 5.17, we find that out-of-distribution images are recognized almost entirely on the basis of texture, either highly textured with an abnormally low likelihood, or very smooth and low-contrast resulting in an abnormally high likelihood. Several other authors (Kirichenko et al., 2020; Serrà et al., 2019) seem in agreement that this behaviour is not due to some shortcoming or artifacts of the model, but simply a result of the structure and composition of the likelihood of such high-dimensional spaces: pixel-level statistics largely determine the magnitude of the likelihood, while properties of interest such as the semantic content of an image, hardly contribute. By decomposing the log-likelihood into its constituent terms ($\log p(z) + \log\det J$), we find that the Jacobian causes the large difference between highly-textured and smooth images. Confirmed by some of the methods in the next section, it seems that textures or

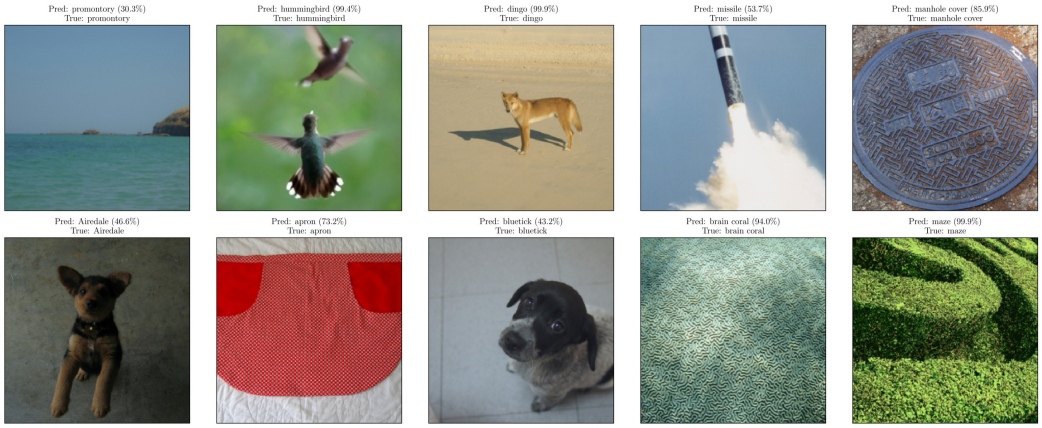


Figure 5.17.: 10 most outlying images according to out-of-distribution detection with a two-tailed quantile hypothesis test using the modeled log-likelihood of the image $\log q(x)$ as a score. This score is generally accepted as a sensible baseline, but leads to unexpected results for high-resolution images, only detecting examples with abnormally high- or low-detail textures. Note how the images are all correctly classified and properly understood by the model.

structures in the image are 'normalized' to a similar amplitude in latent space, i.e. low-contrast textures are amplified (large Jacobian), opposite for high-contrast textures.

By using only the latent log-likelihood $\log p(z)$ as a score, we can expect to avoid this effect, as the Jacobian does not contribute. The detected false positives for this choice are shown in figure 5.18: rather than detecting images with much or little information at a pixel-level, we now recover images with especially high or low information in the latent representation. This corresponds to images with very varied content and many objects in the same image, or none at all, see top right.

Finally, we restrict the score to only use the likelihood in the class-subspace, $\log p(z_c)$. This results in the most surprising findings of all: the model very reliably and reproducibly detects images that contain many instances of the same class. Also note that the predictions are fairly confident and mostly correct, despite being extreme outliers according to this score. It seems this effect occurs due to each object moving the latent code along a vector pointing towards a class. For many of the same object, the latent code is moved far past the class centroid to the outer regions of class-subspace, while still giving a confident prediction (c.f. figure 5.2, pushing the point out of the bottom right corner of the figure).

5.4.4. Explainability

In the following, we demonstrate several examples on how GCs can be used for native and intuitive explanations of the data and the prediction outputs. Certainly, algorithms and approaches exist that can generate similar results for DCs. The point of the following examples is to show that in GCs a range of explanations is available using only the structure of latent space and the learned likelihoods, without requiring additional modifications or algorithms applied in a post-hoc manner.

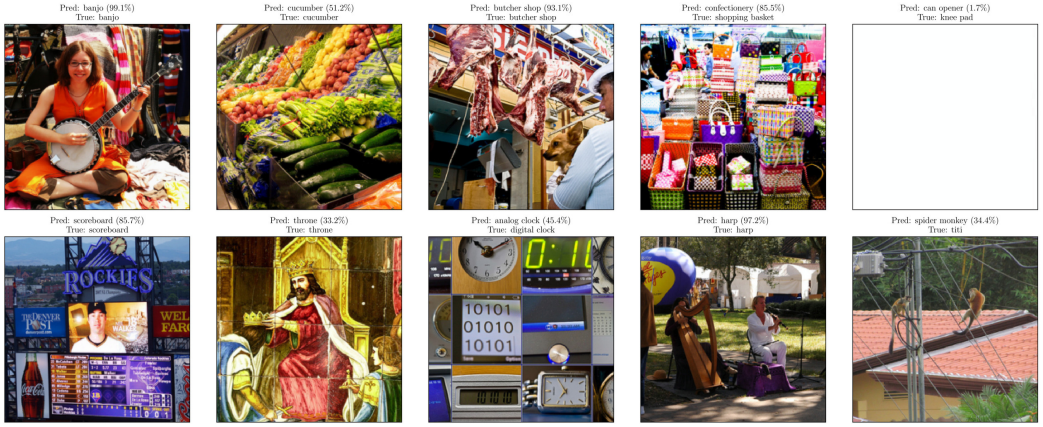


Figure 5.18.: 10 most outlying images when detecting using a two-tailed hypothesis test with either the latent log-likelihood $p(z)$ or the log-likelihood in the style-subspace, $\log p(z_s)$. The results are the same for both, as $\log p(z_c)$ does not make a significant numerical contribution. The detected images are in all cases especially rich or poor in visual details.

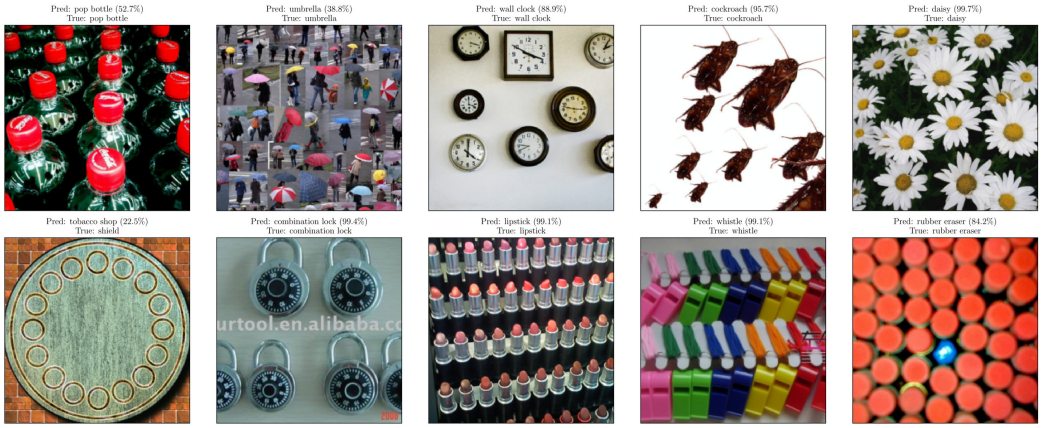


Figure 5.19.: 10 most outlying images when only using the log-likelihood of the class-specific latent subspace $\log p(z_c)$. Perhaps most surprisingly of all, the images detected are one depicting in-distribution objects repeated many times in the same image.

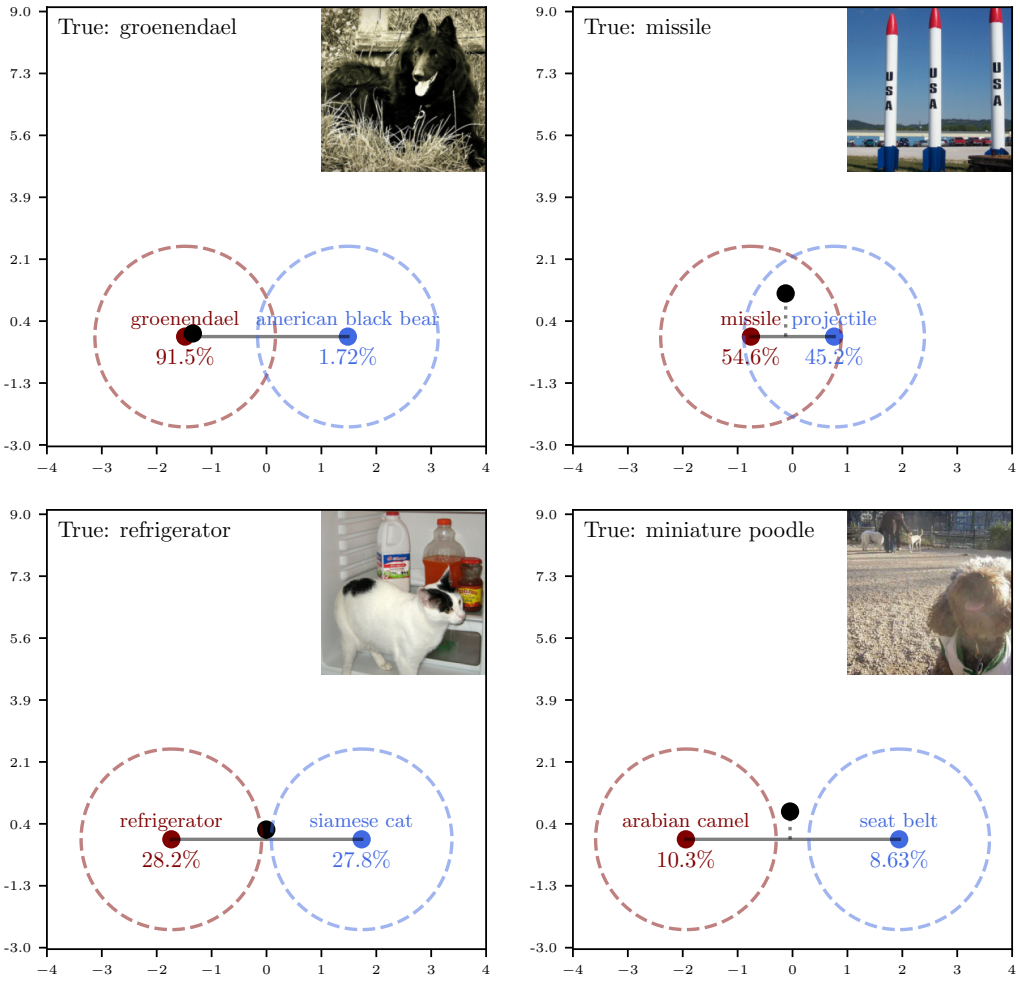


Figure 5.20.: Latent space location of input images (*black point*) in the decision space spanned by the μ_y of the top 5 predicted classes. The horizontal axis of the plot is the axis connecting the top 2 predicted classes (*red and blue points*). The vertical axis of the plot shows the radial distance from the horizontal axis in the 5D space. The illustrative circles are chosen such that in both the vertical and horizontal directions, 90% of the mass of the Gaussian mixture component lies inside. Note that the axes in the plot are scaled differently to make it appear as a circle. Test examples from left to right: a confident in-distribution decision, an uncertain in-distribution decision due to ambiguous classes, an uncertain decision due to multiple plausible image interpretations, an uncertain out-of-distribution decision.

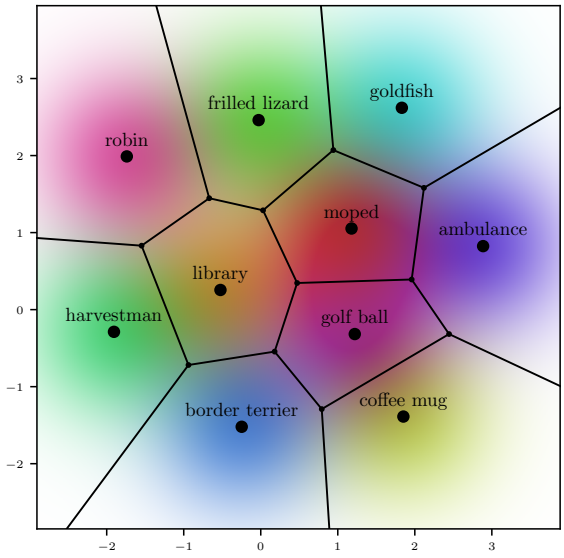


Figure 5.21.: Latent space of a model with only ten classes, where the μ_y (black points) are constrained to a plane. The black lines are the decision boundaries, e.g. all points inside the 'moped'-polygon will be classified as a moped. The background is colored according to the probability density of each mixture component.

Visualizing decision-space:

The properties of a classification decision are fully determined by the latent code of an input image in relation to the surrounding classes. The only difficulty consists in reducing the high-dimensional latent space to a 2D plot. Figure 5.20 shows one possibility: latent codes are visualized in a plane through the centers of the two most probable classes, such that relative distances to the centers and to their connecting axis are preserved. We find that the GC latent space allows making meaningful distinctions between different kinds of uncertainties, as illustrated by the examples.

A second approach is shown in figure 5.21, where the classification among a subset of classes can be fully visualized: First, a subset of 10 ImageNet classes is selected at random for the task. We leave the network as is, and only retrain the μ_y, under the constraint that they lie on a plane. In practice, we achieve this by parametrizing μ_y as a rank-2 matrix. This simplified model with a 2D decision space reaches 90% accuracy for the 10-class classification, while allowing us to show the entire decision space in a single 2D plot. The decision boundaries between all classes form a Voronoi tessellation of the decision space. All latent vectors inside the Voronoi cell of a certain class will have the highest probability under that class. In the full model, where the μ_y are not constrained to a plane and all 1000 classes are used, the behaviour is the same, with high-dimensional polygons for each class, but this can not be readily visualized.

Class similarities:

Building on figure 5.20, we see that different classes have various amounts of overlap, encoding the relationship between them. This is not possible for a feed-forward model, as there is no latent space where the input data is embedded in such a way. We observe that

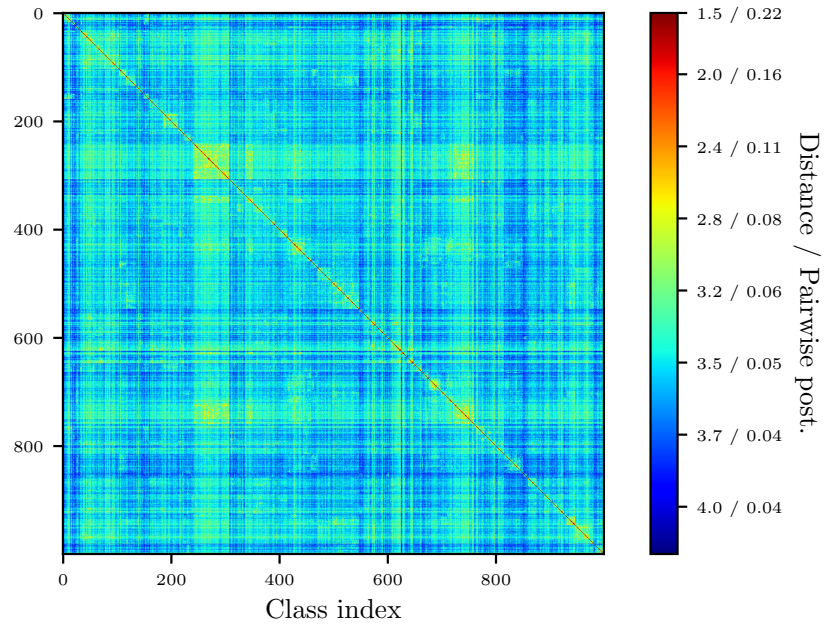


Figure 5.22.: Similarity matrix between all 1000 classes. The two large clusters around class index 250 and 750 are all dog breeds, which are notoriously overrepresented in the selection of ImageNet classes. The color-map indicates the pairwise distance of the μ_y as well as the expected pairwise posterior, meaning e.g. the binary decision between a tabby cat and a tiger cat is associated with 20% expected uncertainty, by construction (see text).

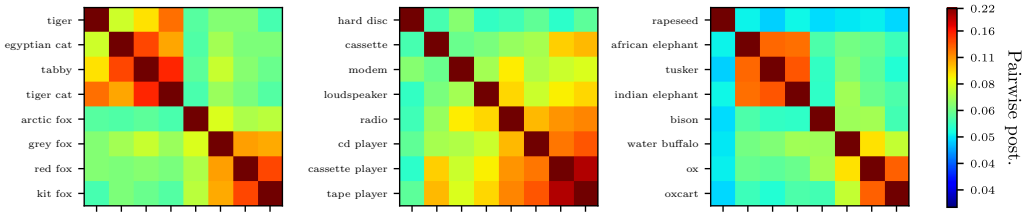


Figure 5.23.: Latent similarity between different classes. The colormap indicates the pairwise distance of the μ_y as well as the expected pairwise posterior, meaning e.g. the binary decision between “tabby cat" and “tiger cat" is associated with 20% expected uncertainty, by construction (see text). The distance on the diagonal is 0 (outside colormap range).

Figure 5.24.: Examples of the prediction heatmaps. Summing the bright areas directly gives the final class prediction. Top: bowtie and sunglasses are located, suit is distributed over a large area. Middle: The head of the bird causes it to be classified as a limpkin, whereas the feathers are more indicative of an eagle or vulture. Bottom: The heatmaps of both Rottweiler and Appenzeller classes are located in the same area (ambiguous classes), while the soccer ball is separate.

the locations μ_y of the Gaussian mixture components are close together for classes that are semantically similar, and far apart for classes that are dissimilar.

Importantly, this also has implications for predictions the model makes. For instance, in figure 5.20, top right, the classes overlap a lot. This means more points will lie in the overlap zone, and consequently more of these decisions will be uncertain, compared to e.g. bottom left, where most inputs will be in only one of two classes. Any inlier of one of the two classes will automatically have a significant probability under the other class.

More precisely, the closer two class centers are, the larger is the overlap, and the larger the proportion of decisions that are uncertain. In fact, if a class A is the top prediction, the expected confidence for any other class B can be worked out explicitly from the distance between μ_A and μ_B in latent space,

see Appendix C.5. We do this pairwise for all 1000 classes, with the results seen in Fig. 5.22. Some more detailed examples are shown in figure 5.23.

These considerations highlight an important fact: the latent mixture model contains a built-in uncertainty between classes. A decision between semantically related classes will always be uncertain, by the structure of the latent space alone. This may be one of the reasons explaining why the predictive uncertainties are better calibrated in such GCs.

Posterior Heatmaps:

To increase the trust in a decision, it is often helpful to show which regions of the image were relevant. Examples are widespread where models base their decision e.g. on the background of the image, not the object in question, or focus only on a spurious specific detail that identifies an object. Approaches such as CAM or GradCAM Zhou et al. (2016); Selvaraju et al. (2017) can be used to generate coarse heatmaps showing regions that are influential for a particular decision. With the IB-INN, we can provide such heatmaps as a direct decomposition of the prediction output, meaning they can be understood simply as a different way of representing the model output, rather than a post-hoc explanation technique.

To produce a spatially structured output, we consider the following: Due to the invertibility of every part of the model, we can start from the output z, and transform it back through the DCT operation. Unlike standard mean-pooling, the DCT pooling does not lose any information in either direction. We define the following for short:

$$\begin{aligned} w^{(y)} &= \mathrm{DCT}^{-1}(z) - \mathrm{DCT}^{-1}(\mu_y) \\ &= \mathrm{DCT}^{-1}\big(z - \mu_y\big). \end{aligned} \tag{5.35}$$

Importantly, $w^{(y)}$ has the spatial structure of the final convolutional outputs: It will have three indices, k, l for the spatial position and m for the feature channels: $w^{(y)}_{klm}$.

Because the DCT is linear and orthogonal, it conserves distances, i.e. $\|z - \mu_y\| = \|w^{(y)}\|$, This allows us to factorize the latent log likelihood over the spatial indices.

$$p(z|y) \propto \exp\left(-\frac{\|w^{(y)}\|^2}{2}\right) = \exp\left(-\sum_{kl}\frac{\left(w^{(y)}_{kl}\right)^2}{2}\right), \tag{5.36}$$

leading to the factorization

$$p(z|y) = \prod_{k,l} p(w_{kl}|y) \tag{5.37}$$

where

$$p(w_{kl}|y) \coloneqq \mathcal{N}(w_{kl}^{(y)}; 0, 1) \tag{5.38}$$

In a similar way, we can also decompose the posterior rather than the latent likelihood using a few extra steps, This leads to our heatmap $Q_{\text{Class}}(k, l, y)$, that sums to the class posterior over space in the same way as in equation (5.36):

$$q_\theta(y|x) = \prod_{kl} \exp\left(Q_{\text{Class}}(k, l, y)\right) \tag{5.39}$$

$$= \exp\left(\sum_{kl} Q_{\text{Class}}(k, l, y)\right). \tag{5.40}$$

Q_{Class} has a single hyperparameter that adjusts the contrast of the heatmaps, but equation above holds in any case. The derivation is given in Appendix C.6. Examples are shown in figure 5.24.

5.5. Conclusions

To conclude, this chapter demonstrated the strength of invertible generative classifiers, that is by enhancing uncertainty quantification, out-of-distribution detection, and explainability.

While the chapter only focussed on classification, we find that the method can be readily extended to other settings. For instance, in their recent work, Liang et al. (2022) extend the framework to the semantic segmentation setting, learning a latent mixture model for each image patch. In theory, the IB-INN also carries over to regression tasks in a straightforward way, although a finding a tractable latent model for this case is still an open question to our knowledge.

The main disadvantage compared to current discriminative classifiers is especially the higher computational requirement, larger networks, and slower training. Further advances in terms of INN architectures are necessary to make generative classifiers viable e. g. on mobile devices.

Overall, we are optimistic that IB-INNs or similar generative classifiers can help make deep learning models safer and more robust in real-world applications in the future, as research and availability of computational resources progress further.

6 Summary & Conclusions

6.1. Contributions in wider context

To begin with, we will relate the contributions of this thesis back to the concepts discussed in chapter 1, specifically how each chapter addresses the open problems outlined in the introduction.

Uncertainty quantification & diverse solutions: In chapters 3 and 4, we demonstrated how invertible generative models enable uncertainty estimates for regression problems, even producing full posteriors beyond the typical mean field solution, i. e. not just a scalar uncertainty value per output dimension. For high-dimensional data, the posteriors represent a diverse solution space. Here, the cINN proved especially powerful on problems such as image synthesis and colorization. In chapter 3, we used standard feed forward networks as part of the generative model in the form of 'conditioning networks', allowing the use of well-established domain-specific or pre-trained architectures. In chapter 4, we showed that the invertibility can even help in the learning of ill-posed inverse problems, improving the training over standard feed forward in addition to the uncertainty estimates.

In chapter 5, we used invertible generative models in the classification setting in a complementary way, namely as generative classifiers (GCs). Here, we demonstrated that the training procedure as well as the model architecture itself improve the uncertainty calibration over standard softmax discriminative classifiers (DCs).

Out-of-distribution data: Chapter 5 was most closely focused on the topic of out-of-distribution (OoD) data. Using invertible generative models as GCs, i. e. modeling the whole data distribution, not only enables native OoD detection on par with existing methods, but also provides a better understanding and interpretation of what OoD means, how it relates to in-distribution data, and how OoD inputs are treated in classification.

Since the first publication of the work discussed in chapter 3, Schmitt et al. (2021) have also extended the cINN to be able to detect out-of-distribution data from the viewpoint of model misspecification.

Explainability: In all chapters of this thesis, we observed that the latent space, which all invertible generative models possess, seemed to be interpretable and disentangled, making the model's data representation inherently explainable. Promising first results by Sorrenson et al. (2020) and others examine the property of disentanglement in INNs more closely and

theoretically. Especially for the IB-INN in chapter 5, the interpretable latent space allowed for new ways of visualizing and understanding decision processes.

Furthermore, the exact invertibility and tractable likelihood allow for manipulating or decomposing the network inputs to visualize the structure of the data distribution, and the models understanding of the data on the input side. This was used in chapter 3, e. g. by transferring colors to new conditioning images or interpolating between images, and in chapter 5, by generating representative images from each class as the network understands them.

6.2. Open questions and challenges

In light of the successful applications in this thesis, it is easy to miss the fact that generative invertible models still face a number of open issues limiting their adoption in some cases.

Firstly, from a purely practical standpoint, the design, training and use of INNs requires more specialized knowledge than standard feed forward networks, and support for INN components in mainstream deep learning frameworks is still somewhat lacking. The FrEIA library (`https://github.com/vislearn/FrEIA`) developed in the course of this thesis attempts to alleviate this to some degree.

Second, INNs are typically computationally more expensive than feed-forward networks, confirmed by our direct comparison in section 5.4 (see also Mackowiak et al., 2021). The high computational cost may be a problem for scientists in other fields without sufficient access to large GPU clusters, or for applications on end-user or mobile field devices. Furthermore, especially for coupling block architectures, training instabilities due to numerical errors are known to occur with the typical 32-bit floating point precision on GPUs (Behrmann et al., 2020).

While the quality of modeled distributions is on par with any other method or even superior for lower-dimensional problems ("lower-dimensional" in this case meaning $\lesssim 10^3$ dimensions), the visual quality of generated results, specifically in image generation, degrades in comparison to competing methods for higher-dimensional data.

Besides these practical limitations, some theoretical questions also remain unanswered. For instance, open question remain concerning the expressive power of coupling blocks, the number of blocks needed for a universal diffeomorphism approximator, and the types of distributions that can be modeled (see e. g. Teshima et al., 2020; Draxler et al., 2020). Many properties of the latent space are also unexplored, beyond the already mentioned work by Sorrenson et al. (2020).

Furthermore, the nature of the probability distribution of high-dimensional real-world data such as natural images is poorly understood in general, regardless of the type of generative model. This concerns questions about the intrinsic dimensionality (Pope et al., 2021), topology (Carlsson et al., 2008), tail behaviour, and more. As a result, out-of-distribution (OoD) detection is also very poorly understood from a theoretical standpoint. It is unclear if it is even possible to clearly define OoD data independent of a specific task, or if it is always dependent on the use case and human interpretation of the data (Kirichenko et al., 2020).

6.3. The future of invertible generative models

We close this chapter with a brief examination of the future potential of generative invertible models and promising directions for improvement considering the points outlined above.

Continued use in lower-dimensional problems A look at recent literature confirms what was suggested in section 3.8: in more and more cases, generative invertible models are being used for lower dimensional problems in different fields of science. Even if generative invertible models were to stay limited to such use cases for practical applications, it would make them a useful and important tool to the wider scientific community.

To further this development, the priority should be to provide easy to use tools and to improve the ease and reliability of training, as well as the computational requirements relative to available hardware. We can expect this to happen as the available software libraries and methods mature, in the same way as has occurred with feed forward networks in the past.

Improved architectures For "high-dimensional" data ($\gtrsim 10^3$ dimensions), most typically image generation, generative invertible models are currently the exception, as they often do not hold up in terms of image quality or computation efficiency when compared e. g. to GANs. One way how this may change in future is the discovery of improvements to the architecture or training process of invertible models.

Some advances have already be made, e. g. by cleverly decomposing the input images and modeling components separately (Yu et al., 2020), adapting the latent distribution for greater stability (Alexanderson and Henter, 2020), or introducing new types of invertible layers or blocks (e.g. Song et al., 2019b; Karami et al., 2019). While some of these approaches seem promising, they have not been able to close the performance gap so far.

Hybrid models Several publications have appeared in recent years that take a different approach: Instead of improving the invertible architecture, they combine generative invertible models with feed-forward architectures in advantageous ways. The cINN from chapter 3 is one such example, where relevant information is extracted by a feed-forward network, reducing the burden on the invertible part. As further examples, invertible generative models have been used to generate lower-resolution image representations that are mapped non-invertibly to high-resolution images (Rombach et al., 2020), or have been used to generate latent codes that are fed into a different (non-invertible) generative model to guide its outputs (Dorkenwald et al., 2021).

Such approaches manage to leverage the advantages of generative invertible models, i. e. full mode coverage, guarantees, latent manipulation, while at the same time working with high resolution data and reasonable hardware requirements. If any further substantial improvements to the invertible architectures themselves fail to materialize, such hybrid approaches may prove to be the most promising in future.

Innovations in the field of diffusion models We also take note of diffusion models in this context: Like INNs, diffusion models can be considered a type of likelihood model, and thus share many of the same advantages, namely full mode coverage, theoretical guarantees, etc.

But even more so than INNs, they are extremely expensive to train and evaluate in many cases, remaining reserved for only the best funded corporate research institutions at Google, OpenAI, etc., who have huge amounts of computational resources (Ramesh et al., 2022).

However, the critical breakthrough in terms of real-world applicability came more recently: (Rombach et al., 2022) presented so-called *latent diffusion models*, that use a vector-quantized auto-encoder to greatly reduce the dimensionality of the input images, and apply diffusion models to the auto-encoder's latent space. Through this simple modification, latent diffusion models have arguably become the most popular method both in conditional and unconditional image generation, considering inference can be run on consumer hardware. Since then, latent diffusion models have have an impact not only in machine learning research, but also in the general public. A slew of services and start-ups allow enthusiasts and laypeople to generate custom images, artworks, and portraits.

We bring up diffusion models here because they attest to the following two points:

Firstly, for practical use, likelihood models seem to be preferable to e.g. GAN-based models. While the image quality of diffusion models is not necessarily superior to that of state-of-the-art GANs (both often border on indistinguishable from real images), diffusion models have been catapulted to a degree of public popularity and use that GAN-based models never achieved. This is precisely due to the properties afforded by their nature as likelihood models: full mode coverage producing varied and interesting results, intuitive bidirectional manipulation and selective editing of generated images, and seemingly overall better generalization and greater ease of conditioning the outputs. These properties are shared with the generative invertible models discussed in this thesis.

Secondly, they serve as inspiration for a possible future of invertible models: Diffusion models, once deemed prohibitively expensive and impractical for real-world use despite their favourable theoretical properties, have suddenly become the primary choice for practical applications of image generation almost over night, mainly due to a simple but effective architectural modification. It seems plausible that a similarly explosive success may be possible for generative invertible models after a similar breakthrough discovery.

The following years will tell how each of these options develops, whether generative invertible models remain an important but niche tool in science and engineering, whether steady improvements or new architecture combinations allow them to flourish in a specialized subset of high-dimensional tasks, or whether a breakthrough development will make generative invertible models an established choice for any kind of generative modeling.

Bibliography

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905, 2018.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168, 2018.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=HyxQzBceg.

Simon Alexanderson and Gustav Eje Henter. Robust model training and generalisation with studentising flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (INNF+ 2020)*, volume 2, pages 25–1, 2020.

Rana Ali Amjad and Bernhard C Geiger. How (not) to train your neural network using the information bottleneck principle. *arXiv preprint arXiv:1802.09766*, 2018.

Rana Ali Amjad and Bernhard Claus Geiger. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *Intl. Conf. on Learning Representations*, 2019a.

Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019b.

Lynton Ardizzone, Jakob Kruse, Carsten Lüth, Niels Bracher, Carsten Rother, and Ullrich Köthe. Conditional invertible neural networks for diverse image-to-image translation. In *DAGM German Conference on Pattern Recognition*, pages 373–387. Springer, 2020a.

Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. *Advances in Neural Information Processing Systems*, 33, 2020b.

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.

David Barber and Felix V Agakov. The im algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, page None, 2003.

Normand J Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*, 2011.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 573–582, 2019. URL http://proceedings.mlr.press/v97/behrmann19a.html.

Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. *arXiv preprint arXiv:2006.09347*, 2020.

Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018. URL http://arxiv.org/abs/1801.04062.

Marco Bellagente, Anja Butter, Gregor Kasieczka, Tilman Plehn, Armand Rousselot, Ramon Winterhalder, Lynton Ardizzone, and Ullrich Köthe. Invertible networks or partons to detector and back again. *SciPost Physics*, 9(5):074, 2020.

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

Siddhant Bhambri, Sumanyu Muku, Avinash Tulasi, and Arun Balaji Buduru. A survey of black-box adversarial attacks on computer vision models. *arXiv preprint arXiv:1912.01667*, 2019.

Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. ISBN 9780387310732.

Christopher M. Bishop and Julia Lasserre. Generative or discriminative? getting the best of both worlds. *Bayesian statistics*, 8(3):3–24, 2007.

Vladimir Igorevich Bogachev, Aleksandr Viktorovich Kolesnikov, and Kirill Vladimirovich Medvedev. Triangular transformations of measures. *Sbornik: Mathematics*, 196(3):309, 2005.

Guillaume Bouchard and Bill Triggs. The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, pages 721–728, 2004.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. From optimal transport to generative modeling: the vegan cookbook. *stat*, 1050:22, 2017.

Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Intl. Conf. on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Jacques Bughin, Eric Hazan, Sree Ramaswamy, Michael Chui, Tera Allas, Peter Dahlstrom, Nicolaus Henke, and Monica Trench. Artificial intelligence: the next digital frontier? 2018.

Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. In *Joint Europ. Conf. on Machine Learning and Knowledge Discovery in Databases*, pages 151–166. Springer, 2017.

Gunnar Carlsson, Tigran Ishkhanov, Vin De Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1): 1–12, 2008.

Benjamin Cheatham, Kia Javanmardian, and Hamid Samandari. Confronting the risks of artificial intelligence. *McKinsey Quarterly*, pages 1–9, 2019.

Gal Chechik, Amir Globerson, Naftali Tishby, and Yair Weiss. Information bottleneck for gaussian variables. *Journal of machine learning research*, 6(Jan):165–188, 2005.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6572–6583, 2018.

Zhe Chen, Shohei Nobuhara, and Ko Nishino. Invertible neural brdf for object inverse rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Hyunsun Choi, Eric Jang, and Alexander A Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in neural information processing systems*, pages 4088–4098, 2017.

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.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

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

Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham J Mysore, Fredo Durand, and William T Freeman. The visual microphone: Passive recovery of sound from video. 2014.

Nicolaas Govert De Bruijn. *Asymptotic methods in analysis*, volume 4. Courier Corporation, 1981.

Gustavo Deco and Wilfried Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995.

Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. Learning diverse image colorization. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6837–6845, 2017.

Terrance DeVries and Graham W. Taylor. Learning confidence for out-of-distribution detection in neural networks. *CoRR*, abs/1802.04865, 2018. URL http://arxiv.org/abs/1802.04865.

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

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

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv:1605.09782*, 2017.

Michael Dorkenwald, Timo Milbich, Andreas Blattmann, Robin Rombach, Konstantinos G Derpanis, and Bjorn Ommer. Stochastic image-to-video synthesis using cinns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3742–3753, 2021.

Felix Draxler, Jonathan Schwarz, Christoph Schnörr, and Ullrich Köthe. Characterizing the role of a single coupling layer in affine normalizing flows. In *DAGM German Conference on Pattern Recognition*, pages 1–14. Springer, 2020.

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

Ethan Fetaya, Jörn-Henrik Jacobsen, and Richard S. Zemel. Conditional generative models are not robust. *CoRR*, abs/1906.01171, 2019. URL http://arxiv.org/abs/1906.01171.

Ethan Fetaya, Jörn-Henrik Jacobsen, Will Grathwohl, and Richard S. Zemel. Understanding the limitations of conditional generative models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1lPleBFvH.

Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. Multivariate information bottleneck. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 152–161. Morgan Kaufmann Publishers Inc., 2001.

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

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*, 34, 2021.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018.

Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

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.

Partha Ghosh, Arpan Losalka, and Michael J. Black. Resisting adversarial attacks using gaussian mixture variational autoencoders. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 541–548. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.3301541. URL https://doi.org/10.1609/aaai.v33i01.3301541.

Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. An information theoretic tradeoff between complexity and accuracy. In *Learning Theory and Kernel Machines*, pages 595–609. Springer, 2003.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

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.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

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.

Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. *arXiv:1705.08868*, 2017.

Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, and Kevin Murphy. Pixcolor: Pixel recursive colorization. *arXiv:1705.07208*, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Jonas Haldemann, Victor Ksoll, Daniel Walter, Yann Alibert, Ralf S Klessen, Willy Benz, Ullrich Koethe, Lynton Ardizzone, and Carsten Rother. Exoplanet characterization using conditional invertible neural networks. *arXiv preprint arXiv:2202.00027*, 2022.

Douglas Hanahan and Robert A. Weinberg. Hallmarks of cancer: The next generation. *Cell*, 144(5):646–674, March 2011. ISSN 00928674.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9995–10006. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/71e9c6620d381d60196ebe694840aaaa-Paper.pdf.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2722–2730, 2019. URL http://proceedings.mlr.press/v97/ho19a.html.

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, pages 12134–12144, 2019.

Emiel Hoogeboom, Victor Garcia Satorras, Jakub Tomczak, and Max Welling. The convolution exponential and generalized sylvester flows. *Advances in Neural Information Processing Systems*, 33:18249–18260, 2020.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Leveraging the feature distribution in transfer-based few-shot learning. *arXiv preprint arXiv:2006.03806*, 2020.

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

Gabriel Huang, Hugo Berard, Ahmed Touati, Gauthier Gidel, Pascal Vincent, and Simon Lacoste-Julien. Parametric adversarial divergences are good losses for generative modeling. *arXiv preprint arXiv:1708.02511*, 2017.

Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks. *arXiv preprint arXiv:1812.08342*, 2018b.

Uiwon Hwang, Dahuin Jung, and Sungroh Yoon. Hexagan: Generative adversarial nets for real world classification. *arXiv preprint arXiv:1902.09913*, 2019.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR'17*, pages 1125–1134, 2017.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard S. Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=BkfbpsAcF7.

Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-RevNet: deep invertible networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJsjkMb0Z.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. *arXiv preprint arXiv:1905.02325*, 2019.

Da Eun Kang, Eric W Pellegrini, Lynton Ardizzone, Ralf S Klessen, Ullrich Koethe, Simon CO Glover, and Victor F Ksoll. Emission-line diagnostics of hii regions using conditional invertible neural networks. *arXiv preprint arXiv:2201.08765*, 2022.

Mahdi Karami, Dale Schuurmans, Jascha Sohl-Dickstein, Laurent Dinh, and Daniel Duckworth. Invertible convolutional flow. *Advances in Neural Information Processing Systems*, 32, 2019.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv:1710.10196*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

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.

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.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2018.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montreal, Canada*, pages 10236–10245, 2018. URL http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.

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.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33, 2020.

Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. *Advances in neural information processing systems*, 31, 2018.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pages 5361–5370. PMLR, 2020.

Artemy Kolchinsky, Brendan D. Tracey, and David H. Wolpert. Nonlinear information bottleneck. *CoRR*, abs/1705.02436, 2017. URL http://arxiv.org/abs/1705.02436.

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.

Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. In *ACM Transactions on Graphics (ToG)*, volume 26, page 96. ACM, 2007.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Jakob Kruse, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Benchmarking invertible architectures on inverse problems. *Workshop on Invertible Neural Nets and Normalizing Flows at ICML*, 2019. URL https://arxiv.org/abs/2101.10763.

Jakob Kruse, Gianluca Detommaso, Ullrich Köthe, and Robert Scheichl. Hint: Hierarchical invertible neural transport for density estimation and bayesian inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8191–8199, 2021.

Victor F Ksoll, Lynton Ardizzone, Ralf Klessen, Ullrich Koethe, Elena Sabbi, Massimo Robberto, Dimitrios Gouliermis, Carsten Rother, Peter Zeidler, and Mario Gennaro. Stellar parameter determination from photometry using invertible neural networks. *arXiv preprint arXiv:2007.08391*, 2020.

Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on graphics (TOG)*, 33(4):1–11, 2014.

Valero Laparra, Gustavo Camps-Valls, and Jesús Malo. Iterative gaussianization: from ica to random rotations. *IEEE transactions on neural networks*, 22(4):537–549, 2011.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Europ. Conf. on Computer Vision*, pages 577–593. Springer, 2016.

Pierre L'Ecuyer. Note: On the interchange of derivative and expectation for likelihood ratio derivative estimators. *Management Science*, 41(4):738–747, 1995.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV)*, pages 35–51, 2018.

Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. *arXiv preprint arXiv:1901.11300*, 2019.

Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *arXiv preprint arXiv:2110.01167*, 2021.

Yingzhen Li, John Bradshaw, and Yash Sharma. Are generative classifiers more robust to adversarial attacks? In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3804–3814. PMLR, 2019. URL http://proceedings.mlr.press/v97/li19a.html.

Chen Liang, Wenguan Wang, Jiaxu Miao, and Yi Yang. Gmmseg: Gaussian mixture based generative semantic segmentation models. *To appear in NeurIPS 2022 proceedings*, 2022.

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.

Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. *Journal of Biomedical Optics*, 19(1):10901, January 2014. ISSN 1560-2281.

Radek Mackowiak, Lynton Ardizzone, Ullrich Kothe, and Carsten Rother. Generative classifiers as a basis for trustworthy image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2971–2981, 2021.

Jens Müller, Robert Schmier, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Learning robust models using the principle of independent causal mechanisms. In *DAGM German Conference on Pattern Recognition*, pages 79–110. Springer, 2021.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*, 5, 2019a.

Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019b. URL https://openreview.net/forum?id=H1xwNhCcYm.

Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 4723–4732, 2019c. URL http://proceedings.mlr.press/v97/nalisnick19b.html.

Whitney K Newey and Daniel McFadden. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245, 1994.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 841–848, 2001.

Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. *Advances in Neural Information Processing Systems*, 33:12685–12696, 2020.

Pablo Noever-Castelos, Lynton Ardizzone, and Claudio Balzani. Model updating of wind turbine blade cross sections with invertible neural networks. *Wind Energy*, 2021.

Jan-Hinrich Nölke, Tim Adler, Janek Gröhl, Thomas Kirchner, Lynton Ardizzone, Carsten Rother, Ullrich Köthe, and Lena Maier-Hein. Invertible neural networks for uncertainty quantification in photoacoustic imaging. In *Bildverarbeitung für die Medizin 2021*, pages 330–335. Springer, 2021.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016b.

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, 2016c.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

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

Jeong Joon Park, Aleksander Holynski, and Steven M Seitz. Seeing the world in a bag of chips. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1417–1427, 2020.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. *arXiv:1903.07291*, 2019.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

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.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021.

Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7949–7958, 2020.

Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Stefan T Radev, Frederik Graw, Simiao Chen, Nico T Mutters, Vanessa M Eichel, Till Bärnighausen, and Ullrich Köthe. Outbreakflow: Model-based bayesian inference of disease outbreak dynamics with invertible neural networks and its application to the covid-19 pandemics in germany. *PLoS computational biology*, 17(10):e1009472, 2021.

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.

Rajat Raina, Yirong Shen, Andrew Mccallum, and Andrew Y Ng. Classification with hybrid generative/discriminative models. In *Advances in neural information processing systems*, pages 545–552, 2004.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Christian Raymond and Giuseppe Riccardi. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

Stefan Reissl, Robert Brauer, and Sebastian Wolf. Radiative transfer with polaris: I. analysis of magnetic fields through synthetic dust continuum polarization measurements. *Astronomy & Astrophysics*, 593, 04 2016.

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

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

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

Robin Rombach, Patrick Esser, and Bjorn Ommer. Network-to-network translation with conditional invertible neural networks. *Advances in Neural Information Processing Systems*, 33:2784–2797, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Amelie Royer, Alexander Kolesnikov, and Christoph H. Lampert. Probabilistic image colorization. In *British Machine Vision Conference (BMVC)*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

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

Akiyoshi Sannai, Yuuki Takai, and Matthieu Cordonnier. Universal approximations of permutation invariant/equivariant functions by deep neural networks. *arXiv preprint arXiv:1903.01939*, 2019.

Robin Schirrmeister, Yuxuan Zhou, Tonio Ball, and Dan Zhang. Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features. *Advances in Neural Information Processing Systems*, 33, 2020.

R.T. Schirrmeister, P. Chrabąszcz, F. Hutter, and T. Ball. Training generative reversible networks. *arXiv:1806.01610*, 2018.

Marvin Schmitt, Paul-Christian Bürkner, Ullrich Köthe, and Stefan T Radev. Bayesflow can reliably detect model misspecification and posterior errors in amortized bayesian inference. *arXiv preprint arXiv:2112.08866*, 2021.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019a. URL https://openreview.net/forum?id=S1EHOsC9tX.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=S1EHOsC9tX.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Robert J Serfling. *Approximation theorems of mathematical statistics*, volume 162. John Wiley & Sons, 2009.

Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*, 2019.

Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Jiaming Song, Yang Song, and Stefano Ermon. Unsupervised out-of-distribution detection with batch normalization. *arXiv preprint arXiv:1910.09115*, 2019a.

Yang Song, Chenlin Meng, and Stefano Ermon. Mintnet: Building invertible neural networks with masked convolutions. *Advances in Neural Information Processing Systems*, 32:11004–11014, 2019b.

Peter Sorrenson, Carsten Rother, and Ullrich Köthe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *arXiv preprint arXiv:2001.04872*, 2020.

Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. Ai for science. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2020.

Haoliang Sun, Ronak Mehta, Hao H Zhou, Zhichun Huang, Sterling C Johnson, Vivek Prabhakaran, and Vikas Singh. Dual-glow: Conditional flow-based generative model for modality transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10611–10620, 2019.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

E. G. Tabak and Cristina V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013. doi: 10.1002/cpa.21423.

Esteban G Tabak, Eric Vanden-Eijnden, et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

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

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *arXiv preprint arXiv:2006.11469*, 2020.

Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2020.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop, ITW 2015, Jerusalem, Israel, April 26 - May 1, 2015*, pages 1–5, 2015. doi: 10.1109/ITW.2015.7133169. URL https://doi.org/10.1109/ITW.2015.7133169.

Naftali Tishby, Fernando C. N. Pereira, and William Bialek. The information bottleneck method. *CoRR*, physics/0004057, 2000. URL http://arxiv.org/abs/physics/0004057.

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

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

Jakub M Tomczak and Max Welling. Improving variational auto-encoders using convex combination linear inverse autoregressive flow. *arXiv:1706.02326*, 2017.

Darya Trofimova, Tim Adler, Lisa Kausch, Lynton Ardizzone, Klaus Maier-Hein, Ulrich Köthe, Carsten Rother, and Lena Maier-Hein. Representing ambiguity in registration problems with conditional invertible neural networks. *arXiv preprint arXiv:2012.08195*, 2020.

Ilkay Ulusoy and Christopher M. Bishop. Comparison of generative and discriminative techniques for object detection and classification. In *Toward Category-Level Object Recognition*, pages 173–195, 2006. doi: 10.1007/11957959_9. URL https://doi.org/10.1007/11957959_9.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844, 2018.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

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.

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.

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*, Lecture Notes in Computer Science, pages 134–141. Springer, Cham, September 2017.

Tailin Wu, Ian Fischer, Isaac Chuang, and Max Tegmark. Learnability for the information bottleneck. In *ICLR 2019 Workshop LLD*, 2019.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. *arXiv preprint arXiv:2002.10689*, 2020.

Jing-Hao Xue and D. M. Titterington. On the generative-discriminative tradeoff approach: Interpretation, asymptotic efficiency and classification performance. *Computational Statistics & Data Analysis*, 54(2):438–451, 2010. doi: 10.1016/j.csda.2009.09.011. URL https://doi.org/10.1016/j.csda.2009.09.011.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *CoRR*, abs/1703.01898, 2017. URL http://arxiv.org/abs/1703.01898.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Jason J Yu, Konstantinos G Derpanis, and Marcus A Brubaker. Wavelet flow: Fast training of high resolution normalizing flows. *Advances in Neural Information Processing Systems*, 33:6184–6196, 2020.

Sophia Zackrisson, SMWY Van De Ven, and SS Gambhir. Light in and sound out: emerging translational strategies for photoacoustic imaging. *Cancer research*, 74(4):979–1004, 2014.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric K Oermann. Confounding variables can degrade generalization performance of radiological deep learning models. *arXiv preprint arXiv:1807.00431*, 2018.

Daniel Zhang, Saurabh Mishra, Erik Brynjolfsson, John Etchemendy, Deep Ganguli, Barbara Grosz, Terah Lyons, James Manyika, Juan Carlos Niebles, Michael Sellitto, et al. The ai index 2021 annual report. *arXiv preprint arXiv:2103.06312*, 2021.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Europ.Conf. on Computer Vision*, pages 649–666, 2016.

Yufeng Zhang, Wanwei Liu, Zhenbang Chen, Ji Wang, Zhiming Liu, Kenli Li, Hongmei Wei, and Zuoning Chen. Out-of-distribution detection with distance guarantee in deep generative models. *arXiv preprint arXiv:2002.03328*, 2020.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV'17*, pages 2223–2232, 2017a.

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, 2017b.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682. PMLR, 2019.

Appendix: Conditional Invertible Neural Networks

A

A.1. Proofs and Assumptions

We further disambiguate the notation used in the main chapter: We include the INNs network parameters θ implicitly in f, and write $f(\cdot; \hat{\theta}) =: \hat{f}(\cdot)$, as well as $q(x \mid c, \theta) =: q(x \mid c)$ and $q(x \mid c, \hat{\theta}) =: \hat{q}(x \mid c)$, to simplify the equations. We also write the loss as a functional depending on f and φ for clarity: $\mathcal{L}_{\mathrm{cML}} = \mathcal{L}_{\mathrm{cML}}[f, \varphi]$. We also understand the argmin and argmax operations to return a set, because the extremum must not necessarily be unique. So, we write $\hat{a} \in \arg\min \mathcal{L}(a)$ instead of '$=$' as in the main chapter. We also restate the propositions according to this notation.

We consider the RVs X, Y jointly: $[X, Y] : \Omega \to (\mathcal{X}, \mu)$ with the measurable space $(\mathcal{X}, \mu)$, where $\mathcal{X} = \mathcal{X}_X \times \mathcal{X}_Y = \mathbb{R}^{d_X + d_Y}$ is the domain of x, y, and μ is the Lebesgue measure. We assume $p(X, Y)$ is absolutely continuous w.r.t. μ. We make use of the differential entropy and differential conditional entropy H, see (Cover and Thomas, 2012, Ch. 8.1) for definitions.

For the loss, we do not take a finite number of training samples into account, and instead assume that the loss $\mathcal{L}_{\mathrm{cML}}[f, \varphi]$ is the exact expecation over the training data distribution $p(X, Y)$. Strong consistency of the loss for the empirical expectation can be shown in the usual way, assuming a compact parameter space and bounded outputs of the INN, and applying Chebyshev's Inequality, but this is beyond the scope of this work, and greatly complicates the form of the propositions.

To outline the proofs in the following section: We first state and discuss some assumptions about the INN f and the conditioning network φ, which are required in our proofs of the propositions. Lemma 1 states two inequalities, which bound the loss from below. Under Assumptions 1, 2 and 3, we show that choices of f and φ exist where each bound is met exactly. Lastly, for each proposition, we show that when the respective bound from Lemma 1 is met exactly, the Proposition holds true.

Assumption 1. *We assume that the INN is chosen from a family of distributional universal approximators $\mathcal{F}$, as defined in Teshima et al. (2020).*

This simply means that the INN is in principle powerful enough to represent any distribution $p(X \mid C)$. Affine coupling block INNs specifically were recently proven to satisfy this in Teshima et al. (2020), under some requirements for the subnetworks. Their result readily generalizes to CCBs, by changing the definition for $\mathcal{H}$ in Sec. D of Teshima et al. (2020). Proving this rigorously is beyond the scope of this work, so we leave the universality of the INN as an assumption.

Assumption 2. *We assume that the conditioning network $\varphi : \mathbb{R}^{d_Y} \to \mathbb{R}^{d_C}$ is chosen from*

the set $\mathcal{G}_0$, which is defined as the set of all functions, for which the pushforward measure $p(C) \equiv p(\varphi(Y)) \coloneqq \varphi\sharp p(Y)$ is absolutely continuous w.r.t. the Lebesgue measure.

What this means intuitively, is that no subset of features can be exactly the same, or otherwise perfectly inter-dependent. This is mostly a formal requirement: if it is not fulfilled, it does damage or alter the outcome of the training, it simply means the MI is ill-defined, and Proposition 1 can not be formulated in the same way. Proposition 2 can still be shown, by slightly altering Lemma 1 to avoid using the MI, see e.g. Beaudry and Renner (2011) for an alternative formulation of the data processing inequality, that applies even if the assumption does not hold.

Assumption 3. *In addition to Assumption 2, we assume that φ is chosen from $\mathcal{G}_1$, where $\mathcal{G}_1$ is a family of universal approximators as defined in Hornik et al. (1989). Secondly, we assume the number of features extracted, $d_C \coloneqq \dim(C)$ is $\geq d_Y \coloneqq \dim(Y)$.*

This assumption is stronger than Assumption 2, and is a sufficient condition to show Proposition 2. Intuitively, it says that φ must have sufficient expressive power so that the features C can be informative enough for the INN to reproduce the true posterior. This includes the number of features, and the network power. If the features are too few, or inaccurate and uninformative, the INN will not be able to model the true posterior.

Lemma 1. *Denoting the pushforward measure as $p(C) \equiv p(\varphi(Y)) \coloneqq \varphi\sharp p(Y)$, the following inequality holds for all choices of φ, f:*

$$\mathcal{L}_{\mathrm{cML}}[f, \varphi] \underset{(a)}{\geq} H\big(X \mid \varphi(Y)\big) \underset{(b)}{\geq} H\big(X \mid Y\big) \tag{A.1}$$

Under Assumption 1, a choice of $f \in \mathcal{F}$ exists where (a) is equal, and under Assumption 3 a choice of $\varphi \in \mathcal{G}_1$ exists where (b) becomes equal.

Proof. Because $\mathcal{L}_{\mathrm{cML}}$ is equivalent to the definition of the differential cross-entropy $H_p(q(X \mid \varphi(Y)))$, (a) follows directly from the inequality that the cross entropy is $\geq$ the entropy with equality iff $p = q$ (Cover and Thomas, 2012, p. 256). Assumption 1 guarantees that f can be chosen so that $p(X \mid \varphi(Y)) = q(X \mid \varphi(Y))$, and therefore (a) becomes '='.

(b) follows from the information processing inequality (Cover and Thomas, 2012, p. 34), whereby

$$I(X, Y) \geq I(X, \varphi(Y)). \tag{A.2}$$

Writing out the terms, and subtracting the constant $H(X)$ from both sides directly yields (b). Therefore, (b) becomes '=' iff Eq. A.2 is also '='. We can construct a φ for which this is the case using Assumption 3 as a sufficient condition: We split up C into $C = [C_s, C_n]$, where $\dim(C_s) = \dim(Y)$ and $\dim(C_n) = \dim(C) - \dim(Y) \geq 0$. According to (Hornik et al., 1989, Corollary 4.2), we can choose φ such that that the mapping $\varphi_s : Y \to C_s$ is a homeomorphism, and C_n only contains constants (or noise), independent of Y. This part can effectively be ignored. Using this construction, we get

$$I(X, \varphi(Y)) = I(X, \varphi_s(Y)) = I(X, Y). \tag{A.3}$$

We used the independence of C_n and Y in the first step, and the invariance of the MI under homeomorphic transforms (Kraskov et al., 2004, Eq. A2) in the second step. □

Proposition 1. *Let $\hat{f}$ be the INN and $\hat{\varphi}$ the conditioning network that jointly minimize $\mathcal{L}_{\mathrm{cML}}$,*

where φ is optimized over $\mathcal{G}_0$ given in Assumption 2, and f over $\mathcal{F}$ given in Assumption 1. Then it holds that

$$I\big(X, \hat{\varphi}(Y)\big) = \max_{\varphi \in \mathcal{G}_0} I\big(X, \varphi(Y)\big) \tag{A.4}$$

Proof. Recall the minimization process

$$\hat{f}, \hat{\varphi} \in \underset{f,\varphi \in \mathcal{F}\times\mathcal{G}_0}{\arg\min}\, \mathcal{L}_{\mathrm{cML}}[f, \varphi] \tag{A.5}$$

By definition, we then have

$$\mathcal{L}_{\mathrm{cML}}[\hat{f}, \hat{\varphi}] = \min_{\varphi \in \mathcal{G}_0} \mathcal{L}_{\mathrm{cML}}[\hat{f}, \varphi] \tag{A.6}$$

Using Assumption 1, we have already shown that (a) in Lemma 1 will be '=' for $\hat{f}$ and arbitrary φ, the loss has its minimum possible value w.r.t. f, and the bound can be reached exactly. We use the equality to substitute $\mathcal{L}_{\mathrm{cML}}$ in Eq. A.6:

$$H(X|\hat{\varphi}(Y)) = \min_{\varphi \in \mathcal{G}_0} H(X|\varphi(Y)) \tag{A.7}$$

As a final step, we subtract the constant $H(X)$ to both sides. It can be written inside the $\min()$ operation, as it does not depend on φ. Writing out the entropies and rearranging the terms directly gives

$$-I(X, \hat{\varphi}(Y)) = \min_{\varphi \in \mathcal{G}_0} -I(X, \varphi(Y)) \tag{A.8}$$

□

Proposition 2. *Assume φ is optimized over $\mathcal{G}_1$ and $\dim(C) \geq \dim(Y)$ (Assumption 3), and f is optimized over $\mathcal{F}$ (Assumption 1). Then the following holds for $\hat{f}, \hat{\varphi} \in \arg\min \mathcal{L}_{\mathrm{cML}}[f, \varphi]$:*

$$\hat{q}(X|\hat{\varphi}(Y)) = p(X|Y) \tag{A.9}$$

Proof. We consider the KL-divergence between the true and learned posterior, write it out, and identify the loss and the conditional entropy:

$$D_{\mathrm{KL}}\big(p(X \mid Y) \| \hat{q}(X \mid \hat{\varphi}(Y))\big) \tag{A.10}$$

$$= \mathcal{L}_{\mathrm{cML}}[f, \varphi] - H(X \mid Y) \tag{A.11}$$

In Lemma 1, we showed that $H(X \mid Y)$ is the global minimum of the loss, and will be reached in training under Assumptions 1, 2, and 3 ((a) and (b) both '='). Then we can immediately see that the KL-divergence is zero, which is the case iff the two distributions are identical. □

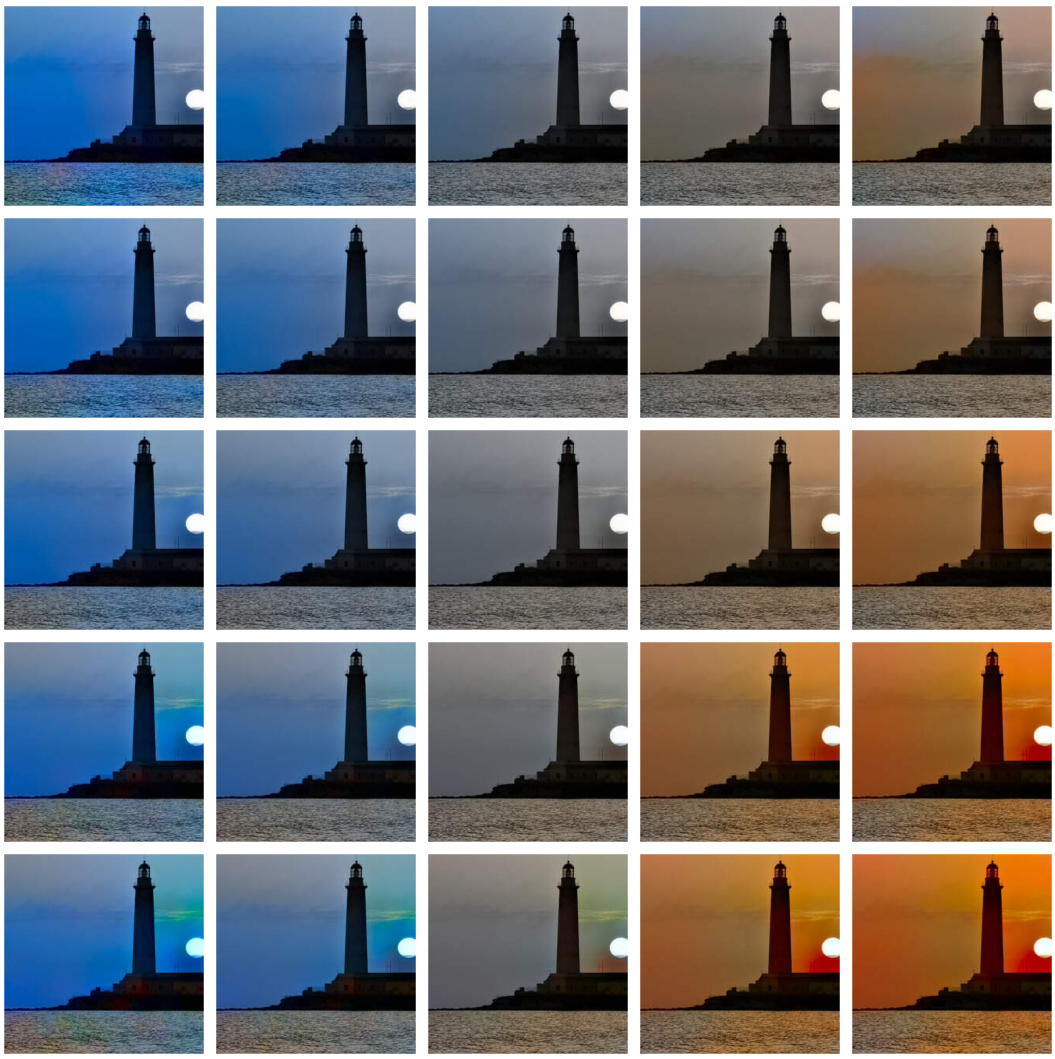
A.2. Additional Figures and Experiments

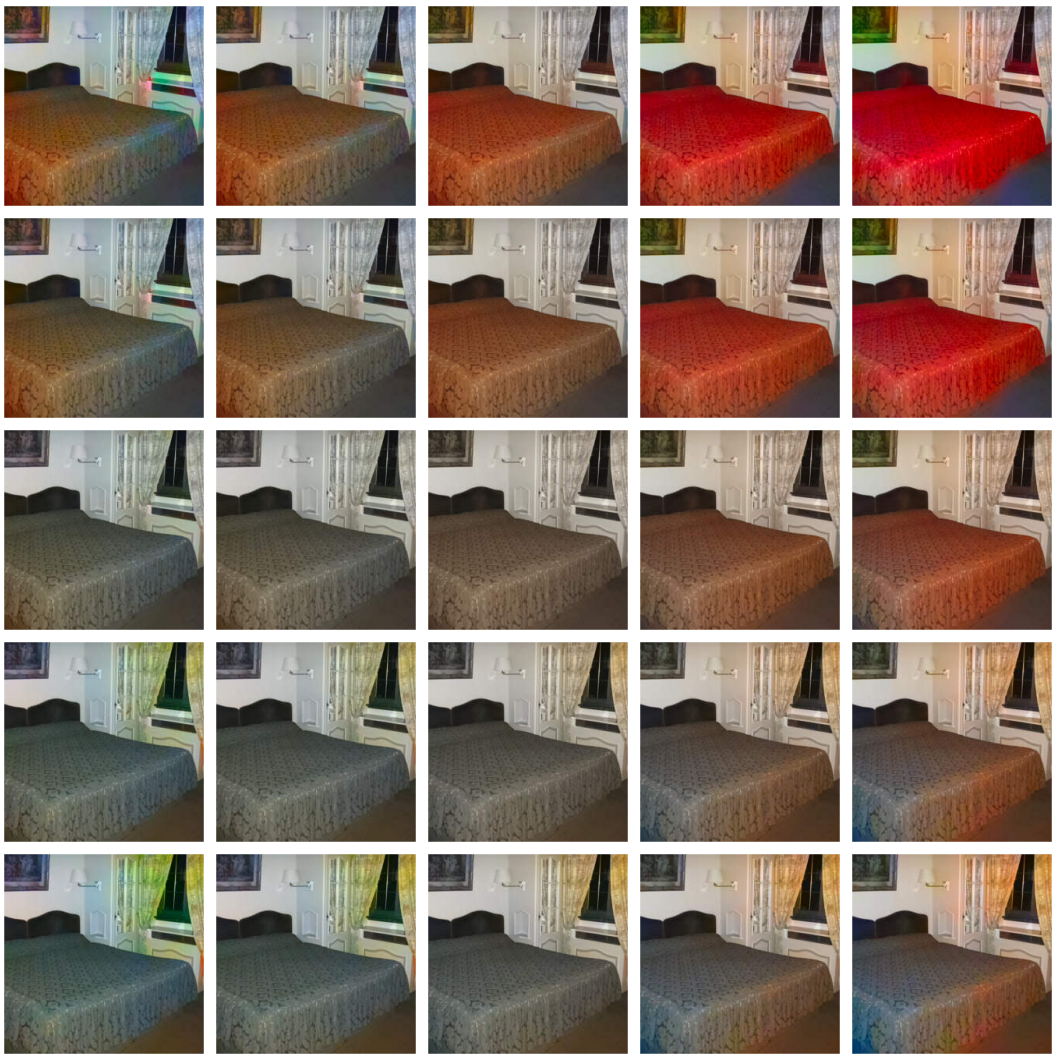
A.2.1. Colorization – Interpolations

In the following, we show 2-dimensional interpolations in latent space. Two random latent vectors $z^{(1)}$, $z^{(2)}$ are linearly combined:

$$z^* = a_1 z^{(1)} + a_2 z^{(2)}$$

with varying a_1, $a_2 \in [-0.9 \dots 0.9]$ across each axis of a grid. The center image has $z^* = 0$. Note that the images in the corners have a larger magnitude than trained for, $\|z^*\|_2 \approx 1.3\,\mathbb{E}\big[\|z\|_2\big]$, leading to some oversaturation artifacts, as in Fig. 12 of the main paper.





A.2.2. Ablation of training improvements

To demonstrate the improved stability and training speed through the improvements from section 3.7, we perform ablations, see Fig. A.1. The ablations for colorization were performed for the LSUN bedrooms task, due to training speed.

We find that for stable training at Adam learning rates of 10^{-3}, the tanh-clamping of s and Haar wavelet downsampling are strictly necessary. Without these, the network has to be trained with much lower learning rates and more careful and specialized initialization, as used e.g. in Kingma and Dhariwal (2018). Beyond this, the noise augmentation and permutations lead to the largest improvement in final result. The effect of the noise is more pronounced for MNIST, possibly becuase large parts of the image are completely black otherwise, additionally leading to a problem with sparse gradients. Note however, that the training curves of the models with and without noise augmentation is not directly comparable, as the loss differs an additional summand $\approx \log(\sigma_{\text{aug.}})$. The effect on the training speed and stability is clearly visible regardless. The initialization only improves the final result by a small margin, but also converges noticeably faster.

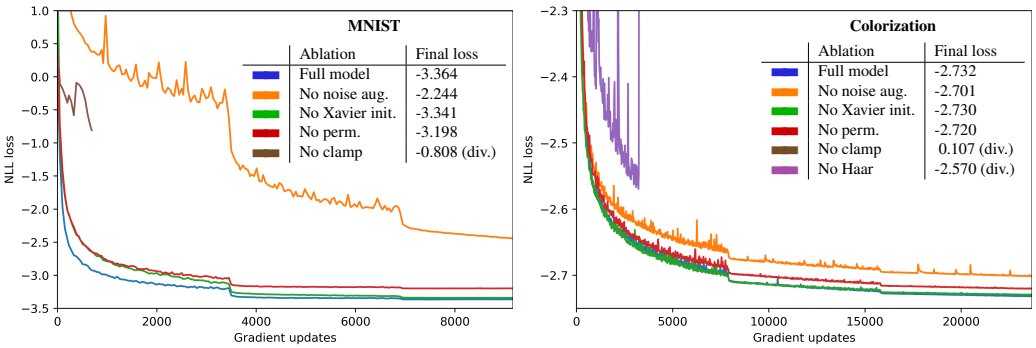


Figure A.1.: Training curves for each task, ablating the different improvements. "div." denotes that the training diverges, and the lowest loss so far is given.

A.2.3. LSUN Bedrooms

To provide a simpler model for more in-depth experiments and ablations, we additionally train a cINN for colorization on the LSUN bedrooms dataset Yu et al. (2015). We use a smaller model than for ImageNet, and train the conditioning network jointly from scratch, without pretraining. Both the conditioning input, as well as the generated color channels have a resolution of 64×64 pixels. The entire model trains in under 4 hours on a single GTX 1080Ti GPU.

To our knowledge, the only diversity-enforcing cGAN architecture previously used for colorization is the colorGAN Cao et al. (2017), which is also trained exclusively on the bedrooms dataset. Training the colorGAN for comparison, we find it requires over 24 hours to converge stably, after multiple restarts. The results are generally worse than those of the cINN, as shown in Fig. A.2. While the resulting pixel-wise color variance is slightly higher for the colorGAN, it is not clear whether this captures the true variance, or whether it is due to unrealistically colorful outputs, such as in the second row in Fig. A.2.

Metric	cINN	colorGAN
MSE best-of-8	**6.14**	6.43
Variance	33.69	**39.46**
FID	**26.48**	28.31

Table A.1.: Quantitative comparison between smaller cINN and colorGAN on LSUN bedrooms. The metrics used are explained in detail in the work of Ardizzone et al. (2020a)

Figure A.2.: Qualitative comparison between smaller cINN and colorGAN on LSUN bedrooms.

Appendix: INNs for Constrained Inverse Problems

B

B.1. Proof of correctness of generated posteriors

Lemma: *If some bijective function $f : x \to z$ transforms a probability density $p_X(x)$ to $p_Z(z)$, then the inverse function f^{-1} transforms $p_Z(z)$ back to $p_X(x)$.*

Proof: We denote the probability density obtained through the reverse transformation as $p^*_X(x)$. Therefore, we have to show that $p^*_X(x) = p_X(x)$. For the forward direction, via the change-of-variables formula, we have

$$p_Z(z) = p_X\Big(x = f^{-1}(z)\Big) \left|\det[\partial_z(f^{-1})]\right| \tag{B.1}$$

with the Jacobian $\partial_z f^{-1} \equiv \partial f_i^{-1}/\partial z_j$. For the reverse transformation, we have

$$p^*_X(x) = p_Z\Big(z = f(x)\Big) \left|\det[\partial_x f]\right|. \tag{B.2}$$

We can substitute p_Z from Eq. B.1 and obtain

$$\begin{aligned} p^*_X(x) &= p_X\Big(x = f^{-1}(f(x))\Big) \left|\det\Big[(\partial_z(f^{-1}))(\partial_x f)\Big]\right| && \text{(B.3)} \\ &= p_X(x) \left|\det\Big[(\partial_z f^{-1})(\partial_x f)\Big]\right| && \text{(B.4)} \\ &= p_X(x) \left|\det[I]\right| = p_X(x). && \text{(B.5)} \end{aligned}$$

In Eq. B.4, the Jacobians cancel out due to the inverse function theorem, i.e. the Jacobian $\partial_z(f^{-1})$ is the matrix inverse of $\partial_x f$.

Theorem: *If an INN $f(x) = [y, z]$ is trained as proposed, and both the supervised loss $\mathcal{L}_y = E[(y - f_y(x))^2]$ and the unsupervised loss $\mathcal{L}_z = D\big(q(y,z), p(y)\,p(z)\big)$ reach zero, sampling according to Eq. 4.1 with $g = f^{-1}$ returns the true posterior $p(x \,|\, y^*)$ for any measurement y^*.*

Proof: We denote the chosen latent distribution as $p_Z(z)$, the distribution of observations as $p_Y(y)$, and the joint distribution of network outputs as $q(y, z)$. As shown by Gretton et al. (2012), if the MMD loss converges to 0, the network outputs follow the prescribed distribution:

$$\mathcal{L}_z = 0 \iff q(y,z) = p_Y(y)\,p_Z(z) \tag{B.6}$$

Suppose we take a posterior conditioned on a fixed y^*, i.e. $p(x \mid y^*)$, and transform it using the forward pass of our perfectly converged INN. From this we obtain an output distribution $q^*(y, z)$. Because $\mathcal{L}_y = 0$, we know that the output distribution of y (marginalized over z) must be $q^*(y) = \delta(y - y^*)$. Also, because of the independence between z and y in the output, the distribution of z-outputs is still $q^*(z) = p_Z(z)$. So the joint distribution of outputs is

$$q^*(y, z) = \delta(y - y^*)\, p_Z(z) \tag{B.7}$$

When we invert the network, and repeatedly input y^* while sampling $z \sim p_Z(z)$, this is the same as sampling $[y, z]$ from the $q^*(y, z)$ above. Using the Lemma from above, we know that the inverted network will output samples from $p(x \mid y^*)$.

Corollary: *If the conditions of the theorem above are fulfilled, the unsupervised reverse loss $\mathcal{L}_x = D\big(q(x), p_X(x)\big)$ between the marginalized outputs of the inverted network, $q(x)$, and the prior data distribution, $p_X(x)$, will also be 0. This justifies using the loss on the prior to speed up convergence, without altering the final results.*

Proof: Due to the theorem, the estimated posteriors generated by the INN are correct, i.e. $q(x \mid y^*) = p(x \mid y^*)$. If they are marginalized over observations y^* from the training data, then $q(x)$ will be equal to $p_X(x)$ by definition. As shown by Gretton et al. (2012), this is equivalent to $\mathcal{L}_x = 0$.

B.2. Artificial data – Gaussian mixture

In section 4.2.1, we demonstrate that the proposed INN can approximate the true posteriors very well and is not hindered by the required coupling block architecture. Here we show how some existing methods do on the same task, using neural networks of similar size as the INN.

cGAN Training a conditional GAN of network size comparable to the INN (counting only the generator) and only two noise dimensions turned out to be challenging. Even with additional pre-training to avoid mode collapse, the individual modes belonging to one label are reduced to nearly one-dimensional structures.

Larger cGAN In order to match the results of the INN, we trained a more complex cGAN with 2M parameters instead of the previous 10K, and a latent dimension of 128, instead of 2. To prevent mode collapse, we introduced an additional regularization: an extra loss term forces the variance of generator outputs to match the variance of the training data prior. With these changes, the cGAN can be seen to recover the posteriors reasonably well.

Generator + MMD Another option is to keep the cGAN generator the same size as our INN, but replace the discriminator with an MMD loss (cf. Sec. 4.1.3). This loss receives a concatenation of the generator output x and the label y it was supplied with, and compares these batch-wise with the concatenation of ground truth (x, y)-pairs. Note that in contrast to this, the corresponding MMD loss of the INN only receives x, and no information about y.

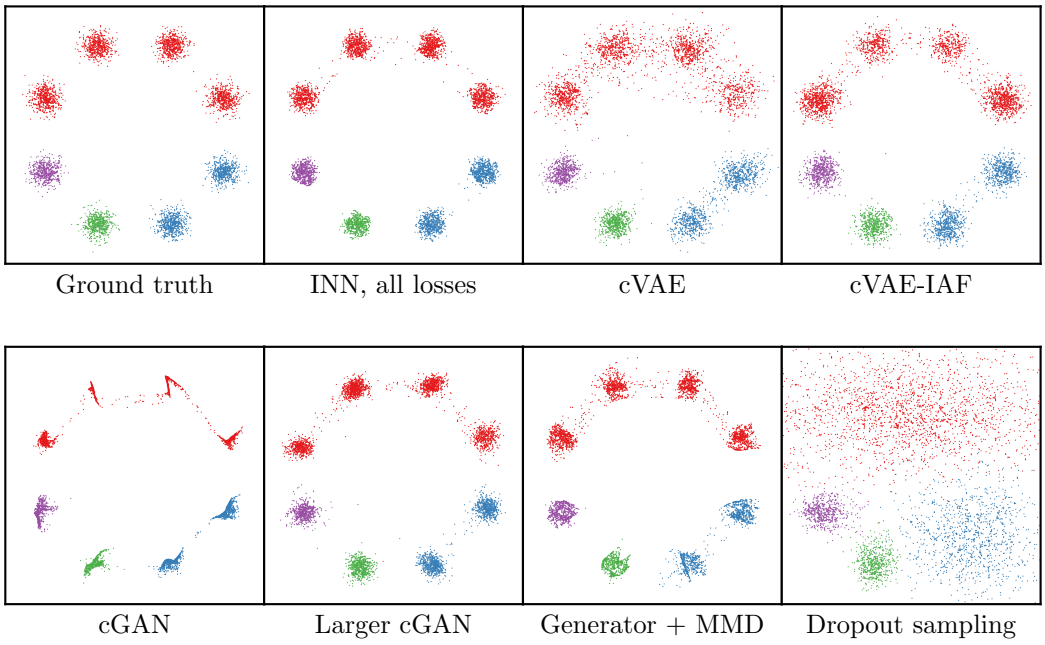


Figure B.1.: Results of several existing methods for the Gaussian mixture toy example.

For this small toy problem, we find that the hand-crafted MMD loss dramatically improves results compared to the smaller learned discriminator.

cVAE We also compare to a conditional Variational Autoencoder of same total size as the INN. There is some similarity between the training setup of our method (Fig. B.2, *right*) and that of cVAE (Fig. B.2, *left*), as the forward and inverse pass of an INN can also be seen as an encoder-decoder pair. The main differences are that the cVAE learns the relationship $x \to y$ only indirectly, since there is no explicit loss for it, and that the INN requires no reconstruction loss, since it is bijective by construction.

cVAE-IAF We adapt the cVAE to use Inverse Autoregressive Flow (Kingma et al., 2016) between the encoder and decoder. On the Gaussian mixture toy problem, the trained cVAE-IAF generates correct posteriors on par with our INN (see Fig. B.1).

Dropout sampling The method of dropout sampling with learned error terms is by construction not able to produce multi-modal outputs, and therefore fails on this task.

B.2.1. Latent space analysis

To analyze how the latent space of our INN is structured for this task, we choose a fixed label y^* and sample z from a dense grid. For each z, we compute x through our inverse network and colorize this point in latent (z) space according to the distance from the closest mode in 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. B.3).

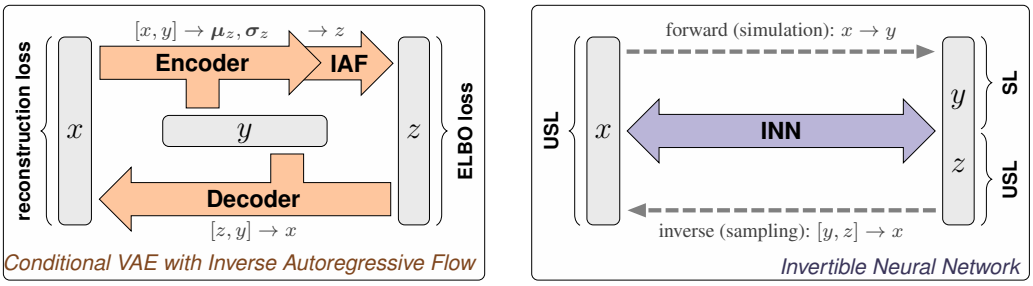


Figure B.2.: Abstraction of the cVAE-IAF training scheme compared to our INN from Fig. 4.1. For the standard cVAE, the IAF component is omitted.

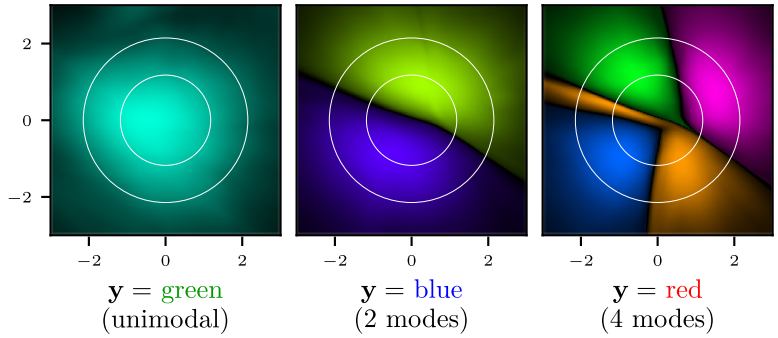


Figure B.3.: Layout of INN latent space for one fixed label y^*, colored by mode closest to $x = g(y^*, z)$. For each latent position z, the hue encodes which mode the corresponding x belongs to and the luminosity encodes how close x is to this mode. Note that colors used here do not relate to those in Fig. 4.2, and encode the position x instead of the label y. The first three columns correspond to labels *green*, *blue* and *red* Fig. 4.2. White circles mark areas that contain 50% and 90% of the probability mass of latent prior $p(z)$.

B.3. Artificial data – inverse kinematics

A short video demonstrating the structure of our INN's latent space can be found under `https://gfycat.com/SoggyCleanHog`, for a slightly different arm setup.

The dataset is constucted using gaussian priors $x_i \sim \mathcal{N}(0, \sigma_i)$, with $\sigma_1 = 0.25$ and $\sigma_2 = \sigma_3 = \sigma_4 = 0.5 \mathrel{\hat{=}} 28.65°$. The forward process is given by

$$y_1 = x_1 + l_1 \sin(x_2) + l_2 \sin(x_3 - x_2) + l_3 \sin(x_4 - x_2 - x_3) \tag{B.8}$$

$$y_2 = l_1 \cos(x_2) + l_2 \cos(x_3 - x_2) + l_3 \cos(x_4 - x_2 - x_3) \tag{B.9}$$

with the arm lenghts $l_1 = 0.5$, $l_2 = 0.5$, $l_3 = 1.0$.

To judge the quality of posteriors, we quantify both the re-simulation error and the calibration error over the test set, as in Sec. 4.2.3 of the paper. Because of the cheap simulation, we average the re-simulation error over the whole posterior, and not only the MAP estimate. In Table B.1, we find that the INN has a clear advantage in both metrics, confirming the observations from Fig. 4.6.

Table B.1.: Quantitative evaluation of the inverse kinematics experiment

Method	Mean re-sim. err.	Median re-sim. err.	Calibration err.
cVAE	0.0368	0.0307	7.78%
cVAE-IAF	0.0368	0.0307	7.81%
INN	0.0139	0.0113	0.96%

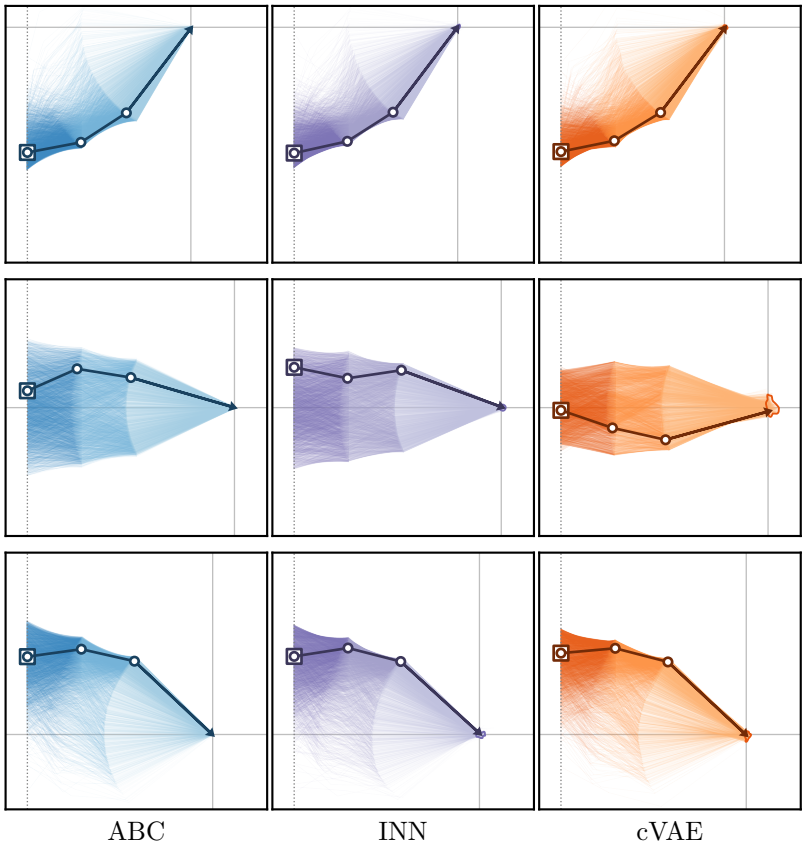


Figure B.4.: Posteriors generated for less challenging observations y^* than in Fig. 4.6.

B.4. Approximate Bayesian computation (ABC)

While there is a whole field of research concerned with ABC approaches and their efficiency-accuracy tradeoffs, our use of the method here is limited to the essential principle of rejection sampling. When we require N samples of x from the posterior $p(x \mid y^*)$ conditioned on some y^*, there are two basic ways to obtain them:

Threshold: We set an acceptance threshold ϵ, repeatedly draw x-samples from the prior, compute the corresponding y-values (via simulation) and keep those where $\text{dist}(y, y^*) < \epsilon$, until we have accepted N samples. The smaller we want ϵ, the more simulations have to be run, which is why we use this approach only for the experiment in Sec. 4.2.2, where we can afford to run the forward process millions or even billions of times.

Quantile: Alternatively, we choose what quantile q of samples shall be accepted, and then run exactly N/q simulations. All sampled pairs (x, y) are sorted by $\text{dist}(y, y^*)$ and the N closest to y^* form the posterior. This allows for a more predictable runtime when the simulations are costly, as in the medical application in Sec. 4.2.3 where $q = 0.005$.

B.5. Details of datasets and network architectures

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

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

Experiment	training data	$\dim(x)$	$\dim(y)$	$\dim(z)$	see also
Gaussian mixture	10^6	2	8	2	
Inverse kinematics	10^6	4	2	2	
Medical data	15 000	13	8	13	Wirkert et al. (2016)
Astronomy	8 772	19	69	17	Pellegrini et al. (2011)

B.5.1. Artificial data – Gaussian mixture

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

Dropout sampling: 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. Initially 100 iterations training with $\mathcal{L} = \frac{1}{N} \sum_i \|g(z_i, y_i) - x_i\|_2^2$, to separate the differently labeled modes, followed by pure GAN training.

Larger cGAN: 2 fully connected layers with 1024 neurons each for discriminator and generator, batch size 512, Adam optimizer with learning rate $8 \cdot 10^{-4}$ for the generator, SGD with learning rate $1.2 \cdot 10^{-3}$ and momentum 0.05 for the discriminator, $1.6 \cdot 10^{-3}$ weight decay for both, 0.25 dropout probabiliy for the generator at training and test time. Equal weighting of discriminator loss and penalty of output variance $\mathcal{L} = (\text{Var}_i[g(z_i, y_i)] - \text{Var}_i[x_i])^2$

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, $h = 0.5$.

cVAE: 3 fully connected layers each for encoder and decoder, ReLU activations, learning rate $2{\cdot}10^{-2}$, decay to $2.5{\cdot}10^{-5}$, Adam optimizer, batch size 25, reconstruction loss weighted 50:1 versus KL divergence loss.

B.5.2. Artificial data – inverse kinematics

INN: 6 affine coupling blocks with 3 fully connected layers each and leaky ReLU activations. Adam optimizer, decaying learning rate from 10^{-2} to 10^{-4}, multiquadratic kernel with $h = 1.2$.

cVAE: 4 fully connected layers each for encoder and decoder, ReLU activations, learning rate $5 \cdot 10^{-3}$, decay to $1.6 \cdot 10^{-5}$, Adam optimizer, batch size 250, reconstruction loss weighted 15:1 versus KL divergence loss.

B.5.3. 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 $h = 1$, weighted MMD terms by observation distance with decaying $\gamma = 0.2$ to 0.

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

cVAE: 4 fully connected layers each for encoder and decoder, ReLU activations, learning rate 10^{-3}, decay to $3.2 \cdot 10^{-6}$, Adam optimizer, batch size 25, reconstruction loss weighted 10^3:1 versus KL divergence loss.

B.5.4. 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: $k(z, z') = \exp(-\|(z - z')/h\|_2)$ with $h = 7.1$. Kernel for x-space: $k(x, x') = -\|x - x'\|_{1/2}^{1/4}$. 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.

Appendix: Information Bottleneck INN

C

C.1. Proofs and Derivations

C.1.1. Assumptions

Assumption 1 *We assume that the the sample space $\mathcal{X}$ belonging to the input RV $X : \mathcal{X} \to \mathbb{R}$ is a compact domain in $\mathbb{R}^d$, and that $p(X \mid y)$ is absolutely continuous $\forall y \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of available classes.*

The compactness of $\mathcal{X}$ is the major aspect here. However, this is always fulfilled for image data, as the pixels can only take certain range of values, and equally fulfilled for most other real-world datasets, as data representations, measurement devices, etc. only have a finite range.

Assumption 2 *We assume g_θ is from a family of universal density estimators, as defined by Definition 3 in Teshima et al. (2020). Moreover, we assume the network parameter space Θ is a compact subdomain of $\mathbb{R}^n$, g_θ and J_θ are uniformly bounded, and that the lower bound of $|\det J_\theta|$ is > 0. We also assume that g_θ and J_θ are continuous and differentiable in both X and θ.*

This is a fairly mild set of assumptions, as it is fulfilled by construction with most existing INN architectures using standard multi-layer subnetworks. See e.g. Behrmann et al. (2020); Virmaux and Scaman (2018) for details. Specifically, it holds for our tanh-clamped coupling block design (see section 3.5 and **??** for details). Note that some properties directly follow from Assumption 2: Firstly, as J_θ is uniformly bounded, this implies that g_θ is uniformly Lipschitz-continuous. Second, using Assumption 1, the domain of $Z = g_\theta(X)$ is compact, and $p(Z)$ is absolutely continuous.

C.1.2. Mutual Cross-Information as Estimator for MI

In our case, we only require $CI(X, Z_\mathbb{E})$ and $CI(Y, Z_\mathbb{E})$, but we show the correspondence for two unspecified random variables U, V, as it may be of general interest. However, note that our estimator will likely not be particularly useful outside of our specific use-case, and other methods should be preferred (e.g. MINE, Belghazi et al., 2018). Our approach has the specific advantage, that we estimate the MI of the model using the model itself. For e.g. MINE, we would require three models, one generative model, and two models that only serve to estimate the MI. Secondly, it is not clear how the large constant $d \log(\sigma)$ can be cancelled out using other approaches.

For the joint input space $\Omega = \mathcal{U} \times \mathcal{V}$, we assume that $\mathcal{U}$ is a compact domain in $\mathbb{R}^d$, and $\mathcal{V}$ is either also a compact domain in $\mathbb{R}^l$ (Case 1), or discrete, i.e. a finite subset of $\mathbb{N}$ (Case 2). In Case 1, we assume that $p(U, V)$ is absolutely continuous with respect to the Lebesgue measure, and in Case 2, $p(U|v)$ is absolutely continuous for all values of $v \in \mathcal{V}$. This is in agreement with Assumption 1.

In Case 1, $q(U)$, $q(V)$, $q(U, V)$, the densities can all be modeled separately, by three flow networks $g_\theta^{(U)}(u)$, $g_\theta^{(V)}(v)$, $g_\theta^{(UV)}(u, v)$. Although in our formulation, we are later able to approximate the latter two through the first.

In Case 2, we only model $q(U|V)$, and assume that $q(V)$ is either known beforehand and set to $p(V)$ (e.g. label distribution), or the probabilities are parametrized directly. Either way, $q(U, V) = q(U|V)q(V)$ and $q(U) = \sum_{v \in \mathcal{V}} q(U, v)$.

1 Assume that the $q(.)$ densities can be chosen from a sufficiently rich model family (e.g. a universal density estimator). Then for every $\eta > 0$ there is a model such that

$$\left|I(U, V) - CI(U, V)\right| < \eta \tag{C.1}$$

and $I(U, V) = CI(U, V)$ if $p(U, V) = q(U, V)$.

Writing out the definitions explicitly, and rearranging, we find

$$\begin{aligned} CI(U, V) = I(U, V) &+ D_{\mathrm{KL}}\big(p(U, V)\big\|q(U, V)\big) \\ &- D_{\mathrm{KL}}\big(p(U)\big\|q(U)\big) - D_{\mathrm{KL}}\big(p(V)\big\|q(V)\big) \end{aligned} \tag{C.2}$$

Shortening the KL terms to D_1, D_2 and D_3 for convenience:

$$|CI(U, V) - I(U, V)| = |D_1 - D_2 - D_3| \tag{C.3}$$

$$\leq D_1 + D_2 + D_3 \tag{C.4}$$

$$\leq 3 \max(D_1, D_2, D_3) \tag{C.5}$$

At this point, we can simply apply results from measure transport: if the g_θ are from a family of universal density estimators, we can choose θ^* to make $\max(D_1, D_2, D_3)$ arbitrarily small by matching p and q. This was shown in general for increasing triangular maps, e.g. in Hyvärinen and Pajunen (1999), Theorem 1 for an accessible proof, or Bogachev et al. (2005) for a more in-depth approach (specifically Corollary 4.2). Generality was also proven for several concrete architectures, e.g. Teshima et al. (2020); Jaini et al. (2019); Huang et al. (2018a).

For the second part of the Proposition, we note the following: if $p(U, V) = q(U, V)$, we have $D_1 = D_2 = D_3 = 0$, and therefore $CI(U, V) = I(U, V)$.

C.1.3. Loss Function $\mathcal{L}_X$

In the following, we use the subscript-notation for the cross entropy:

$$h_q(U) = \mathbb{E}_{u \sim p(U)} \left[-\log q(u)\right], \tag{C.6}$$

to avoid confusion with the joint entropy that arises with the usual notation ($h(p(U), q(U))$).

2 For the case given in the paper, that $Z_{\mathbb{E}} = g_\theta(X + \mathbb{E})$, it holds that $I(X, Z_{\mathbb{E}}) \leq CI(X, Z_{\mathbb{E}})$.

In the following, we first use the invariance of the (cross-)information to homeomorphic transforms (see e.g. Cover and Thomas (2012) Sec. 8.6). Then, we use $p(X+\mathbb{E}|X) = q(X+\mathbb{E}|X) = p(\mathbb{E})$ (known exactly) and write out all the terms, most of which cancel. Finally, we use the inequality that the cross entropy is larger than the entropy, $h_q(U) \geq h(U)$ regardless of q. The equality holds iff the two distributions are the same.

$$\begin{aligned} CI(X, Z_{\mathbb{E}}) - I(X, Z_{\mathbb{E}}) &= CI(X, X{+}\mathbb{E}) - I(X, X{+}\mathbb{E}) && \text{(C.7)} \\ &= h_q(X) - h(X) + 0 && \text{(C.8)} \\ &\geq 0 && \text{(C.9)} \end{aligned}$$

With equality iff $p(X) = q(X)$.

We now want to show that the network optimization procedure that arises from the empirical loss, in particular the gradients w.r.t. network parameters θ, are consistent with those of $CI(X, Z_{\mathbb{E}})$:

3 The defined loss is a consistent estimator for $CI(X, Z_{\mathbb{E}})$ up to a known constant, and a consistent estimator for the gradients. Specifically, for any $\epsilon_1, \epsilon_2 > 0$ and $0 < \delta < 1$ there are $\sigma_0 > 0$ and $N_0 \in \mathbb{N}$, such that $\forall N \geq N_0$ and $\forall \sigma < \sigma_0$,

$$\Pr\left(\left| CI(X, Z_{\mathbb{E}}) + d \log \sqrt{2\pi e \sigma^2} - \mathcal{L}_X^{(N)} \right| < \epsilon_1\right) > 1 - \delta$$

and

$$\Pr\left(\left\| \frac{\partial}{\partial \theta} CI(X, Z_{\mathbb{E}}) - \frac{\partial}{\partial \theta} \mathcal{L}_X^{(N)} \right\| < \epsilon_2\right) > 1 - \delta$$

holds uniformly for all model parameters θ.

The loss function is as defined in the paper:

$$\mathcal{L}_X = h_q(Z_{\mathbb{E}}) - \mathbb{E}_{x \sim p(X+\mathbb{E})}\Big[\log\big|\det J_\theta(x)\big|\Big] \tag{C.10}$$

as well as its empirical estimate using N samples, $\mathcal{L}_X^{(N)}$.

We split the proof into two Lemmas, which we will later combine.

Lemma 1 *For any $\eta_1, \eta_2 > 0$ and $\delta > 0$ there is an $N_0 \in \mathbb{N}$ so that*

$$\begin{aligned} \Pr\Big(\big|\mathcal{L}_X^{(N)} - \mathcal{L}_X\big| < \eta_1\Big) &> 1 - \delta && \text{(C.11)} \\ \Pr\Big(\big|\frac{\partial}{\partial \theta}\mathcal{L}_X^{(N)} - \frac{\partial}{\partial \theta}\mathcal{L}_X\big| < \eta_2\Big) &> 1 - \delta && \text{(C.12)} \\ & && \forall N \geq N_0 \end{aligned}$$

For the first part (Eq. C.11), we simply have to show that the uniform law of large numbers applies, specifically that all expressions in the expectations are bounded and change continuously with θ. For the Jacobian term in the loss, this is the case by definition. For the $h_q(Z_{\mathbb{E}})$-term, we can show the boundedness of $\log q$ occurring in the expectation by inserting the GMM explicitly. We find

$$-\log(q(z)) \leq \max_y[(z - \mu_y)^2/2] + const. \tag{C.13}$$

while we know that $z = g_\theta(x)$ is bounded. Therefore, the uniform law of large numbers (Newey and McFadden, 1994, Lemma 2.4) guarantees existence of an N_1 to satisfy the condition for all $\theta \in \Theta$.

For the second part (Eq. C.12), we will show that the gradient w.r.t. θ and the expectation can be exchanged, as the gradient is also bounded by the same arguments as before. We find that the conditions for exchanging expectation and gradient are trivially satisfied, again due to the bounded gradients (see L'Ecuyer (1995), assumption A1, with Γ set to the upper bound). This results in an $N_2 \in \mathbb{N}$ for which Eq. C.12 is satisfied. As a last step, we simply define $N_0 := \max(N_1, N_2)$.

Lemma 2 *For any $\eta_1, \eta_2 > 0$ there is an $\sigma_0 > 0$, so that*

$$\left\| CI_\theta(X, Z_\mathbb{E}) + d \log \sqrt{2\pi e \sigma^2} - \mathcal{L}_X \right\| < \eta_2 \tag{C.14}$$

$$\left\| \frac{\partial}{\partial \theta} \Big(CI_\theta(X, Z_\mathbb{E}) - \mathcal{L}_X \Big) \right\| < \eta_2 \tag{C.15}$$

$$\forall \sigma < \sigma_0$$

In the following proof, we make use of the $O(\cdot)$ notation, see e.g. De Bruijn (1981):

We write $f(\sigma) = O(g(\sigma)) \quad (\sigma \to 0)$ iff there exists a σ_0 and an $M \in \mathbb{R}$, $M > 0$ so that

$$\|f(\sigma)\| < M\, g(\sigma) \quad \forall \sigma \leq \sigma_0. \tag{C.16}$$

Furthermore, to discuss the limit case, it is necessary we reparametrize the noise variable $\mathbb{E}$ in terms of noise S with a fixed standard normal distribution:

$$\mathbb{E} = \sigma S \quad \text{with} \quad p(S) = \mathcal{N}(0, 1) \tag{C.17}$$

To begin with, we use the invariance of CI under the homeomorphic transform g_θ. This can be easily verified by inserting the change-of-variables formula into the definition. See e.g. Cover and Thomas (2012) Sec. 8.6. This results in

$$CI(X, Z_\mathbb{E}) = CI(Z, Z_\mathbb{E}) = h_q(Z_\mathbb{E}) - h_q(Z_\mathbb{E}|Z) \tag{C.18}$$

Next, we series expand $Z_\mathbb{E}$ around $\sigma = 0$. We can use Taylor's theorem to write

$$Z_\mathbb{E} = Z + J_\theta(Z)\mathbb{E} + O(\sigma^2) \tag{C.19}$$

We have written the Jacobian dependent on Z, but note that it is still $\partial g_\theta / \partial X$, and we simply substituted the argument. We put this into the second entropy term $h_q(Z_\mathbb{E}|Z)$ in Eq. C.18, and then perform a zero-order von Mises expansion of h_q. In general, the identity is

$$h_q(W + \xi) = h_q(W) + O(\|\xi\|) \quad (\|\xi\| \to 0), \tag{C.20}$$

and we simply put $\xi = O(\sigma^2)$ (the identity applies in the same way to the *conditional* cross-entropy). Intuitively, this is what we would expect: the entropy of an RV with a small

perturbation should be approximately the same without the perturbation. See e.g. Serfling (2009), Sec. 6 for details. Effectively, this allows us to write the residual outside the entropy:

$$h_q(Z_{\mathbb{E}}|Z) = h_q\big(Z + J_\theta(Z)\mathbb{E} + O(\sigma^2)\big|Z\big) \tag{C.21}$$
$$= h_q\big(Z + J_\theta(Z)\mathbb{E}\big|Z\big) + O(\sigma^2) \tag{C.22}$$
$$= h_q\big(J_\theta(Z)\mathbb{E}\big|Z\big) + O(\sigma^2) \tag{C.23}$$

At this point, note that $q_\theta(J_\theta(Z)\mathbb{E}|Z)$ is simply a multivariate normal distribution, due to the conditioning on Z. In this case, we can use the entropy of a multivariate normal distribution, and simplify to obtain the following:

$$-h_q(J_\theta\mathbb{E}|Z) = \mathbb{E}\left[\frac{1}{2}\log\left(\det(2\pi\sigma^2 J_\theta J_\theta^T)\right)\right] \tag{C.24}$$
$$= \mathbb{E}\left[\frac{1}{2}\log\left((2\pi\sigma^2)^d \det(J_\theta)^2\right)\right] \tag{C.25}$$
$$= d\log\sqrt{2\pi e\sigma^2} + \mathbb{E}\left[\log|\det J_\theta|\right]. \tag{C.26}$$

Here, we exploited the fact that $J_\theta(Z)$ is an invertible matrix, and used $d = \dim(Z)$. Finally, as in practice we only want to evaluate the model once, we use the differentiability of J_θ to replace

$$\mathbb{E}\left[\log|\det J_\theta(Z)|\right] = \mathbb{E}\left[\log|\det J_\theta(Z_{\mathbb{E}})|\right] + O(\sigma). \tag{C.27}$$

The residual can be written outside of the expectation as we know it is bounded from our assumptions about g_θ and J_θ (Dominated Convergence theorem).

Putting the terms together, we obtain

$$CI(X, Z_{\mathbb{E}}) = h_q(Z_{\mathbb{E}}) - d\log\sqrt{2\pi e\sigma^2} - \mathbb{E}\left[\log|\det J_\theta|\right] + O(\sigma) \tag{C.28}$$
$$= \mathcal{L}_X - d\log\sqrt{2\pi e\sigma^2} + O(\sigma) \tag{C.29}$$

Through the definition of $O(\cdot)$, Eq. C.14 is satisfied. To show that the gradients also agree (Eq. C.15), we must ensure that the $O(\sigma)$ term is uniformly convergent to 0 over θ, i.e. there is a single constant M in the definition of $O(\cdot)$ that applies for all $\theta \in \Theta$. This is directly the case, as g_θ is Lipschitz continuous and the outputs are bounded (Arzela - Ascoli theorem).

We can now combine the two Lemmas 1 and 2, to show Proposition 3.

Proposition 3 - Proof. The Proposition follows directly from Lemmas 1 and 2: for a given ϵ_1, ϵ_2 and δ, we choose each $\eta_i = \epsilon_i/2$, and apply the triangle inequality, meaning there exists an N_0 and σ_0 so that

$$\left|CI(X, Z_{\mathbb{E}}) + d\log\sqrt{2\pi e\sigma^2} - \mathcal{L}_X^{(N)}\right|$$
$$\leq \left|CI(X, Z_{\mathbb{E}}) + d\log\sqrt{2\pi e\sigma^2} - \mathcal{L}_X\right| + \left|\mathcal{L}_X - \mathcal{L}_X^{(N)}\right|$$
$$< \frac{\epsilon_1}{2} + \frac{\epsilon_1}{2}$$

And therefore $\Pr(\dots) > 1 - \delta$. Equivalently for the gradients.

C.1.4. Density Error through Noise Augmentation

For the derivation of the losses, we only assumed that X and $X + \mathbb{E} =: X_{\mathbb{E}}$ are both RVs on a domain $\mathcal{X}$, and required no further assumptions about a possible quantization of X. However, *if* X is quantized, which is mostly the case in practice, we can exploit this fact to derive a bound on the additional modeling error caused by the augmentation. To demonstrate this, we introduce the discrete, quantized data W. This is essentially the same as X, but is only defined on a finite, discrete set $\mathcal{W}$. With F regular quantization steps in each of the d dimensions, spaced by the quanitzation step size ΔX, we write

$$\mathcal{W} = \left\{0, 1\Delta X, 2\Delta X \ldots, (F-1)\Delta X\right\}^d \subset \mathcal{X}, \tag{C.30}$$

We denote probabilities of this discrete variable as upper case P and Q for true and modeled probabilities, respectively. We index the finite number of elements in $\mathcal{W}$ as w_i. For convenience, we also introduce the following notation:

$$P(w_i) =: P_i \qquad Q(w_i) =: Q_i. \tag{C.31}$$

Furthermore, we denote the noise distribution used for augmentation as $r(\mathbb{E})$ in the following, as this simplifies the notation and avoids ambiguities (it was denoted $p(\mathbb{E})$ instead for the loss derivation). From this, we can see how the distribution $p(X_{\mathbb{E}})$, which is used to train the network, can be expressed in terms of $P(W)$ and $r(\mathbb{E})$:

$$p(X_{\mathbb{E}}) = \sum_i P_i \; r(X_{\mathbb{E}} - w_i) \tag{C.32}$$

At test time, we want to recover an estimate Q_i. For standard normalizing flows, this is generally computed as

$$\tilde{Q}_i \coloneqq \frac{q(X_{\mathbb{E}} = w_i)}{r(0)} \tag{C.33}$$

Among other things, this is used to measure the bits/dim. In the most general case, $\tilde{Q}$ will not sum to 1, so it is not guaranteed to be a valid probability, indicated by the tilde. Nevertheless, we can see why this definition is sensible by considering the noise distribution r used by most normalizing flows: hereby the support of r in each dimension is smaller or equal to the quantization step size. Then, only one term in the sum in Eq. C.32 is $\neq 0$ at any point. As a result, we obtain

$$q(X_{\mathbb{E}}) = p(X_{\mathbb{E}}) \implies \tilde{Q}(W) = P(W). \tag{C.34}$$

This means that in principle a standard normalizing flow can learn the true underlying discrete distribution from the noisy augmented distribution. In other words, the augmentation process does not introduce an additional error to the density estimation.

We now apply these definitions to our setting of a Gaussian noise distribution, $r(\mathbb{E}) = \mathcal{N}(0, \sigma^2\mathbb{I})$. We consider the case where the model learns the training data distribution perfectly, i.e. $q(X_{\mathbb{E}}) = p(X_{\mathbb{E}})$. We find that Eq. C.34 no longer holds for the Gaussian case, but that the error between $\tilde{Q}(W)$ and $P(W)$ has a known bound that decreases exponentially for small σ. For convenience, we write $A \coloneqq \mathcal{N}(0; 0, \sigma^2\mathbb{I}) = (2\pi\sigma^2)^{-d/2}$. From this, we

get

$$\tilde{Q}_j = \frac{q(X_{\mathbb{E}} = w_j)}{A} = \frac{p(X_{\mathbb{E}} = w_j)}{A} \tag{C.35}$$

$$= \frac{1}{A} \sum_i P_i \mathcal{N}(w_j - w_i; 0, \sigma^2 \mathbb{I}) \tag{C.36}$$

$$= \frac{P_j \mathcal{N}(0; 0, \sigma^2 \mathbb{I})}{A} + \frac{1}{A} \sum_{i \neq j} P_i \mathcal{N}(w_j - w_i; 0, \sigma^2 \mathbb{I}) \tag{C.37}$$

$$= P_j + \underbrace{\frac{1}{A} \sum_{i \neq j} P_i \mathcal{N}(w_j - w_i; 0, \sigma^2 \mathbb{I})}_{:= \Delta P_j} \tag{C.38}$$

We are now interested in determining a bound for the error ΔP_j. Because $\|w_i - w_j\| \geq \Delta X$ for $i \neq j$, we know

$$\mathcal{N}(w_i - w_j; 0, \sigma^2 \mathbb{I}) \leq A \exp\left(-\frac{\Delta X^2}{2\sigma^2}\right). \tag{C.39}$$

From that, we obtain the following bound:

$$\Delta P_j \leq \left(\sum_{i \neq j} P_i\right) \frac{1}{A} A \exp\left(-\frac{\Delta X^2}{2\sigma^2}\right) \tag{C.40}$$

$$\leq \exp\left(-\frac{\Delta X^2}{2\sigma^2}\right) \tag{C.41}$$

C.2. Practical Loss Implementation

In the following, we provide the explicit loss implementations, as there are some considerations to make with regards to numerical tractability. Specifically, we make use of the operations `softmax, log_softmax, logsumexp` provided by major deep learning frameworks, as they avoid the most common pitfalls.

The class probabilities $q(Y)$ can be characterized through a vector Φ, with

$$q(y) = \mathrm{softmax}_y(\Phi), \tag{C.42}$$

where the subscript of the softmax operator denotes which index is selected for the enumerator. The use of the softmax ensures that w_y stay positive and sum to one. For our work, $q(Y) = p(Y)$ is known beforehand, so we leave Φ fixed to 0 (equal probability for each class). However, we also find it is possible to learn Φ as a free parameter during training. In this case, only the gradients of the $\mathcal{L}_X$ loss w.r.t. Φ should be taken, as the $\mathcal{L}_Y$ loss is no longer a lower bound, and can be exploited by sending $\Phi_y \to \infty$ for some fixed y, and $\Phi_k \to -\infty$ for all $k \neq y$. If only $\mathcal{L}_X$ is backpropagated w.r.t. Φ, this is avoided and Φ converges to the correct class weights. We use the shorthand $w_y := \log p(y)$ in the following.

With $z \coloneqq g_\theta(x + \varepsilon)$, we also have

$$\bullet \log q(y) = w_y = \text{logsoftmax}_y(\Phi) \tag{C.43}$$

$$\bullet \log q(z|y) = -\frac{1}{2}\|z - \mu_y\|^2 + const. \tag{C.44}$$

$$\bullet \log q(z) =_{y'} \left(-\frac{\|z - \mu_{y'}\|^2}{2} + w_{y'} \right) + const. \tag{C.45}$$

With this, the loss functions are evaluated as

$$\mathcal{L}_X(x) =_{y'} \left(\frac{\|z - \mu'_y\|^2}{2} - w_{y'} \right) - \log J(x) \tag{C.46}$$

$$\mathcal{L}_Y(x, y) = \text{logsoftmax}_y\Big(-\frac{\|z - \mu_{y'}\|^2}{2} + w_{y'} \Big) - w_y. \tag{C.47}$$

The constants have been dropped for convenience. The use of the `logsumexp` and `logsoftmax` operations above is especially important. Otherwise when explicitly performing the exp and log operations with 32 bit floating point numbers, the values become too large, and the loss numerically ill-defined (`NaN`).

C.3. Calibration Error Measures

In the following, we make use of the Iverson bracket:

$$\big[C\big] \coloneqq \begin{cases} 1 & \text{if } C \text{ is true;} \\ 0 & \text{otherwise,} \end{cases} \tag{C.48}$$

Firstly, we define the bin edges b_i, with $i \in \{1, \dots, K+1\}$, so that $b_1 = 0$, $b_{K+1} = 1$, and $b_{i+1} > b_i$. In practice, we choose the b_i be spaced more tightly near high and low confidences, as this is where the bulk of the predictions are made:

```
concatenate(range(0.00, 0.05, stepsize=0.01),
            range(0.05, 0.95, stepsize=0.1),
            range(0.95, 1.00, stepsize=0.01))
```

The bins themselves are then half-open intervals between the bin edges: $B_i = [b_i, b_{i+1})$ with $i \in \{1, \dots, K\}$. We now define $n^{(i)}$, the count of predictions within a confidence bin; as well as $n_c^{(i)}$, the count of *correct* predictions in that bin:

$$n^{(i)} \coloneqq \sum_{x_j} \sum_{y'} \big[p(y'|x_j) \in B_i\big] \tag{C.49}$$

$$n_c^{(i)} \coloneqq \sum_{(x_j, y_j)} \sum_{y'} \big[p(y'|x_j) \in B_i\big] \cdot \big[\arg\max_{y'}(p(y'|x_j) = y_j\big] \tag{C.50}$$

where x_j and the (x_j, y_j)-pairs are from the test set.

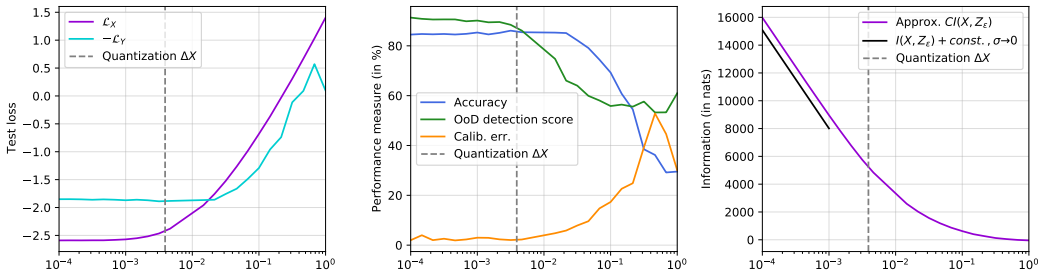


Figure C.1.: From left to right: Changes in test-loss, performance metrics, and a comparison between approximation and known slope of the true mutual information for varying values of σ (x-axis)

We define the confidence P as the center of each bin, and the achieved accuracy in this bin as Q:

$$P_i = \frac{b_i + b_{i+1}}{2} \tag{C.51}$$

$$Q_i = \frac{n_c^{(i)}}{n^{(i)}} \tag{C.52}$$

Finally, using Q and P, we define the calibration error measures, in agreement with Guo et al. (2017):

$$\text{ECE} = \sum_i \frac{n^{(i)}}{n_{\text{tot}}} |P_i - Q_i| \qquad \text{(Expected calib. err.)} \tag{C.53}$$

$$\text{MCE} = \max_i |P_i - Q_i| \qquad \text{(Maximum calib. err.)} \tag{C.54}$$

$$\text{ICE} = \sum_i (b_{i+1} - b_i) |P_i - Q_i| \qquad \text{(Integrated calib. err.)} \tag{C.55}$$

using the shorthand $n_{\text{tot}} \coloneqq \sum_i n^{(i)}$.

C.4. Additional Experiments

C.4.1. Choice of σ

Fig. C.1 shows the behaviour for 25 different models trained with σ between 10^{-4} and 10^{0} (x-axis), and fixed $\gamma = 0.2$. We find that the loss values (left) and performance characteristics (middle) do not depend on σ below a threshold that is about a factor 4 smaller than the qantization step size ΔX. Contrary to expectations from existing work on normalizing flows, the models performance does not decrease even when σ is 50 times smaller than ΔX. Detrimental effects might occur more easily if the quantization steps are larger, e.g. $\Delta X = 1/32$ as used by Kingma and Dhariwal (2018), or if the model were more powerful or less regularized (e.g. from the tanh-clamping we employ). The rightmost plot compares our approximation of $CI(X, Z_\varepsilon)$ with the asymptotic $I(X, Z_\varepsilon) + const.$ for $\sigma \to 0$, where the constant is unknown. The slope of the approximation agrees well for small σ, but breaks down for larger values.

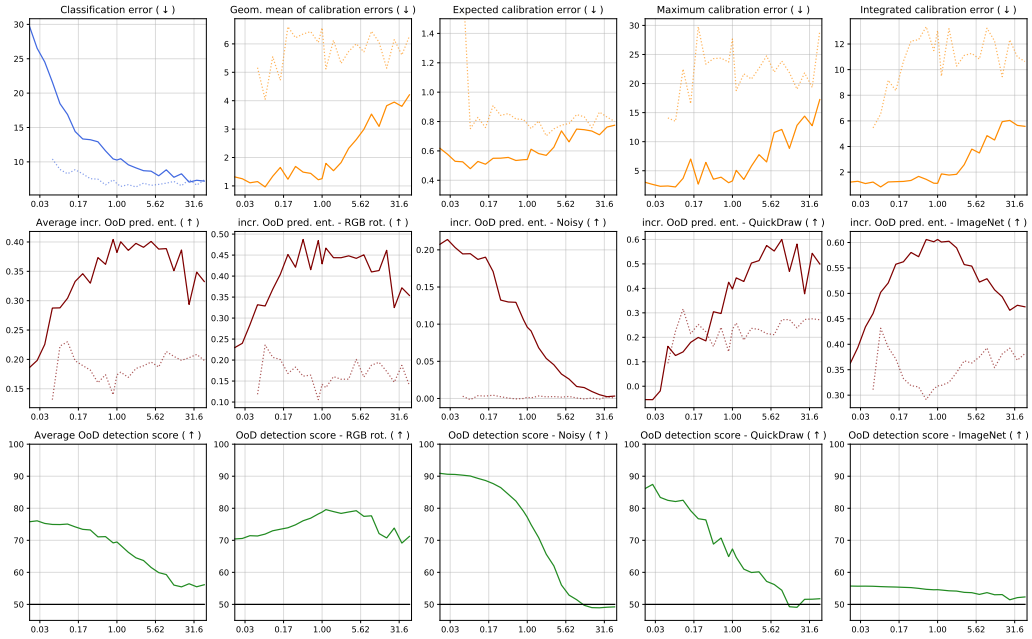


Figure C.2.: Effect of changing the parameter $\tilde{\beta}$ (x-axis) on the different performance measures (y-axis). The arrows indicate if a larger or smaller score is better. The black horizontal line in the last row indicates random performance. Details are explained in the paper. The VIB results are added as dotted lines. The VIB does not converge reliably for values of $\gamma < 0.2$, producing some otiliers e.g. for expected calibration error. This is not to claim that the IB-INN is better than the VIB or vice versa. The comparison serves to show how the IB affects GCs and DCs differently.

C.4.2. Further experiments

Figure C.2 provides all the performance metrics discussed in the paper over the range of γ.

Figure C.3 shows samples generated by the model, using different values of γ. In general, we find the quality of generated images degrades faster with γ than the interpolations between existing images. We see indications that the mass of points in latent space is offset from the learned μ_y, meaning the regions that are sampled from have not seen much training data. In contrast to the IB-INN, the standard class-NLL trained model generates fairly generic looking images for all classes, due to the collapse of class-components in latent space.

C.5. Class similarity

For the pairwise predictive uncertainty, we only consider two classes, $y \in \{1, 2\}$. We denote the distance of the class centers as $\Delta\mu = \|\mu_1 - \mu_2\|$. We assume $y = 1$ is the top prediction. This is just for simplification, as 1 and 2 can be swapped in the derivation if $y = 2$ is the top prediction. The prediction confidence c for any latent vector z is then between 0.5 and 1.0, computed as

$$c(z) = \frac{q(z \mid y{=}1)}{q(z \mid y{=}1) + q(z \mid y{=}2)} \tag{C.56}$$

Figure C.3.: Samples from four different models, each using the same architecture but a different loss. From top to bottom: class-NLL, $\gamma = 0.02$, $\gamma = 0.2$, $\gamma = 1$. In each subfigure, each row corresponds to one class. The classes from top to bottom are: plane, car, bird, cat, deer, dog, frog, horse, boat, truck.

The model's latent density is

$$q(Z) = \frac{1}{2}\mathcal{N}(\mu_1; 1) + \frac{1}{2}\mathcal{N}(\mu_2; 1) \tag{C.57}$$

This allows us to explicitly work out how the confidences will be distributed through the change-of-variables formula. Note that z can be expressed in cylindrical coordinates oriented along the line connecting μ_1 and μ_2. All the radial parts integrate out, only the position along this line is relevant. After some substitutions and simplifications, we obtain

$$p(c) = \frac{1}{A} \underbrace{\left(c - c^2\right)^{-3/2} \exp\left(-\frac{1}{2\Delta\mu^2} \log^2\left(\frac{1}{c} - 1\right)\right)}_{:=\rho(c)} \tag{C.58}$$

A is the normalization constant and has no closed form:

$$A = \int_{1/2}^{1} \rho(c) \mathrm{d}c \tag{C.59}$$

And we simply call the unnormalized density $\rho(c)$. Finally, the expected confidence $\overline{C}$ can be readily computed as

$$\overline{C} = \frac{\int c\rho(c)\mathrm{d}c}{\int \rho(c)\mathrm{d}c} \tag{C.60}$$

The expected *uncertainty* as opposed to the confidence is simply $1 - \overline{C}$.

C.6. Posterior Heatmaps

We consider the class prediction:

$$q(y|x) = \frac{q(z|y)}{\sum_{y'} q(z|y')} =: \frac{q(z|z)}{S(z)}, \tag{C.61}$$

where $p(y) = 1/M$ and the Jacobian $|\det J|$ both cancel out. We therefore plot for any class the following 'class posterior heatmap':

$$Q_{\text{Class}}(k, l, y) = \log p(w_{kl}|y) - S_{kl} \quad \text{s.t.} \quad \sum_{kl} S_{kl} = S \tag{C.62}$$

The $-S_{kl}$ term means a fixed 'image' is subtracted from each heatmap, representing the denominator, which is constant for all classes. There is some freedom to choose S_{kl}, as long as it sums to S. When distributing it evenly over space, the differences in the heatmaps between classes are hard to see by eye, compared to the common differences within the heatmaps shared across classes, which are larger by magnitude. Heuristically, we instead find the best contrast when we choose the relative weight of each S_{kl} in the following way:

$$S_{kl} = S\, \frac{r_{kl} + 0.03}{\sum_{kl}(r_{kl} + 0.03)} \tag{C.63}$$

where r_{kl} is the same as $\log p(w_{kl})$ but normalized to the $[0, 1]$-range over each image.

Comparing to Eq. C.61, we see that summing Q_{Class} over feature-space pixels gives exactly the log-prediction $\log q_\theta(y|x)$. So Q_{Class} represents a spatial decomposition of the actual predictive output:

$$q(y|x) = \exp\left(\sum_{kl} Q_{\text{Class}}(k, l, y)\right) \tag{C.64}$$